Oracle8™ Time Series Cartridge

User's Guide

Release 8.0.4

November, 1997

Part No. A57501-01

ORACLE®

Enabling the Information Age™

Oracle8 Time Series Cartridge User's Guide

Part No. A57501-01

Release 8.0.4

Contents

Send Us Your Comments xi

Preface xiii

1 Introduction

1.1 Data Cartridges 1-1
1.2 Object Relational Technology 1-2
1.3 Storing and Accessing Data 1-2
1.4 Installing the Kit 1-3
1.5 Creating Public Synonyms for Time Series Packages 1-4
1.6 Time Series Cartridge Demos (Demonstrations) 1-4
1.6.1 Running the Usage Demo 1-5
1.6.2 Usage Demo Files 1-5
1.6.3 Tables and Views in the Usage Demo 1-6

2 Time Series Concepts

2.1 Overview of Time Series Data 2-1
2.1.1 Data Generation for a Time Series 2-2
2.1.2 Historical Data 2-4
2.2 Calendars 2-5
2.2.1 Frequency and Precision 2-6
2.2.2 Calendar Datatypes 2-7
2.2.3 Overview of Calendar Definition 2-8
2.2.4 Deriving Calendar Exceptions from Time Series Data 2-9

2.3 Time Series Cartridge Architecture 2-10
2.4 Storage of Time Series Data 2-11
2.4.1 Flat IOT Storage 2-12
2.5 Interfaces to Time Series and Time Scaling Functions 2-12
2.5.1 Instance-Based Interface 2-13
2.5.2 Reference-Based Interface 2-15
2.6 Consistency of Time Series Data 2-18
2.6.1 Rules for Time Series Consistency 2-18
2.6.2 Enforcing Time Series Consistency with Security Views 2-18
2.6.3 Bulk Loading and Consistency 2-20
2.7 Calendar Functions 2-20
2.7.1 End-User Functions 2-21
2.7.2 Product-Developer Functions 2-21
2.8 Time Series Functions 2-23
2.8.1 Time Series Datatypes 2-23
2.8.2 Conventions and Semantics 2-24
2.8.3 Extraction, Retrieval, and Trim Functions 2-26
2.8.4 Shift Functions 2-27
2.8.5 SQL Formatting Functions 2-27
2.8.6 Aggregate Functions 2-28
2.8.7 Arithmetic Functions 2-28
2.8.8 Cumulative Sequence Functions 2-29
2.8.9 Moving Average and Sum Functions 2-29
2.8.10 Conversion Functions 2-30
2.9 Time Scaling Functions 2-31
2.9.1 Time Scaling on Collections 2-32
2.9.2 Time Scaling in the GROUP BY Clause 2-33

3 Time Series Usage

3.1 Using the Cartridge 3-1
3.1.1 Step 1: Create the Underlying Storage (Table) 3-1
3.1.2 Step 2: Define a Calendar 3-2
3.1.3 Step 3: Load Time Series Data 3-4
3.1.4 Step 4: Create a Security View and INSTEAD OF Triggers 3-4
3.1.5 Step 5: Create a Reference-Based View 3-6

3.1.6 Step 6: Validate Time Series Consistency 3-7
3.1.7 Step 7: Formulate Time Series Queries 3-7
3.2 Loading Time Series Data 3-9
3.2.1 Bulk Loading 3-10
3.2.2 Incremental Loading 3-12
3.3 Deriving Calendar Exceptions 3-13
3.3.1 Deriving Exceptions Using a Calendar and Table of Dates (Approach 1) 3-13
3.3.2 Deriving Exceptions Using Two Time Series Parameters (Approach 2) 3-14
3.4 Using Product-Developer Functions 3-16

4 Calendar Functions: Reference

CombineCals 4-2
DeleteExceptions 4-7
DisplayValCal Procedure 4-10
EqualCals 4-17
GetOffset 4-20
InsertExceptions 4-23
IntersectCals 4-27
InvalidTimeStampsBetween 4-31
IsValidCal 4-34
IsValidDate 4-40
NumInvalidTimeStampsBetween 4-43
NumOffExceptions 4-46
NumOnExceptions 4-49
NumTimeStampsBetween 4-52
OffsetDate 4-55
SetPrecision 4-58
TimeStampsBetween 4-61
UnionCals 4-65
ValidateCal 4-69

5 Time Series and Time Scaling Functions: Reference

Cavg 5-3
Cmax 5-5
Cmin 5-7
Cprod 5-9
Csum 5-11
DeriveExceptions 5-13
Display 5-15
DisplayValTS Procedure 5-18
ExtractCal 5-26
ExtractDate 5-28
ExtractTable 5-30
ExtractValue 5-32
Fill 5-34
First 5-40
FirstN 5-42
GetDatedElement 5-44
GetNthElement 5-46
GetSeries 5-48
IsValidTS 5-51
Lag 5-59
Last 5-62
LastN 5-64
Lead 5-66
Mavg 5-69
Msum 5-72
Scaleup 5-74
ScaleupAvg 5-76
ScaleupCount 5-78
ScaleupFirst 5-80
ScaleupLast 5-82

ScaleupMax 5-84
ScaleupMin 5-86
ScaleupSum 5-88
TrimSeries 5-90
TSAdd 5-93
TSAvg 5-97
TSCount 5-99
TSDivide 5-101
TSMax 5-105
TSMaxN 5-107
TSMedian 5-109
TSMin 5-111
TSMinN 5-113
TSMultiply 5-115
TSProd 5-119
TSStdDev 5-121
TSSubtract 5-123
TSSum 5-127
TSVariance 5-129
ValidateTS 5-131

A Error Messages

Examples

2–1	Overview of Calendar Definition	2-8
3–1	Create a Stock Data Table	3-1
3–2	Create a Calendar of Business Days	3-3
3–3	Create a Security View	3-4
3–4	Create an INSTEAD OF Trigger	3-4
3–5	Create a Reference-Based View	3-6
3–6	Formulate Time Series Queries	3-7

Figures

1–1 Tables and Views in the Time Series Usage Demo 1-7
2–1 Data Generation in Equities Markets 2-3
2–2 Historical Data for Stocks 2-4
2–3 Time Series Architecture 2-11
2–4 Example of ORDTNumTab Collection Type 2-14
2–5 Relationship of Input and Output Time Series in Moving Average/Sum 2-30
2–6 Time Scaling from Daily to Monthly Frequency 2-31

Tables

1–1	Time Series Cartridge Demos	1-4
1–2	Time Series Cartridge Usage Demo Files	1-5
2–1	Frequencies	2-6
2–2	Precisions	2-7
2–3	End-User Calendar Functions	2-21
2–4	Product-Developer Calendar Functions	2-22
2–5	Extraction Functions	2-26
2–6	Retrieval and Trim Functions	2-27
2–7	Shift Functions	2-27
2–8	SQL Formatting Functions	2-27
2–9	Aggregate Functions	2-28
2–10	Arithmetic Functions	2-29
2–11	Cumulative Sequence Functions	2-29
2–12	Moving Average and Sum Functions	2-30
2–13	Conversion Functions	2-31
2–14	Scaleup Functions for Collections	2-33
4–1	SetPrecision and Timestamp of 19-Sep-1997 09:09:09	4-58
4–2	Errors Repaired by ValidateCal	4-70
5–1	Lagging a Time Series by Two Days	5-60
5–2	Leading a Time Series by Two Days	5-67

Send Us Your Comments

Oracle8 Time Series Cartridge User's Guide, Release 8.0.4

Part No. A57501-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

You can send comments to us in the following ways:

- e-mail: nedc_doc@us.oracle.com
- fax: 603.897.3269 Attn: Data Cartridge Documentation
- postal service:
  Oracle Corporation
  Data Cartridge Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you would like a reply, please include your name and contact information.

Preface

This manual describes how to use the Oracle8 Time Series Cartridge.

Intended Audience

This manual is intended for anyone who is interested in storing, retrieving, and manipulating time series data in an Oracle database, including developers of specialized cartridges.

Structure

This manual contains the following chapters and appendix:

Chapter 1	Introduces data cartridges, object types, and the contents of the Time Series cartridge.
Chapter 2	Explains time series concepts and operations.
Chapter 3	Explains important procedures for using the cartridge.
Chapter 4	Provides reference information on calendar functions.
Chapter 5	Provides reference information on time series and time scaling functions.
Appendix A	Lists potential errors, their causes, and user actions to correct them.

Related Documents

For information added after the production of this manual, see the README file in the following directory:

- ORACLE_HOME/ord/ts/admin (Solaris systems)

- ORACLE_HOME\ord80\ts\admin (Windows NT systems)

The location of the README file is platform-dependent.

For more information, see the following manuals in the Oracle8 documentation set:

- *PL/SQL User's Guide and Reference*
- *Oracle Call Interface Programmer's Guide*
- *Oracle8 Application Developer's Guide*

Conventions

In this manual, the Oracle8 Time Series Cartridge is sometimes referred to as the Time Series cartridge.

The following conventions are also used in this manual:

Convention	Meaning
. . .	A vertical ellipsis in an example means that lines not directly related to the example have been omitted.
. . .	A horizontal ellipsis in an example means that part of the statement or command not directly related to the example has been omitted
boldface text	Boldface type in text indicates a term defined in the text.
italicized text	Italic type in text indicates emphasis or a user-defined variable, schema name, or object datatype.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Introduction

The Oracle8 Time Series Cartridge is an extension to Oracle8 that provides storage and retrieval of timestamped data through object types. The cartridge is a building block for applications rather than being an end-user application in itself. It consists of datatypes along with related functions for managing and processing time series data.

For example, applications can use this cartridge to process historical data derived from financial market transactions, such as trades of stocks, bonds, and mutual fund shares. In such applications, the functions included with the Time Series cartridge let you conveniently perform operations ranging from the simple to the complex, such as:

- finding the opening, closing, low, and high prices for a stock on a specific date
- calculating monthly volumes for a stock for a specific year
- deriving the 30-day moving average for a stock over a year

Time series applications have certain distinct requirements and some degree of commonality. The time series datatypes accommodate the commonality and support extensions that address application-specific requirements. With this cartridge, time series data can be managed more conveniently and efficiently than is possible using only traditional datatypes and user-defined functions.

With the Time Series cartridge, you can use or adapt existing tables for time series applications, or you can create new tables. You can also extend the capabilities of the cartridge to add or modify functions and to create customized calendars.

1.1 Data Cartridges

Within the Oracle Network Computing Architecture™ (NCA), data cartridges facilitate the storage and retrieval of complex datatypes required by nontraditional

database applications, such as geographic information systems, imaging, workflow, document management, and digital libraries. These applications are built using components or modules that support the capture, input, processing, analysis, storage, retrieval, and display of the complex datatypes.

A **data cartridge** is the mechanism by which clients, application-specific servers, and database servers can be easily and reliably extended. The Oracle8 Time Series Cartridge provides support for time series domain-specific types, functions, and interfaces. The Time Series cartridge focuses on a set of time series data representation and access mechanisms sufficient to support many applications and the development of more specialized cartridges.

1.2 Object Relational Technology

The objects option makes Oracle8 an **object-relational** database management system, which means that users can define additional kinds of data -- specifying both the structure of the data and the ways of operating on it -- and use these types within the relational model. This approach adds value to the data stored in a database.

Oracle8 with the objects option stores structured business data in its natural form and allows applications to retrieve it that way. For that reason it works efficiently with applications developed using object-oriented programming techniques.

1.3 Storing and Accessing Data

The Time Series cartridge can store time series data in the database under transactional control.

Once stored in the database, this data can be queried and retrieved by finding a row in a table that contains the primary key (which includes the timestamp) using the various alphanumeric columns (attributes) of the table. Typical queries might include the following:

- Select the closing price from a stock market data table where the ticker (stock symbol) is XYZ and the date is 30-May-1997.
- Select the 30-day moving average of stock XYZ for the month of May 1997.

Applications access and manipulate time series data using SQL or PL/SQL™. See the *Oracle8 SQL Reference* manual for information on SQL syntax.

1.4 Installing the Kit

The exact procedure for installing the Time Series cartridge is platform-dependent. For more detailed information, see the README.txt file for your platform which can be found either in $ORACLE_HOME/ord/ts/admin (UNIX systems) or $ORACLE_HOME\ord80\ts\admin (Windows NT systems). On many platforms, automated default installation is available using Oracle Installer; however, on some platforms manual installation of the database objects is required.

This section describes the steps required to manually install the Time Series cartridge. It is intended for users who want to customize the database installation or who simply want a better understanding of the Time Series installation process.

To use the Time Series cartridge, at least the following software components must be installed: Oracle8 Enterprise Edition (RDBMS), PL/SQL (on platforms on which it is a separate installation option), and the Oracle8 Time Series Cartridge. These components can be installed all at once, or the Oracle8 Time Series Cartridge can be added to an existing Oracle8 Enterprise Edition release 8.0.4 installation that includes PL/SQL.

Follow these steps to perform a manual installation of the Time Series cartridge:

1. If the ORDSYS user does not already exist, decide on a password.

   ORDSYS is the standard schema for Oracle-supplied cartridges, and it has special privileges for data cartridges.

2. Create and start the database.

   For detailed information about database creation and startup, see the *Oracle8 Installation and Configuration Guide* for your operating system, the *Oracle8 Administrator's Guide*, and the *Oracle8 Concepts* manual.

3. Install the UTLREF package, which is needed by the Time Series cartridge:

   ```
   SVRMGR> @$ORACLE_HOME/rdbms/admin/utlrefld.sql
   ```

4. If the ORDSYS account does not already exist, create it:

   ```
   SVRMGR> CREATE USER ORDSYS IDENTIFIED BY <password>;
   ```

   The <password> is the password you chose in step 1.

5. Set the privileges for ORDSYS user:

   ```
   SVRMGR> GRANT connect,resource,create library TO ORDSYS;
   ```

6. Install the Time Series cartridge datatypes and stored procedures:

```
SVRMGR> CONNECT ORDSYS/<password>
SVRMGR> @$ORACLE_HOME/ord/ts/admin/ordtinst.sql
```

The user group PUBLIC is granted execute privilege on all Time Series cartridge datatypes and packages.

1.5 Creating Public Synonyms for Time Series Packages

All Time Series cartridge packages and datatypes are installed under the ORDSYS schema, and all users must include the ORDSYS schema name when referring to these packages and datatypes. However, to simplify references to packages, you can define public synonyms for the Calendar and TimeSeries packages.

To create public synonyms, run the ordtsyn.sql file supplied with the Time Series cartridge in the admin directory. The ordtsyn.sql file creates the following public synonyms:

```
CREATE PUBLIC SYNONYM TimeSeries for ORDSYS.TimeSeries;
CREATE PUBLIC SYNONYM Calendar for ORDSYS.Calendar;
```

1.6 Time Series Cartridge Demos (Demonstrations)

Table 1–1 shows the demos (SQL scripts and related files that demonstrate capabilities) included with the Time Series cartridge. This table includes a description of each demo and the default directory in which its files are installed. (The exact location and directory syntax are platform-dependent.)

Table 1–1 Time Series Cartridge Demos

Description	Directory
Usage demo for end users and product developers who want to use existing cartridge feature	demo/usage
Advanced-developer demo for those who want to extend the cartridge features	demo/extend
OCI demo showing how to call cartridge functions using the Oracle Call Interface	demo/oci
PRO*C/C++ demo showing how to call cartridge functions in applications created using the Oracle Pro*C/C++ Precompiler	demo/proc
Developer/2000™ demo showing how to call cartridge functions in an Oracle Forms™ application	demo/dev2k

Each demo is described in a README file in its directory.

The usage demo, described in this section, is a working example of the usage of the cartridge. The example models a historical database of stock pricing and provides sample queries over this stock database.

The usage demo is designed to guide you through the cartridge in a step-by-step fashion. It includes example code for creating and populating tables and calendars, constructing security views, constructing views to synthesize the interface to time series functions, and running some example queries.

1.6.1 Running the Usage Demo

After the cartridge has been installed, you can run the usage demo by going to the appropriate directory (see Table 1–1) and invoking the demo.sql procedure, as follows:

```
% svrmgrl
SVRMGR> @demo
```

1.6.2 Usage Demo Files

The usage demo files include examples of bulk and incremental loading, defining tables, calendars, and views for the time series cartridge, and running example queries. These files are listed in Table 1–2.

Table 1–2 Time Series Cartridge Usage Demo Files

File	Description
demo.sql	Main procedure file
stockdat.ctl	SQL*Loader® control file
stockdat.dat	SQL*Loader data file for time series data
tables.sql	DDL for tables
popcal.sql	Defines calendars and populates calendar table
securevw.sql	Example security view
refvw.sql	Example reference-based view
queries.sql	Example time series queries (SQL)
queplsql.sql	Example time series queries (PL/SQL)
calqueries.sql	Example calendar queries (PL/SQL)

Table 1–2 Time Series Cartridge Usage Demo Files (Cont.)

File	Description
incload.sql	Incremental load script
stockinc.ctl	SQL*Loader control file for incremental load
stockinc.dat	SQL*Loader data file for incremental time series data
cleanup.sql	Deletes database objects created by the demo

1.6.3 Tables and Views in the Usage Demo

The stock database consists of three tables:

- *stockdemo* stores historical time series pricing data.
- *stockdemo_calendars* stores instances of calendars.
- *stockdemo_metadata* maintains mapping between time series (here, for tickers) and calendars. This metadata table would not be needed if all tickers were associated with a private calendar that is named by the ticker or if a single calendar were shared by all tickers. The *stockdemo_metadata* table allows, for example, five tickers to be associated with one calendar, two tickers with another calendar, and so on.

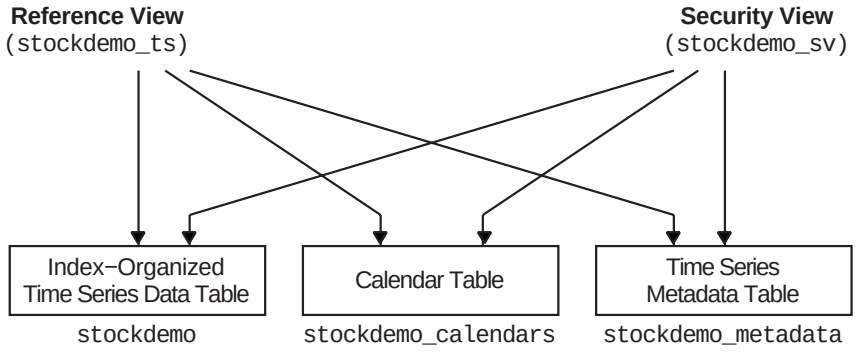
To maintain time series consistency and provide a collection-based interface for time series functions, two views are constructed using these tables.

- *stockdemo_sv* is a security view. A security view should be used for any insert, update, and delete operations to time series data.
- *stockdemo_ts* is a reference-based view. A reference-based view provides an object model of a time series, and it can be used for efficient read-only access using Time Series cartridge functions.

The security view ensures that insert, update, and delete operations maintain a time series that is consistent with the associated calendar. (Time series consistency is explained in Section 2.6.) The security view and the reference-based view access the three underlying tables. The reference-based view synthesizes references to collections. (Reference-based views are explained in Section 2.5.2.)

Figure 1–1 shows the layering and interdependencies between underlying tables and the security and reference-based views.

Figure 1–1 Tables and Views in the Time Series Usage Demo



2

Time Series Concepts

This chapter explains concepts related to the Oracle8 Time Series Cartridge, and it provides information on using the cartridge. It covers the following topics:

- overview of time series data
- calendars
- Time Series cartridge architecture
- storage of time series data
- the collection-based interface to functions
- consistency of time series data
- the reference-based interface
- overviews of each major function type: calendar, time series, time scaling
- using the Time Series cartridge

2.1 Overview of Time Series Data

A time series is a set of timestamped data entries. A time series allows a natural association of data collected over intervals of time. For example, summaries of stock market trading or banking transactions are typically collected daily, and are naturally modeled with time series.

A time series can be regular or irregular, depending on how predictably data points arrive or occur:

- In a **regular** time series, the data arrives predictably at predefined intervals. For example, daily summaries of stock market data form a regular time series, and one such time series might be the set of trade volumes and opening, high, low, and closing prices for stock XYZ for the year 1997.

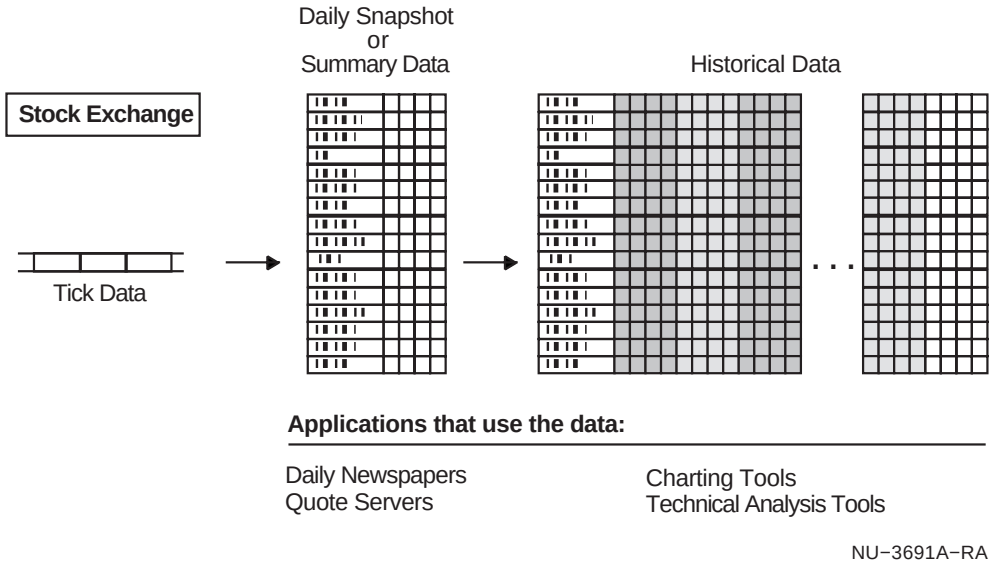
- In an **irregular** time series, unpredictable bursts of data arrive at unspecified points in time, or most timestamps cannot be characterized by a repeating pattern. For example, account deposits and withdrawals from a bank automated teller machine (ATM) form an irregular time series. An irregular time series may have long periods with no data or short periods with bursts of data.

2.1.1 Data Generation for a Time Series

Data generation for a time series begins with individual transactions, such as trades on a stock exchange or purchases of products. Each transaction has a timestamp and sufficient information to identify that transaction uniquely (such as a stock ticker or a product ID), as well as other pertinent information (such as the price and information to identify the party initiating the purchase or sale).

Individual transaction data is typically *rolled up* to produce summary data for a meaningful time period, such as a daily summary indicating the trade volume and the opening, high, low, and closing prices for each stock traded that day. This summary data is collected to produce historical data, such as a table of all daily volumes and opening, high, low, and closing prices for all stocks traded for the year 1997. For example, Figure 2–1 shows how data related to securities on a stock exchange is generated.

Figure 2–1 Data Generation in Equities Markets



In Figure 2–1, each trade on the stock exchange includes several items of information, including a ticker and a price (for example, stock XYZ at 37.50). The daily summary data includes the opening, high, low, and closing prices for each ticker (for example, for XYZ: 37.75, 38.25, 37.00, 37.625). The daily data for each ticker is appended to the historical data for the ticker. The daily data is used for such purposes as quote server applications and listing in the next day's newspapers; the historical data is used by such applications as price and volume charting and technical analysis.

The data-collection model for historical data has the following characteristics:

- At daily intervals, historical data is updated with daily summary data (main update cycle).
- At some period after the main update cycle, corrections of the daily summary data may need to be applied.
- Queries may be executed at any time, even during the update cycle.
- Queries do not observe the current day's summary information until after the main update cycle has completed.

This historical data is modeled using multiple regular time series.

The Time Series cartridge and the Oracle8 utilities, with their bulk-loading capabilities and transactional semantics, are well suited for the requirements of time series data generation.

2.1.2 Historical Data

The Time Series cartridge is especially useful in dealing with historical data. This type of data typically has relatively simple metadata but massive data storage requirements. That is, the data attributes (columns) are relatively few and easy to understand (such as ticker, volume, and opening, high, low, and closing prices); however, the number of rows is enormous (for example, data for all listed stocks for all trading days for several years). Moreover, the number of functions that users might want to perform on the data is large: for example, finding various sums, counts, maximum and minimum values, averages, number of trading days between two dates, moving average, and so on.

Figure 2–2 shows an example of historical data stored in a database.

Figure 2–2 Historical Data for Stocks

Ticker	Tstamp	Open	High	Low	Close	Volume
XYZ	01-02-1997	21.75	22.75	21.50	22.00	352,000
XYZ	01-03-1997	22.125	22.50	21.00	21.75	530,000
XYZ	01-06-1997	21.625	22.00	21.625	21.875	490,000
...	...	...	...	...	...	...
YZA	01-02-1997	44.25	44.25	43.50	43.875	125,000
YZA	01-03-1997	43.75	44.25	43.75	44.125	97,000
YZA	01-06-1997	44.25	44.50	44.125	44.125	107,000
...	...	...	...	...	...	...

Stock market historical databases have the following general characteristics:

- Multiple stocks, each identified by the ticker symbol, can be stored in the database.
- Each stock can have multiple attributes (*ticker, tstamp, open, high, low, close, volume*).
- Each stock has one or more associated time series.

- Each time series has an associated calendar (see Section 2.2).

This kind of financial historical data is used in examples in this manual and in the usage demo (see Section 1.6) provided with the Time Series cartridge.

2.2 Calendars

A calendar maps human-meaningful time values to underlying machine representations of time. Typically, a calendar is associated with a time series.

For example, a business day calendar can define the days of the week on which stocks are traded. The holidays when trading does not occur are also in the calendar. The following are key components of a calendar:

- Frequency

  A **frequency** specifies the granularity of the calendar representation. Examples of frequencies are *second, minute, hour, day, month,* and *year*.

- Pattern

  The **pattern** specifies the repeating pattern of frequencies and an anchor date that identifies the date of an occurrence of the frequency. For example, if the frequency is set to *day*, the pattern can define which days of the week are included in the calendar. For example, a pattern of '0,1,1,1,1,1,0' over a *day* frequency defines a calendar over all weekdays. If an anchor date of 01-Jun-1997 (or any Sunday) is specified, then the 7-day pattern begins each Sunday; and Sunday and Saturday (0) are excluded from the calendar, while Monday through Friday (1) are included in the calendar.

- Exceptions

  Exceptions are timestamps that do not conform to the calendar pattern but that are significant for the calendar definition. There are two kinds of exceptions: off-exceptions and on-exceptions:

  – An **off-exception** is an exception to the 1-bits in the pattern, and thus is a timestamp to be excluded from the calendar. For example, to ensure that Wednesday, 25-Dec-1996, is excluded from the calendar when Wednesdays normally are included, define that date as an off-exception.

  – An **on-exception** is an exception to the 0-bits in the pattern, and thus is a timestamp to be included in the calendar. For example, to ensure that Saturday, 28-Jun-1997, is included in the calendar when Saturdays are excluded, define that date as an on-exception.

On-exceptions can also be used without a pattern, that is, when a zero pattern is specified for the time series. An irregular time series can be constructed by defining a frequency, a zero (0) pattern, and an on-exceptions list. In this case, the on-exceptions list defines the timestamps that constitute the irregular time series. An example of such an irregular time series is a calendar of dates for quarterly dividend payments or earnings announcements.

2.2.1 Frequency and Precision

Each frequency has an associated integer code that is used in function calls. Table 2–1 lists the supported frequencies and their integer codes.

Table 2–1 Frequencies

Frequency (Every:)	Integer Code
second	1
minute	2
hour	3
day	4
month	6
year	8

Each frequency has an associated precision. Time Series cartridge functions require that input timestamps be of the precision of the frequency associated with the calendar. (The SetPrecision function is the exception: this function takes a frequency and a timestamp and returns a timestamp that conforms to the associated calendar.)

A timestamp that is not consistent with the frequency is said to be **imprecise**. For example, a timestamp of 09-Sep-1997 is imprecise if it is input to a function that is dealing with a calendar whose frequency is 6 (*month*) or 8 (*year*). When you create a calendar, all timestamps used in the calendar definition (the anchor date for the pattern, the *minDate* and *maxDate*, and all off- and on-exceptions) must be precise with respect to the frequency. For example, the calendar will not be valid if you specify a frequency of *month* and an anchor date (*patAnchor*) of 02-Jan. (The calendar datatypes and their attributes are presented in Section 2.2.2.)

Table 2–2 shows the frequencies, their precision conventions, and an example timestamp of each precision.

Table 2–2 Precisions

Frequency (Every:)	Precision Convention	Example Result
second	MM-DD-YYYY HH24:MI:SS	09-09-1997 09:09:09
minute	MM-DD-YYYY HH24:MI:00	09-09-1997 09:09:00
hour	MM-DD-YYYY HH24:00:00	09-09-1997 09:00:00
day	MM-DD-YYYY 00:00:00 (midnight)	09-09-1997 00:00:00
month	MM-01-YYYY 00:00:00 (midnight of first day of month)	09-01-1997 00:00:00
year	01-01-YYYY 00:00:00 (midnight of first day of year)	01-01-1997 00:00:00

2.2.2 Calendar Datatypes

The Time Series cartridge provides the following calendar datatypes. (Time series datatypes are described in Section 2.8.1.)

- calendar (ORDTCalendar) (Sections 2.2.3 and 3.1.2 contain calendar definitions with explanatory notes.)

```
CREATE TYPE ORDSYS.ORDTCalendar AS OBJECT (
  caltype INTEGER,
  name VARCHAR2(256),
  frequency INTEGER,
  pattern ORDSYS.ORDTPattern,
  minDate DATE,
  maxDate DATE,
  offExceptions ORDSYS.ORDTExceptions,
  onExceptions ORDSYS.ORDTExceptions);
```

- pattern (ORDTPatternBits and ORDTPattern)

```
CREATE TYPE ORDSYS.ORDTPatternBits AS VARRAY(32500) OF INTEGER;

CREATE TYPE ORDSYS.ORDTPattern AS OBJECT (
  patBits ORDSYS.ORDTPatternBits,
  patAnchor DATE);
```

- exception (ORDTExceptions)

```
CREATE TYPE ORDSYS.ORDTExceptions AS VARRAY(32500) OF DATE;
```

All Time Series cartridge datatypes are installed under the ORDSYS schema, and all users must include the ORDSYS schema name when referring to these datatypes.

2.2.3 Overview of Calendar Definition

To define a calendar, you create a table in which to store calendar definitions and then store a row for each calendar to be defined.

The following example creates a table named *stockdemo_calendars* and defines a calendar named *BusinessDays*. The *BusinessDays* calendar includes Mondays through Fridays, but excludes 28-Nov-1996 and 25-Dec-1996. Explanatory notes follow the example.

Example 2–1 Overview of Calendar Definition

```
CREATE TABLE stockdemo_calendars of ORDSYS.ORDTCalendar (
    name CONSTRAINT calkey PRIMARY KEY);

INSERT INTO stockdemo_calendars VALUES(
  ORDSYS.ORDTCalendar( ❶
        0 ❷
        'BusinessDays', ❸
        4, ❹
        ORDSYS.ORDTPattern( ❺
                ORDSYS.ORDTPatternBits(0,1,1,1,1,1,0),
                     TO_DATE('01-JAN-1995','DD-MON-YYYY')),
        TO_DATE('01-JAN-1990','DD-MON-YYYY'), ❻
        TO_DATE('01-JAN-2001','DD-MON-YYYY'),

        ORDSYS.ORDTExceptions(TO_DATE('28-NOV-1996','DD-MON-YYYY'),
              TO_DATE('25-DEC-1996','DD-MON-YYYY')), ❼
        ORDSYS.ORDTExceptions() ❽
        ));
```

Notes on Example 2–1:

❶ The *stockdemo_calendars* table has rows of type ORDTCalendar, which is described in Section 2.2.2.

❷ 0 indicates that this is an exception-based calendar (the only type of calendar currently supported).

❸ *BusinessDays* is the name of this calendar.

❹ 4 is the frequency code for *day*.

❺ The calendar's pattern consists of an excluded occurrence followed by five included occurrences followed by an excluded occurrence (`0,1,1,1,1,1,0`). Because the frequency is daily and because the anchor date (01-Jan-1995) is a Sunday, Sundays are excluded, Mondays through Fridays are included, and Saturdays are excluded.

❻ The calendar begins at the start of 01-Jan-1990 and ends at the start of 01-Jan-2001.

❼ 28-Nov-1996 and 25-Dec-1996 are off-exceptions (that is, excluded from the calendar).

Note: All exceptions (off- and on-) must be specified in ascending sorted order.

---

❽ *ORDSYS.ORDTExceptions()* indicates that there are no on-exceptions (that is, no Saturday or Sunday dates to be included in the calendar).

2.2.4 Deriving Calendar Exceptions from Time Series Data

When you want to create calendars that conform to time series data, you can use the DeriveExceptions function to simplify the process. You can use one of two approaches with DeriveExceptions, depending on your needs and the requirements for each approach:

- The first approach uses a DeriveExceptions call with input parameters of a calendar and an ORDTDateTab instance. (An ORDTDateTab instance is a collection of dates; these dates can be compared with the set of valid timestamps implied by the calendar.) A calendar is returned with the appropriate exception lists populated. This returned calendar is defined on the pattern and frequency of the input calendar, and it is consistent with the timestamps of the ORDTDateTab instance.
- The second approach uses a DeriveExceptions call with two time series as input parameters. The first time series is essentially an expansion of a pattern-only calendar. As in the first approach, a calendar is returned with the appropriate exception lists populated. The returned calendar is defined on the pattern and frequency of the calendar of the first input time series, and it is consistent with the timestamps of the second input time series.

While the first approach can be performed in a single step, the second approach requires an additional step (before DeriveExceptions is called) in order to construct the first time series.

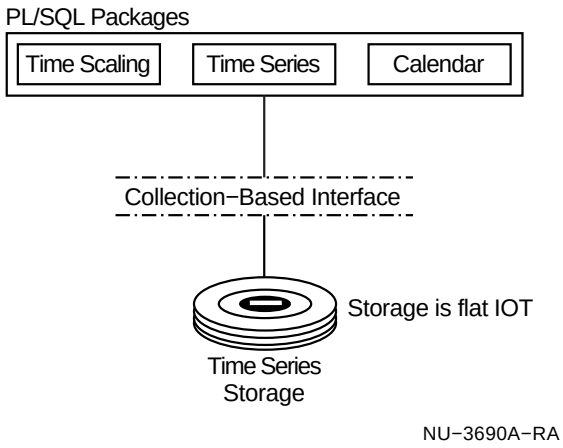
Although the first approach is simpler in practice, the second approach has significant performance advantages when you need to define multiple calendars that have the same frequency and pattern but different exception lists. The first approach is less efficient than the second approach in this case, because the internal implementation of the first approach generates a collection of dates based on the input calendar parameter. If you need to derive exceptions for multiple calendars defined on the same frequency and pattern, this date-generation operation is performed multiple times. You can avoid these multiple date-generation operations by using the second approach.

Section 3.3 contains more detailed information about using each approach to deriving calendar exceptions.

2.3 Time Series Cartridge Architecture

Figure 2–3 shows the Time Series cartridge architecture. At the lowest level, a storage option is required, and for the initial release this must be a flat index-organized table (IOT). For future releases, nested IOT and VARRAY storage options are planned. The actual cartridge consists of PL/SQL packages for calendar, time series, and time scaling functions. In addition, a collection-based interface between time series storage and the packaged functions is provided.

Figure 2–3 Time Series Architecture



The rest of this chapter describes this architecture, working from bottom to top in Figure 2–3:

- storage of time series data
- interfaces (instance-based and reference-based) to time series and time scaling functions
- calendar functions
- time series functions
- time scaling functions

The chapter concludes with the steps for using the Time Series cartridge.

2.4 Storage of Time Series Data

The underlying storage model must satisfy the following requirements for time series data:

- The data must be stored in timestamp order, because many of the analytical functions require access to the data in this order.
- It must be possible to couple a time series with a calendar. In general, each time series is associated with a distinct instance of a calendar. Time series and time

scaling functions generally require an input parameter that is typed to contain both a time series and its associated calendar.

2.4.1 Flat IOT Storage

A time series is stored as multiple rows in a flat index-organized table (IOT).[1] Each row stores a ticker, a timestamp, and composite data. This storage option is shown in Figure 2–2.

The flat IOT storage model provides efficient utilization of disk storage (for example, the timestamp data is stored once per composite entry) and allows flexibility in queries; however, it does require that users perform certain manual actions:

- An object view must be defined to form the collection and to couple a time series with a calendar (see Section 2.5.2).
- A security view should be defined to ensure data integrity (see Section 2.6.2).

2.5 Interfaces to Time Series and Time Scaling Functions

The interfaces to the time series and time scaling functions rely on the following aspects of the Time Series cartridge architecture:

- Time series data is stored as relational data (in a flat IOT), one timestamp per row.
- Calendars are stored in object tables.
- Time series and time scaling functions expect time series data and calendars to be formatted as objects. A time series object is typically the first parameter to a function.

Two basic interfaces to time series and time scaling functions are defined:

- an instance-based object interface

  In the instance-based interface, the first input parameter to a time series function is an instance of a time series (for example, ORDTNumSeries).

- a reference-based object interface

  In the reference-based interface, the first input parameter to a time series function is a reference to a time series (for example, ORDTNumSeriesIOTRef). The reference-based interface requires that you provide enough descriptive infor-

[1] A time series could be stored in a standard table; however, for performance reasons it is *highly* recommended that you use an IOT rather than a standard table.

mation to enable the functions to execute dynamic SQL to obtain an instance of a time series.

The datatypes related to the instance-based and reference-based interfaces (for example, ORDTNumSeries and ORDTNumSeriesIOTRef) are discussed in Sections 2.5.1 and 2.5.2.

Note that both types of interfaces return only instances of time series (for example, ORDTNumSeries). Also, because nesting of time series functions is allowed (for example, SELECT (Lead(Mavg, ...) ...)), the instance-based interface is used internally for the second and subsequent levels of nesting.

When possible, you should use the reference-based interface. Although this interface may be difficult to understand initially, it offers significant performance advantages over the instance-based interface. The examples in this manual emphasize the reference-based interface.

2.5.1 Instance-Based Interface

Time series functions operate on instances of time series objects (for example, an ORDTNumSeries). An instance of a time series object includes a name field, an instance of a calendar, and an instance of a time series. For example, as the following type definitions for a numeric time series show, ORDTNumTab defines a collection and ORDTNumSeries bundles a calendar instance with a collection:

```
CREATE TYPE ORDSYS.ORDTNumCell AS OBJECT (tstamp DATE, value NUMBER);
CREATE TYPE ORDSYS.ORDTNumTab AS TABLE OF ORDTNumCell;
CREATE TYPE ORDSYS.ORDTNumSeries AS OBJECT (
     name VARCHAR2(256),
     cal ORDTCalendar,
     series ORDTNumTab
);
```

For a numeric time series, the time series data is contained in the ORDTNumTab structure. This structure is a table of a DATE column and a NUMBER column, and is also known as a *collection*.

Figure 2–4 shows an example of an ORDTNumTab collection type

Figure 2–4 Example of ORDTNumTab Collection Type

Tstamp	Value
01-01-1996	22.00
01-02-1996	23.00
...	...
12-31-1996	...

Functions such as Mavg (Moving Average, described in Section 2.8.9) use the ORDTNumTab structure as the source data for performing computations, and they use the ORDTCalendar type to enable navigation through the time series data. The calendar-based navigation is especially useful for functions such as Mavg, which has as input parameters the starting date (*startDate*) and ending date (*endDate*) for which to return moving averages and an integer (*k*) indicating the look-back window (*k* denoting the number of timestamps, including the current one, over which to compute the moving average). Calendar-based navigation is used to determine the date that is *k-1* timestamps previous to *startDate*.

Although time series functions operate on time series instances, they are invoked from SQL using a REF to a time series. For a numeric time series, this type is an ORDTNumSeriesIOTRef. (Section 2.5.2 explains the use of REFs in the reference-based interface.) The REF contains enough information so that time series functions can derive the instance (ORDTNumSeries) at runtime (using dynamic SQL).

The convention of defining an interface on a DATE column and a *single* NUMBER column provides a uniform interface for time series functions. Because the underlying IOT that stores time series data may have multiple NUMBER columns, the view defining the REF also maps the underlying storage to conform to the two-column interface defined by the ORDTNumSeries type.

The following are the key aspects of the instance-based interface to time series functions:

- The input parameter of a time series function is a REF to a time series object (for example, ORDTNumSeriesIOTRef).
- Time series functions operate on time series instances (for example, ORDTNumSeries).

Note: In addition to numeric series, a character time series is also provided, with the types ORDTVarchar2Series and ORDTVarchar2SeriesIOTRef.

- You should use a view to construct the reference descriptor.
- The REF couples the calendar with the time series.
- Instances of calendars are typically stored in a table separate from time series data.
- It is important to ensure and maintain consistency between time series data and the corresponding calendar. Section 2.6 discusses consistency of time series data, including ways of ensuring consistency.

2.5.2 Reference-Based Interface

The Time Series cartridge provides a reference-based interface for time series and time scaling functions.

This interface provides efficient performance, especially when only a portion of the time series is accessed. The performance benefit of this interface results from the fact that at runtime the reference-based interface materializes only those rows within the specified date range, as opposed to materializing the entire collection of rows from the time series.

The reference-based interface uses the ORDTNumSeriesIOTRef and ORDTVarchar2SeriesIOTRef types, which include a REF to a calendar, plus several literal values. At runtime, reference-based time series functions use these literal values to form and execute a SQL statement (using dynamic SQL) that derives an instance of a time series that contains only the timestamps needed for this instance. The time series function determines which timestamps are needed based on the *minDate* and *maxDate* parameters to the function.

The ORDTNumSeriesIOTRef type is defined as follows:

```
CREATE TYPE ORDSYS.ORDTNumSeriesIOTRef AS OBJECT
   (
   name                VARCHAR2(256),
   cal                 REF ORDSYS.ORDTCalendar,
   table_name          VARCHAR2(256),
   tstamp_colname      VARCHAR2(30),
   value_colname       VARCHAR2(30),
   qualifier_colname   VARCHAR2(30),
```

```
  qualifier_value     VARCHAR2(4000)
  );
```

The attributes of the ORDTNumSeriesIOTRef type are as follows:

- *name* is the name of the time series.
- *cal* is a REF to the calendar.
- *table_name* is the fully qualified name of the flat IOT.
- *tstamp_colname* is the name of tstamp column in the flat IOT.
- *value_colname* is the name of the value column in the flat IOT (for example, *close* for the closing price).
- *qualifier_colname* is the name of the column that identifies a time series instance (for example, *ticker*).
- *qualifier_value* is the value of the column that identifies a time series instance (for example, *ACME*, which is the ticker for Acme Corporation).

In the Time Series cartridge usage demo, the view *stockdemo_ts* uses the reference-based interface to time series functions. The *stockdemo_ts* view determines which calendar should be coupled with the time series by accessing the calendar (*stockdemo_calendars*) and metadata (*stockdemo_metadata*) tables. The pricing data is accessed through the underlying table containing historical time series pricing data (*stockdemo*). For an illustration of the relationship between the reference-based view and the underlying tables in the Time Series cartridge usage demo, see Figure 1–1 in Chapter 1.

The *stockdemo_ts* view is defined as follows:

```
CREATE OR REPLACE VIEW stockdemo_ts(ticker,open,high,low,close,volume) AS
  SELECT meta.tickername,
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' open NumSeries',
              Ref(cal), 'ordtdev.stockdemo',
              'tstamp', 'open', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' high NumSeries',
              Ref(cal), 'ordtdev.stockdemo',
              'tstamp', 'high', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' low NumSeries',
               Ref(cal), 'ordtdev.stockdemo',
               'tstamp', 'low', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
```

```
      substr(meta.tickername, 1, 230) || ' close NumSeries',
             Ref(cal), 'ordtdev.stockdemo',
             'tstamp', 'close', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
      substr(meta.tickername, 1, 230) || ' volume NumSeries',
             Ref(cal), 'ordtdev.stockdemo',
             'tstamp', 'volume', 'ticker', meta.tickername)
  FROM stockdemo_metadata meta, stockdemo_calendars cal
  WHERE meta.calendarname = cal.name;
```

Depending on which column is selected, a different literal value is applied as an attribute of the ORDTNumSeriesIOTRef type. For example, for the following query:

```
SELECT ORDSYS.TimeSeries.Mavg(close,
                              to_date('02-DEC-96','DD-MON-YY'),
                              to_date('31-DEC-96','DD-MON-YY'),
                              10)
FROM ORDTDEV.stockdemo_ts
WHERE ticker='ACME';
```

The literal value *close* is used as the *value_colname* column name. The other attributes of the ORDTNumSeriesIOTRef type include the timestamp column name (*tstam*p), a qualifying column name (*ticker*), and the actual value of the qualifying column (*meta.tickername*).

The implementation of time series functions uses the information stored in the ORDTNumSeriesIOTRef type to generate the appropriate dynamic SQL statement at runtime. Using the preceding example, to instantiate a time series object (that is, to convert an ORDTNumSeriesIOTRef to an ORDTNumSeries), the Mavg function generates a query that performs the following action (with the logic shown, not the exact syntax):

```
SELECT tstamp, close
FROM ordtdev.stockdemo_ts
WHERE ticker='ACME' and tstamp BETWEEN <a date range adjusted
                  to reflect the 10-day window and the
                  calendar, including any holidays>;
```

The Mavg function computes the moving average and returns the result as a time series instance (ORDTNumSeries). For more information about the Mavg function, see Section 2.8.9.

2.6 Consistency of Time Series Data

Most time series and time scaling functions rely on calendars that are consistent with time series data.[1] By assuming a time series is consistent with its calendar, time series and time scaling functions can use the calendar as a basis for navigation of time series data.

Time series consistency must be maintained; otherwise, functions might raise exceptions or return incorrect results.

2.6.1 Rules for Time Series Consistency

For a time series to be consistent, the following must be true:

- All timestamps are sorted in ascending sequence.
- There are no duplicate timestamps.
- All timestamps match the precision of the calendar.
- No timestamps are beyond the bounds of the calendar (*minDate* and *maxDate*).
- All timestamps conform to the pattern specification, except those listed in the off-exceptions list or the on-exceptions list.
- The time series data is contiguous. That is, between the smallest (earliest) and largest (latest) timestamps in the time series, the time series data contains timestamps for all valid calendar timestamps.

If some mechanism is not used to enforce these consistency rules, accidental or malicious actions could destroy the integrity of the time series data. For example, a user might delete rows from the middle of the time series, rather than being restricted to deleting rows at the beginning and the end of the date range for the time series.

2.6.2 Enforcing Time Series Consistency with Security Views

Enforcing time series consistency can be accomplished with a security view. A **security view** is a relational view of time series data that uses INSTEAD OF triggers to maintain time series consistency. (For an explanation of INSTEAD OF triggers, see the *Oracle8 Server Concepts* manual.) The security view is intended to be used for limited or moderate insert, update, and delete operations; it is not intended for bulk changes to time series data.

[1] An exception is the Fill function, which can be used to add pairs of timestamps and values to make a time series consistent with the calendar.

The cartridge demo (see Section 1.6) includes a security view defined in the file demo/usage/securevw.sql. This security view:

- enables view updates to be propagated to the underlying table
- ensures that the underlying table can only be updated using a view mechanism, provided that users are granted update access to the security view and not granted update access to the underlying table
- ensures that update, delete, and insert operations affecting time series data are constrained to conform to the calendar associated with the time series
- purifies timestamps to match the precision of the calendar

2.6.2.1 Precision

Timestamps are *purified* to match the precision of the calendar. For example, for a calendar with a *day* frequency, any hour, minute, and second values in the input timestamp are set to zero. Only purified timestamps are inserted into a time series, and timestamps are purified if necessary before insert and delete operations.

2.6.2.2 INSTEAD OF Triggers

INSTEAD OF triggers enforce rules on insert, delete, and update operations. These rules maintain time series data that conforms to the associated calendar.

For *insert* operations, the following rules apply:

- For an empty time series, the new timestamp must be a valid date in the calendar.
- For a non-empty time series, an insertion is allowed immediately after the last timestamp or immediately before the first timestamp, but nowhere else.

For *delete* operations, the following rules apply:

- For an empty time series, an exception is raised.
- For a non-empty time series, only the first or last timestamp can be deleted.

For *update* operations, the following rules apply:

- The timestamp must exist in the time series.
- Updates are not allowed to the timestamp and qualifier columns (for example, *tstamp* and *ticker* in the usage demo security view).

INSTEAD OF triggers in a security view enable you to ensure that a time series meets the consistency requirements described in Section 2.6.1.

INSTEAD OF triggers allow for multiple timestamps to be inserted or deleted in a single query, given that the group of timestamps inserted or deleted are in the proper order. For example, a specified number of timestamps can be deleted from the beginning of a time series by using a simple range restriction on the timestamp. A specified number of timestamps can be inserted at the end of a time series by using a subquery that references another table containing time series data.

2.6.3 Bulk Loading and Consistency

The SQL*Loader utility is useful for loading large amounts of data into a table. For better performance, you should perform bulk loads on underlying tables instead of on security views. However, after you load data into the tables, you must ensure time series consistency by using one of the following approaches:

- Adjust calendars to be consistent with the time series.

  If you are sure that all timestamps are correct, it is safe to adjust the calendar to be consistent with the time series. This strategy is normally appropriate when there is a unique calendar per time series.

  The DeriveExceptions function is useful for adjusting a calendar to be consistent with the time series.

- Validate that each time series is consistent with the calendar.

  If you expect time series data to adhere to a predefined calendar, validating each time series is the better approach. This approach is particularly useful if the same calendar is used for all time series data being loaded.

  The IsValidTimeSeries function can be used to check if the time series is consistent with the calendar.

  For better performance in the case of a shared calendar for all time series, you may want to customize time series validation using PL/SQL. This involves writing custom utility functions that call Time Series cartridge product-developer calendar functions (see Section 2.7.2) to test and maintain time series consistency.

Section 3.2 contains additional information and examples of bulk and incremental loading of time series data.

2.7 Calendar Functions

The Time Series cartridge provides calendar functions for querying and modifying calendars. The calendar functions can be divided into the following categories:

- *End-user* functions allow application developers to use the main calendar-related features of the Time Series cartridge.
- *Product-developer* functions allow developers to modify or supplement the Time Series cartridge capabilities by creating value-added enhancements.

Reference information for all calendar functions is in Chapter 4.

2.7.1 End-User Functions

End-user functions let you use the main calendar-related features of the Time Series cartridge. If you do not need to modify or expand the Time Series cartridge capabilities, you probably can limit your use of calendar functions to those listed in Table 2–3.

Table 2–3 End-User Calendar Functions

Function	Description
	Calendar-Related Functions
EqualCals	Returns 1 if the two calendars are equivalent. If a date range is provided, tests only equivalence between the supplied dates.
IntersectCals	Intersects two calendars.
UnionCals	Returns the union of two calendars.
ValidateCal	Validates a calendar; repairs errors where possible.
	Exception-Related Functions
InsertExceptions	Inserts a list of timestamps into the appropriate exceptions list(s).
DeleteExceptions	Deletes a list of timestamps from the appropriate exceptions list(s).

2.7.2 Product-Developer Functions

Product-developer functions let you modify and expand the Time Series cartridge capabilities. For example, you could use product-developer calendar functions in creating a new function that modified the information returned for the moving average or that returned a net present value for a portfolio of stocks at a specified date.

Note: It is recommended that you not modify the functions provided with the Time Series cartridge. If you want a function with a behavior different from an existing function, create a new function with a different name or put the function in a different package, or do both. For example, if you work for XYZ Corporation and create a modified moving average function, you could name the function *MavgXYZ* and put it in a package named *XYZPackage*.

Table 2–4 lists the product-developer calendar functions.

Table 2–4 Product-Developer Calendar Functions

Function	Description
Calendar-Related Functions	
CombineCals	Combines two calendars. Similar to IntersectCals, except patterns must be identical.
Exception-Related Functions	
NumOffExceptions	Returns the number of off-exceptions between two dates.
NumOnExceptions	Returns the number of on-exceptions between two dates.
Date and Index-Related Functions	
IsValidDate	Determines if a supplied date is valid.
OffsetDate	Returns a date which is k dates in the future (or k in the past if k is negative) of the supplied date.
NumInvalidTstampsBetween	Returns the number of invalid timestamps between two dates.
NumTstampsBetween	Returns the number of valid timestamps between two dates.
TstampsBetween	Returns the valid timestamps between two dates.
InvalidTstampsBetween	Returns the invalid timestamps between two dates.
SetPrecision	Sets the precision of the input timestamp to correspond to the input frequency.

For an example of using product-developer functions, see Section 3.4.

2.8 Time Series Functions

Time series functions operate on a time series. A time series type is always used as the input parameter to a time series function.

Reference information for all time series functions is in Chapter 5.

2.8.1 Time Series Datatypes

Time series functions are defined over datatypes that contain a calendar and a collection. The Time Series cartridge provides the following time series datatypes. (Calendar datatypes are described in Section 2.2.2.)

```
CREATE TYPE ORDSYS.ORDTNumCell AS OBJECT
   (tstamp DATE, value NUMBER);

CREATE TYPE ORDSYS.ORDTNumTab AS TABLE OF
   ORDSYS.ORDTNumCell;

CREATE TYPE ORDSYS.ORDTNumSeries AS OBJECT
   (
   name         VARCHAR2(256),
   cal          ORDSYS.ORDTCalendar,
   series       ORDSYS.ORDTNumTab
   );

CREATE TYPE ORDSYS.ORDTNumSeriesIOTRef AS OBJECT
   (
   name                VARCHAR2(256),
   cal                 REF ORDSYS.ORDTCalendar,
   table_name          VARCHAR2(256),
   tstamp_colname      VARCHAR2(30),
   value_colname       VARCHAR2(30),
   qualifier_colname   VARCHAR2(30),
   qualifier_value     VARCHAR2(4000)
   );

CREATE TYPE ORDSYS.ORDTVarchar2Cell AS OBJECT
   (tstamp DATE, value VARCHAR2(4000));

CREATE TYPE ORDSYS.ORDTVarchar2Tab AS TABLE OF
   ORDSYS.ORDTVarchar2Cell;

CREATE TYPE ORDSYS.ORDTVarchar2Series AS OBJECT
   (
```

```
   name         VARCHAR2(256),
   cal          ORDSYS.ORDTCalendar,
   series       ORDSYS.ORDTVarchar2Tab
   );

CREATE TYPE ORDSYS.ORDTVarchar2SeriesIOTRef AS OBJECT
   (
   name               VARCHAR2(256),
   cal                REF ORDSYS.ORDTCalendar,
   table_name         VARCHAR2(256),
   tstamp_colname     VARCHAR2(30),
   value_colname      VARCHAR2(30),
   qualifier_colname  VARCHAR2(30),
   qualifier_value    VARCHAR2(4000)
   );

CREATE TYPE ORDSYS.ORDTDateTab AS TABLE OF DATE;
```

The preceding statements show the definition of a numeric time series and a character time series (instance-based and reference-based interfaces), each composed of a calendar instance and a collection. The collection (ORDTxxxTab) is defined as a table of ORDTxxxCell (except for ORDTDateTab, which is a table of DATE). Time Series cartridge datatypes, such as ORDTNumSeries and ORDTVarchar2Series, are input and output parameters of time series functions.

2.8.2 Conventions and Semantics

For time series functions that accept two time series, both time series must be defined on calendars that have the same frequency and the same pattern. The calendars may have different exceptions lists and different starting and ending dates.

2.8.2.1 Semantics of Null Operands

A number of time series functions perform arithmetic, comparison, and grouping operations. When nulls are encountered in this context, the default behavior is to mirror SQL:

- *Group functions* ignore nulls. When all values encountered are null, a null is returned.

  For example, the sum of (1, NULL, NULL, 3) returns 4. The sum of (NULL, NULL, NULL, NULL) returns null.

- *Functions that operate on time series* ignore nulls, but return a null if all values encountered are null. Such functions include Mavg (Moving Average) and Msum (Moving Sum)

  For example, if there are 5 nulls in the last 30 timestamps for (and including) a specific date, the 30-day moving average on that date is computed using only 25 values (that is, adding only the non-null values and dividing by 25). However, if all 30 dates (the date and the 29 previous dates) have nulls, the moving average for that date is null.

- Any *arithmetic expression* containing a null returns a null.

  For example, 10 + NULL returns null.

- A *comparison operator* that encounters a null returns a null.

  For example, a GT comparison of 30-Jun-1997 and null returns null.

  Note that because PL/SQL does not implement UNKNOWN, these semantics are slightly different than the SQL treatment of comparisons with nulls. In SQL, a comparison operator that encounters a null returns UNKNOWN, which is like a null, except that operations on UNKNOWN return UNKNOWN.

- *Scaleup* functions return a null if all timestamps for a scaling interval contain nulls.

  For example, if you are scaling up daily data from 01-Jan-1997 through 30-Jun-1997 to monthly data, and if there are no values for the month of February, a null is returned for February and scaled data is returned for the other months. (Note that this behavior differs from the standard GROUP BY scaling in SQL, in which February would be missing in the scaled results.)

Some functions allow alternate semantics in the form of an option. The reference information for each function describes any alternate semantics options.

2.8.2.2 Semantics of Off-Exception Operands

In comparisons of two time series, it is possible that a timestamp valid for one time series is not valid for the other time series. Operations on two time series having similar calendars return a time series that is defined over a new calendar. This new calendar is derived from the two input calendars, using all of the following:

- the union of the off-exceptions
- the intersection of the on-exceptions
- bounded by [min(maxDate1, maxDate2), max(minDate1, minDate2)]

For example, assume the following two calendars:

- Calendar 1: 01-Jan-1997 through 01-Dec-1997; daily pattern '0,1,1,1,1,1,0' (Monday through Friday), off-exception 01-May; on-exceptions 29-Mar and 29-Jun.
- Calendar 2: 01-Feb-1997 through 01-Jan-1998; daily pattern '0,1,1,1,1,1,0' (Monday through Friday), off-exceptions 01-May and 14-Jul; on-exceptions 29-Jun and 28-Sep.

The new (derived) calendar is: 01-Feb-1997 through 01-Dec-1997; daily pattern '0,1,1,1,1,1,0' (Monday through Friday), off-exceptions 01-May and 14-Jul; on-exception 29-Jun.

2.8.3 Extraction, Retrieval, and Trim Functions

Time series extraction, retrieval, and trim functions operate on any time series type. Extraction functions return one or more time series rows, while retrieval and trim functions return a time series.

Table 2–5 lists the extraction functions.

Table 2–5 Extraction Functions

Function	Description
DeriveExceptions	Returns a calendar populated with exceptions derived from either a calendar and a table of dates or two time series.
ExtractCal	Returns a calendar that is the same as the calendar on which the time series is based.
ExtractDate	Gets the date from an element in a time series.
ExtractTable	Returns the time series table (ORDTNumTab or ORDTVarchar2Tab) associated with a time series.
ExtractValue	Gets the value stored in an element in a time series.
First	Gets the first element in a time series.
GetDatedElement	Gets the element of a time series at a supplied date.
GetNthElement	Gets the Nth element of a time series.
Last	Gets the last element in a time series.

Table 2–6 lists the retrieval and trim functions.

Table 2–6 Retrieval and Trim Functions

Function	Description
FirstN	Gets the first *n* elements in a time series.
GetSeries	Returns the entire time series.
LastN	Gets the last *n* elements in a time series.
TrimSeries	Returns the time series data between the supplied dates.

2.8.4 Shift Functions

Shift functions lead or lag a time series by a specified number of units, where units reflects the frequency of the calendar for the time series.

Table 2–7 Shift Functions

Function	Description
Lead	Leads a time series by the specified number of units.
Lag	Lags a time series by the specified number of units.

2.8.5 SQL Formatting Functions

When called from a SQL SELECT expression, a time series function returns an instance of a time series datatype, which is not displayable. The SQL formatting functions facilitate format conversions that allow time series to be displayed.

Table 2–8 SQL Formatting Functions

Function	Description
ExtractCal	Given a time series, returns a calendar that is the same as the calendar on which the time series is based.
ExtractDate	Given an element in a time series, returns the date.
ExtractTable	Given a time series, returns the time series table (ORDTNumTab or ORDTVarchar2Tab) associated with the time series.
ExtractValue	Given an element in a time series, returns the value stored in it.

2.8.6 Aggregate Functions

Aggregate functions return scalar or ORDTNumTab values. Each aggregate function can be used in either of the following ways:

- The function accepts a numeric time series, ORDTNumSeries, and operates on all elements of the collection.
- The function accepts a numeric time series, ORDTNumSeries, and a date range, bounded by date1 and date2. The function is computed on the time series defined by the date range.

Thus, each aggregate function is of the form:

```
f(ts ORTDNumSeries, [date1 DATE, date2 DATE])
```

Table 2–9 Aggregate Functions

Function	Returns
TSAvg	Average (mean) of a time series
TSCount	Number of elements in a time series
TSMax	Maximum value of a time series
TSMaxN	Specified number of top (highest) values in a time series
TSMedian	Middle element of a time series
TSMin	Minimum value of a times series
TSMinN	Specified number of bottom (lowest) values in a time series
TSProd	Product of the elements of a time series
TSStddev	Standard deviation (square root of VAR)
TSSum	Sum of the elements of a time series
TSVariance	Variance (analogous to the SQL group function VAR)

2.8.7 Arithmetic Functions

Arithmetic functions accept two time series (ORDTNumSeries1,ORDTNumSeries2) or a time series and a constant (ORDTNumSeries1, Const), and perform a pairwise arithmetic operation on each element of the time series. This operation determines the value of each element of the returned time series:

```
Algorithm for f(ts1, ts2)
ForAll i, tsRet(i) = ts1(i) op ts2(i);
```

Table 2–10 Arithmetic Functions

Function	Description
TSAdd	Time series addition
TSDivide	Time series division
TSMultiply	Time series multiplication
TSSubtract	Time series subtraction

2.8.8 Cumulative Sequence Functions

Cumulative sequence functions operate on successive elements of a time series, accumulating the result into the current element of the output time series. For example, CSUM((1,2,3,4,5)) => (1,3,6,10,15). In this example, the result time series (f(i)), is computed from the input time series (I(i)) as follows:

```
f(1) = I(1)
ForAll i > 1, f(i) = f(i - 1) + I(i)
```

Table 2–11 Cumulative Sequence Functions

Function	Returns
Cavg	Cumulative average
Cmax	Cumulative maximum
Cmin	Cumulative minimum
Cprod	Cumulative product
Csum	Cumulative sum

2.8.9 Moving Average and Sum Functions

The Moving Average (Mavg) function returns a time series that contains the averages of values from each successive timestamp for a specified interval over a range of dates. For example, the 30-day moving average for a stock is the average of the closing price for the specified date and the 29 trading days preceding it.

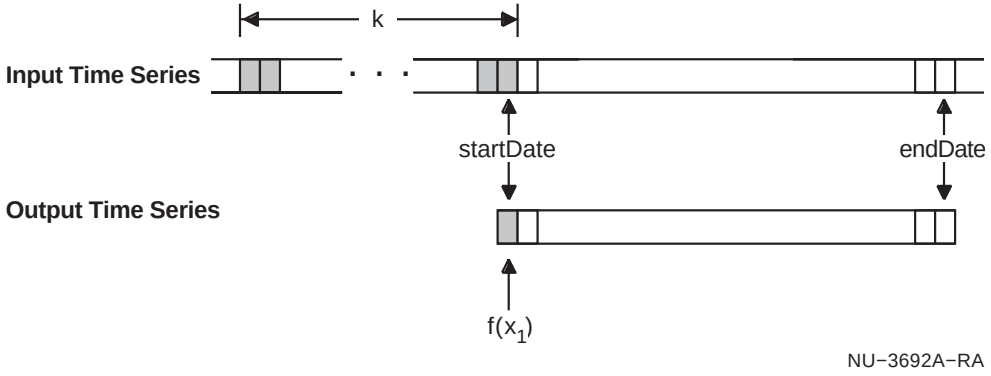
The Moving Sum (Msum) function returns a sum of values from each successive timestamp for a specified interval over a range of dates. For example, the 30-day moving sum of trading volumes for a stock is the sum of the volume for the specified date and for 29 trading days preceding it.

Table 2–12 Moving Average and Sum Functions

Function	Returns
Mavg	Moving average
Msum	Moving sum

The relationship between the input and output time series in the computation of a moving average or sum is illustrated in Figure 2–5. The figure focuses on the common invocation of moving average or sum, where *k* is the number of timestamps in the look-back window (for example, 30) and a date range (*startDate* and *endDate*) is supplied. (For more information about the parameters, see the Mavg function description in Chapter 5.)

Figure 2–5 Relationship of Input and Output Time Series in Moving Average/Sum



2.8.10 Conversion Functions

Conversion functions fill missing elements of a numeric time series (ORDTNumSeries). Missing elements are those where their timestamps are defined by the calendar and are in the range of the current time series, but they are not currently in the time series.

Table 2–13 Conversion Functions

Function	Description
Fill	Fills a time series based on the calendar and fill type.

2.9 Time Scaling Functions

The Time Series cartridge provides functions to scale up time series data. **Scaleup** functions produce summary information from finer granularity information, for example, monthly data based on daily data. Scaleup is also known as *rollup*.

The relationship between the input and output time series in a scaleup operation is illustrated in Figure 2–6, which shows a mapping when scaling from a daily frequency to a monthly frequency.

Figure 2–6 Time Scaling from Daily to Monthly Frequency

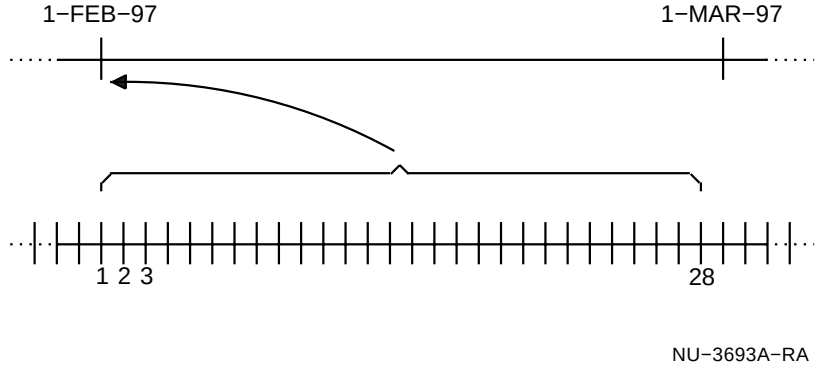


Figure 2–6 shows all days in February being mapped to the month of February. This mapping also suggests the importance of the precision of timestamps of different frequencies. In the example shown in this figure:

- The month timestamp for February 1997 is represented as 1-FEB-97 00:00:00.
- The day timestamps for February 1997 are represented as 1-FEB-97 00:00:00, 2-FEB-97 00:00:00,... 28-FEB-97 00:00:00.

Note: Scaledown functions are not included in the initial release, but they are planned for a future release. **Scaledown** functions generate finer-granularity information from coarser-granularity information. For example, quarterly data can be converted to a daily time series.

Two interfaces to time scaling are supported: the collection-based interface (operations on collections) and the GROUP BY interface (SQL GROUP BY clause). Section 2.9.1 discusses the collection-based interface for time scaling, and Section 2.9.2 discusses the GROUP BY interface.

Note: You should use the collection-based interface for most time scaling queries. Although the GROUP BY interface is useful for certain advanced queries (see Section 2.9.2), the collection-based interface offers much better performance in most cases.

2.9.1 Time Scaling on Collections

The scaleup functions accept as input a numeric time series and a destination calendar. A numeric time series is returned, which is scaled based on the destination calendar.

For example, the following statement returns the last closing prices for stock SAMCO for the months of October, November, and December of 1996:

```
select * from the
  (select cast(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupLast(
                       ts.close,
                       sc.calendar,
                       to_date('01-OCT-1996','DD-MON-YYYY'),
                       to_date('01-JAN-1997','DD-MON-YYYY')
                       )
              ) as ORDSYS.ORDTNumTab)
      from ordtdev.stocks_ts ts, ordtdev.scale sc
      where ts.ticker='SAMCO' and
            sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-OCT-96     42.375
01-NOV-96      38.25
01-DEC-96      39.75
3 rows selected.
```

Note that each timestamp reflects the first date of the month in the calendar (following the rules explained in Section 2.2.1), and each value in this case reflects the closing price on the last date for that month in the calendar.

Scaleup functions ignore nulls. For example, ScaleupAvg returns a time series reflecting the average value of each scaled group of non-null values.

Table 2–14 Scaleup Functions for Collections

Function	Description
ScaleupAvg	Returns the average value of each group.
ScaleupCount	Returns the count of timestamps in each group.
ScaleupSum	Returns the sum of each group.
ScaleupMin	Returns the minimum of each group.
ScaleupMax	Returns the maximum of each group.
ScaleupFirst	Returns the first value of each group.
ScaleupLast	Returns the last value of each group.

2.9.2 Time Scaling in the GROUP BY Clause

Time scaling in the GROUP BY clause supports statements such as the following:

```
SELECT sum(volume), max(high), min(low)
FROM StockTab, CalendarTab cal
WHERE ticker = 'XYZ' AND cal.name = 'Monthly'
GROUP BY ORDSYS.TimeSeries.Scaleup(tstamp, cal)
```

This statement scans the daily data stored in *StockTab*, and sums the volume attribute on a monthly basis. The calendar to be scaled up to is a parameter of the Scaleup function, and is extracted from a table of calendars, *CalendarTab*, which is of the form:

```
CREATE TABLE CalendarTab of ORDTCalendar;
```

The Scaleup function accepts a timestamp and a calendar, and returns a timestamp. If the input timestamp is a valid timestamp of the calendar, the input timestamp is returned; otherwise, the closest timestamp in that calendar that precedes the input timestamp is returned.

Only SQL aggregate functions are supported in the GROUP BY interface.

3

Time Series Usage

This chapter explains important procedures related to using the Oracle8 Time Series Cartridge. It covers the following topics:

- using the cartridge (major steps)
- loading time series data
- deriving calendar exceptions from time series data
- using product-developer functions

3.1 Using the Cartridge

This section provides a technical overview of using the Time Series cartridge. It presents the major steps, with examples.

For more detailed explanations of the concepts and terminology, see Chapter 2.

3.1.1 Step 1: Create the Underlying Storage (Table)

Create the table to hold the time series data. Example 3–1 shows the table definition for a stock trading database.

Example 3–1 Create a Stock Data Table

```
/* Table Creation (user) */

CREATE TABLE stockdemo ❶
  (ticker VARCHAR2(5),
  tstamp DATE,
  open NUMBER,
  high NUMBER,
  low NUMBER,
```

```
  close NUMBER,
  volume INTEGER,
  CONSTRAINT pk_stockdemo PRIMARY KEY (ticker, tstamp)) ❷
  ORGANIZATION INDEX; ❸
```

Notes on Example 3–1:

❶ The table is named *stockdemo* and has the columns for the ticker (stock symbol), the timestamp (date on which stocks are traded), that day's opening, high, low, and closing prices, and the trading volume.

❷ The constraint named *pk_stockdemo* defines the primary key as the ticker plus the timestamp.

❸ ORGANIZATION INDEX indicates that this is an index-organized table.

The CREATE TABLE statement can also include other keywords, such as TABLESPACE and STORAGE.

3.1.2 Step 2: Define a Calendar

If the calendar does not already exist, create it by inserting its definition in a table of calendars. If the table of calendars does not already exist, create it first.

Your calendar will be based on the system-defined datatype ORDTCalendar, which is supplied with the cartridge. ORDTCalendar has the following definition:

```
/* System-Defined Calendar Datatype */

CREATE TYPE ORDSYS.ORDTCalendar AS OBJECT (
  caltype INTEGER,
  name VARCHAR2(256),
  frequency INTEGER,
  pattern ORDSYS.ORDTPattern,
  minDate DATE,
  maxDate DATE,
  offExceptions ORDSYS.ORDTExceptions,
  onExceptions ORDSYS.ORDTExceptions);
```

The following example creates a table named *stockdemo_calendars* and defines a calendar named *BusinessDays*. The *BusinessDays* calendar includes Mondays through Fridays in 1997, but excludes 04-Jul-1997 and 25-Dec-1997. Explanatory notes follow the example.

Example 3–2 Create a Calendar of Business Days

```
CREATE TABLE stockdemo_calendars of ORDSYS.ORDTCalendar;

INSERT INTO stockdemo_calendars (1)
VALUES(
   ORDSYS.ORDTCalendar(
       0, (2)
       'BusinessDays', (3)
             4, (4)
      ORDSYS.ORDTPattern(ORDTPatternBits(0,1,1,1,1,1,0), (5)
         (to_date('01-05-97','MM-DD-YY'))),
      to_date('01-01-97','MM-DD-YY'), (6)
      to_date('01-01-98','MM-DD-YY'),
       ORDSYS.ORDTExceptions(to_date('07-04-97','MM-DD-YY'), (7)
         to_date('12-25-97','MM-DD-YY')),
      NULL)); (8)
```

Notes on Example 3–2:

(1) *stockdemo_calendars* is a table of ORDSYS.ORDTCalendar objects. The ORDTCalendar datatype is described in Section 2.2.2.

(2) *0* (zero) for calendar type (*caltype*) indicates that this is an exception-based calendar. (This is the only calendar type currently supported.)

(3) *BusinessDays* is the name of this calendar.

(4) *4* is the frequency code for *day*.

(5) The pattern is defined as an excluded occurrence followed by five included occurrences followed by an excluded occurrence (`0,1,1,1,1,1,0`). Because the frequency is daily and because the anchor date (05-Jan-1997) is a Sunday, Sundays are excluded, Mondays through Fridays are included, and Saturdays are excluded.

(6) The calendar begins at the start of 01-Jan-1997 and ends at the start of 01-Jan-1998.

(7) 04-Jul-1997 and 25-Dec-1997 are off-exceptions (that is, excluded from the calendar).

(8) NULL indicates that there are no on-exceptions (that is, no Saturday or Sunday dates to be included in the calendar).

3.1.3 Step 3: Load Time Series Data

Perform a bulk load of the time series data in order to populate the underlying data storage tables. Follow the guidelines and instructions for bulk loading in Section 3.2.

3.1.4 Step 4: Create a Security View and INSTEAD OF Triggers

Create a security view and INSTEAD OF triggers, to ensure the consistency and integrity of time series data, as explained in Section 2.6.2.

Example 3–3 creates a security view (*stockdemo_sv*) to get all ticker values.

Example 3–3 Create a Security View

```
CREATE OR REPLACE VIEW stockdemo_sv AS SELECT * FROM stockdemo;
```

After you create the view, create INSTEAD OF triggers using the definitions in the securevw.sql demo file as examples or templates. Example 3–4 creates an INSTEAD OF trigger (*stockdemo_sv_delete*) that ensures the following:

- An exception is raised if a delete operation is attempted on an empty time series.
- For a non-empty time series, a delete operation is allowed on either the first or last timestamp in an existing series.

If you grant users access to the security view and deny access to the underlying tables, you can ensure that all delete operations are checked and performed by the trigger. (Similar INSTEAD OF triggers can be written to allow safe insert and update operations. For more information about using INSTEAD OF triggers with security views, see Section 2.6.2.2.)

Example 3–4 Create an INSTEAD OF Trigger

```
CREATE OR REPLACE TRIGGER stockdemo_sv_delete
  INSTEAD OF DELETE on stockdemo_sv
  REFERENCING old AS o
  FOR EACH row
  DECLARE
       cal           ORDSYS.ORDTCalendar := NULL;
       purifieddate  DATE;
       startdate     DATE;
       enddate       DATE;
  BEGIN
    --
```

```
-- Retrieve the calendar that maps to the stock ticker.
--
BEGIN
SELECT VALUE(c) INTO cal
  FROM stockdemo_calendars c, stockdemo_metadata m
  WHERE m.tickername = :o.ticker AND c.name = m.calendarname;
EXCEPTION
  when NO_DATA_FOUND THEN
    raise_application_error(-20000,'Could not find calendar');
END;
IF cal IS null THEN
  raise_application_error(-20000, 'NULL calendar found');
END IF;
--
-- Set the precision of timestamp to correspond to the precision
-- of the calendar.
--
purifieddate := ORDSYS.Calendar.SetPrecision(:o.tstamp,cal.frequency);
--
-- Retrieve the current startdate AND enddate for the stock ticker;
--
SELECT max(tstamp),min(tstamp) INTO enddate,startdate
  FROM stockdemo_sv
  WHERE ticker = :o.ticker;
--
-- There are three cases of deletion to consider:
--
-- Case 1:  The table does not have any existing time series
--          entries for the given ticker.  In this case the
--          trigger raises an exception.
--
-- Case 2:  The tstamp is equal to the current startdate.
--          This routine verifies this and then deletes
--          the row.
--
-- Case 3:  The tstamp is equal to the current enddate.
--          This routine verifies this and then deletes the row.
--
--          If the time series is not empty and if the row being
--          deleted is not the startdate or enddate, an exception
--          is raised.
--
IF startdate IS null THEN
  raise_application_error(-20000,'Timeseries is empty');
ELSE
```

```
      IF (NOT ((purifieddate = startdate) or (purifieddate = enddate))) THEN
         raise_application_error(-20000, 'Timestamp date not startdate or
                         enddate');
      END IF;
    END IF;
    --
    -- Delete the row in the time series.
    --
    DELETE FROM stockdemo
      WHERE ticker = :o.ticker AND
            tstamp = purifieddate;
  END;
--
-- Next, create two other triggers: one update-specific (for
-- example, stockdemo_sv_update) and the other insert-specific
-- (for example, stockdemo_sv_insert). See the Time Series
-- cartridge usage demo for an example.
-- After creating all appropriate triggers, grant SELECT, DELETE,
-- UPDATE, and INSERT privileges on the security view to the
-- appropriate users. For example:
--
GRANT SELECT,DELETE,UPDATE,INSERT on stockdemo_sv TO ordtuser;
```

3.1.5 Step 5: Create a Reference-Based View

Create a reference-based view for convenient and efficient access to time series data, as explained in Section 2.5.2.

Example 3–5 creates a reference-based view for stock price data.

Example 3–5 Create a Reference-Based View

```
CREATE VIEW stockdemo_ts(ticker,open,high,low,close,volume) AS
  SELECT meta.tickername,
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' open NumSeries',
              Ref(cal), 'ORDTDEV.stockdemo',
              'tstamp', 'open', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' high NumSeries',
              Ref(cal), 'ORDTDEV.stockdemo',
              'tstamp', 'high', 'ticker', meta.tickername),
  ORDSYS.ORDTNumSeriesIOTRef(
       substr(meta.tickername, 1, 230) || ' low NumSeries',
               Ref(cal), 'ORDTDEV.stockdemo',
```

```
            'tstamp', 'low', 'ticker', meta.tickername),
ORDSYS.ORDTNumSeriesIOTRef(
     substr(meta.tickername, 1, 230) || ' close NumSeries',
            Ref(cal), 'ORDTDEV.stockdemo',
            'tstamp', 'close', 'ticker', meta.tickername),
ORDSYS.ORDTNumSeriesIOTRef(
     substr(meta.tickername, 1, 230) || ' volume NumSeries',
            Ref(cal), 'ORDTDEV.stockdemo',
            'tstamp', 'volume', 'ticker', meta.tickername)
FROM stockdemo_metadata meta, stockdemo_calendars cal
WHERE meta.calendarname = cal.name;
```

The refvw.sql demo file creates a reference-based view.

3.1.6 Step 6: Validate Time Series Consistency

Choose one of the following approaches to ensuring the consistency of time series data, using the guidelines in Section 2.6.3:

- Adjust calendars to be consistent with the time series.

  Use the DeriveExceptions function in adjusting a calendar to be consistent with the time series. See Section 2.2.4 for more information about this approach.

- Validate that each time series is consistent with the calendar.

  Use the IsValidTS function to check that the time series is consistent with the calendar. See the IsValidTS function reference information in Chapter 5.

3.1.7 Step 7: Formulate Time Series Queries

Formulating time series queries involves invoking time series or time scaling functions, or both. Example 3–6 uses the Mavg time series function to obtain 30-day moving averages for stock ACME, and it uses the ScaleupSum time scaling function to obtain monthly volumes for stock ACME. (The results shown in the example reflect sample data for the cartridge usage demo.)

The queries in this step use the reference-based view (*stockdemo_ts*) that was created in step 6.

Example 3–6 Formulate Time Series Queries

```
SELECT * FROM THE(
     SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
          ORDSYS.TimeSeries.Mavg(close,
               to_date('11-01-96','MM-DD-YY'),
```

```
            to_date('12-31-96','MM-DD-YY'),
            10))
        as ORDSYS.ORDTNumtab)
    FROM stockdemo_ts
    WHERE ticker = 'ACME');

TSTAMP    VALUE
--------- ----------
01-NOV-96
04-NOV-96
05-NOV-96
06-NOV-96
07-NOV-96
08-NOV-96
11-NOV-96
12-NOV-96
13-NOV-96
14-NOV-96       63.5
15-NOV-96       64.5
18-NOV-96       65.5
19-NOV-96       66.5
20-NOV-96       67.5
21-NOV-96       68.5
22-NOV-96       69.5
25-NOV-96       70.5
26-NOV-96       71.5
27-NOV-96       72.5
29-NOV-96       73.5
02-DEC-96       74.5
03-DEC-96       75.5
04-DEC-96       76.5
05-DEC-96       77.5
06-DEC-96       78.5
09-DEC-96       79.5
10-DEC-96       80.5
11-DEC-96       81.5
12-DEC-96       82.5
13-DEC-96       83.5
16-DEC-96       84.5
17-DEC-96       85.5
18-DEC-96       86.5
19-DEC-96       87.5
20-DEC-96       88.5
23-DEC-96       89.5
24-DEC-96       90.5
```

```
26-DEC-96       91.5
27-DEC-96       92.5
30-DEC-96       93.5
31-DEC-96       94.5
41 rows selected.

SELECT * FROM THE(
     SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
          ORDSYS.TimeSeries.ScaleupSum(volume,value(cal)))
          as ORDSYS.ORDTNumtab)
     FROM stockdemo_ts, stockdemo_calendars cal
     WHERE ticker = 'ACME' AND cal.name = 'MONTHLY');

TSTAMP    VALUE
--------- ----------
01-NOV-96      20000
01-DEC-96      21000
2 rows selected.
```

3.2 Loading Time Series Data

This section describes how to use the SQL*Loader utility to perform bulk loading and incremental loading of time series data.

To ensure the consistency of time series data during loading, you must choose one of the approaches described in Section 2.6.3:

- Adjust calendars to be consistent with the time series, if you are sure that all timestamps are correct.

  This strategy is normally appropriate when there is a unique calendar per time series.

- Validate that each time series is consistent with the calendar, if you expect time series data to adhere to a predefined calendar.

  This approach is particularly useful if the same calendar is used for all time series data being loaded.

This section describes how to perform bulk loading using these two approaches, and it also describes how to perform incremental loading.

The loading of time series data is usually performed under controlled circumstances, so it is safe to perform these loads directly to an underlying table instead of to a security view.

3.2.1 Bulk Loading

After you create an index-organized table (IOT) to hold time series data (such as for the *stockdemo* demo database), you must populate the table with data. For a database of stock information, you may need to load millions of rows of daily summary information into the IOT.

SQL*Loader is recommended for loading large amounts of time series data. The following example shows a SQL*Loader script, with an excerpt from the sample data (stockdat.dat) and the SQL*Loader control file (stockdat.ctl). For complete information about SQL*Loader, see the *Oracle8 Server Utilities* manual.

The SQL*Loader script contains the following:

```
% sqlldr userid=ordtdev/ordtdev control=stockdat.ctl
       log=stockdat.log bad=stockdat.bad errors=1000
```

The stockdat.dat sample data file includes the following:

```
ACME  01-NOV-96  59.00  60.00  58.00  59.00  1000
ACME  04-NOV-96  60.00  61.00  59.00  60.00  1000
ACME  05-NOV-96  61.00  62.00  60.00  61.00  1000
          ...
```

The stockdat.ctl file contains the following

```
options (direct=true)
unrecoverable
load data
infile 'stockdat.dat'
replace
into table stockdemo
sorted indexes (StockTabx)
fields terminated by whitespace
(ticker, tstamp DATE(13) "DD-MON-YY", open, high, low, close, volume)
```

SQL*Loader can handle many file formats and delimiters, as documented in the *Oracle8 Utilities* manual.

After the load has completed, you may want to choose one of the following approaches for ensuring calendar consistency:

- Adjust calendars to conform to time series data (see Section 3.2.1.1).
- Validate that the time series conforms to the calendar (see Section 3.2.1.2).

In either case, you may need to update the exception lists of your calendars.

3.2.1.1 Adjusting Calendars to Conform to Time Series Data

Often you will want to create calendars that conform to the time series data that you are receiving. In this case, you usually know the frequency and the pattern of a calendar, but not the specific on- or off-exceptions. You can extract these exceptions from the data by using the DeriveExceptions function.

3.2.1.2 Validating That the Time Series Conforms to the Calendar

Often you will want to ensure that the time series data extracted from the incoming data conforms to a predefined calendar. To do this, insert the exceptions either when you create the calendar or afterward with the InsertExceptions functions (or do both, creating the calendar with some exceptions and then adding others); then use the IsValidTimeSeries function to check that the time series is consistent with the calendar.

You can insert exceptions when you define the calendar. For example, the following statement specifies 28-Nov-1996 and 25-Dec-1996 as off-exceptions in the calendar named *BUSINESS-96*:

```
INSERT INTO stockdemo_calendars VALUES(
       ORDSYS.ORDTCalendar(
             0,
             'BUSINESS-96',
             4,
             ORDSYS.ORDTPattern(
                      ORDSYS.ORDTPatternBits(0,1,1,1,1,1,0),
                           TO_DATE('01-JAN-1995','DD-MON-YYYY')),
             TO_DATE('01-JAN-1990','DD-MON-YYYY'),
             TO_DATE('01-JAN-2001','DD-MON-YYYY'),
             ORDSYS.ORDTExceptions(
                           TO_DATE('28-NOV-1996','DD-MON-YYYY'),
                           TO_DATE('25-DEC-1996','DD-MON-YYYY')),
             ORDSYS.ORDTExceptions()
             ));
```

You can also add exceptions after the calendar is defined by using the InsertExceptions function. For example, the following statement adds 01-Jan-1997, 17-Feb-1997, and 26-May-1997 as off-exceptions:

```
UPDATE stockdemo_calendars cal
   SET cal = (SELECT ORDSYS.Calendar.InsertExceptions(
                     VALUE(cal),
                     ORDSYS.ORDTDateTab(
                           to_date('01-JAN-97','DD-MON-YY'),
                           to_date('17-FEB-97','DD-MON-YY'),
```

```
                              to_date('26-MAY-97','DD-MON-YY')))
          FROM dual)
WHERE cal.name = 'BUSINESS-96';
```

After you have defined the calendar and populated the exception lists, you can use the IsValidTimeSeries function to check that the time series is consistent with the calendar.

3.2.2 Incremental Loading

After you have performed the bulk load of time series data and have started using the Time Series cartridge, you will probably want to add data periodically. For example, every trading day after the stock exchange closes, that day's data for each ticker becomes available.

As with bulk loading, incremental loading is typically done in a controlled environment. You know which timestamps will become off-exceptions, and you can explicitly update the exception lists of the appropriate calendars. The following example demonstrates such an update:

```
UPDATE stockdemo_calendars cal
 SET cal = (SELECT ORDSYS.Calendar.InsertExceptions(
                          VALUE(cal),
                          to_date('01-JAN-97','DD-MON-YY'))
              FROM dual)
        WHERE cal.name = 'XCORP';
```

The SQL*Loader utility is recommended for performing an incremental load of such additional data. The following example shows a SQL*Loader script, with an excerpt from the sample daily data (stockinc.dat) and the SQL*Loader control file (stockinc.ctl).

The SQL*Loader script contains the following:

```
sqlldr userid=ordtdev/ordtdev control=stockinc.ctl
        log=stockinc.log bad=stockinc.bad errors=1000
```

The stockinc.dat sample data file includes the following:

```
ACME  02-JAN-97 100.00 101.00  99.00 100.00  1000
FUNCO 02-JAN-97  25.00  25.00  25.00  25.00  2000
SAMCO 02-JAN-97  39.00  40.00  38.00  39.50 30000
        ...
```

The stockinc.ctl file contains the following:

```
load data
infile 'stockinc.dat'
append
into table stockdemo
fields terminated by whitespace
(ticker, tstamp DATE(13) "DD-MON-YY", open, high, low, close, volume)
```

Note the following differences in the control file for incremental loading as opposed to bulk loading:

- The *conventional path* is used instead of the direct path. That is, the control file for incremental loading does *not* contain the line *options (direct=true)*.

  The conventional path is better for incremental loading because the amount of new data (daily stock information) is small relative to the total amount of data. For an explanation of conventional and direct paths, including situations in which the conventional path is necessary or preferable, see the SQL*Loader documentation in the *Oracle8 Server Utilities* manual.

- The APPEND keyword is specified, so that the new data is appended to the existing tabular data.

3.3 Deriving Calendar Exceptions

This section explains in greater detail the two approaches to deriving calendar exceptions from time series data. These two approaches were introduced in Section 2.2.4; see that section for information on concepts related to exceptions and the reasons for choosing a particular approach.

3.3.1 Deriving Exceptions Using a Calendar and Table of Dates (Approach 1)

The first approach to deriving exceptions takes a calendar and an ORDTDateTab (that is, a table of dates) as input parameters, using the following form of the function:

DeriveExceptions(cal ORDTCalendar, DateTab ORDTDateTab)

The table of dates (*DateTab* parameter) includes all dates in the time series, for example, all dates on which stock XYZ traded. A calendar is returned that is defined on the same pattern and frequency as the input calendar, and the exception lists of the returned calendar are populated to be consistent with the time series data in *DateTab*. The exception lists are updated based on finding timestamps that are in the calendar pattern or in the table of dates, but not in both. (A timestamp is in the calendar pattern if it is within the date range of the calendar and maps to an *on* (1) bit in the pattern.)

The returned calendar's on- and off- exceptions are populated based on the calendar pattern and the table of dates, as follows:

- All timestamps that are in the calendar pattern but not in the table of dates become off-exceptions.

  For example, 04-Jul-1997 (Friday) is in the pattern of a stock trading calendar, but it is not a date on which U.S. stocks were traded.

- All timestamps that are in the table of dates but are not in the calendar pattern become on-exceptions.

The following example derives the exceptions for all time series in the *stockdemo* table and updates the corresponding calendars in the *stockdemo_calendars* table:

```
UPDATE stockdemo_calendars cal
  SET cal = (SELECT ORDSYS.Calendar.DeriveExceptions(
                VALUE(cal),
                CAST(multiset(
                  SELECT s.tstamp
                  FROM stockdemo s
                  WHERE cal.name = s.ticker) AS ORDSYS.ORDTDateTab))
             FROM dual);
```

This approach (Approach 1) to deriving calendar exceptions has the following requirements:

- The input table of dates must be sorted in ascending timestamp order before the call to the DeriveExceptions function.
- The precision of the timestamps of the dates in the table must conform to the frequency of the input calendar.

3.3.2 Deriving Exceptions Using Two Time Series Parameters (Approach 2)

The second approach to deriving exceptions takes two time series references as input parameters, using the following form of the function:

DeriveExceptions(series1 ORDTNumSeriesIOTRef,

series2 ORDTNumSeriesIOTRef)

or

DeriveExceptions(series1 ORDTVarchar2SeriesIOTRef,

series2 ORDTVarchar2SeriesIOTRef)

This overloading of the DeriveExceptions function allows the input parameters to be time series REFs (either two ORDTNumSeriesIOTRef parameters or two ORDTVarchar2SeriesIOTRef parameters).

Before calling DeriveExceptions, you must construct a time series based on a reference calendar. This time series will contain all the timestamps within the date range (*minDate* through *maxDate*) of the calendar.

The following example builds a reference time series based on a calendar named *PATTERN-ONLY*. An INSERT statement populates the time series named *PATTERN-ONLY* with the valid timestamps between the starting and ending dates of the calendar.

```
INSERT INTO stocks(ticker,tstamp)
SELECT 'PATTERN-ONLY',
       t1.c1
       FROM
       (SELECT column_value c1 FROM the
        (SELECT CAST(ORDSYS.Calendar.TimeStampsBetween(VALUE(cal),
                                                cal.mindate,
                                                cal.maxdate)
              AS ORDSYS.ORDTDateTab)
        FROM stock_calendars cal
        WHERE cal.name = 'PATTERN-ONLY')) t1;
```

The insertion is made directly into the underlying table, not into the security view. Using the underlying table is safe here because the time series is presumed to be correct, so the mechanisms for ensuring consistency between the time series and the calendar provided by the security view are not needed in this case.

The *PATTERN-ONLY* calendar should have no exceptions. If this calendar has any exceptions, the resulting time series will have non-null exception lists, which will cause the DeriveExceptions function to report an error.

After you create the reference time series, call the DeriveExceptions function with the reference time series as the first parameter (*series1*). DeriveExceptions compares the dates in *series1* with the dates in *series2*, and it returns the calendar of *series1* with the exceptions created as follows:

- All timestamps that are in *series1* but not in *series2* become off-exceptions.

  For example, if *series2* contains dates on which stock XYZ traded and 04-Jul-1997 (Friday) is not in that time series, then 04-Jul-1997 is added to the calendar as an off-exception.

- All timestamps that are in *series2* but not in *series1* become on-exceptions.

The following example uses the reference time series created in the preceding statement to update the exception lists of every other calendar in the *stockdemo_calendars* table, with the exceptions for each calendar derived from the timestamps in the associated time series. (This example assumes that each calendar maps to a time series with the same name.)

```
UPDATE stockdemo_calendars cal
 SET cal = (SELECT ORDSYS.TimeSeries.DeriveExceptions(ts1.open,ts2.open)
           FROM stocks_ts ts1, stocks_ts ts2
             WHERE ts1.ticker = 'PATTERN-ONLY' and ts2.ticker = cal.name)
WHERE cal.name <> 'PATTERN-ONLY';
```

This approach (Approach 2) to deriving calendar exceptions has the following requirements:

- The input parameters to the DeriveExceptions function must be either two ORDTNumSeriesIOTRef parameters or two ORDTVarchar2SeriesIOTRef parameters. ORDTNumSeries and ORDTVarchar2Series variants are not supported for this function.
- Calendars of the time series input parameters must have the same frequency and pattern.
- The first time series parameter (*PATTERN-ONLY* time series) must have no exceptions.
- The starting date (*minDate*) of the calendar of the second time series must be greater (later) than or equal to the starting date of the calendar of the first time series.
- The ending date (*maxDate*) of the calendar of the second time series must be less (earlier) than or equal to the ending date of the calendar of the first time series.

3.4 Using Product-Developer Functions

Product-developer functions, described in Section 2.7.2, let you modify and expand the Time Series cartridge capabilities. For example, an ISV could develop additional time series analysis functions by calling product-developer functions.

The following example shows the use of the IsValidDate, NumTstampsBetween, and OffsetDate product-developer functions in a PL/SQL implementation of the Lead function. The Lead function inputs a time series and a lead_date, and returns a time series where the starting timestamp is the lead_date. (Note that to simplify the presentation, some error checking has been omitted.)

```
create function Lead (ts ORDSYS.ORDTNumSeries, lead_date date)
     return ORDSYS.ORDTNumSeries is
i integer;
outts ORDSYS.ORDTNumSeries; /* Temporary Storage for Result */
new_tstamp date;   /* Changeable version of lead_date */
last_lead_date date; /* Last timestamp of the output time series*/
first_tstamp date; /* First timestamp of
                      the input time series */
last_index integer; /* Last index of the input time series */
last_tstamp date;       /* Last timestamp of the input time series */
units integer;      /* Number of timestamps between input and
                       output time series */

ERR_LEAD_TSTAMP_BOUNDS constant integer := 20540;
ERR_LEAD_TSTAMP_BOUNDS_MSG constant varchar2(100) :=
         'Projected lead timestamp beyond calendar bounds';

begin
     first_tstamp :=ts.series(1).tstamp;
     last_index :=ts.series.last;
     last_tstamp :=ts.series(last_index).tstamp;

     if ORDSYS.Calendar.IsValidDate(ts.cal, lead_date) = 0 then
          Raise_Application_Error(ERR_LEAD_TSTAMP_BOUNDS,
                  ERR_LEAD_TSTAMP_BOUNDS_MSG);
     end if;

     /* units is the number of timestamps between the first timestamp of
     the input time series and lead_date. */
     units := ORDSYS.Calendar.NumTimeStampsBetween(ts.cal, first_tstamp,
              lead_date);

     last_lead_date := ORDSYS.Calendar.OffsetDate(ts.cal, last_tstamp,
                       units);
     if last_lead_date is null then
          Raise_Application_Error(ERR_LEAD_TSTAMP_BOUNDS,
                    ERR_LEAD_TSTAMP_BOUNDS_MSG);
     end if;

     /* Instantiate output time series. */
     outts := ORDSYS.ORDTNumSeries('Lead Result', ts.cal, ORDSYS.ORDTNumTab());
     outts.series.extend(last_index);

     /* Assign the first timestamp of the output time series to
     first_lead_date. Copy value from input time series to output
```

```
     time series. */
     new_tstamp := lead_date;
     outts.series(1) := ORDSYS.ORDTNumCell(new_tstamp, ts.series(1).value);

     /* Assign subsequent timestamps by calling OffsetDate with the
     previous date and an offset of 1. */
     for i in 2..outts.series.last loop
          new_tstamp := ORDSYS.Calendar.OffsetDate(ts.cal,
            outts.series(i-1).tstamp, 1);
          outts.series(i) := ORDSYS.ORDTNumCell(new_tstamp,
            ts.series(i).value);
     end loop;

     return(outts);
end;
```

For other examples of using product-developer functions, see the files for the advanced-developer demo (described briefly in Table 1–1 in Section 1.6).

4

Calendar Functions: Reference

The Oracle8 Time Series Cartridge library consists of the following:

- datatypes (described in Section 2.2.2)
- calendar functions (described in this chapter)
- time series and time scaling functions (described in Chapter 5)

Two separate reference chapters are provided for the functions, because the functions described in each are typically done at different times in the application development cycle and by people performing different job roles:

- Calendar functions are mainly used by product developers, such as ISVs, to develop new time series functions and to administer and modify calendars.
- Time Series and time scaling functions are used mainly by application developers and some end users after the associated calendar or calendars have been defined.

Syntax notes:

- The ORDSYS schema name and the package name must be used with the function name, although public synonyms can be created to eliminate the need for specifying the schema name (see Section 1.4). Each function is included in a PL/SQL package, such as Calendar or TimeSeries. The ORDSYS schema name and the package name are included in the Format and in any examples.
- Function calls are not case sensitive, except for any quoted literal values. For example, the following code line excerpts are valid and semantically identical:

```
select CAST(TimeSeries.ExtractTable(close) AS ORDTNumTab)
select cast(TIMESERIES.extracttable(close) as ordtnumtab)
select cast(TiMeSeRiEs.eXtRaCtTaBlE(ClosE) As ordtNUMtab)
```

CombineCals

Format

```
ORDSYS.Calendar.CombineCals(
    cal1 ORDSYS.ORDTCalendar,
    cal2 ORDSYS.ORDTCalendar,
    [startDate DATE,
    endDate DATE,]
    equalFlag OUT INTEGER
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Combines two calendars. The CombineCals function is provided primarily for use in developing functions that operate on two time series (such as the TSAdd function).

Parameters

cal1
The first calendar to be combined.

cal2
The second calendar to be combined.

startDate
Starting date for the resulting calendar. If *startDate* is not specified, the starting date is the starting date for the calendars, or the higher (later) of the starting dates if they are different.

endDate
Ending date for the resulting calendar. If *endDate* is not specified, the ending date is the ending date for the calendars, or the lower (earlier) of the ending dates if they are different.

equalFlag
Contains 1 if the input calendars are equal, and 0 if the input calendars are not equal.

Usage

If the frequencies of the two calendars are not equal, the function returns NULL.

If the aligned patterns of the two calendars are not equal, the function returns NULL.

If *startDate* is not specified, the starting date of the resulting calendar is the later of the starting dates of the two calendars, that is, resulting minDate = max(minDate1, minDate2).

If *endDate* is not specified, the ending date of the resulting calendar is the earlier of the ending dates of the two calendars, that is, resulting maxDate = min(maxDate1, maxDate2).

The function intersects the on-exception lists of the two calendars. For example, if *cal1* has 30-Mar and 29-Jun as on-exceptions and *cal2* has 29-Jun and 28-Sep as on-exceptions, the resulting calendar has only 29-Jun as an on-exception.

The function performs a union of the off-exceptions of the two calendars. For example, if *cal1* has 01-Jan and 04-Jul as off-exceptions and *cal2* has 01-Jan and 14-Jul as off-exceptions, the resulting calendar has 01-Jan, 04-Jul, and 14-Jul as off-exceptions.

CombineCals and IntersectCals differ as follows:

- CombineCals requires the frequencies and the aligned patterns of the two calendars to be equal, whereas IntersectCals requires only that the frequencies be equal. However, IntersectCals does require that the patterns be of the same length.
- CombineCals lets you specify starting and ending dates for the resulting calendar, whereas IntersectCals does not let you specify starting and ending dates.

Example

Combine two calendars (*GENERIC-CAL1* and *GENERIC-CAL2*), then intersect the two calendars:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
```

```
tstCal1 ORDSYS.ORDTCalendar;
tstCal2 ORDSYS.ORDTCalendar;
resultCal ORDSYS.ORDTCalendar;
equalFlag INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select the calendars GENERIC-CAL1 into tstCal1
 -- and GENERIC-CAL2 into tstCal2
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal1
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';
 SELECT value(cal) INTO tstCal2
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL2';

 -- Display the calendars tstCal1 and tstCal2.
 SELECT ORDSYS.TimeSeries.Display(tstCal1) INTO dummyVal FROM dual;
 SELECT ORDSYS.TimeSeries.Display(tstCal2) INTO dummyVal FROM dual;

 -- Combine tstCal1 and tstCal2
 resultCal := ORDSYS.Calendar.CombineCals(tstCal1, tstCal2, equalFlag);
 SELECT ORDSYS.TimeSeries.Display(resultCal, 'result of CombineCals')
 INTO dummyVal
 FROM dual;
 DBMS_OUTPUT.PUT_LINE('equalFlag = ' || equalFlag);

 -- Intersect tstCal1 and tstCal2
 resultCal := ORDSYS.Calendar.IntersectCals(tstCal1, tstCal2);
 SELECT ORDSYS.TimeSeries.Display(resultCal, 'result of IntersectCals')
 INTO dummyVal
 FROM dual;

END;
/
```

This example might produce the following output:

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
```

```
         0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Calendar Name = GENERIC-CAL2
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1997 00:00:00
 patBits:
         1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
     10/13/1996 00:00:00     11/10/1996 00:00:00     12/14/1996 00:00:00
     01/04/1997 00:00:00     02/09/1997 00:00:00     03/08/1997 00:00:00
     04/05/1997 00:00:00     05/11/1997 00:00:00     06/08/1997 00:00:00
 offExceptions :
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
     01/01/1997 00:00:00     02/12/1997 00:00:00     03/04/1997 00:00:00
     04/07/1997 00:00:00     05/05/1997 00:00:00     06/09/1997 00:00:00

result of CombineCals :

 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
         0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
equalFlag = 0
```

```
result of IntersectCals :

 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
```

DeleteExceptions

Format

```
ORDSYS.Calendar.DeleteExceptions(
    inputCal IN ORDSYS.ORDTCalendar,
    delExcDate IN DATE
    ) RETURN ORDSYS.ORDTCalendar;
```

or

```
ORDSYS.Calendar.DeleteExceptions(
    inputCal IN ORDSYS.ORDTCalendar,
    delExcTab IN ORDSYS.ORDTDateTab
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Deletes from the specified calendar all exceptions that either match a specified date (*delExcDate*) or are included in a table of dates (*delExcTab*), and returns the resulting calendar.

Parameters

inputCal
The calendar from which one or more exceptions are to be deleted.

delExcDate
The date to be deleted from the exceptions of the calendar.

delExcTab
A table of dates to be deleted from the exceptions of the calendar.

Usage

If a date to be deleted is in either the on-exception list or off-exception list of the calendar, the function deletes the date from the appropriate list.

If *delExcDate* is not in either the on-exception list or off-exception list of the calendar, the function returns the input calendar with no changes.

For any date in *delExcTab* that is not in either the on-exception list or off-exception list of the calendar, the function ignores the date. If no date in *delExcTab* is in either the on-exception list or off-exception list of the calendar, the function returns the input calendar with no changes.

Example

Delete some exceptions from a calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDTab ORDSYS.ordtDateTab;
resultCal ORDSYS.ORDTCalendar;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Delete some exceptions in tstCal.
 tstDTab := ORDSYS.ORDTDateTab(
                 '01/21/1996', -- ON  Exception
                 '05/08/1996', -- OFF Exception
                 '08/04/1996', -- ON  Exception
                 '07/09/1996');-- OFF Exception
 SELECT ORDSYS.TimeSeries.Display(tstDTab, 'Input DateTab')
 INTO dummyVal
 FROM dual;
 resultCal := ORDSYS.Calendar.DeleteExceptions(tstCal, tstDTab);
```

```
 SELECT ORDSYS.TimeSeries.Display(resultCal) INTO dummyVal
 FROM dual;

END;
/
```

This example might produce the following output. The second display of information about *GENERIC-CAL1* does not include the deleted on-exceptions and off-exceptions.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Input DateTab :

     01/21/1996 00:00:00     05/08/1996 00:00:00     08/04/1996 00:00:00
     07/09/1996 00:00:00

Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     02/03/1996 00:00:00     03/24/1996 00:00:00     04/27/1996 00:00:00
     05/19/1996 00:00:00     06/23/1996 00:00:00     07/07/1996 00:00:00
     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     06/25/1996 00:00:00
```

DisplayValCal Procedure

Format

```
ORDSYS.Calendar.DisplayValCal(
    validFlag IN INTEGER,
    outMessage IN VARCHAR2,
    invOnExc IN ORDSYS.ORDTDateTab,
    invOffExc IN ORDSYS.ORDTDateTab,
    impOnExc IN ORDSYS.ORDTDateTab,
    impOffExc IN ORDSYS.ORDTDateTab,
    inputCal IN ORDSYS.ORDTCalendar,
    mesg IN VARCHAR2
    );
```

Description

Displays the results returned by the ValidateCal function.

> **Note:** DisplayValCal is a procedure, not a function. Procedures do not return values.

Parameters

validFlag
The return value from the ValidateCal function call:

Value	Meaning
0	The calendar is valid. No errors were found.
1	Correctable errors were found and corrected. The resulting calendar is valid.
-1	Uncorrectable errors were found. The calendar is not valid.

outMessage
Message output by ValidateCal describing how the calendar was repaired (if the return value = 1) or why the calendar could not be repaired (if the return value = -1).

invOnExc
Table of the invalid on-exceptions found in the calendar.

invOffExc
Table of the invalid off-exceptions found in the calendar.

impOnExc
Table of the imprecise on-exceptions found in the calendar.

impOffExc
Table of the imprecise off-exceptions found in the calendar.

inputCal
The calendar returned by ValidateCal (repaired if necessary).

mesg
Optional message.

Usage

This procedure is intended to be used with the ValidateCal function. See the information on ValidateCal in this chapter.

Example

Use the IsValidCal and ValidateCal functions and the DisplayValCal procedure with an invalid calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
outMessage  varchar2(32750);
invOnExc    ORDSYS.ORDTDateTab;
invOffExc   ORDSYS.ORDTDateTab;
impOnExc    ORDSYS.ORDTDateTab;
impOffExc   ORDSYS.ORDTDateTab;
dummyval     integer;
```

```
validFlag     integer;
tstCal1     ORDSYS.ORDTCalendar :=
               ORDSYS.ORDTCalendar(
                  0,
                  'CALENDAR FOO',
                  4,
                  ORDSYS.ORDTPattern(ORDSYS.ORDTPatternBits(1,1,1,1,1,0,0),
                                                TO_DATE('01-08-1996 01:01:01')),
                  TO_DATE('01-01-1975'),
  TO_DATE('01-01-1999'),
                  ORDSYS.ORDTExceptions(
            TO_DATE('02-03-1969'), -- Date < minDate,
            TO_DATE('02-14-1969'), -- Date < minDate,
            TO_DATE('02-03-1999'), -- Date > maxDate,
            TO_DATE('02-17-1999'), -- Date > maxDate,
            TO_DATE('12-31-1995'), -- Maps to 0 in pattern (Sunday)
            TO_DATE('01-13-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('02-24-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('03-30-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('02-02-1996 01:01:01'), -- Imprecise
            TO_DATE('03-04-1996 01:01:01'), -- Imprecise
            TO_DATE('04-05-1996 02:02:02'), -- Imprecise
            TO_DATE('03-25-1996'), -- Valid off-exception
            TO_DATE('01-22-1996'), -- Valid, but out of sequence
            TO_DATE('02-12-1996'),
            TO_DATE('04-30-1996'),
            NULL,                  -- Null date
            TO_DATE('02-12-1996'), -- Duplicate date within OFFs
            NULL,                  -- Null date
            TO_DATE('04-30-1996'), -- Duplicate off-exception
            NULL,                  -- Null date
            TO_DATE('03-25-1996'), -- Duplicate off-exception
            TO_DATE('01-22-1996'), -- Duplicate off-exception
            TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
            TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
            TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('02-02-1996'),
            TO_DATE('03-04-1996'),
            TO_DATE('05-06-1997')),
    ORDSYS.ORDTExceptions(
            TO_DATE('02-08-1969'), -- Date < minDate,
            TO_DATE('02-15-1969'), -- Date < minDate,
            TO_DATE('02-13-1999'), -- Date > maxDate,
            TO_DATE('02-20-1999'), -- Date > maxDate,
```

```
            TO_DATE('01-03-1996'), -- Maps to 1 in pattern (Wednesday)
            TO_DATE('02-19-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-18-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('05-27-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-23-1996 01:01:01'), -- Imprecise
            TO_DATE('02-18-1996 01:01:01'), -- Imprecise
            TO_DATE('05-26-1996 01:01:01'), -- Imprecise
            TO_DATE('01-13-1996'), -- Valid on-exception
            TO_DATE('01-14-1996'), -- Valid on-exception
            NULL,                  -- Null date
            NULL,                  -- Null date
            TO_DATE('02-24-1996'), -- Valid on-exception
            TO_DATE('03-23-1996'), -- Valid on-exception
            TO_DATE('01-13-1996'), -- Duplicate on-exception
            TO_DATE('01-14-1996'), -- Duplicate on-exception
            TO_DATE('02-24-1996'), -- Duplicate on-exception
            TO_DATE('03-23-1996'), -- Duplicate on-exception
            TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
            TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
            TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('01-06-1996'), -- Valid, but out of sequence
            TO_DATE('02-03-1996'),
            TO_DATE('05-04-1997'))
                  );
BEGIN
   SELECT ORDSYS.TIMESERIES.Display(tstCal1, 'tstCal1') INTO dummyval
   FROM dual;
   validFlag := ORDSYS.CALENDAR.IsValidCal(tstCal1);
   IF(validFlag = 0)
   THEN
       validFlag := ORDSYS.CALENDAR.ValidateCal(
                tstCal1, outMessage, invOnExc, invOffExc, impOnExc, impOffExc
                );

       ORDSYS.TIMESERIES.DisplayValCal(
             validFlag,
             outMessage,
             invOnExc,
             invOffExc,
             impOnExc,
             impOffExc,
             tstCal1,
             'Your Message'
             );
```

```
   END IF;
END;
/
```

This example might produce the following output:

```
tstCal1 :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
 MaxDate = 01/01/1999 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 01:01:01
 onExceptions  :
     02/08/1969 00:00:00     02/15/1969 00:00:00     02/13/1999 00:00:00
     02/20/1999 00:00:00     01/03/1996 00:00:00     02/19/1996 00:00:00
     03/18/1996 00:00:00     05/27/1996 00:00:00     03/23/1996 01:01:01
     02/18/1996 01:01:01     05/26/1996 01:01:01     01/13/1996 00:00:00
     01/14/1996 00:00:00
     02/24/1996 00:00:00     03/23/1996 00:00:00     01/13/1996 00:00:00
     01/14/1996 00:00:00     02/24/1996 00:00:00     03/23/1996 00:00:00
     01/17/1996 00:00:00     05/28/1996 00:00:00     06/18/1996 00:00:00
     04/23/1996 00:00:00     01/06/1996 00:00:00     02/03/1996 00:00:00
     05/04/1997 00:00:00
 offExceptions :
     02/03/1969 00:00:00     02/14/1969 00:00:00     02/03/1999 00:00:00
     02/17/1999 00:00:00     12/31/1995 00:00:00     01/13/1996 00:00:00
     02/24/1996 00:00:00     03/30/1996 00:00:00     02/02/1996 01:01:01
     03/04/1996 01:01:01     04/05/1996 02:02:02     03/25/1996 00:00:00
     01/22/1996 00:00:00     02/12/1996 00:00:00     04/30/1996 00:00:00
          02/12/1996 00:00:00
     04/30/1996 00:00:00          03/25/1996 00:00:00
     01/22/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00     02/02/1996 00:00:00
     03/04/1996 00:00:00     05/06/1997 00:00:00

DisplayValCal Your Message:

TS-WRN: the input calendar has rectifiable errors. See the message for details

message output by validateCal:

TS-WRN: fixed precision of the pattern anchor date
```

```
TS-WRN: removed superfluous dates in the on exception list (refer invalidOnExc)
TS-WRN: fixed imprecise dates in the on exception list (refer impreciseOnExc)
TS-WRN: removed null dates in the on exception list
TS-WRN: sorted the on exceptions list
TS-WRN: removed duplicate dates in the on exceptions list
TS-WRN: removed superfluous dates in off exceptions list (refer invalidOffExc)
TS-WRN: fixed imprecise dates in the off exception list (refer impreciseOffExc)
TS-WRN: removed null dates in the off exception list
TS-WRN: sorted the off exceptions list
TS-WRN: removed duplicate dates in the off exceptions list
TS-WRN: the on exceptions list was trimmed between calendar minDate & maxDate
TS-WRN: the off exceptions list was trimmed between calendar minDate & maxDate

list of invalid on exceptions :

     01/03/1996 00:00:00     02/19/1996 00:00:00     03/18/1996 00:00:00
     05/27/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00

list of invalid off exceptions :

     12/31/1995 00:00:00     01/13/1996 00:00:00     02/24/1996 00:00:00
     03/30/1996 00:00:00

list of imprecise on exceptions :

     03/23/1996 01:01:01     02/18/1996 01:01:01     05/26/1996 01:01:01
list of imprecise off exceptions :

     02/02/1996 01:01:01     03/04/1996 01:01:01     04/05/1996 02:02:02

the validated calendar :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
 MaxDate = 01/01/1999 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     01/06/1996 00:00:00     01/13/1996 00:00:00     01/14/1996 00:00:00
     02/03/1996 00:00:00     02/18/1996 00:00:00     02/24/1996 00:00:00
     03/23/1996 00:00:00     05/26/1996 00:00:00     05/04/1997 00:00:00
 offExceptions :
```

```
01/17/1996 00:00:00     01/22/1996 00:00:00     02/02/1996 00:00:00
02/12/1996 00:00:00     03/04/1996 00:00:00     03/25/1996 00:00:00
04/05/1996 00:00:00     04/23/1996 00:00:00     04/30/1996 00:00:00
05/28/1996 00:00:00     06/18/1996 00:00:00     05/06/1997 00:00:00
```

EqualCals

Format

```
ORDSYS.Calendar.EqualCals(
    cal1 ORDSYS.ORDTCalendar,
    cal2 ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN BINARY_INTEGER;
```

Description

Checks if two calendars (completely or within a specified date range) are equal.

Parameters

cal1
The first calendar to be checked.

cal2
The second calendar to be checked.

startDate
Starting date for the checking. If *startDate* is not specified, the starting date is the starting date for the calendars, or the higher (later) of the starting dates if they are different.

endDate
Ending date for the checking. If *endDate* is not specified, the ending date is the ending date for the calendars, or the lower (earlier) of the ending dates if they are different.

Usage

The function checks if the frequencies, off-exceptions, on-exceptions, and aligned patterns are the same for the two calendars. If they are all the same, the function returns 1; if they are not all the same, the function returns 0.

The function does not require the calendars to have the same starting and ending dates.

Example

Check if two calendars (*GENERIC-CAL1* and *GENERIC-CAL2*) are equal:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal1 ORDSYS.ORDTCalendar;
tstCal2 ORDSYS.ORDTCalendar;
resultCal ORDSYS.ORDTCalendar;
equalFlag INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select the calendars GENERIC-CAL1 into tstCal1
 -- and GENERIC-CAL2 into tstCal2
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal1
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';
 SELECT value(cal) INTO tstCal2
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL2';

 -- Display the calendars tstCal1 and tstCal2.
 SELECT ORDSYS.TimeSeries.Display(tstCal1) INTO dummyVal FROM dual;
 SELECT ORDSYS.TimeSeries.Display(tstCal2) INTO dummyVal FROM dual;

 -- Compare tstCal1 and tstCal2 for equality.
 DBMS_OUTPUT.NEW_LINE;
 equalFlag := ORDSYS.Calendar.EqualCals(tstCal1, tstCal2);
 DBMS_OUTPUT.PUT_LINE('EqualCals(GENERIC-CAL1, GENERIC-CAL2) = ' || equalFlag);

END;
/
```

This example might display the following output. In this example, the returned value of 0 indicates that the calendars are not equal.

```
Calendar Name = GENERIC-CAL1
```

```
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Calendar Name = GENERIC-CAL2
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1997 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
     10/13/1996 00:00:00     11/10/1996 00:00:00     12/14/1996 00:00:00
     01/04/1997 00:00:00     02/09/1997 00:00:00     03/08/1997 00:00:00
     04/05/1997 00:00:00     05/11/1997 00:00:00     06/08/1997 00:00:00
 offExceptions :
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
     01/01/1997 00:00:00     02/12/1997 00:00:00     03/04/1997 00:00:00
     04/07/1997 00:00:00     05/05/1997 00:00:00     06/09/1997 00:00:00

EqualCals(GENERIC-CAL1, GENERIC-CAL2) = 0
```

GetOffset

Format

```
ORDSYS.TimeSeries.GetOffset(
    inputCal IN ORDSYS.ORDTCalendar,
    origin_date IN DATE,
    reference_date IN DATE
    ) RETURN INTEGER;
```

Description

Given a calendar, one date (*origin_date*), and another date (*reference_date*), returns the number of timestamps that the second date is offset from the first.

Parameters

inputCal
The input calendar.

origin_date
Date from which the offset is to be computed.

reference_date
Date whose offset from *origin_date* is to be returned.

Usage

The function considers the frequency, pattern, and exceptions of the calendar.

The returned integer is positive if *reference_date* is one or more timestamps in the future with respect to *origin_date*, and negative if it is in the past with respect to *origin_date*. For example, assume that the calendar includes Mondays through Fridays, that 04-Jul-1997 (Friday) is an off-exception, and that 03-Jul-1997 (Thursday) is the *origin_date*. If 10-Jul-1997 (Thursday) is the *reference_date*, the returned offset is 4; if the *reference_date* is 01-Jul-1997 (Monday), the returned offset is -2.

If *origin_date* and *reference_date* are the same, the function returns 0 (zero).

An exception is returned if the calendar has an empty or null pattern.

Example

Return the offset of 05-Jun-1996 from 04-Mar-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get offset of 05-JUN-1996 from 04-MAR-1996.
 tstDate1 := TO_DATE('04/03/1996');
 tstDate2 := TO_DATE('06/05/1996');
 result  := ORDSYS.Calendar.GetOffset(tstCal,tstDate1, tstDate2);
 DBMS_OUTPUT.PUT_LINE('GetOffset(' || tstDate1 ||' , ' || tstDate2
                                                  || ') = ' || result);
END;
/
```

This example might produce the following output. In this example, 05-Jun-1996 is 45 timestamps later than 04-Mar-1996.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
```

```
    01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
    04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
    07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
    01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
    04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
    07/09/1996 00:00:00

GetOffset(04/03/1996 00:00:00 , 06/05/1996 00:00:00) = 45
```

InsertExceptions

Format

```
ORDSYS.Calendar.InsertExceptions(
    inputCal IN ORDSYS.ORDTCalendar,
    newExcDate IN DATE
    ) RETURN ORDSYS.ORDTCalendar;
```

or

```
ORDSYS.Calendar.InsertExceptions(
    inputCal IN ORDSYS.ORDTCalendar,
    newExcTab IN ORDSYS.ORDTDateTab
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Inserts into the specified calendar all exceptions that either match a specified date (*newExcDate*) or are included in a table of dates (*newExcTab*), and returns the resulting calendar.

Parameters

inputCal
The calendar into which one or more exceptions are to be inserted.

newExcDate
The date to be inserted as an exception in the calendar.

newExcTab
A table of dates to be inserted as exceptions in the calendar.

Usage

For each date to be inserted, the function inserts it in the appropriate list (off-exceptions or on-exceptions), according to the frequency and pattern of the calendar.

If a date to be inserted is already an exception in the calendar, the function ignores the request to insert the date.

If *newExcDate* or *newExcTab* is empty or null, or if all dates to be inserted already exist in the calendar as exceptions, the function returns the input calendar with no changes.

Example

Insert some exceptions into a calendar.

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDTab ORDSYS.ordtDateTab;
resultCal ORDSYS.ORDTCalendar;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Populate tstDTab with some on- and off-exceptions.
 tstDTab := ORDSYS.ORDTDateTab(
                 '02/10/1996', -- ON  Exception
                 '07/09/1996', -- OFF Exception
                 '03/17/1996', -- ON  Exception
                 '04/08/1996');-- OFF Exception
 SELECT ORDSYS.TimeSeries.Display(tstDTab, 'Input DateTab')
 INTO dummyVal
 FROM dual;


 -- Insert some exceptions in tstCal.
```

```
 resultCal := ORDSYS.Calendar.InsertExceptions(tstCal, tstDTab);
 SELECT ORDSYS.TimeSeries.Display(resultCal) INTO dummyVal
 FROM dual;

END;
/
```

This example might produce the following output. The second display of information about *GENERIC-CAL1* includes the added on-exceptions and off-exceptions.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Input DateTab :

     02/10/1996 00:00:00     07/09/1996 00:00:00     03/17/1996 00:00:00
     04/08/1996 00:00:00

Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     02/10/1996 00:00:00
     03/17/1996 00:00:00     03/24/1996 00:00:00     04/27/1996 00:00:00
     05/19/1996 00:00:00     06/23/1996 00:00:00     07/07/1996 00:00:00
     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
```

```
04/04/1996 00:00:00     04/08/1996 00:00:00     05/08/1996 00:00:00
06/25/1996 00:00:00     07/09/1996 00:00:00
```

IntersectCals

Format

```
ORDSYS.Calendar.IntersectCals(
    cal1 ORDSYS.ORDTCalendar,
    cal2 ORDSYS.ORDTCalendar
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Returns the intersection of two calendars.

Parameters

cal1
The first calendar to be intersected.

cal2
The second calendar to be intersected.

Usage

The function performs an intersection of the two input calendars, as follows:

- The starting date of the resulting calendar is the later of the starting dates of the two calendars, that is, resulting minDate = max(minDate1, minDate2).
- The ending date of the resulting calendar is the earlier of the ending dates of the two calendars, that is, resulting maxDate = min(maxDate1, maxDate2).
- The intersection of the aligned patterns is computed. For example, if both calendars have a *day* frequency with Sunday as the first day, and if *cal1* has a pattern of '0,1,1,1,1,1,0' and *cal2* has a pattern of '0,0,1,1,1,1,1', the resulting pattern is '0,0,1,1,1,1,0' (that is, the calendar includes only Tuesdays, Wednesdays, Thursdays, and Fridays).
- The intersection of the on-exception lists of the two calendars is computed. For example, if *cal1* has 30-Mar and 29-Jun as on-exceptions and *cal2* has 29-Jun and 28-Sep as on-exceptions, the resulting calendar has only 29-Jun as an on-exception.

- The union of the off-exceptions of the two calendars is computed. For example, if *cal1* has 01-Jan and 04-Jul as off-exceptions and *cal2* has 01-Jan and 14-Jul as off-exceptions, the resulting calendar has 01-Jan, 04-Jul, and 14-Jul as off-exceptions.

If the frequencies of the two calendars are not equal, the function returns NULL.

Contrast this function with UnionCals, which performs a union of two calendars.

IntersectCals and CombineCals differ as follows:

- CombineCals requires the frequencies and the aligned patterns of the two calendars to be equal, whereas IntersectCals requires only that the frequencies be equal. However, IntersectCals does require that the patterns be of the same length.
- CombineCals lets you specify starting and ending dates for the resulting calendar, whereas IntersectCals does not let you specify starting and ending dates.

Example

Combine two calendars (*GENERIC-CAL1* and *GENERIC-CAL2*), then intersect the two calendars:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal1 ORDSYS.ORDTCalendar;
tstCal2 ORDSYS.ORDTCalendar;
resultCal ORDSYS.ORDTCalendar;
equalFlag INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select the calendars GENERIC-CAL1 into tstCal1
 -- and GENERIC-CAL2 into tstCal2
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal1
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';
 SELECT value(cal) INTO tstCal2
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL2';
```

```
 -- Display the calendars tstCal1 and tstCal2.
 SELECT ORDSYS.TimeSeries.Display(tstCal1) INTO dummyVal FROM dual;
 SELECT ORDSYS.TimeSeries.Display(tstCal2) INTO dummyVal FROM dual;

 -- Combine tstCal1 and tstCal2.
 resultCal := ORDSYS.Calendar.CombineCals(tstCal1, tstCal2, equalFlag);
 SELECT ORDSYS.TimeSeries.Display(resultCal, 'result of CombineCals')
 INTO dummyVal
 FROM dual;
 DBMS_OUTPUT.PUT_LINE('equalFlag = ' || equalFlag);

 -- Intersect tstCal1 and tstCal2.
 resultCal := ORDSYS.Calendar.IntersectCals(tstCal1, tstCal2);
 SELECT ORDSYS.TimeSeries.Display(resultCal, 'result of IntersectCals')
 INTO dummyVal
 FROM dual;

END;
/
```

This example might produce the following output:

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Calendar Name = GENERIC-CAL2
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1997 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
```

```
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
     10/13/1996 00:00:00     11/10/1996 00:00:00     12/14/1996 00:00:00
     01/04/1997 00:00:00     02/09/1997 00:00:00     03/08/1997 00:00:00
     04/05/1997 00:00:00     05/11/1997 00:00:00     06/08/1997 00:00:00
 offExceptions :
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
     01/01/1997 00:00:00     02/12/1997 00:00:00     03/04/1997 00:00:00
     04/07/1997 00:00:00     05/05/1997 00:00:00     06/09/1997 00:00:00

result of CombineCals :

 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
equalFlag = 0

result of IntersectCals :

 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
```

InvalidTimeStampsBetween

Format

```
ORDSYS.Calendar.InvalidTimeStampsBetween(
    inputCal IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN ORDSYS.ORDTDateTab;
```

Description

Given starting and ending input timestamps, returns a table (ORDTDateTab) containing the invalid timestamps within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used to determine whether a timestamp is valid or invalid.

startDate
Starting date in the range to be checked for invalid timestamps.

endDate
Ending date in the range to be checked for invalid timestamps.

Usage

A timestamp is invalid if any of the following conditions is true:

- It is outside the date range of the calendar.
- It is an off-exception in the calendar.
- It is imprecise (for example, a timestamp of 02-Jul-1997 if the calendar frequency is *month*).
- It is null.

startDate and *endDate* are included in the check for invalid timestamps.

If there are no invalid timestamps in the date range, the function returns an empty ORDTDateTab.

If *startDate* is greater (later) than *endDate*, an exception is raised.

Contrast this function with TimeStampsBetween, which returns a table containing the valid timestamps in a date range.

Example

Return a table of invalid timestamps between 03-Mar-1996 and 03-Jun-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
resultDTab ORDSYS.ordtDateTab;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get all the invalid timestamps between 03-MAR-1996 and 03-JUN-1996.
 tstDate1 := TO_DATE('03/03/1996');
 tstDate2 := TO_DATE('06/03/1996');
 resultDTab := ORDSYS.Calendar.InvalidTimeStampsBetween
                                    (tstCal, tstDate1, tstDate2);
 SELECT ORDSYS.TimeSeries.Display(resultDTab, 'InValid timestamps')
 INTO dummyVal
 FROM dual;
```

```
END;
/
```

This example might produce the following output:

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
         0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

InValid timestamps :

     03/03/1996 00:00:00     03/05/1996 00:00:00     03/09/1996 00:00:00
     03/10/1996 00:00:00     03/16/1996 00:00:00     03/17/1996 00:00:00
     03/23/1996 00:00:00     03/30/1996 00:00:00     03/31/1996 00:00:00
     04/04/1996 00:00:00     04/06/1996 00:00:00     04/07/1996 00:00:00
     04/13/1996 00:00:00     04/14/1996 00:00:00     04/20/1996 00:00:00
     04/21/1996 00:00:00     04/28/1996 00:00:00     05/04/1996 00:00:00
     05/05/1996 00:00:00     05/08/1996 00:00:00     05/11/1996 00:00:00
     05/12/1996 00:00:00     05/18/1996 00:00:00     05/25/1996 00:00:00
     05/26/1996 00:00:00     06/01/1996 00:00:00     06/02/1996 00:00:00
```

IsValidCal

Format

```
ORDSYS.Calendar.IsValidCal(
    inputCal IN ORDSYS.ORDTCalendar
    ) RETURN BINARY_INTEGER
```

Description

Returns 1 if the calendar is valid and 0 if the calendar is not valid.

Parameters

inputCal
The calendar to be checked for validity.

Usage

A calendar is invalid (not valid) if it contains any errors. This function does not correct any errors or perform any repair operations on the calendar.

Contrast this function with the ValidateCal function, which checks the validity of the calendar and repairs any correctable errors. For detailed information on calendar errors, see the information on ValidateCal in this chapter.

If the IsValidCal function returns 0, you should do the following before you attempt to use the calendar:

1. Use the ValidateCal function to repair any correctable errors.
2. If there are any errors that ValidateCal cannot correct, correct these errors yourself.
3. Repeat steps 1 and 2 as often as necessary until the resulting calendar is valid.

Example

Use the IsValidCal and ValidateCal functions and the DisplayValCal procedure with an invalid calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
```

```
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
outMessage  varchar2(32750);
invOnExc    ORDSYS.ORDTDateTab;
invOffExc   ORDSYS.ORDTDateTab;
impOnExc    ORDSYS.ORDTDateTab;
impOffExc   ORDSYS.ORDTDateTab;
dummyval     integer;
validFlag     integer;
tstCal1     ORDSYS.ORDTCalendar :=
               ORDSYS.ORDTCalendar(
                  0,
                  'CALENDAR FOO',
                  4,
                  ORDSYS.ORDTPattern(ORDSYS.ORDTPatternBits(1,1,1,1,1,0,0),
                                                 TO_DATE('01-08-1996 01:01:01')),
                  TO_DATE('01-01-1975'),
  TO_DATE('01-01-1999'),
                  ORDSYS.ORDTExceptions(
           TO_DATE('02-03-1969'), -- Date < minDate,
           TO_DATE('02-14-1969'), -- Date < minDate,
           TO_DATE('02-03-1999'), -- Date > maxDate,
           TO_DATE('02-17-1999'), -- Date > maxDate,
           TO_DATE('12-31-1995'), -- Maps to 0 in pattern (Sunday)
           TO_DATE('01-13-1996'), -- Maps to 0 in pattern (Saturday)
           TO_DATE('02-24-1996'), -- Maps to 0 in pattern (Saturday)
           TO_DATE('03-30-1996'), -- Maps to 0 in pattern (Saturday)
           TO_DATE('02-02-1996 01:01:01'), -- Imprecise
           TO_DATE('03-04-1996 01:01:01'), -- Imprecise
           TO_DATE('04-05-1996 02:02:02'), -- Imprecise
           TO_DATE('03-25-1996'), -- Valid off-exception
           TO_DATE('01-22-1996'), -- Valid, but out of sequence
           TO_DATE('02-12-1996'),
           TO_DATE('04-30-1996'),
           NULL,                  -- Null date
           TO_DATE('02-12-1996'), -- Duplicate date within OFFs
           NULL,                  -- Null date
           TO_DATE('04-30-1996'), -- Duplicate off-exception
           NULL,                  -- Null date
           TO_DATE('03-25-1996'), -- Duplicate off-exception
           TO_DATE('01-22-1996'), -- Duplicate off-exception
           TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
           TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
           TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
```

```
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('02-02-1996'),
            TO_DATE('03-04-1996'),
            TO_DATE('05-06-1997')),
        ORDSYS.ORDTExceptions(
            TO_DATE('02-08-1969'), -- Date < minDate,
            TO_DATE('02-15-1969'), -- Date < minDate,
            TO_DATE('02-13-1999'), -- Date > maxDate,
            TO_DATE('02-20-1999'), -- Date > maxDate,
            TO_DATE('01-03-1996'), -- Maps to 1 in pattern (Wednesday)
            TO_DATE('02-19-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-18-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('05-27-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-23-1996 01:01:01'), -- Imprecise
            TO_DATE('02-18-1996 01:01:01'), -- Imprecise
            TO_DATE('05-26-1996 01:01:01'), -- Imprecise
            TO_DATE('01-13-1996'), -- Valid on-exception
            TO_DATE('01-14-1996'), -- Valid on-exception
            NULL,                  -- Null date
            NULL,                  -- Null date
            TO_DATE('02-24-1996'), -- Valid on-exception
            TO_DATE('03-23-1996'), -- Valid on-exception
            TO_DATE('01-13-1996'), -- Duplicate on-exception
            TO_DATE('01-14-1996'), -- Duplicate on-exception
            TO_DATE('02-24-1996'), -- Duplicate on-exception
            TO_DATE('03-23-1996'), -- Duplicate on-exception
            TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
            TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
            TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('01-06-1996'), -- Valid, but out of sequence
            TO_DATE('02-03-1996'),
            TO_DATE('05-04-1997'))
                  );
BEGIN
   SELECT ORDSYS.TIMESERIES.Display(tstCal1, 'tstCal1') INTO dummyval
   FROM dual;
   validFlag := ORDSYS.CALENDAR.IsValidCal(tstCal1);
   IF(validFlag = 0)
   THEN
       validFlag := ORDSYS.CALENDAR.ValidateCal(
                tstCal1, outMessage, invOnExc, invOffExc, impOnExc, impOffExc
                );

       ORDSYS.TIMESERIES.DisplayValCal(
```

```
            validFlag,
            outMessage,
            invOnExc,
            invOffExc,
            impOnExc,
            impOffExc,
            tstCal1,
            'Your Message'
            );
   END IF;
END;
/
```

This example might produce the following output:

```
tstCal1 :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
 MaxDate = 01/01/1999 00:00:00
 patBits:
         1111100
 patAnchor = 01/08/1996 01:01:01
 onExceptions  :
     02/08/1969 00:00:00     02/15/1969 00:00:00     02/13/1999 00:00:00
     02/20/1999 00:00:00     01/03/1996 00:00:00     02/19/1996 00:00:00
     03/18/1996 00:00:00     05/27/1996 00:00:00     03/23/1996 01:01:01
     02/18/1996 01:01:01     05/26/1996 01:01:01     01/13/1996 00:00:00
     01/14/1996 00:00:00
     02/24/1996 00:00:00     03/23/1996 00:00:00     01/13/1996 00:00:00
     01/14/1996 00:00:00     02/24/1996 00:00:00     03/23/1996 00:00:00
     01/17/1996 00:00:00     05/28/1996 00:00:00     06/18/1996 00:00:00
     04/23/1996 00:00:00     01/06/1996 00:00:00     02/03/1996 00:00:00
     05/04/1997 00:00:00
 offExceptions :
     02/03/1969 00:00:00     02/14/1969 00:00:00     02/03/1999 00:00:00
     02/17/1999 00:00:00     12/31/1995 00:00:00     01/13/1996 00:00:00
     02/24/1996 00:00:00     03/30/1996 00:00:00     02/02/1996 01:01:01
     03/04/1996 01:01:01     04/05/1996 02:02:02     03/25/1996 00:00:00
     01/22/1996 00:00:00     02/12/1996 00:00:00     04/30/1996 00:00:00
          02/12/1996 00:00:00
     04/30/1996 00:00:00          03/25/1996 00:00:00
     01/22/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00     02/02/1996 00:00:00
```

```
     03/04/1996 00:00:00     05/06/1997 00:00:00

DisplayValCal Your Message:

TS-WRN: the input calendar has rectifiable errors. See the message for details

message output by validateCal:

TS-WRN: fixed precision of the pattern anchor date
TS-WRN: removed superfluous dates in the on exception list (refer invalidOnExc)
TS-WRN: fixed imprecise dates in the on exception list (refer impreciseOnExc)
TS-WRN: removed null dates in the on exception list
TS-WRN: sorted the on exceptions list
TS-WRN: removed duplicate dates in the on exceptions list
TS-WRN: removed superfluous dates in off exceptions list (refer invalidOffExc)
TS-WRN: fixed imprecise dates in the off exception list (refer impreciseOffExc)
TS-WRN: removed null dates in the off exception list
TS-WRN: sorted the off exceptions list
TS-WRN: removed duplicate dates in the off exceptions list
TS-WRN: the on exceptions list was trimmed between calendar minDate & maxDate
TS-WRN: the off exceptions list was trimmed between calendar minDate & maxDate

list of invalid on exceptions :

     01/03/1996 00:00:00     02/19/1996 00:00:00     03/18/1996 00:00:00
     05/27/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00

list of invalid off exceptions :

     12/31/1995 00:00:00     01/13/1996 00:00:00     02/24/1996 00:00:00
     03/30/1996 00:00:00

list of imprecise on exceptions :

     03/23/1996 01:01:01     02/18/1996 01:01:01     05/26/1996 01:01:01
list of imprecise off exceptions :

     02/02/1996 01:01:01     03/04/1996 01:01:01     04/05/1996 02:02:02

the validated calendar :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
```

```
MaxDate = 01/01/1999 00:00:00
patBits:
        1111100
patAnchor = 01/08/1996 00:00:00
onExceptions  :
    01/06/1996 00:00:00     01/13/1996 00:00:00     01/14/1996 00:00:00
    02/03/1996 00:00:00     02/18/1996 00:00:00     02/24/1996 00:00:00
    03/23/1996 00:00:00     05/26/1996 00:00:00     05/04/1997 00:00:00
offExceptions :
    01/17/1996 00:00:00     01/22/1996 00:00:00     02/02/1996 00:00:00
    02/12/1996 00:00:00     03/04/1996 00:00:00     03/25/1996 00:00:00
    04/05/1996 00:00:00     04/23/1996 00:00:00     04/30/1996 00:00:00
    05/28/1996 00:00:00     06/18/1996 00:00:00     05/06/1997 00:00:00
```

IsValidDate

Format

```
ORDSYS.Calendar.IsValidDate(
    inputCal IN ORDSYS.ORDTCalendar,
    checkDate IN DATE
    ) RETURN BINARY_INTEGER;
```

Description

Checks whether an input date is valid or invalid according to the specified calendar.

Parameters

inputCal
The calendar to be used to determine whether the input timestamp is valid or invalid.

checkDate
The timestamp to be checked for validity according to the calendar.

Usage

If *checkDate* is valid, the function returns 1; if *checkDate* is invalid, the function returns 0.

A timestamp is invalid if any of the following conditions is true:

- It is outside the date range of the calendar.
- It is an off-exception in the calendar.
- It is not sufficiently precise (for example, a timestamp of 01-Jul-1997 if the calendar has a frequency of year).
- It is null.

Example

Check if 02-Jan-1996 is a valid timestamp for a calendar (*GENERIC-CAL1*):

```
CONNECT ORDTUSER/ORDTUSER
```

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Verify if 02-JAN-1996 (a Monday) is a valid date and display the result.
 tstDate1 := TO_DATE('01/02/1996');
 result  := ORDSYS.Calendar.IsValidDate(tstCal,tstDate1);
 DBMS_OUTPUT.PUT_LINE('IsValidDate(' || tstDate1 || ') = ' || result);

END;
/
```

This example might produce the following output. In this example, the returned value of 1 indicates that 02-Jan-1996 is a valid timestamp for the *BUSINESS-96* calendar.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
```

```
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

IsValidDate(01/02/1996 00:00:00) = 1
```

NumInvalidTimeStampsBetween

Format

```
ORDSYS.Calendar.NumInvalidTimeStampsBetween(
    inputCal IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN INTEGER;
```

Description

Given starting and ending input timestamps, returns the number of invalid timestamps within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used to determine whether a timestamp is valid or invalid.

startDate
Starting date in the range to be checked for invalid timestamps.

endDate
Ending date in the range to be checked for invalid timestamps.

Usage

A timestamp is invalid if any of the following conditions is true:

- It is outside the date range of the calendar.
- It is an off-exception in the calendar.
- It is not sufficiently precise (for example, a timestamp of 01-Jul-1997 if the calendar has a frequency of year).
- It is null.

startDate and *endDate* are included in the check for invalid timestamps.

If there are no invalid timestamps in the date range, the function returns 0 (zero).

If *startDate* is greater (later) than *endDate*, an exception is raised.

Contrast this function with NumTimeStampsBetween, which returns the number of valid timestamps in a date range.

Example

Return the number of invalid timestamps between 03-Feb-1996 and 16-May-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get the number of invalid timestamps between 03-FEB-1996 and 16-MAY-1996.
 tstDate1 := TO_DATE('02/03/1996');
 tstDate2 := TO_DATE('05/16/1996');
 result  := ORDSYS.Calendar.NumInvalidTimeStampsBetween(
                                                    tstCal,tstDate1, tstDate2);
 DBMS_OUTPUT.PUT_LINE('NumInvalidTimeStampsBetween(' || tstDate1 ||' , ' ||
                                                   tstDate2|| ') = ' || result);
END;
/
```

This example might produce the following output. In this example, there are 30 invalid timestamps in the specified date range.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

NumInvalidTimeStampsBetween(02/03/1996 00:00:00 , 05/16/1996 00:00:00) = 30
```

NumOffExceptions

Format

```
ORDSYS.Calendar.NumOffExceptions(
    inputCal  IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN INTEGER;
```

Description

Given starting and ending input timestamps, returns the number of off-exceptions within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used in computing the number of off-exceptions.

startDate
Starting date in the range to be checked for off-exceptions.

endDate
Ending date in the range to be checked for off-exceptions.

Usage

startDate and *endDate* are included in the check for off-exceptions. (For an explanation of off-exceptions and on-exceptions, see Section 2.2.)

If *startDate* is greater (later) than *endDate*, an exception is raised.

Example

Return the number of off-exceptions between 02-Feb-1996 and 07-Jul-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
```

```
DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get the number of off-exceptions between 02-FEB-1996 and 07-JUL-1996.
 tstDate1 := TO_DATE('02/02/1996');
 tstDate2 := TO_DATE('07/07/1996');
 result  := ORDSYS.Calendar.NumOffExceptions(tstCal,tstDate1, tstDate2);
 DBMS_OUTPUT.PUT_LINE('NumOffExceptions(' || tstDate1 ||' , ' || tstDate2
                                                 || ') = ' || result);
END;
/
```

This example might produce the following output. As the last line of the output indicates, there are five off-exceptions in the specified date range (02-Feb-1996 through 07-Jul-1996).

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
```

```
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

NumOffExceptions(02/02/1996 00:00:00 , 07/07/1996 00:00:00) = 5
```

NumOnExceptions

Format

```
ORDSYS.Calendar.NumOnExceptions(
    inputCal  IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN INTEGER;
```

Description

Given starting and ending input timestamps, returns the number of on-exceptions within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used in computing the number of on-exceptions.

startDate
Starting date in the range to be checked for on-exceptions.

endDate
Ending date in the range to be checked for on-exceptions.

Usage

startDate and *endDate* are included in the check for on-exceptions. (For an explanation of off-exceptions and on-exceptions, see Section 2.2.)

If *startDate is* greater (later) than *endDate,* an exception is raised.

Example

Return the number of on-exceptions between 02-Feb-1996 and 07-Jul-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
```

```
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get the number of ON Exceptions between 02-FEB-1996 and 07-JUL-1996.
 tstDate1 := TO_DATE('02/02/1996');
 tstDate2 := TO_DATE('07/07/1996');
 result  := ORDSYS.Calendar.NumOnExceptions(tstCal,tstDate1, tstDate2);
 DBMS_OUTPUT.PUT_LINE('NumOnExceptions(' || tstDate1 ||' , ' || tstDate2
                                                 || ') = ' || result);
END;
/
```

This example might produce the following output. As the last line of the output indicates, there are six on-exceptions in the specified date range (02-Feb-1996 through 07-Jul-1996).

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
```

```
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

NumOnExceptions(02/02/1996 00:00:00 , 07/07/1996 00:00:00) = 6
```

NumTimeStampsBetween

Format

```
ORDSYS.Calendar.NumTimeStampsBetween(
    inputCal  IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN INTEGER;
```

Description

Given starting and ending input timestamps, returns the number of valid timestamps within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used to determine whether a timestamp is valid or invalid.

startDate
Starting date in the range to be checked for invalid timestamps.

endDate
Ending date in the range to be checked for invalid timestamps.

Usage

A timestamp is invalid (not valid) if any of the following conditions is true:

- It is outside the date range of the calendar.
- It is an off-exception in the calendar.
- It is not sufficiently precise (for example, a timestamp of 01-Jul-1997 if the calendar has a frequency of year).
- It is null.

startDate and *endDate* are included in the check for valid timestamps.

If there are no valid timestamps in the date range, the function returns 0 (zero).

If *startDate* is greater (later) than *endDate*, an exception is raised.

Contrast this function with NumInvalidTimeStampsBetween, which returns the number of invalid timestamps in a date range.

Example

Return the number of valid timestamps between 03-Feb-1996 and 16-May-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
result INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get the number of Valid timestamps between 03-FEB-1996 and 16-MAY-1996.
 tstDate1 := TO_DATE('02/03/1996');
 tstDate2 := TO_DATE('05/16/1996');
 result  := ORDSYS.Calendar.NumTimeStampsBetween(tstCal,tstDate1, tstDate2);
 DBMS_OUTPUT.PUT_LINE('NumTimeStampsBetween(' || tstDate1 ||' , ' || tstDate2
                                                 || ') = ' || result);
END;
/
```

This example might produce the following output. In this example, there are 74 valid timestamps in the specified date range.

```
Calendar Name = GENERIC-CAL1
```

```
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
         0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

NumTimeStampsBetween(02/03/1996 00:00:00 , 05/16/1996 00:00:00) = 74
```

OffsetDate

Format

```
ORDSYS.Calendar.OffsetDate(
    inputCal  IN ORDSYS.ORDTCalendar,
    origin IN DATE,
    relOffset IN INTEGER
    ) RETURN DATE;
```

Description

Given a reference date (*origin*) and an offset with respect to the origin (*relOffset*), returns the timestamp corresponding to the offset input.

Parameters

inputCal
Calendar from which the date is to be returned.

origin
The date to which the offset value (relOffset) is to be applied in computing the returned date.

relOffset
The relative offset of the returned date with respect to the origin.

Usage

The function returns the date of the timestamp at the *relOffset* number of timestamps from the *origin* date. If *relOffset* is positive, the returned date is later than *origin*; if *relOffset* is negative, the returned date is earlier than *origin*. If *relOffset* is zero (0), the returned date is *origin* if *origin* is a valid date; however, if *relOffset* is zero (0) and *origin* is not a valid date, the function returns NULL.

For example, assume a Monday through Friday business day calendar for 1997 with 04-Jul-1997 (Friday) defined as an off-exception, and assume that *origin* is 02-Jul-1997 (Wednesday):

- If *relOffset* = 2, the returned date is 07-Jul-1997 (Monday).

- If *relOffset* = -2, the returned date is 30-Jun-1997 (Monday).
- If *relOffset* = 0, the returned date is 02-Jul-1997 (Wednesday).

If the *origin* date is not in the calendar (*inputCal*), the next later date is used if *relOffset* is positive or zero, and the next earlier date is used if *relOffset* is negative. Using the calendar in the preceding example, if *origin* is specified as 04-Jul-1997 and if *relOffset* = 2, then 07-Jul-1997 (Monday, the next business day) is used as *origin*, and the returned date is 09-Jul-1997 (Wednesday).

If the calendar pattern is empty or null, an exception is raised.

Example

Get the dates 20 timestamps later and 20 timestamps earlier than 03-Mar-1996 in the GENERIC-CAL1 calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
resultDate date;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Offset 03-MAR-1996 by 20.
 tstDate1 := TO_DATE('03/03/1996');
 relOffset := 20;
 resultDate := ORDSYS.Calendar.OffsetDate(tstCal, tstDate1, relOffset);
 DBMS_OUTPUT.PUT_LINE('OffsetDate(' || tstDate1 || ' , ' || relOffset
                                                  || ') = ' || resultDate);
```

```
 DBMS_OUTPUT.NEW_LINE;

 -- Offset 03-MAR-1996 by -20.
 tstDate1 := TO_DATE('03/03/1996');
 relOffset := -20;
 resultDate := ORDSYS.Calendar.OffsetDate(tstCal, tstDate1, relOffset);
 DBMS_OUTPUT.PUT_LINE('OffsetDate(' || tstDate1 || ' , ' || relOffset
                                                 || ') = ' || resultDate);
 DBMS_OUTPUT.NEW_LINE;

END;
/
```

This example might produce the following output. In this example, 29-Mar-1996 is 20 timestamps later than 03-Mar-1996, and 05-Feb-1996 is 20 timestamps earlier than 03-Mar-1996.

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

OffsetDate(03/03/1996 00:00:00 , 20) = 03/29/1996 00:00:00

OffsetDate(03/03/1996 00:00:00 , -20) = 02/05/1996 00:00:00
```

SetPrecision

Format

```
ORDSYS.Calendar.SetPrecision(
    timestamp IN DATE,
    frequency IN INTEGER
    ) RETURN DATE;
```

Description

Given a timestamp and a frequency, returns a timestamp that reflects the level of precision implied by the frequency.

Parameters

timestamp
Timestamp whose precision is to be set.

frequency
Frequency to be applied in setting the precision.

Usage

The returned timestamp reflects the precision implied by the frequency, as explained in Section 2.2.1. For example, if the input timestamp is 29-Dec-1997 12:45:00 and *frequency* is 6 (*month*), the returned timestamp is 01-Dec-1997 00:00:00. Table 4–1 shows the frequencies, their precision conventions, and the resulting precision if an input timestamp of 19-Sep-1997 09:09:09 is supplied.

Table 4–1 SetPrecision and Timestamp of 19-Sep-1997 09:09:09

Frequency (Every:)	Precision Convention	Result
second	MM-DD-YYYY HH24:MI:SS	09-19-1997 09:09:09
minute	MM-DD-YYYY HH24:MI:00	09-19-1997 09:09:00

Table 4–1 SetPrecision and Timestamp of 19-Sep-1997 09:09:09 (Cont.)

Frequency (Every:)	Precision Convention	Result
hour	MM-DD-YYYY HH24:00:00	09-19-1997 09:00:00
day	MM-DD-YYYY 00:00:00 (midnight)	09-19-1997 00:00:00
month	MM-01-YYYY 00:00:00 (midnight of first day of month)	09-01-1997 00:00:00
year	01-01-YYYY 00:00:00 (midnight of first day of year)	01-01-1997 00:00:00

If the *frequency* value is not valid, an exception is raised.

Example

Set the precision of an imprecise timestamp (here, a timestamp containing hour, minute, and second values where the calendar has a *day* frequency):

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
resultDate date;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

-- Set the precision of an imprecise date.
```

```
 tstDate1 := TO_DATE('03/03/1996 01:01:01');
 resultDate := ORDSYS.Calendar.SetPrecision(tstDate1, tstCal.frequency);
 DBMS_OUTPUT.PUT_LINE('SetPrecision(' ||
                            TO_CHAR(tstDate1) ||
                            ' , ' || tstCal.frequency || ') = ' ||
                            TO_CHAR(resultDate) );
END;
/
```

This example might produce the following output. In this example, the hour, minute, and second components of the timestamp are set to zeroes because the calendar frequency is 4 (*day*).

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

SetPrecision(03/03/1996 01:01:01 , 4) = 03/03/1996 00:00:00
```

TimeStampsBetween

Format

```
ORDSYS.Calendar.TimeStampsBetween(
    inputCal IN ORDSYS.ORDTCalendar,
    startDate IN DATE,
    endDate IN DATE
    ) RETURN ORDSYS.ORDTDateTab;
```

Description

Given starting and ending input timestamps, returns a table (ORDTDateTab) containing the valid timestamps within that range according to the specified calendar.

Parameters

inputCal
The calendar to be used to determine whether a timestamp is valid or invalid.

startDate
Starting date in the range to be checked for valid timestamps.

endDate
Ending date in the range to be checked for valid timestamps.

Usage

A timestamp is invalid if any of the following conditions is true:

- It is outside the date range of the calendar.
- It is an off-exception in the calendar.
- It is not sufficiently precise (for example, a timestamp of 01-Jul-1997 if the calendar has a frequency of year).
- It is null.

startDate and *endDate* are included in the check for valid timestamps.

If there are no valid timestamps in the date range, the function returns an empty ORDTDateTab.

If *startDate* is greater (later) than *endDate*, an exception is raised.

Contrast this function with InvalidTimeStampsBetween, which returns a table containing the invalid timestamps in a date range.

Example

Return a table of valid timestamps between 03-Mar-1996 and 03-Jun-1996 in the *GENERIC-CAL1* calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal ORDSYS.ORDTCalendar;
tstDate1 DATE;
tstDate2 DATE;
resultDTab ORDSYS.ordtDateTab;
dummyVal INTEGER;
relOffset INTEGER;

BEGIN

 -- Select a calendar (say, GENERIC-CAL1) into tstCal
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';

 -- Display the calendar.
 SELECT ORDSYS.TimeSeries.Display(tstCal) INTO dummyVal FROM dual;
 DBMS_OUTPUT.NEW_LINE;

 -- Get all the valid timestamps between 03-MAR-1996 and 03-JUN-1996.
 tstDate1 := TO_DATE('03/03/1996');
 tstDate2 := TO_DATE('06/03/1996');
 resultDTab := ORDSYS.Calendar.TimeStampsBetween(tstCal, tstDate1, tstDate2);
 SELECT ORDSYS.TimeSeries.Display(resultDTab, 'Valid timestamps')
 INTO dummyVal
 FROM dual;

END;
```

```
/
```

This example might produce the following output:

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
         0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Valid timestamps :

     03/04/1996 00:00:00     03/06/1996 00:00:00     03/07/1996 00:00:00
     03/08/1996 00:00:00     03/11/1996 00:00:00     03/12/1996 00:00:00
     03/13/1996 00:00:00     03/14/1996 00:00:00     03/15/1996 00:00:00
     03/18/1996 00:00:00     03/19/1996 00:00:00     03/20/1996 00:00:00
     03/21/1996 00:00:00     03/22/1996 00:00:00     03/24/1996 00:00:00
     03/25/1996 00:00:00     03/26/1996 00:00:00     03/27/1996 00:00:00
     03/28/1996 00:00:00     03/29/1996 00:00:00     04/01/1996 00:00:00
     04/02/1996 00:00:00     04/03/1996 00:00:00     04/05/1996 00:00:00
     04/08/1996 00:00:00     04/09/1996 00:00:00     04/10/1996 00:00:00
     04/11/1996 00:00:00     04/12/1996 00:00:00     04/15/1996 00:00:00
     04/16/1996 00:00:00     04/17/1996 00:00:00     04/18/1996 00:00:00
     04/19/1996 00:00:00     04/22/1996 00:00:00     04/23/1996 00:00:00
     04/24/1996 00:00:00     04/25/1996 00:00:00     04/26/1996 00:00:00
     04/27/1996 00:00:00     04/29/1996 00:00:00     04/30/1996 00:00:00
     05/01/1996 00:00:00     05/02/1996 00:00:00     05/03/1996 00:00:00
     05/06/1996 00:00:00     05/07/1996 00:00:00     05/09/1996 00:00:00
     05/10/1996 00:00:00     05/13/1996 00:00:00     05/14/1996 00:00:00
     05/15/1996 00:00:00     05/16/1996 00:00:00     05/17/1996 00:00:00
     05/19/1996 00:00:00     05/20/1996 00:00:00     05/21/1996 00:00:00
     05/22/1996 00:00:00     05/23/1996 00:00:00     05/24/1996 00:00:00
     05/27/1996 00:00:00     05/28/1996 00:00:00     05/29/1996 00:00:00
     05/30/1996 00:00:00     05/31/1996 00:00:00     06/03/1996 00:00:00
```

Section 3.3.2 contains an example showing the use of TimeStampsBetween to create a time series for use with the DeriveExceptions function.

UnionCals

Format

```
ORDSYS.Calendar.UnionCals(
    cal1 ORDSYS.ORDTCalendar,
    cal2 ORDSYS.ORDTCalendar
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Returns a calendar that is the union of two input calendars.

Parameters

cal1
The first calendar on which the union operation is to be performed.

cal2
The second calendar on which the union operation is to be performed.

Usage

The function performs a union of the two input calendars, as follows:

- The starting date of the resulting calendar is the later of the starting dates of the two calendars, that is, resulting minDate = max(minDate1, minDate2).
- The ending date of the resulting calendar is the earlier of the ending dates of the two calendars, that is, resulting maxDate = min(maxDate1, maxDate2).
- The union of the aligned patterns is computed. For example, if both calendars have a *day* frequency with Sunday as the first day, and if *cal1* has a pattern of '0,1,1,1,1,1,0' and *cal2* has a pattern of '0,0,1,1,1,1,1', the resulting pattern is '0,1,1,1,1,1,1' (that is, the calendar includes Mondays through Saturdays).
- The union of the on-exception lists is computed. For example, if *cal1* has 30-Mar and 29-Jun as on-exceptions and *cal2* has 29-Jun and 28-Sep as on-exceptions, the resulting calendar has 30-Mar, 29-Jun, and 28-Sep on-exceptions.

- The intersection of the off-exception lists is computed. For example, if *cal1* has 01-Jan and 04-Jul as off-exceptions and *cal2* has 01-Jan and 14-Jul as off-exceptions, the resulting calendar has only 01-Jan as an off-exception.

If the frequencies of the two calendars are not equal, the function returns NULL.

Contrast this function with IntersectCals, which intersects two calendars.

Example

Perform a union of two calendars:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
tstCal1 ORDSYS.ORDTCalendar;
tstCal2 ORDSYS.ORDTCalendar;
resultCal ORDSYS.ORDTCalendar;
equalFlag INTEGER;
dummyVal INTEGER;

BEGIN

 -- Select the calendars GENERIC-CAL1 into tstCal1
 -- and GENERIC-CAL2 into tstCal2
 -- from stockdemo_calendars.
 SELECT value(cal) INTO tstCal1
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL1';
 SELECT value(cal) INTO tstCal2
 FROM ORDTDEV.stockdemo_calendars cal
 WHERE cal.name = 'GENERIC-CAL2';

 -- Display the calendars tstCal1 and tstCal2.
 SELECT ORDSYS.TimeSeries.Display(tstCal1) INTO dummyVal FROM dual;
 SELECT ORDSYS.TimeSeries.Display(tstCal2) INTO dummyVal FROM dual;

 -- Union tstCal1 and tstCal2.
 resultCal := ORDSYS.Calendar.Unioncals(tstCal1, tstCal2);
 SELECT ORDSYS.TimeSeries.Display(resultCal, 'result of UnionCals')
 INTO dummyVal
 FROM dual;

END;
```

```
/
```

This example might produce the following output:

```
Calendar Name = GENERIC-CAL1
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
     01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
     04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
 offExceptions :
     01/08/1996 00:00:00     02/02/1996 00:00:00     03/05/1996 00:00:00
     04/04/1996 00:00:00     05/08/1996 00:00:00     06/25/1996 00:00:00
     07/09/1996 00:00:00

Calendar Name = GENERIC-CAL2
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1997 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 00:00:00
 onExceptions  :
     07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
     10/13/1996 00:00:00     11/10/1996 00:00:00     12/14/1996 00:00:00
     01/04/1997 00:00:00     02/09/1997 00:00:00     03/08/1997 00:00:00
     04/05/1997 00:00:00     05/11/1997 00:00:00     06/08/1997 00:00:00
 offExceptions :
     07/09/1996 00:00:00     08/05/1996 00:00:00     09/10/1996 00:00:00
     10/23/1996 00:00:00     11/19/1996 00:00:00     12/12/1996 00:00:00
     01/01/1997 00:00:00     02/12/1997 00:00:00     03/04/1997 00:00:00
     04/07/1997 00:00:00     05/05/1997 00:00:00     06/09/1997 00:00:00

result of UnionCals :

 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 12/31/1996 00:00:00
 patBits:
          1111100
```

```
patAnchor = 01/08/1996 00:00:00
onExceptions  :
    01/21/1996 00:00:00     02/03/1996 00:00:00     03/24/1996 00:00:00
    04/27/1996 00:00:00     05/19/1996 00:00:00     06/23/1996 00:00:00
    07/07/1996 00:00:00     08/04/1996 00:00:00     09/15/1996 00:00:00
    10/13/1996 00:00:00     11/10/1996 00:00:00     12/14/1996 00:00:00
offExceptions :
    07/09/1996 00:00:00
```

ValidateCal

Format

```
ORDSYS.Calendar.ValidateCal(
    cal INOUT ORDSYS.ORDTCalendar,
    outMessage OUT VARCHAR2,
    invOnExc OUT ORDTDateTab,
    invOffExc OUT ORDTDateTab,
    impOnExc OUT ORDTDateTab,
    impOffExc OUT ORDTDateTab
    ) RETURN BINARY_INTEGER;
```

Description

Validates a calendar and, if necessary, repairs the calendar and outputs information related to the problems and repairs.

Parameters

cal
The calendar to be validated and (if necessary) repaired.

outMessage
Message describing how the calendar was repaired (if the return value = 1) or why the calendar could not be repaired (if the return value = -1).

invOnExc
Table of the invalid on-exceptions found in the calendar.

invOffExc
Table of the invalid off-exceptions found in the calendar.

impOnExc
Table of the imprecise on-exceptions found in the calendar.

impOffExc
Table of the imprecise off-exceptions found in the calendar.

Usage

This function returns one of the following values:

Value	Meaning
0	The calendar is valid. No errors were found.
1	Correctable errors were found and corrected. The resulting calendar is valid.
-1	Uncorrectable errors were found. The calendar is not valid.

Errors in the input calendar make it invalid. Depending on the error, it may be correctable or uncorrectable. Correctable errors are repaired by the ValidateCal function. If all errors are correctable, the resulting calendar is valid.

For a calendar to be valid, all timestamps in the off-exception and on-exception lists must be consistent with the defined pattern for the calendar. If one or more exception timestamps are not consistent with the pattern, the calendar is invalid. For example, if 04-Jan-1997 (Saturday) is in the off-exception list of a calendar whose pattern includes only Mondays through Fridays as normal business days, 04-Jan-1997 is an invalid off-exception (because as a Saturday is would normally be an "off" day).

Imprecise exception timestamps are repaired. For an explanation of precision, see Section 2.2.1.

Table 4–2 lists correctable errors and the repair actions taken by the ValidateCal function:

Table 4–2 Errors Repaired by ValidateCal

Error	Repair Action
Imprecise anchor date	The precision is adjusted.
Character other than 1 or 0 in the pattern	All pattern characters other than 0 or 1 are set to 1.
Imprecise date	The precision is adjusted.
Superfluous date (for example, a regular valid date in the on-exceptions list)	The date is removed from the exceptions list.
Null date	The date is removed from the calendar.

Table 4–2 Errors Repaired by ValidateCal (Cont.)

Error	Repair Action
Unsorted dates	The dates are sorted.
Duplicate dates in the on-exceptions or off-exceptions list	Duplicates are removed; the date appears only once in the list.
Date appearing in both the on-exceptions and off-exceptions lists	The date is removed from the inappropriate list, depending on the pattern and the anchor date.
Date outside the date range of the calendar	The date is removed from the exceptions list.

The following errors are not correctable. The function returns -1 if one or more of these errors are found:

- The frequency is not valid.
- The starting date is later than the ending date.
- The pattern is null or empty
- All pattern bits are empty.
- One or more pattern bits are null.
- The anchor date is null and the pattern is not "all ones" or "all zeroes" (for example, a pattern of '0,1,1,1,1,1,0' but no anchor date specified).

If the function returns -1, you should not use the calendar until you have fixed the errors that ValidateCal could not fix. Then use ValidateCal again, and use the calendar only if the function returns 0 or 1.

You can use the DisplayValCal procedure to display the information returned by the ValidateCal function. See the information on DisplayValCal in this chapter.

The IsValidCal function (described in this chapter) checks the validity of the calendar but does not perform any repair operations.

Example

Use the IsValidCal and ValidateCal functions and the DisplayValCal procedure with an invalid calendar:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
```

```
DECLARE
outMessage  varchar2(32750);
invOnExc    ORDSYS.ORDTDateTab;
invOffExc   ORDSYS.ORDTDateTab;
impOnExc    ORDSYS.ORDTDateTab;
impOffExc   ORDSYS.ORDTDateTab;
dummyval     integer;
validFlag     integer;
tstCal1     ORDSYS.ORDTCalendar :=
               ORDSYS.ORDTCalendar(
                  0,
                  'CALENDAR FOO',
                  4,
                  ORDSYS.ORDTPattern(ORDSYS.ORDTPatternBits(1,1,1,1,1,0,0),
                                              TO_DATE('01-08-1996 01:01:01')),
                  TO_DATE('01-01-1975'),
  TO_DATE('01-01-1999'),
                  ORDSYS.ORDTExceptions(
            TO_DATE('02-03-1969'), -- Date < minDate,
            TO_DATE('02-14-1969'), -- Date < minDate,
            TO_DATE('02-03-1999'), -- Date > maxDate,
            TO_DATE('02-17-1999'), -- Date > maxDate,
            TO_DATE('12-31-1995'), -- Maps to 0 in pattern (Sunday)
            TO_DATE('01-13-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('02-24-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('03-30-1996'), -- Maps to 0 in pattern (Saturday)
            TO_DATE('02-02-1996 01:01:01'), -- Imprecise
            TO_DATE('03-04-1996 01:01:01'), -- Imprecise
            TO_DATE('04-05-1996 02:02:02'), -- Imprecise
            TO_DATE('03-25-1996'), -- Valid off-exception
            TO_DATE('01-22-1996'), -- Valid, but out of sequence
            TO_DATE('02-12-1996'),
            TO_DATE('04-30-1996'),
            NULL,                  -- Null date
            TO_DATE('02-12-1996'), -- Duplicate date within OFFs
            NULL,                  -- Null date
            TO_DATE('04-30-1996'), -- Duplicate off-exception
            NULL,                  -- Null date
            TO_DATE('03-25-1996'), -- Duplicate off-exception
            TO_DATE('01-22-1996'), -- Duplicate off-exception
            TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
            TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
            TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('02-02-1996'),
```

```
            TO_DATE('03-04-1996'),
            TO_DATE('05-06-1997')),
     ORDSYS.ORDTExceptions(
            TO_DATE('02-08-1969'), -- Date < minDate,
            TO_DATE('02-15-1969'), -- Date < minDate,
            TO_DATE('02-13-1999'), -- Date > maxDate,
            TO_DATE('02-20-1999'), -- Date > maxDate,
            TO_DATE('01-03-1996'), -- Maps to 1 in pattern (Wednesday)
            TO_DATE('02-19-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-18-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('05-27-1996'), -- Maps to 1 in pattern (Monday)
            TO_DATE('03-23-1996 01:01:01'), -- Imprecise
            TO_DATE('02-18-1996 01:01:01'), -- Imprecise
            TO_DATE('05-26-1996 01:01:01'), -- Imprecise
            TO_DATE('01-13-1996'), -- Valid on-exception
            TO_DATE('01-14-1996'), -- Valid on-exception
            NULL,                  -- Null date
            NULL,                  -- Null date
            TO_DATE('02-24-1996'), -- Valid on-exception
            TO_DATE('03-23-1996'), -- Valid on-exception
            TO_DATE('01-13-1996'), -- Duplicate on-exception
            TO_DATE('01-14-1996'), -- Duplicate on-exception
            TO_DATE('02-24-1996'), -- Duplicate on-exception
            TO_DATE('03-23-1996'), -- Duplicate on-exception
            TO_DATE('01-17-1996'), -- Added to on- and off-exceptions
            TO_DATE('05-28-1996'), -- Added to on- and off-exceptions
            TO_DATE('06-18-1996'), -- Added to on- and off-exceptions
            TO_DATE('04-23-1996'), -- Added to on- and off-exceptions
            TO_DATE('01-06-1996'), -- Valid, but out of sequence
            TO_DATE('02-03-1996'),
            TO_DATE('05-04-1997'))
                  );
BEGIN
   SELECT ORDSYS.TIMESERIES.Display(tstCal1, 'tstCal1') INTO dummyval
   FROM dual;
   validFlag := ORDSYS.CALENDAR.IsValidCal(tstCal1);
   IF(validFlag = 0)
   THEN
       validFlag := ORDSYS.CALENDAR.ValidateCal(
                tstCal1, outMessage, invOnExc, invOffExc, impOnExc, impOffExc
                );

       ORDSYS.TIMESERIES.DisplayValCal(
             validFlag,
             outMessage,
```

```
            invOnExc,
            invOffExc,
            impOnExc,
            impOffExc,
            tstCal1,
            'Your Message'
            );
   END IF;
END;
/
```

This example might produce the following output:

```
tstCal1 :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
 MaxDate = 01/01/1999 00:00:00
 patBits:
          1111100
 patAnchor = 01/08/1996 01:01:01
 onExceptions  :
     02/08/1969 00:00:00     02/15/1969 00:00:00     02/13/1999 00:00:00
     02/20/1999 00:00:00     01/03/1996 00:00:00     02/19/1996 00:00:00
     03/18/1996 00:00:00     05/27/1996 00:00:00     03/23/1996 01:01:01
     02/18/1996 01:01:01     05/26/1996 01:01:01     01/13/1996 00:00:00
     01/14/1996 00:00:00
     02/24/1996 00:00:00     03/23/1996 00:00:00     01/13/1996 00:00:00
     01/14/1996 00:00:00     02/24/1996 00:00:00     03/23/1996 00:00:00
     01/17/1996 00:00:00     05/28/1996 00:00:00     06/18/1996 00:00:00
     04/23/1996 00:00:00     01/06/1996 00:00:00     02/03/1996 00:00:00
     05/04/1997 00:00:00
 offExceptions :
     02/03/1969 00:00:00     02/14/1969 00:00:00     02/03/1999 00:00:00
     02/17/1999 00:00:00     12/31/1995 00:00:00     01/13/1996 00:00:00
     02/24/1996 00:00:00     03/30/1996 00:00:00     02/02/1996 01:01:01
     03/04/1996 01:01:01     04/05/1996 02:02:02     03/25/1996 00:00:00
     01/22/1996 00:00:00     02/12/1996 00:00:00     04/30/1996 00:00:00
          02/12/1996 00:00:00
     04/30/1996 00:00:00          03/25/1996 00:00:00
     01/22/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00     02/02/1996 00:00:00
     03/04/1996 00:00:00     05/06/1997 00:00:00
```

```
DisplayValCal Your Message:

TS-WRN: the input calendar has rectifiable errors. See the message for details

message output by validateCal:

TS-WRN: fixed precision of the pattern anchor date
TS-WRN: removed superfluous dates in the on exception list (refer invalidOnExc)
TS-WRN: fixed imprecise dates in the on exception list (refer impreciseOnExc)
TS-WRN: removed null dates in the on exception list
TS-WRN: sorted the on exceptions list
TS-WRN: removed duplicate dates in the on exceptions list
TS-WRN: removed superfluous dates in off exceptions list (refer invalidOffExc)
TS-WRN: fixed imprecise dates in the off exception list (refer impreciseOffExc)
TS-WRN: removed null dates in the off exception list
TS-WRN: sorted the off exceptions list
TS-WRN: removed duplicate dates in the off exceptions list
TS-WRN: the on exceptions list was trimmed between calendar minDate & maxDate
TS-WRN: the off exceptions list was trimmed between calendar minDate & maxDate

list of invalid on exceptions :

     01/03/1996 00:00:00     02/19/1996 00:00:00     03/18/1996 00:00:00
     05/27/1996 00:00:00     01/17/1996 00:00:00     05/28/1996 00:00:00
     06/18/1996 00:00:00     04/23/1996 00:00:00

list of invalid off exceptions :

     12/31/1995 00:00:00     01/13/1996 00:00:00     02/24/1996 00:00:00
     03/30/1996 00:00:00

list of imprecise on exceptions :

     03/23/1996 01:01:01     02/18/1996 01:01:01     05/26/1996 01:01:01
list of imprecise off exceptions :

     02/02/1996 01:01:01     03/04/1996 01:01:01     04/05/1996 02:02:02

the validated calendar :

Calendar Name = CALENDAR FOO
 Frequency = 4
 MinDate = 01/01/1975 00:00:00
 MaxDate = 01/01/1999 00:00:00
 patBits:
```

```
        1111100
patAnchor = 01/08/1996 00:00:00
onExceptions  :
    01/06/1996 00:00:00     01/13/1996 00:00:00     01/14/1996 00:00:00
    02/03/1996 00:00:00     02/18/1996 00:00:00     02/24/1996 00:00:00
    03/23/1996 00:00:00     05/26/1996 00:00:00     05/04/1997 00:00:00
offExceptions :
    01/17/1996 00:00:00     01/22/1996 00:00:00     02/02/1996 00:00:00
    02/12/1996 00:00:00     03/04/1996 00:00:00     03/25/1996 00:00:00
    04/05/1996 00:00:00     04/23/1996 00:00:00     04/30/1996 00:00:00
    05/28/1996 00:00:00     06/18/1996 00:00:00     05/06/1997 00:00:00
```

5

Time Series and Time Scaling Functions: Reference

The Oracle8 Time Series Cartridge library consists of:

- datatypes (described in Section 2.2.2)
- calendar functions (described in Chapter 4)
- time series and time scaling functions (described in this chapter)

Two separate reference chapters are provided for the functions because the functions described in each are typically done at different times in the application development cycle and by people performing different job roles:

- Calendar functions are mainly used by product developers, such as ISVs, to develop new time series functions and to administer and modify calendars.
- Time series and time scaling functions are used mainly by application developers and some end users after the associated calendar or calendars have been defined.

Syntax notes:

- The ORDSYS schema name and the package name must be used with the function name, although public synonyms can be created to eliminate the need for specifying the schema name (see Section 1.4). Each function is included in a PL/SQL package, such as Calendar or TimeSeries. The ORDSYS schema name and the package name are included in the Format and in any examples.
- Function calls are not case sensitive, except for any quoted literal values. For example, the following code line excerpts are valid and semantically identical:

```
select CAST(TimeSeries.ExtractTable(close) AS ORDTNumTab)
select cast(TIMESERIES.extracttable(close) as ordtnumtab)
```

```
select cast(TiMeSeRiEs.eXtRaCtTaBlE(ClosE) As ordtNUMtab)
```

- The syntax and examples show the reference-based interface (types ORDT-NumSeriesIOTRef and ORDTVarchar2SeriesIOTRef).

Usage note:

All time series functions accept both references and instances as parameters. (For example, an ORDTNumSeriesIOTRef parameter could also be ORDTNumSeries.) All time series functions return instances. Thus, if you nest functions, such as *Cmax(Cmax(...), ...)*, the innermost nesting accepts a reference and outputs an instance, and any other functions in the nesting accept an instance and output an instance.

For an explanation of the reference-based interface, see Section 2.5.2.

Cavg

Format

```
ORDSYS.TimeSeries.Cavg(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, returns an ORDTNumSeries with each element containing the cumulative average up to and including the corresponding element in the input ORDTNumSeries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the cumulative average is to be computed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the cumulative average is to be computed. If *endDate* is specified, *startDate* must also be specified.

Usage

Only non-null values are considered in computing the cumulative average.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate is* earlier than *startDate*.

If *startDate* and *endDate* are specified, the time series is trimmed to the date range before the cumulative average is computed.

Example

Return the cumulative average of the closing price of stock ACME for November 1996:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Cavg(ts.close,to_date('01-NOV-96','DD-MON-YY'),
                           to_date('30-NOV-96','DD-MON-YY'))
               ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-NOV-96         59
04-NOV-96       59.5
05-NOV-96         60
06-NOV-96       60.5
07-NOV-96         61
08-NOV-96       61.5
11-NOV-96         62
12-NOV-96       62.5
13-NOV-96         63
14-NOV-96       63.5
15-NOV-96         64
18-NOV-96       64.5
19-NOV-96         65
20-NOV-96       65.5
21-NOV-96         66
22-NOV-96       66.5
25-NOV-96         67
26-NOV-96       67.5
27-NOV-96         68
29-NOV-96       68.5
20 rows selected.
```

Cmax

Format

```
ORDSYS.TimeSeries.Cmax(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, returns an ORDTNumSeries with each element containing the cumulative maximum up to and including the corresponding element in the input ORDTNumSeries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the cumulative maximum is to be returned. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the cumulative maximum is to be returned. If *endDate* is specified, *startDate* must also be specified.

Usage

Only non-null values are considered in determining the cumulative maximum.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate is* earlier than *startDate*.

If *startDate* and *endDate* are specified, the time series is trimmed to the date range before the cumulative maximum is computed.

Example

Return the cumulative maximum of the closing price of stock ACME for November 1996:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Cmax(ts.close,to_date('01-NOV-96','DD-MON-YY'),
                           to_date('30-NOV-96','DD-MON-YY'))
               ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output. (Note that this output reflects the simplified artificial data in the usage demo database, where the closing price rises one point each day.)

```
TSTAMP    VALUE
--------- ----------
01-NOV-96         59
04-NOV-96         60
05-NOV-96         61
06-NOV-96         62
07-NOV-96         63
08-NOV-96         64
11-NOV-96         65
12-NOV-96         66
13-NOV-96         67
14-NOV-96         68
15-NOV-96         69
18-NOV-96         70
19-NOV-96         71
20-NOV-96         72
21-NOV-96         73
22-NOV-96         74
25-NOV-96         75
26-NOV-96         76
27-NOV-96         77
29-NOV-96         78
20 rows selected.
```

Cmin

Format

```
ORDSYS.TimeSeries.Cmin(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, returns an ORDTNumSeries with each element containing the cumulative minimum up to and including the corresponding element in the input ORDTNumSeries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the cumulative minimum is to be returned. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the cumulative minimum is to be returned. If *endDate* is specified, *startDate* must also be specified.

Usage

Only non-null values are considered in determining the cumulative minimum.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate is* earlier than *startDate*.

If *startDate* and *endDate* are specified, the time series is trimmed to the date range before the cumulative minimum is computed.

Example

Return the cumulative minimum of the closing price of stock ACME for November 1996:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Cmin(ts.close,to_date('01-NOV-96','DD-MON-YY'),
                           to_date('30-NOV-96','DD-MON-YY'))
               ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output. (Note that this output reflects the simplified artificial data in the usage demo database, where the closing price rises one point each day.)

```
TSTAMP    VALUE
--------- ----------
01-NOV-96         59
04-NOV-96         59
05-NOV-96         59
06-NOV-96         59
07-NOV-96         59
08-NOV-96         59
11-NOV-96         59
12-NOV-96         59
13-NOV-96         59
14-NOV-96         59
15-NOV-96         59
18-NOV-96         59
19-NOV-96         59
20-NOV-96         59
21-NOV-96         59
22-NOV-96         59
25-NOV-96         59
26-NOV-96         59
27-NOV-96         59
29-NOV-96         59
20 rows selected.
```

Cprod

Format

```
ORDSYS.TimeSeries.Cprod(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, returns an ORDTNumSeries with each element containing the cumulative product of multiplication up to and including the corresponding element in the input ORDTNumSeries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the cumulative product is to be computed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the cumulative product is to be computed. If *endDate* is specified, *startDate* must also be specified.

Usage

Only non-null values are considered in computing the cumulative product.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate is* earlier than *startDate*.

If *startDate* and *endDate* are specified, the time series is trimmed to the date range before the cumulative product is computed.

Example

Return the cumulative product of the daily volume of stock ACME for the first four trading days of November 1996. (This example is presented merely to illustrate the function; the results of this query have no practical value for financial analysis.)

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Cprod(ts.volume,to_date('01-NOV-96','DD-MON-YY'),
                          to_date('06-NOV-96','DD-MON-YY'))
               ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-NOV-96       1000
04-NOV-96    1000000
05-NOV-96 1000000000
06-NOV-96 1.0000E+12
4 rows selected.
```

Csum

Format

```
ORDSYS.TimeSeries.Csum(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, returns an ORDTNumSeries with each element containing the cumulative sum up to and including the corresponding element in the input ORDTNumSeries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the cumulative sum is to be computed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the cumulative sum is to be computed. If *endDate* is specified, *startDate* must also be specified.

Usage

Only non-null values are considered in computing the cumulative sum.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate is* earlier than *startDate*.

If *startDate* and *endDate* are specified, the time series is trimmed to the date range before the cumulative sum is computed.

Example

Return the cumulative sum of the daily volume of stock ACME for November 1996:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Csum(ts.volume,to_date('01-NOV-96','DD-MON-YY'),
                           to_date('30-NOV-96','DD-MON-YY'))
               ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-NOV-96       1000
04-NOV-96       2000
05-NOV-96       3000
06-NOV-96       4000
07-NOV-96       5000
08-NOV-96       6000
11-NOV-96       7000
12-NOV-96       8000
13-NOV-96       9000
14-NOV-96      10000
15-NOV-96      11000
18-NOV-96      12000
19-NOV-96      13000
20-NOV-96      14000
21-NOV-96      15000
22-NOV-96      16000
25-NOV-96      17000
26-NOV-96      18000
27-NOV-96      19000
29-NOV-96      20000
20 rows selected.
```

DeriveExceptions

Format

Approach 1:

```
ORDSYS.TimeSeries.DeriveExceptions(
    inputCal  IN ORDSYS.ORDTCalendar,
    DateTab IN ORDSYS.ORDTDateTab
    ) RETURN ORDSYS.ORDTCalendar;
```

Approach 2:

```
ORDSYS.TimeSeries.DeriveExceptions(
    series1 ORDTNumSeriesIOTRef,
    series2 ORDTNumSeriesIOTRef
    ) RETURN ORDSYS.ORDTCalendar;
```

or

```
ORDSYS.TimeSeries.DeriveExceptions(
    series1 ORDTVarchar2SeriesIOTRef,
    series2 ORDTVarchar2SeriesIOTRef
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Derives calendar exceptions from a calendar and a table of dates (Approach 1) or from two time series (Approach 2).

Parameters

inputCal
The calendar that contains no exceptions and for which exceptions are to be derived.

DateTab
The table of dates that includes all dates in the time series (for example, all dates on which stock XYZ traded).

series1
The "reference" time series that contains no exceptions and all valid timestamps from the calendar (for example, all Monday through Friday dates within the date range of the calendar).

series2
The time series that contains the timestamps to be used in deriving the exceptions for the resulting calendar (for example, all dates on which stock XYZ traded).

Usage

See Section 2.2.4 for a detailed explanation of the two approaches to using this function.

Example

See Sections 3.3.1 and 3.3.2 for examples of the two approaches to using this function.

Display

Format

```
ORDSYS.TimeSeries.Display(
    ts  ORDSYS.[see parameter description]
    [,mesg VARCHAR2]
    ) RETURN INTEGER;
```

Description

Displays various information (see the description of the *ts* parameter) using DBMS_OUTPUT routines.

Parameters

ts
The object to be displayed. Because the function is overloaded, this parameter can be any of the following datatypes:

- ORDTNumSeriesIOTRef or ORDTNumSeries
- ORDTVarchar2SeriesIOTRef or ORDTVarcharSeries
- ORDTNumTab
- ORDTVarchar2Tab
- ORDTNumCell
- ORDTVarchar2Cell
- ORDTDateTab
- ORDTCalendar
- ORDTExceptions
- ORDTPattern

mesg
Optional message text to be included in the display heading ("Timeseries dump for <mesg>").

Usage

Use the SET SERVEROUTPUT ON statement to view the output of the Display function. However, the default display buffer of 2000 bytes is often too small to display a large time series. In such cases you must use the ENABLE procedure of the DBMS_OUTPUT package to specify a larger display buffer size. For example:

```
DBMS_OUTPUT.ENABLE(1000000);
```

You should use Display only for development and debugging. Specify a display buffer larger than 2000 only when necessary, because the display buffer uses shared system resources, and a large value might affect the performance of other users.

Because the Display function uses DBMS_OUTPUT routines, it is subject to the limitations of these routines. These limitations include the following:

- Output cannot exceed 1 megabyte.
- The Display function cannot be used with the OCI.
- SQL*Plus does not support DBMS_OUTPUT in the context of a SELECT statement, but it does support DBMS_OUTPUT for anonymous PL/SQL blocks.

Example

Display the output for a query that returns the 10 highest closing prices for stock AONE for the month of January 1996:

```
SET SERVEROUTPUT ON
DECLARE
     tmp INTEGER;
BEGIN
SELECT ORDSYS.TimeSeries.Display(
         ORDSYS.TimeSeries.TSMaxN(close,10,
            to_date('01011996','MMDDYYYY'),
            to_date('01311996','MMDDYYYY')))
         INTO tmp
FROM ORDTDEV.stocks_ts
WHERE ticker ='AONE';
END;
/
```

This example might produce the following output:

```
Tab Data:
 ----------------------------
```

```
Date                    Value
01/24/1996 00:00:00      43.9138
01/25/1996 00:00:00      42.9925
01/31/1996 00:00:00      42.9925
01/26/1996 00:00:00      42.7413
01/30/1996 00:00:00      42.7413
01/29/1996 00:00:00      42.5738
01/23/1996 00:00:00      41.9875
01/22/1996 00:00:00      41.82
01/19/1996 00:00:00      41.485
01/18/1996 00:00:00      40.815
----------------------------
```

The preceding example works from both SQL*Plus and the Server Manager (svrmgrl) prompt. The following version of the example works from the Server Manager prompt but not from SQL*Plus:

```
SET SERVEROUTPUT ON
SELECT ORDSYS.TimeSeries.Display(
         ORDSYS.TimeSeries.TSMaxN(close,10,
            to_date('01011996','MMDDYYYY'),
            to_date('01311996','MMDDYYYY')))
FROM ORDTDEV.stocks_ts
WHERE ticker ='AONE';
```

See the TSMaxN function for an example that returns the same information, but that uses a subquery instead of the Display function.

DisplayValTS Procedure

Format

```
ORDSYS.TimeSeries.DisplayValTS(
    validFlag IN INTEGER,
    outMessage IN VARCHAR2,
    loDateTab IN ORDSYS.ORDTDateTab,
    hiDateTab  IN ORDSYS.ORDTDateTab,
    impreciseDateTab IN ORDSYS.ORDTDateTab,
    duplicateDateTab IN ORDSYS.ORDTDateTab,
    extraDateTab IN ORDSYS.ORDTDateTab,
    missingDateTab IN ORDSYS.ORDTDateTab,
    mesg IN VARCHAR2
    );
```

Description

Displays the results returned by the ValidateTS function.

> **Note:** DisplayValTS is a procedure, not a function. Procedures do not return values.

Parameters

validFlag
The return value from the ValidateTS function.

outMessage
The diagnostic returned by the ValidateTS function.

loDateTab
A table of dates before the starting date of the calendar associated with the time series.

hiDateTab
A table of dates after the starting date of the calendar associated with the time series.

impreciseDateTab
A table of the imprecise dates found in the time series.

duplicateDateTab
A table of the duplicate dates (dates that appear more than once in the time series).

extraDateTab
A table of dates that are included in the time series but that should be excluded based on the calendar definition (for example, a Saturday timestamp that is in a Monday-Friday calendar and that is not an on-exception).

missingDateTab
A table of dates that are excluded from the time series but that should be included based on the calendar definition (for example, a Wednesday date that is not a holiday in a Monday-Friday calendar and for which there is no data). Such dates can be considered as "holes" in the time series.

mesg
Optional message.

Usage

This procedure is intended to be used with the ValidateTS function. See the information on ValidateTS in this chapter.

The DisplayValTS procedure uses the DBMS_OUTPUT package. See the Usage information for the Display function for limitations relating to the use of DBMS_OUTPUT.

Example

Use the IsValidTS and ValidateTS functions and the DisplayValTS procedure with an invalid time series:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
 numTS  ORDSYS.ORDTNumSeries;
 tempVal integer;
```

```
 retIsValid  integer;
 retValTS    integer;
 loDateTab  ORDSYS.ORDTDateTab := NULL;
 hiDateTab  ORDSYS.ORDTDateTab := NULL;
 impDateTab ORDSYS.ORDTDateTab := NULL;
 dupDateTab ORDSYS.ORDTDateTab := NULL;
 extraDateTab ORDSYS.ORDTDateTab := NULL;
 missingDateTab ORDSYS.ORDTDateTab := NULL;
 outMesg varchar2(2000);

BEGIN

   --  Set the buffer size
   DBMS_OUTPUT.ENABLE(100000);

   --
   -- NOTE: Here an instance of the time series is materialized
   -- so that it could be modified to generate an invalid time series.
   --
   SELECT ORDSYS.TIMESERIES.GetSeries(ts.open) INTO numTS
   FROM ordtdev.stockdemo_ts ts
   WHERE ts.ticker = 'ACME';

   -- Example of validating a valid time series.
   SELECT ordsys.timeseries.display(numTS, 'A VALID TIME SERIES') INTO tempVal
   FROM dual;
   retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
   retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg, loDateTab,
                           hiDateTab, impDateTab, dupDateTab,
                           extraDateTab, missingDateTab);
   DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
   DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
   ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                         impDateTab, dupDateTab, extraDateTab, missingDateTab,
                         'Testing DisplayValTS');
   DBMS_OUTPUT.NEW_LINE;

   -- For illustration let us first create an invalid timeseries.
   --
   -- Here we are adjusting the calendar's minDate and maxDate to avoid
   -- getting a huge list of missing dates.
   --
   numTS.cal.minDate := TO_DATE('10/28/1996');
   numTS.cal.maxDate := TO_DATE('01/05/1997');
```

```
   -- Add Dates Before numTS.cal.minDate
   numTS.series(10).tstamp := numTS.cal.minDate - 1;
   numTS.series(11).tstamp := numTS.cal.minDate - 2;

   -- Add Dates Beyond numTS.cal.maxDate
   numTS.series(12).tstamp := numTS.cal.maxDate + 1;
   numTS.series(13).tstamp := numTS.cal.maxDate + 2;

   -- Add some null timestamps
   numTS.series(14).tstamp := NULL;
   numTS.series(15).tstamp := NULL;

   -- Add some imprecise dates (some are duplicated)
   numTS.series(17).tstamp := numTS.series(16).tstamp + 1/24;
   numTS.series(18).tstamp := numTS.series(16).tstamp + 15/24;

   -- Add some duplicate timestamps
   numTS.series(19).tstamp := numTS.series(18).tstamp;
   numTS.series(21).tstamp := numTS.series(20).tstamp;

   -- Add some extra dates in the middle
   numTS.series(37).tstamp := TO_DATE('12/28/1996');
   numTS.series(36).tstamp := TO_DATE('12/29/1996');

   -- Add some holes at the end
   numTS.series(numTS.series.count).tstamp := TO_DATE('01/04/1997');

   -- Example of validating an invalid time series.
   SELECT ordsys.timeseries.display(numTS, 'AN INVALID TIME SERIES')
   INTO tempVal FROM dual;
   retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
   retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg,
                        loDateTab, hiDateTab, impDateTab,
                        dupDateTab, extraDateTab, missingDateTab);
   DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
   DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
   ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                     impDateTab, dupDateTab, extraDateTab, missingDateTab,
                     'Testing DisplayValTS');
END;
/
```

This example might produce the following output:

```
A VALID TIME SERIES :
```

```
Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 11/14/1996 00:00:00      68
 11/15/1996 00:00:00      69
 11/18/1996 00:00:00      70
 11/19/1996 00:00:00      71
 11/20/1996 00:00:00      72
 11/21/1996 00:00:00      73
 11/22/1996 00:00:00      74
 11/25/1996 00:00:00      75
 11/26/1996 00:00:00      76
 11/27/1996 00:00:00      77
 11/29/1996 00:00:00      78
 12/02/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
```

```
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/23/1996 00:00:00      94
 12/24/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 12/31/1996 00:00:00      99
 ----------------------------

Value returned by IsValid  = 1
Value returned by ValidateTS  = 1

DisplayValTS: Testing DisplayValTS:

TS-SUC: the input time series is a valid time series


AN INVALID TIME SERIES :

Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 10/28/1996 00:00:00
 MaxDate = 01/05/1997 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
```

```
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 10/27/1996 00:00:00      68
 10/26/1996 00:00:00      69
 01/06/1997 00:00:00      70
 01/07/1997 00:00:00      71
       72
       73
 11/22/1996 00:00:00      74
 11/22/1996 01:00:00      75
 11/22/1996 15:00:00      76
 11/22/1996 15:00:00      77
 11/29/1996 00:00:00      78
 11/29/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/29/1996 00:00:00      94
 12/28/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 01/04/1997 00:00:00      99
 ----------------------------

Value returned by IsValid  = 0
Value returned by ValidateTS  = 0

DisplayValTS: Testing DisplayValTS:

TS-WRN: the input time series has errors. See the message for details

message output by validateTS:
```

```
TS-ERR: the input time series is unsorted
TS-ERR: the time series has null timestamps
TS-ERR: the time series has timestamps < calendar minDate (refer LoDateTab)
TS-ERR: the time series has timestamps > calendar maxDate (refer HiDateTab)
TS-ERR: the time series has imprecise timestamps (refer impreciseDateTab)
TS-ERR: the time series has duplicate timestamps (refer DuplicateDateTab)

list of dates < calendar minDate - lowDateTab :

     10/26/1996 00:00:00     10/27/1996 00:00:00


list of dates > calendar maxDate - hiDateTab :
     01/06/1997 00:00:00     01/07/1997 00:00:00

list of imprecise dates - impreciseDateTab :

     11/22/1996 01:00:00     11/22/1996 15:00:00

list of duplicate dates - duplicateDateTab :

     11/22/1996 15:00:00     11/29/1996 00:00:00

ExtraDateTab :

     12/28/1996 00:00:00     12/29/1996 00:00:00     01/04/1997 00:00:00

MissingDateTab :

     10/28/1996 00:00:00     10/29/1996 00:00:00     10/30/1996 00:00:00
     10/31/1996 00:00:00     11/14/1996 00:00:00     11/15/1996 00:00:00
     11/18/1996 00:00:00     11/19/1996 00:00:00     11/20/1996 00:00:00
     11/21/1996 00:00:00     11/25/1996 00:00:00     11/26/1996 00:00:00
     11/27/1996 00:00:00     12/02/1996 00:00:00     12/23/1996 00:00:00
     12/24/1996 00:00:00     12/31/1996 00:00:00     01/01/1997 00:00:00
     01/02/1997 00:00:00     01/03/1997 00:00:00
```

ExtractCal

Format

```
ORDSYS.TimeSeries.ExtractCal(
    ts ORDSYS.ORDTNumSeriesIOTRef
    ) RETURN ORDSYS.ORDTCalendar;
```

or

```
ORDSYS.TimeSeries.ExtractCal(
    ts ORDSYS.ORDTVarchar2SeriesIOTRef
    ) RETURN ORDSYS.ORDTCalendar;
```

Description

Given a time series, returns a calendar that is the same as the calendar on which the time series is based.

Parameters

ts
The input time series.

Usage

The function returns a calendar that has the same starting and ending timestamps, pattern, frequency, and exceptions (on- and off-) as the calendar on which the specified time series is based.

An exception is returned if the time series (*ts*) is null.

Example

Return a calendar that matches the one on which the time series for the ACME ticker is based:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
```

```
dummyval INTEGER;

BEGIN

 SELECT ORDSYS.TimeSeries.Display(
     ORDSYS.TimeSeries.ExtractCal(ts.open), 'ExtractCal Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
/
```

This example might produce the following output:

```
ExtractCal Results :

Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
```

ExtractDate

Format

ORDSYS.TimeSeries.ExtractDate(

cell ORDSYS.ORDTNumCell

) RETURN DATE;

or

ORDSYS.TimeSeries.ExtractDate(

cell ORDSYS.ORDTVarchar2Cell

) RETURN DATE;

Description

Given an element in a time series, returns the date.

Parameters

cell
The time series element for which you want the date.

Usage

The time series element must first be identified, such as by using the GetNthElement function.

An exception is returned if the time series element (cell) is null.

Example

Return the date associated with the tenth element in a specified time series:

```
SELECT to_char( ORDSYS.TimeSeries.ExtractDate(
       ORDSYS.TimeSeries.GetNthElement(open, 10)),
             'MM/DD/YYYY HH24:MI:SS')
       FROM ORDTDEV.stocks_ts
       WHERE ticker = 'AONE';
```

This example might produce the following output:

```
TO_CHAR(ORDSYS.TIME
-------------------
01/15/1990 00:00:00
1 row selected.
```

ExtractTable

Format

ORDSYS.TimeSeries.ExtractTable(

ts ORDSYS.ORDTNumSeriesIOTRef

) RETURN ORDSYS.ORDTNumTab;

or

ORDSYS.TimeSeries.ExtractTable(

ts ORDSYS.ORDTVarchar2SeriesIOTRef

) RETURN ORDSYS.ORDTVarchar2Tab;

Description

Given a time series, returns the time series table (ORDTNumTab or ORDTVarchar2Tab) associated with the time series.

Parameters

ts
The input time series.

Usage

The function returns the time series table (ORDTNumTab or ORDTVarchar2Tab) associated with the time series.

An exception is returned if the time series (ts) is null.

Example

Return the closing prices for stock ACME:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(ts.close)
               as ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-NOV-96         59
04-NOV-96         60
05-NOV-96         61
   ...                    ...
31-DEC-96         99
41 rows selected.
```

ExtractValue

Format

ORDSYS.TimeSeries.ExtractValue(

cell ORDSYS.ORDTNumCell

) RETURN NUMBER;

or

ORDSYS.TimeSeries.ExtractValue(

cell ORDSYS.ORDTVarchar2Celf

) RETURN VARCHAR2;

Description

Given an element in a time series, returns the value stored in it.

Parameters

cell
The time series element for which you want the value.

Usage

The time series element must first be identified, such as by using the GetNthElement function.

An exception is returned if the time series element (cell) is null.

Example

Return the value of the tenth opening price in the *stocks_ts* table:

```
SELECT ORDSYS.TimeSeries.ExtractValue(
         ORDSYS.TimeSeries.GetNthElement(open, 10))
  FROM ORDTDEV.stocks_ts
  WHERE ticker = 'AONE';
```

This example might produce the following output:

```
ORDSYS.TIM
```

```
----------
   15.1875
1 row selected.
```

Fill

Format

```
ORDSYS.TimeSeries.Fill(
    ts ORDSYS.ORDTNumSeriesIOTRef
    [, fill_type INTEGER]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series and optionally a fill type, returns a time series in which values for missing dates are inserted. A missing date is a date that is defined by the calendar and within the time series bounds, but that is not in the current time series.

Parameters

ts
The input time series.

fill_type
One of the following integers indicating how missing values are to be filled:

- 0 = null: Insert nulls.
- 1 = forward repeat: Use the values from the preceding (most recent) timestamp.
- 2 = backward repeat: Use the values from the following (next in the future) timestamp.

If *fill_type* is omitted, 0 is assumed.

Usage

The function inserts timestamps and associated values for timestamps that are included in a calendar but for which no entries exist in the time series.

The *fill_type* parameter lets you choose the manner in which missing values will be defaulted. For example, assume that data for 30-Jan-1997 (Thursday) is missing from a time series and that it should be included because this date is within the calendar definition. Assume the following closing prices for stock XYZ:

- 49 on 29-Jan-1997
- 50 on 31-Jan-1997

The following table shows the closing price that would be inserted for 30-Jan-1997 with each of the *fill_type* parameter values:

fill_type	Closing Price for 30-Jan-1997
0	null
1	49
2	50

Some potential uses for this function include:

- deriving the price of a stock for a nontrading day

  For example, you may want to compare prices for a stock that trades on several stock exchanges, where the exchanges have different trading days.

- converting a quarterly time series to a daily time series

  For example, earnings per share (EPS) is computed quarterly, and stocks trade daily. To compute a price-earnings (PE) ratio, earnings per share is first converted to a daily time series using forward repeat. Then, the daily PE ratio is calculated by dividing the daily price time series value by the corresponding daily EPS time series value.

An exception is returned if the specified *fill_type* value is not 0, 1, or 2.

Example

Return a time series illustrating each *fill_type* value:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
-- For illustrating Fill we need a timeseries with missing dates.
-- In the following example, the timeseries 'FOO' has some missing dates
-- (07-DEC-1996 and 08-DEC-1996). Also, note that the calendar associated
-- with 'FOO' has an 'all one' pattern.
--
DECLARE
tstCal ORDSYS.ORDTCalendar;
ts     ORDSYS.ordtnumseries :=
            ORDSYS.ordtnumseries(
                 'FOO',
```

```
                    ORDSYS.ORDTCalendar(
                                    0,
                                    'FOO CALENDAR',
                                    4,
                                    ORDSYS.ORDTPattern(
                                        ORDSYS.ORDTPatternBits(1,1,1,1,1,1,1),
                                           TO_DATE('01/07/1996')),
                                    TO_DATE('01/01/1996'),
                                    TO_DATE('01/01/1997'),
                      ORDSYS.ORDTExceptions(),
                      ORDSYS.ORDTExceptions()
                      ),
                   ORDSYS.ordtnumtab(
                           ORDSYS.ordtnumcell(TO_DATE('12/02/1996'), 1),
                           ORDSYS.ordtnumcell(TO_DATE('12/03/1996'), 2),
                           ORDSYS.ordtnumcell(TO_DATE('12/04/1996'), 3),
                           ORDSYS.ordtnumcell(TO_DATE('12/05/1996'), 4),
                           ORDSYS.ordtnumcell(TO_DATE('12/06/1996'), 5),
                           ORDSYS.ordtnumcell(TO_DATE('12/09/1996'), 6),
                           ORDSYS.ordtnumcell(TO_DATE('12/10/1996'), 7),
                           ORDSYS.ordtnumcell(TO_DATE('12/11/1996'), 8),
                           ORDSYS.ordtnumcell(TO_DATE('12/12/1996'), 9),
                           ORDSYS.ordtnumcell(TO_DATE('12/13/1996'), 10))
                   );

dummyval INTEGER;

BEGIN

 -- Generate a timeseries by from XCORP's high (repeat forward).
 SELECT ORDSYS.TimeSeries.Display(
           ORDSYS.TimeSeries.Fill(ts, 1),
           'Fill Forward') INTO dummyval
 FROM dual;

 -- Generate a timeseries by from XCORP's high (repeat backward).
 SELECT ORDSYS.TimeSeries.Display(
           ORDSYS.TimeSeries.Fill(ts, 2),
           'Fill Backward') INTO dummyval
 FROM dual;

 -- Generate a timeseries by from XCORP's high (null fill).
 SELECT ORDSYS.TimeSeries.Display(
           ORDSYS.TimeSeries.Fill(ts, 0),
           'Null Fill') INTO dummyval
```

```
 FROM dual;

END;
/
```

This example might produce the following output:

```
Fill Forward :

Calendar Data:
Calendar Name = FOO CALENDAR
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 01/01/1997 00:00:00
 patBits:
          1111111
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
Series Data:
 ----------------------------
 Date                    Value
 12/02/1996 00:00:00      1
 12/03/1996 00:00:00      2
 12/04/1996 00:00:00      3
 12/05/1996 00:00:00      4
 12/06/1996 00:00:00      5
 12/07/1996 00:00:00      5
 12/08/1996 00:00:00      5
 12/09/1996 00:00:00      6
 12/10/1996 00:00:00      7
 12/11/1996 00:00:00      8
 12/12/1996 00:00:00      9
 12/13/1996 00:00:00      10
 ----------------------------

Fill Backward :

Calendar Data:
Calendar Name = FOO CALENDAR
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 01/01/1997 00:00:00
 patBits:
          1111111
```

```
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
Series Data:
 ----------------------------
 Date                    Value
 12/02/1996 00:00:00      1
 12/03/1996 00:00:00      2
 12/04/1996 00:00:00      3
 12/05/1996 00:00:00      4
 12/06/1996 00:00:00      5
 12/07/1996 00:00:00      6
 12/08/1996 00:00:00      6
 12/09/1996 00:00:00      6
 12/10/1996 00:00:00      7
 12/11/1996 00:00:00      8
 12/12/1996 00:00:00      9
 12/13/1996 00:00:00      10
 ----------------------------

Null Fill :

Calendar Data:
Calendar Name = FOO CALENDAR
 Frequency = 4
 MinDate = 01/01/1996 00:00:00
 MaxDate = 01/01/1997 00:00:00
 patBits:
          1111111
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
Series Data:
 ----------------------------
 Date                    Value
 12/02/1996 00:00:00      1
 12/03/1996 00:00:00      2
 12/04/1996 00:00:00      3
 12/05/1996 00:00:00      4
 12/06/1996 00:00:00      5
 12/07/1996 00:00:00
 12/08/1996 00:00:00
 12/09/1996 00:00:00      6
 12/10/1996 00:00:00      7
 12/11/1996 00:00:00      8
```

```
12/12/1996 00:00:00      9
12/13/1996 00:00:00      10
----------------------------
```

First

Format

ORDSYS.TimeSeries.First(

ts ORDSYS.ORDTNumSeriesIOTRef

) RETURN ORDSYS.ORDTNumCell;

Description

Given a time series, returns the first element in it.

Parameters

ts
The input time series.

Usage

A null is returned if the time series (*ts*) is empty.

An exception is returned if the time series (*ts*) is null.

Example

Return the first timestamp and opening price for stock ACME in the *stockdemo_ts* time series:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;

BEGIN

 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.First(ts.open), 'First Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
```

```
/
```

This example might produce the following output:

```
First Results :

   Timestamp : 11/01/1996 00:00:00
       Value : 59
```

FirstN

Format

ORDSYS.TimeSeries.FirstN(

ts ORDSYS.ORDTNumSeriesIOTRef,

NumValues NUMBER

) RETURN ORDSYS.ORDTNumSeries;

Description

Given a time series and a number of elements (*NumValues*) to return, returns the first *NumValues* elements in the time series.

Parameters

ts
The input time series.

NumValues
Number of elements from the beginning of the time series to be returned.

Usage

The function returns a time series populated with the first *NumValues* cells from the input time series (*ts*). The calendar of the output time series is the same as that of the input time series.

An exception is returned if the time series (*ts*) is null or if *NumValues* is zero (0) or negative.

Example

Return the first 10 timestamps and opening prices in the time series for stock ACME.:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;
```

```
BEGIN

 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.FirstN(ts.open, 10), 'FirstN Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
/
```

This example might produce the following output:

```
FirstN Results :

Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 11/14/1996 00:00:00      68
 ----------------------------
```

GetDatedElement

Format

```
ORDSYS.TimeSeries.GetDatedElement (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    target_date date
    ) RETURN ORDSYS.ORDTNumCell;
```

Description

Given a time series and a date, returns the time series element for that date.

Parameters

ts
The input time series.

target_date
Positive integer specifying the date of the element to be returned.

Usage

The function returns the cell from the input time series (*ts*) at the specified date (*target_date*). If there is no data in *ts* at *target_date*, the function returns a null.

An exception is returned if the time series (*ts*) is null.

Example

Return the timestamp and opening price for 26-Nov-1996 for stock ACME:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;
tstDate date;

BEGIN

 -- Get the cell for 26-NOV-1996 from ACME's open and display it
```

```
 tstDate := TO_DATE('11/26/1996');

 SELECT ORDSYS.TimeSeries.Display(
    ORDSYS.TimeSeries.GetDatedElement(ts.open, tstDate),
                                    'GetDatedElement Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
/
```

This example might produce the following output:

```
GetDatedElement Results :

   Timestamp : 11/26/1996 00:00:00
       Value : 76
```

GetNthElement

Format

```
ORDSYS.TimeSeries.GetNthElement
    (ts ORDSYS.ORDTNumSeriesIOTRef,
    target_index INTEGER
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumCell;
```

Description

Given a time series, a number (*target_index*), and optionally a date range, returns the Nth element (element whose position corresponds to *target_index*) in the specified time series, or within the date range if one is specified.

Parameters

ts
The input time series.

target_index
Positive integer specifying the position of the element to be returned.

startDate
Starting date within the time series to which *target_index* is to be applied. If *target_index* = 1, the function returns the element for *startDate*. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series to which *target_index* is to be applied. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.

- *target_index* is not an integer, or is zero (0) or a negative number.
- *endDate* is earlier than *startDate*.

Example

Return the tenth opening price for stock AONE:

```
SELECT ORDSYS.TimeSeries.ExtractValue(
         ORDSYS.TimeSeries.GetNthElement(open, 10))
  FROM ORDTDEV.stocks_ts
  WHERE ticker = 'AONE';
```

This example might produce the following output:

```
ORDSYS.TIM
----------
   15.1875
1 row selected.
```

GetSeries

Format

ORDSYS.TimeSeries.GetSeries(

ts ORDSYS.ORDTNumSeriesIOTRef

) RETURN ORDSYS.ORDTNumSeries;

or

ORDSYS.TimeSeries.GetSeries(

ts ORDSYS.ORDTVarchar2SeriesIOTRef

) RETURN ORDSYS.ORDTVarchar2Series;

Description

Given a reference to a time series of references (ORDTNumSeriesIOTRef or ORDTVarchar2SeriesIOTRef), returns a time series instance (ORDTNumSeries or ORDTVarchar2Series).

Parameters

ts
The input time series.

Usage

The function materializes the input time series.

An exception is returned if the time series (*ts*) is null.

Example

Return an instance of a specified time series (opening prices for stock ACME):

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;

BEGIN
```

```
 SELECT ORDSYS.TimeSeries.Display(
     ORDSYS.TimeSeries.GetSeries(ts.open), 'GetSeries Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
/
```

This example might produce the following output:

```
GetSeries Results :

Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 11/14/1996 00:00:00      68
 11/15/1996 00:00:00      69
 11/18/1996 00:00:00      70
 11/19/1996 00:00:00      71
 11/20/1996 00:00:00      72
 11/21/1996 00:00:00      73
 11/22/1996 00:00:00      74
 11/25/1996 00:00:00      75
```

```
11/26/1996 00:00:00      76
11/27/1996 00:00:00      77
11/29/1996 00:00:00      78
12/02/1996 00:00:00      79
12/03/1996 00:00:00      80
12/04/1996 00:00:00      81
12/05/1996 00:00:00      82
12/06/1996 00:00:00      83
12/09/1996 00:00:00      84
12/10/1996 00:00:00      85
12/11/1996 00:00:00      86
12/12/1996 00:00:00      87
12/13/1996 00:00:00      88
12/16/1996 00:00:00      89
12/17/1996 00:00:00      90
12/18/1996 00:00:00      91
12/19/1996 00:00:00      92
12/20/1996 00:00:00      93
12/23/1996 00:00:00      94
12/24/1996 00:00:00      95
12/26/1996 00:00:00      96
12/27/1996 00:00:00      97
12/30/1996 00:00:00      98
12/31/1996 00:00:00      99
----------------------------
```

IsValidTS

Format

```
ORDSYS.TimeSeries.IsValidTS(
    ts IN ORDSYS.ORDTNumSeriesIOTRef
    ) RETURN INTEGER
```

or

```
ORDSYS.TimeSeries.IsValidTS(
    ts IN ORDSYS.ORDTVarchar2SeriesIOTRef
    ) RETURN INTEGER
```

Description

Returns 1 if the time series is valid and 0 if the time series is invalid.

Parameters

ts
The input time series.

Usage

A time series is invalid if one or more of the following conditions are true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- The calendar associated with the time series is invalid.
- The timestamps are not sorted.
- One or more timestamps are null, imprecise, or outside the date range of the calendar.
- One or more timestamps are included in the time series but should be excluded based on the calendar definition (for example, a Saturday timestamp that is in a Monday-Friday calendar and that is not an on-exception).

- One or more timestamps are excluded from the time series but should be included based on the calendar definition (for example, a Wednesday date that is not a holiday in a Monday-Friday calendar and for which there is no data). Such dates can be considered as "holes" in the time series.

Contrast this function with ValidateTS, which checks whether a time series is valid, and if the time series is not valid, outputs a diagnostic message and tables with timestamps that are causing the time series to be invalid.

Example

Use the IsValidTS and ValidateTS functions and the DisplayValTS procedure with an invalid time series:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
 numTS  ORDSYS.ORDTNumSeries;
 tempVal integer;
 retIsValid  integer;
 retValTS    integer;
 loDateTab  ORDSYS.ORDTDateTab := NULL;
 hiDateTab  ORDSYS.ORDTDateTab := NULL;
 impDateTab ORDSYS.ORDTDateTab := NULL;
 dupDateTab ORDSYS.ORDTDateTab := NULL;
 extraDateTab ORDSYS.ORDTDateTab := NULL;
 missingDateTab ORDSYS.ORDTDateTab := NULL;
 outMesg varchar2(2000);

BEGIN

   --  Set the buffer size
   DBMS_OUTPUT.ENABLE(100000);

   --
   -- NOTE: Here an instance of the time series is materialized
   -- so that it could be modified to generate an invalid time series.
   --
   SELECT ORDSYS.TIMESERIES.GetSeries(ts.open) INTO numTS
   FROM ordtdev.stockdemo_ts ts
   WHERE ts.ticker = 'ACME';

   -- Example of validating a valid time series.
   SELECT ordsys.timeseries.display(numTS, 'A VALID TIME SERIES') INTO tempVal
```

```
FROM dual;
retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg, loDateTab,
                          hiDateTab, impDateTab, dupDateTab,
                          extraDateTab, missingDateTab);
DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                       impDateTab, dupDateTab, extraDateTab, missingDateTab,
                       'Testing DisplayValTS');
DBMS_OUTPUT.NEW_LINE;

-- For illustration let us first create an invalid timeseries.
--
-- Here we are adjusting the calendar's minDate and maxDate to avoid
-- getting a huge list of missing dates.
--
numTS.cal.minDate := TO_DATE('10/28/1996');
numTS.cal.maxDate := TO_DATE('01/05/1997');

-- Add Dates Before numTS.cal.minDate
numTS.series(10).tstamp := numTS.cal.minDate - 1;
numTS.series(11).tstamp := numTS.cal.minDate - 2;

-- Add Dates Beyond numTS.cal.maxDate
numTS.series(12).tstamp := numTS.cal.maxDate + 1;
numTS.series(13).tstamp := numTS.cal.maxDate + 2;

-- Add some null timestamps
numTS.series(14).tstamp := NULL;
numTS.series(15).tstamp := NULL;

-- Add some imprecise dates (some are duplicated)
numTS.series(17).tstamp := numTS.series(16).tstamp + 1/24;
numTS.series(18).tstamp := numTS.series(16).tstamp + 15/24;

-- Add some duplicate timestamps
numTS.series(19).tstamp := numTS.series(18).tstamp;
numTS.series(21).tstamp := numTS.series(20).tstamp;

-- Add some extra dates in the middle
numTS.series(37).tstamp := TO_DATE('12/28/1996');
numTS.series(36).tstamp := TO_DATE('12/29/1996');

-- Add some holes at the end
```

```
   numTS.series(numTS.series.count).tstamp := TO_DATE('01/04/1997');

   -- Example of validating an invalid time series.
   SELECT ordsys.timeseries.display(numTS, 'AN INVALID TIME SERIES')
   INTO tempVal FROM dual;
   retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
   retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg,
                        loDateTab, hiDateTab, impDateTab,
                        dupDateTab, extraDateTab, missingDateTab);
   DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
   DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
   ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                     impDateTab, dupDateTab, extraDateTab, missingDateTab,
                     'Testing DisplayValTS');
END;
/
```

This example might produce the following output:

```
A VALID TIME SERIES :

Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
```

```
 11/14/1996 00:00:00      68
 11/15/1996 00:00:00      69
 11/18/1996 00:00:00      70
 11/19/1996 00:00:00      71
 11/20/1996 00:00:00      72
 11/21/1996 00:00:00      73
 11/22/1996 00:00:00      74
 11/25/1996 00:00:00      75
 11/26/1996 00:00:00      76
 11/27/1996 00:00:00      77
 11/29/1996 00:00:00      78
 12/02/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/23/1996 00:00:00      94
 12/24/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 12/31/1996 00:00:00      99
 ----------------------------

Value returned by IsValid  = 1
Value returned by ValidateTS  = 1

DisplayValTS: Testing DisplayValTS:

TS-SUC: the input time series is a valid time series


AN INVALID TIME SERIES :

Name = ACME open NumSeries
```

```
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 10/28/1996 00:00:00
 MaxDate = 01/05/1997 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 10/27/1996 00:00:00      68
 10/26/1996 00:00:00      69
 01/06/1997 00:00:00      70
 01/07/1997 00:00:00      71
       72
       73
 11/22/1996 00:00:00      74
 11/22/1996 01:00:00      75
 11/22/1996 15:00:00      76
 11/22/1996 15:00:00      77
 11/29/1996 00:00:00      78
 11/29/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
```

```
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/29/1996 00:00:00      94
 12/28/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 01/04/1997 00:00:00      99
 ----------------------------

Value returned by IsValid  = 0
Value returned by ValidateTS  = 0

DisplayValTS: Testing DisplayValTS:

TS-WRN: the input time series has errors. See the message for details

message output by validateTS:

TS-ERR: the input time series is unsorted
TS-ERR: the time series has null timestamps
TS-ERR: the time series has timestamps < calendar minDate (refer LoDateTab)
TS-ERR: the time series has timestamps > calendar maxDate (refer HiDateTab)
TS-ERR: the time series has imprecise timestamps (refer impreciseDateTab)
TS-ERR: the time series has duplicate timestamps (refer DuplicateDateTab)

list of dates < calendar minDate - lowDateTab :

     10/26/1996 00:00:00     10/27/1996 00:00:00


list of dates > calendar maxDate - hiDateTab :
     01/06/1997 00:00:00     01/07/1997 00:00:00

list of imprecise dates - impreciseDateTab :

     11/22/1996 01:00:00     11/22/1996 15:00:00

list of duplicate dates - duplicateDateTab :

     11/22/1996 15:00:00     11/29/1996 00:00:00

ExtraDateTab :
```

```
     12/28/1996 00:00:00     12/29/1996 00:00:00     01/04/1997 00:00:00

MissingDateTab :

     10/28/1996 00:00:00     10/29/1996 00:00:00     10/30/1996 00:00:00
     10/31/1996 00:00:00     11/14/1996 00:00:00     11/15/1996 00:00:00
     11/18/1996 00:00:00     11/19/1996 00:00:00     11/20/1996 00:00:00
     11/21/1996 00:00:00     11/25/1996 00:00:00     11/26/1996 00:00:00
     11/27/1996 00:00:00     12/02/1996 00:00:00     12/23/1996 00:00:00
     12/24/1996 00:00:00     12/31/1996 00:00:00     01/01/1997 00:00:00
     01/02/1997 00:00:00   01/03/1997 00:00:00
```

Lag

Format

```
ORDSYS.TimeSeries.Lag (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    units INTEGER
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.Lag (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    lead_date DATE
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a positive or negative number (*units*) or a date (*lead_date*), and optionally a starting and ending timestamp within the time series, returns a time series that lags or (for negative numeric values) leads the input time series by the appropriate number of timestamps.

Parameters

ts
The input time series.

units
Integer specifying the number of timestamps by which the output time series is to be adjusted. If *units* is positive, each element in the output time series is the same as the element in the input time series for that relative position minus the *units*. If

units is negative, each element in the output time series is the same as the element in the input time series for that relative position plus the *units*.

lead_date
The date relative to the starting date reflecting the number of timestamps by which the output time series is to be adjusted. The function calculates the number of timestamps between *lead_date* and *startDate*, and then uses that number as if it were a *units* parameter value. (If *lead_date* is later than *startDate*, the effective units value is positive; if *lead_date* is before the starting date, the effective units value is negative.)

startDate
Starting date to be used in calculating the lead or lag value; also the starting date in the input time series for which the output time series is to be created. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date in the input time series for which the output time series is to be created. If *endDate* is specified, *startDate* must also be specified.

Usage

The function creates a time series whose elements reflect an input time series adjusted by a number of timestamps. For example, using the United States stock trading calendar for 1997, if the first timestamp in the input time series is 06-Jan-1997 (Monday) and the units value is 2, the first timestamp in the output time series is 02-Jan-1997 (Thursday) and its associated value (such as closing price) is the same as that for 06-Jan-1997 in the input time series. Subsequent elements of the output time series reflect the timestamp adjustment.

For example, assuming the United States stock trading calendar for 1997, Table 5–1 shows some time series data with a two-day lag period.

Table 5–1 Lagging a Time Series by Two Days

Input Time Series:		**Output Time Series:**	
Timestamp	**Closing Price**	**Timestamp**	**Closing Price**
06-Jan-1997	49.50	02-Jan-1997	49.50
07-Jan-1997	49.25	03-Jan-1997	49.25
08-Jan-1997	50.00	06-Jan-1997	50.00
...	...	...	...

For convenience, both the Lead and Lag functions are provided.The functions operate identically, except that they interpret the sign of the *units* value in opposite ways. For example, Lead with -10 for *units* is equivalent to Lag with 10 for *units*. Moreover, because of the way the *lead_date* parameter is interpreted, Lead and Lag with a *lead_date* operate identically.

Example

Return a time series starting with 03-Mar-1997 using closing prices from the time series from 01-Nov-1996 through 30-Nov-1996 for stock ACME. The returned time series has the same number of timestamps as are in the specified date range (*startDate* through *endDate*).

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
          ORDSYS.TimeSeries.Lag(ts.close,
               to_date('03-MAR-97','DD-MON-YY'),
               to_date('01-NOV-96','DD-MON-YY'),
               to_date('30-NOV-96','DD-MON-YY'))
           ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
03-MAR-97         59
04-MAR-97         60
05-MAR-97         61
06-MAR-97         62
07-MAR-97         63
10-MAR-97         64
  ...            ...
27-MAR-97         77
28-MAR-97         78
20 rows selected.
```

Last

Format

```
ORDSYS.TimeSeries.Last(
    ts ORDSYS.ORDTNumSeriesIOTRef
    ) RETURN ORDSYS.ORDTNumCell;
```

Description

Given a time series, returns the last element in it.

Parameters

ts
The input time series.

Usage

A null is returned if the time series (*ts*) is empty.

An exception is returned if the time series (*ts*) is null.

Example

Return the last timestamp and opening price for stock ACME in the *stockdemo_ts* time series:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;

BEGIN

 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.Last(ts.open), 'Last Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
```

```
/
```

This example might produce the following output:

```
Last Results :

   Timestamp : 12/31/1996 00:00:00
       Value : 99
```

LastN

Format

```
ORDSYS.TimeSeries.LastN(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    NumValues NUMBER
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series and a number of elements (*NumValues*) to return, returns the last *NumValues* elements in the time series.

Parameters

ts
The input time series.

NumValues
Number of elements from the end of the time series to be returned.

Usage

The function returns a time series populated with the last *NumValues* cells from the input time series (*ts*). The calendar of the output time series is the same as that of the input time series.

An exception is returned if the time series (*ts*) is null or if *NumValues* is zero (0) or negative.

Example

Return the last 10 timestamps and opening prices in the time series for stock ACME.:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
dummyval INTEGER;
```

```
BEGIN

 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.LastN(ts.open, 10), 'LastN Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts
 WHERE ts.ticker='ACME';

END;
/
```

This example might produce the following output:

```
LastN Results :

Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/23/1996 00:00:00      94
 12/24/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 12/31/1996 00:00:00      99
 ----------------------------
```

Lead

Format

```
ORDSYS.TimeSeries.Lead (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    units INTEGER
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.Lead (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    lead_date DATE
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a positive or negative number (*units*) or a date (*lead_date*), and optionally a starting and ending timestamp within the time series, returns a time series that leads or (for negative numeric values) lags the input time series by the appropriate number of timestamps.

Parameters

ts
The input time series.

units
Integer specifying the number of timestamps by which the output time series is to be adjusted. If *units* is positive, each element in the output time series is the same as the element in the input time series for that relative position plus the *units*. If *units*

is negative, each element in the output time series is the same as the element in the input time series for that relative position minus the *units*.

lead_date
The date relative to the starting date reflecting the number of timestamps by which the output time series is to be adjusted. The function calculates the number of timestamps between *lead_date* and *startDate*, and then uses that number as if it were a *units* parameter value. (If *lead_date* is later than *startDate*, the effective *units* value is positive; if *lead_date* is before *startDate*, the effective *units* value is negative.)

startDate
Starting date to be used in calculating the lead or lag value; also the starting date in the input time series for which the output time series is to be created. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date in the input time series for which the output time series is to be created. If *endDate* is specified, *startDate* must also be specified.

Usage

The function creates a time series whose elements reflect an input time series adjusted by a number of timestamps. For example, using the United States stock trading calendar for 1997, if the first timestamp in the input time series is 02-Jan-1997 (Thursday) and the units value is 2, the first timestamp in the output time series is 06-Jan-1997 (Monday) and its associated value (such as closing price) is the same as that for 02-Jan-1997 in the input time series. Subsequent elements of the output time series reflect the timestamp adjustment.

For example, assuming the United States stock trading calendar for 1997, Table 5–2 shows some time series data with a two-day lead period:

Table 5–2 Leading a Time Series by Two Days

Input Time Series:		**Output Time Series:**	
Timestamp	**Closing Price**	**Timestamp**	**Closing Price**
02-Jan-1997	49.00	06-Jan-1997	49.00
03-Jan-1997	50.00	07-Jan-1997	50.00
06-Jan-1997	49.50	08-Jan-1997	49.50
...	...	...	...

For convenience, both the Lead and Lag functions are provided. The functions operate identically, except that they interpret the sign of the *units* value in opposite ways. For example, Lead with -10 for *units* is equivalent to Lag with 10 for *units*. Moreover, because of the way the *lead_date* parameter is interpreted, Lead and Lag with a *lead_date* operate identically.

Example

Return a time series starting with 03-Mar-1997 using closing prices from the time series from 01-Nov-1996 through 30-Nov-1996 for stock ACME. The returned time series has the same number of timestamps as are in the specified date range (*startDate* through *endDate*).

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
          ORDSYS.TimeSeries.Lead(ts.close,
               to_date('03-MAR-97','DD-MON-YY'),
               to_date('01-NOV-96','DD-MON-YY'),
               to_date('30-NOV-96','DD-MON-YY'))
           ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
03-MAR-97         59
04-MAR-97         60
05-MAR-97         61
06-MAR-97         62
07-MAR-97         63
10-MAR-97         64
  ...            ...
27-MAR-97         77
28-MAR-97         78
20 rows selected.
```

Mavg

Format

```
ORDSYS.TimeSeries.Mavg(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    [startDate DATE, endDate DATE,]
    k INTEGER
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given an input ORDTNumSeries, returns a moving average for the time series, or for the date range if one is specified. Each value in the returned time series is the average of the value for the current timestamp plus the value for each of the previous specified number of timestamps minus one.

For example, a 30-day moving average of closing prices for a stock on any given date is the average of that day's closing price and the 29 preceding closing prices.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which to return moving averages. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which to return moving averages. If *endDate* is specified, *startDate* must also be specified.

k
Positive integer specifying the look-back window (number of timestamps, including the current one, over which to compute the moving average).

Usage

The returned time series has nulls for any entry where there are not *k-1* timestamps preceding it in the calendar. For example, if a stock trading calendar for 1997 starts on 02-Jan-1997, the series of 5-day moving averages of the closing price for a stock for the year has nulls for the closing price for the first four timestamps (02-Jan, 03-Jan, 06-Jan, and 07-Jan), because there are insufficient timestamps for computing the average.

Any nulls in the entries for the *k* timestamps are ignored, as explained in Section 2.8.2.1.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return a table of 10-day moving average values of the closing price for stock ACME for the month of December 1996:

```
SELECT * FROM the
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
             ORDSYS.TimeSeries.Mavg(ts.close,to_date('02-DEC-96','DD-MON-YY'),
                  to_date('31-DEC-96','DD-MON-YY'),10)
              ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stockdemo_ts ts
       WHERE ts.ticker='ACME');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
02-DEC-96       74.5
03-DEC-96       75.5
04-DEC-96       76.5
05-DEC-96       77.5
06-DEC-96       78.5
09-DEC-96       79.5
10-DEC-96       80.5
11-DEC-96       81.5
12-DEC-96       82.5
13-DEC-96       83.5
```

```
16-DEC-96       84.5
17-DEC-96       85.5
18-DEC-96       86.5
19-DEC-96       87.5
20-DEC-96       88.5
23-DEC-96       89.5
24-DEC-96       90.5
26-DEC-96       91.5
27-DEC-96       92.5
30-DEC-96       93.5
31-DEC-96       94.5
21 rows selected.
SVRMGR>
```

Msum

Format

```
ORDSYS.TimeSeries.Msum(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    [startDate DATE, endDate DATE,]
    k INTEGER
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given an input ORDTNumSeries, returns a moving sum for the time series, or for the date range if one is specified. Each value in the returned time series is the sum of the value for the current timestamp plus the value for each of the previous specified number of timestamps minus one.

For example, a 30-day moving sum for a stock's daily trading volume on any given date is the sum of that day's volume and the 29 preceding daily volumes.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which to return moving sums. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which to return moving sums. If *endDate* is specified, *startDate* must also be specified.

k
Positive integer specifying the look-back window (number of timestamps, including the current one, over which to compute the moving sum).

Usage

The returned time series has nulls for any entry where there are not *k-1* timestamps preceding it in the calendar. For example, if a stock trading calendar for 1997 starts on 02-Jan-1997, the series of 5-day moving sums of the trading volume for a stock for the year has nulls for the volume for the first four timestamps (02-Jan, 03-Jan, 06-Jan, and 07-Jan), because there are insufficient timestamps for computing the sum.

Any nulls in the entries for the *k* timestamps are ignored, as explained in Section 2.8.2.1.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return a table of 30-day moving sum values of trading volume for stock AONE for 1996:

```
SELECT * FROM THE(
  SELECT CAST(ORDTS.Extract(ORDTS.MSUM(volume,
     to_date('01-01-96','MM-DD-YY'),
     to_date('12-31-96','MM-DD-YY'),
     30)) AS ORDTNumTab)
  FROM StockTabView
  WHERE ticker = 'AONE');
```

Scaleup

Format

```
ORDSYS.TimeSeries.Scaleup(
    inDate DATE,
    calendar ORDSYS.ORDTCalendar
    ) RETURN DATE;
```

Description

Given an input ORDTCalendar and a date, returns a scaled date.

Parameters

inDate
The date to be used for scaling.

calendar
The calendar to be used for scaling the date.

Usage

For an explanation of concepts related to time scaling, see Section 2.9.

This function is used in a SQL GROUP BY clause for scaling of dates.

An exception is returned if *inDate* or *calendar* is null.

Example

For all tickers accessible through the *stockdemo_sv* view (ACME, FUNCO, SAMCO, and XCORP), scale daily data to monthly summary data for the summed volume and average closing price.

```
--
-- Scaleup - Group By interface
-- For all tickers in stockdemo, scale daily data to monthly
-- summary data, reporting summed volumes and average closes.
--
SELECT ticker, ORDSYS.TimeSeries.Scaleup(sv.tstamp,
value(cal)),sum(volume),avg(close)
```

```
  FROM ORDTDEV.stockdemo_sv sv, ordtdev.stockdemo_calendars cal
  WHERE cal.name = 'MONTHLY'
  GROUP BY ticker,ORDSYS.TimeSeries.Scaleup(sv.tstamp, value(cal));
```

This example might produce the following output:

```
TICKE ORDSYS.OR SUM(VOLUME AVG(CLOSE)
----- --------- ---------- ----------
ACME  01-NOV-96      20000       68.5
ACME  01-DEC-96      21000         89
FUNCO 01-NOV-96      20000     23.823
FUNCO 01-DEC-96      21000 23.8257143
SAMCO 01-NOV-96   10207000   39.83125
SAMCO 01-DEC-96    3719450 38.2738095
XCORP 01-OCT-96   10270250 79.1458333
XCORP 01-NOV-96  100243350 84.6973684
XCORP 01-DEC-96  141838350 91.9572368
9 rows selected.
```

ScaleupAvg

Format

```
ORDSYS.TimeSeries.ScaleupAvg(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the average value of each scaled group of non-null values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

Nulls are ignored in computing the average for each group of values.

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the average closing prices for stock SAMCO for each month for the entire time series:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupAvg(
                                ts.close,
                                sc.calendar
                                                )
             ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-JAN-90 29.7074045
01-FEB-90 29.0477211
01-MAR-90 30.7003091
  .
  .
  .
01-OCT-96 42.7717391
01-NOV-96   39.83125
01-DEC-96 38.2738095
84 rows selected.
```

ScaleupCount

Format

```
ORDSYS.TimeSeries.ScaleupCount(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the count of non-null timestamps in each scaled group.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

Nulls are ignored in computing the count for each group of values.

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the quarterly count of daily closing prices for stock SAMCO for the period 01-June-1996 through 31-December 1996:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
            ORDSYS.TimeSeries.ScaleupCount(
                  ts.close,
                  sc.calendar,
                  to_date('01-JUL-1996','DD-MON-YYYY'),
                  to_date('31-DEC-1996','DD-MON-YYYY')
                  )
             ) AS ORDSYS.ORDTNumTab)
      FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
      WHERE ts.ticker='SAMCO' and
            sc.name  ='QUARTERLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-JUL-96         64
01-OCT-96         64
2 rows selected.
```

ScaleupFirst

Format

```
ORDSYS.TimeSeries.ScaleupFirst(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the first non-null value of each scaled group of values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the first closing prices for stock SAMCO for the months of October, November, and December of 1996:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupFirst(
                       ts.close,
                       sc.calendar,
                       to_date('01-OCT-1996','DD-MON-YYYY'),
                       to_date('01-JAN-1997','DD-MON-YYYY')
                       )
              ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-OCT-96      42.75
01-NOV-96     41.875
01-DEC-96     38.125
3 rows selected.
```

ScaleupLast

Format

```
ORDSYS.TimeSeries.ScaleupLast(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the last non-null value of each scaled group of values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the last closing prices for stock SAMCO for the months of October, November, and December of 1996:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupLast(
                       ts.close,
                       sc.calendar,
                       to_date('01-OCT-1996','DD-MON-YYYY'),
                       to_date('01-JAN-1997','DD-MON-YYYY')
                       )
              ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-OCT-96     42.375
01-NOV-96      38.25
01-DEC-96      39.75
3 rows selected.
```

Note that each timestamp reflects the first date of the month in the calendar (following the rules explained in Section 2.2.1), and each value in this case reflects the closing price on the last date for that month in the calendar.

ScaleupMax

Format

```
ORDSYS.TimeSeries.ScaleupMax(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the maximum value of each scaled group of values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the highest (maximum) closing prices for stock SAMCO for each month in the entire time series:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupMax(
                                ts.close,
                                sc.calendar
                                                )
             ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-JAN-90    31.2813
01-FEB-90    29.7813
01-MAR-90    31.1875
01-APR-90    31.5938
01-MAY-90     32.875
01-JUN-90    33.7813
01-JUL-90    34.6875
01-AUG-90     31.875
  ...         ...
01-OCT-96     43.375
01-NOV-96      43.75
01-DEC-96      39.75
84 rows selected.
```

ScaleupMin

Format

```
ORDSYS.TimeSeries.ScaleupMin(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the minimum value of each scaled group of values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.

- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the lowest (minimum) closing prices for stock SAMCO for each month in the entire time series:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupMin(
                                ts.close,
                                sc.calendar
                                                )
             ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-JAN-90    27.6875
01-FEB-90    28.2813
01-MAR-90    30.0938
01-APR-90    30.1875
01-MAY-90    30.7813
01-JUN-90    32.0938
01-JUL-90    32.2813
01-AUG-90    28.5938
  ...         ...
01-OCT-96         42
01-NOV-96     37.375
01-DEC-96     37.875
84 rows selected.
```

ScaleupSum

Format

```
ORDSYS.TimeSeries.ScaleupSum(
    ts ORDSYS.ORDTNumSeriesIOTRef,
    calendar ORDSYS.ORDTCalendar
    [, startDate DATE
    , endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given a time series, a calendar to be used for scaling, and optionally starting and ending dates, returns a time series reflecting the sum of each scaled group of values.

Parameters

ts
The input time series.

calendar
The calendar to be used for the scaling.

startDate
The starting date to be used. If *startDate* is specified, *endDate* must also be specified.

endDate
The ending date to be used. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned for any of the following conditions:

- The input time series, the calendar on which the input time series in based, or the specified calendar is null.
- The frequency of the calendar on which the time series is based is greater than the frequency of the specified calendar (for example, the time series calendar's frequency is *month* and the specified calendar's frequency is *day*).

For an explanation of concepts related to time scaling, see Section 2.9.

Example

Return the sum of the daily trade volume for stock SAMCO for each month in the entire time series:

```
SELECT * FROM THE
  (SELECT CAST(ORDSYS.TimeSeries.ExtractTable(
                ORDSYS.TimeSeries.ScaleupSum(
                                ts.volume,
                                sc.calendar
                                                )
             ) AS ORDSYS.ORDTNumTab)
       FROM ORDTDEV.stocks_ts ts, ordtdev.scale sc
       WHERE ts.ticker='SAMCO' and
             sc.name  ='MONTHLY');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
01-JAN-90    3117750
01-FEB-90    2036500
01-MAR-90    1424375
01-APR-90     981500
01-MAY-90    1348875
01-JUN-90    1395875
01-JUL-90    1088125
01-AUG-90    1503000
  ...          ...
01-OCT-96    1615350
01-NOV-96   10207000
01-DEC-96    3719450
84 rows selected.
```

TrimSeries

Format

```
ORDSYS.TimeSeries.TrimSeries(ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.TrimSeries(ts ORDSYS.ORDTVarchar2SeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTVarchar2Series;
```

Description

Given an input ORDT series, returns an ORDT series of the same type with all data outside of the given date range removed. The calendar of the returned series will be the same as that of the original series.

Parameters

ts
The input time series.

startDate
Starting date within the time series. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.

- *endDate* is earlier than *startDate*.

Example

Return the opening prices for stock AONE for dates in the calendar from 01-Dec-1996 through 31-Dec-1996:

```
SET SERVEROUTPUT ON
DECLARE
 tmp  INTEGER;
 tstDate1  DATE;
 tstDate2  DATE;
BEGIN
-- Set tstDate values
    tstDate1 := TO_DATE('12/01/1996 00:00:00','MM/DD/YYYY HH24:MI:SS');
    tstDate2 := TO_DATE('12/31/1996 00:00:00','MM/DD/YYYY HH24:MI:SS');
    SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.TrimSeries(open, tstDate1, tstDate2))
      INTO tmp
    FROM ORDTDEV.stocks_ts
    WHERE ticker = 'AONE';
END;
/
```

This statement might produce the following output:

```
Calendar Data:
Calendar Name = AONE
 Frequency = 4
 MinDate = 01-JAN-80
 MaxDate = 01-JAN-01
 patBits:
          0111110
 patAnchor = 06-APR-97
 onExceptions  :
 offExceptions :
     19-FEB-90     13-APR-90     28-MAY-90
     04-JUL-90     03-SEP-90     22-NOV-90
     25-DEC-90     01-JAN-91     18-FEB-91
     29-MAR-91     27-MAY-91     04-JUL-91
     02-SEP-91     28-NOV-91     25-DEC-91
     01-JAN-92     17-FEB-92     17-APR-92
     25-MAY-92     03-JUL-92     07-SEP-92
     26-NOV-92     25-DEC-92     01-JAN-93
     15-FEB-93     09-APR-93     31-MAY-93
     05-JUL-93     06-SEP-93     25-NOV-93
```

```
     24-DEC-93     21-FEB-94     01-APR-94
     27-APR-94     30-MAY-94     04-JUL-94
     05-SEP-94     24-NOV-94     26-DEC-94
     02-JAN-95     20-FEB-95     14-APR-95
     29-MAY-95     04-JUL-95     04-SEP-95
     23-NOV-95     25-DEC-95     01-JAN-96
     19-FEB-96     05-APR-96     27-MAY-96
     04-JUL-96     02-SEP-96     17-OCT-96
     28-NOV-96     25-DEC-96     27-DEC-96
Series Data:
 ----------------------------
 Date                    Value
 02-DEC-96      59.875
 03-DEC-96      60.875
 04-DEC-96      60.625
 05-DEC-96      57.75
 06-DEC-96      56.5
 09-DEC-96      57
 10-DEC-96      60.875
 11-DEC-96      59.625
 12-DEC-96      59.75
 13-DEC-96      54.875
 16-DEC-96      55.625
 17-DEC-96      53.25
 18-DEC-96      54.375
 19-DEC-96      53.875
 20-DEC-96      53.375
 23-DEC-96      54.375
 24-DEC-96      53.5
 26-DEC-96      54.375
 30-DEC-96      54.125
 31-DEC-96      52.875
 ----------------------------
```

TSAdd

Format

```
ORDSYS.TimeSeries.TSAdd (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    ts2 ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.TSAdd (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    k NUMBER
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given two input time series or a time series and a constant, and optionally starting and ending dates, returns a time series that reflects the addition of the first two parameters.

Parameters

ts1
The time series (or first time series) whose elements are to be added either to corresponding elements in the second time series or to a constant.

ts2
The time series whose elements are to be added to corresponding elements in the first time series.

k
A constant to be added to corresponding elements in the first time series.

startDate
Starting date within the time series for which the addition is to be performed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the addition is to be performed. If *endDate* is specified, *startDate* must also be specified.

Usage

The function performs a pairwise addition operation on each element of the time series. This operation determines the value of each element of the returned time series. For example:

- If two time series contain daily trade volumes for two stocks, each element of the returned time series contains the sum of the trade volumes for the two stocks for that day.
- If a time series (*ts1*) contains closing prices for a stock and if a constant (*k*) of 1 is specified, each element of the returned time series contains the closing price of *ts1* incremented by 1.

If *ts1* and *ts2* are specified, the function returns a time series whose calendar is the result of using the CombineCals function on the calendars associated with these two time series.

An exception is returned if any of the following conditions is true:

- An input time series is null.
- An input time series does not have an associated calendar.
- The calendars associated with *ts1* and *ts2* do not have the same frequency and aligned pattern.
- *endDate* is earlier than *startDate*.

Example

Add the high price for stock ACME and the low price for stock FUNCO for each trading day from 14-Nov-1996 through 14-Dec-1996:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
```

```
tstCal ORDSYS.ORDTCalendar;
startDate date;
endDate   date;
dummyval INTEGER;

BEGIN

 startDate := TO_DATE('11/14/1996');
 endDate   := TO_DATE('12/14/1996');
 SELECT ORDSYS.TimeSeries.Display(
           ORDSYS.TimeSeries.TSAdd(ts1.high, ts2.low, startDate, endDate),
           'TSAdd Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts1, ORDTDEV.stockdemo_ts ts2
 WHERE ts1.ticker='ACME' and ts2.ticker='FUNCO';

END;
/
```

This example might produce the following output:

```
TSAdd Results :

Calendar Data:
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/14/1996 00:00:00      92.87
 11/15/1996 00:00:00      93.84
 11/18/1996 00:00:00      94.87
 11/19/1996 00:00:00      95.85
 11/20/1996 00:00:00      96.82
 11/21/1996 00:00:00      97.84
 11/22/1996 00:00:00      98.85
 11/25/1996 00:00:00      99.81
 11/26/1996 00:00:00      100.78
 11/27/1996 00:00:00      101.71
```

```
11/29/1996 00:00:00      102.75
12/02/1996 00:00:00      103.88
12/03/1996 00:00:00      105.03
12/04/1996 00:00:00      106.02
12/05/1996 00:00:00      107.13
12/06/1996 00:00:00      107.75
12/09/1996 00:00:00      108.77
12/10/1996 00:00:00      109.8
12/11/1996 00:00:00      110.5
12/12/1996 00:00:00      111.41
12/13/1996 00:00:00      112.4
----------------------------
```

TSAvg

Format

```
ORDSYS.TimeSeries.TSAvg (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the average of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the average is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the average is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the average, variance, and standard deviation of the closing price of stock ACME:

```
--
-- Compute various aggregate statistics.
--
SELECT ORDSYS.TimeSeries.TSAvg(close), ORDSYS.TimeSeries.TSVariance(close),
ORDSYS.TimeSeries.TSStdDev(close)
  FROM ORDTDEV.stockdemo_ts
  WHERE ticker='ACME';
```

This example might produce the following output:

```
ORDSYS.ORD ORDSYS.ORD ORDSYS.ORD
---------- ---------- ----------
        79      143.5 11.9791486
```

1 row selected.

TSCount

Format

```
ORDSYS.TimeSeries.TSCount (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the count of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the count is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the count is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

Nulls are ignored in computing the count.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the total number of daily closing prices for stock AONE for the month of January 1990:

```
SELECT ORDSYS.TimeSeries.TSCount(close,
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSCount
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSCOUNT
----------
        22
1 row selected.
```

TSDivide

Format

```
ORDSYS.TimeSeries.TSDivide (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    ts2 ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.TSDivide (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    k NUMBER
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given two input time series or a time series and a constant, and optionally starting and ending dates, returns a time series that reflects the division of the first parameter by the second parameter.

Parameters

ts1
The time series (or first time series) whose elements are to be divided by either the corresponding elements in the second time series or a constant.

ts2
The time series whose elements are to be divided into corresponding elements in the first time series.

k
A constant to be divided into corresponding elements in the first time series.

startDate
Starting date within the time series for which the division is to be performed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the division is to be performed. If *endDate* is specified, *startDate* must also be specified.

Usage

The function performs a pairwise division operation on each element of the time series (or first time series) by the corresponding element in the second time series or by a constant. This operation determines the value of each element of the returned time series. For example:

- If two time series contain daily trade volumes for two stocks, each element of the returned time series contains the result of dividing the volume in the first time series by the volume in the second time series for that day.
- If a time series (*ts1*) contains closing prices for a stock and if a constant (*k*) of 2 is specified, each element of the returned time series contains the closing price of *ts1* divided by 2.

If *ts1* and *ts2* are specified, the function returns a time series whose calendar is the result of using the CombineCals function on the calendars associated with these two time series.

An exception is returned if any of the following conditions is true:

- An input time series is null.
- An input time series does not have an associated calendar.
- The calendars associated with *ts1* and *ts2* do not have the same frequency and aligned pattern.
- *endDate* is earlier than *startDate*.

Example

Divide the high price for stock ACME by the low price for stock FUNCO for each trading day from 14-Nov-1996 through 14-Dec-1996:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
```

```
DECLARE
tstCal ORDSYS.ORDTCalendar;
startDate date;
endDate   date;
dummyval INTEGER;

BEGIN

 startDate := TO_DATE('11/14/1996');
 endDate   := TO_DATE('12/14/1996');
 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.TSDivide(ts1.high, ts2.low, startDate, endDate),
     'TSDivide Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts1, ORDTDEV.stockdemo_ts ts2
 WHERE ts1.ticker='ACME' and ts2.ticker='FUNCO';

END;
/
```

This example might produce the following output:

```
TSDivide Results :

Calendar Data:
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/14/1996 00:00:00      2.8906577293674067867616254713028906577 3
 11/15/1996 00:00:00      2.9362416107382550335570469798657718120 8
 11/18/1996 00:00:00      2.9744449099287808965228320067029744449 1
 11/19/1996 00:00:00      3.0188679245283018867924528301886792452 8
 11/20/1996 00:00:00      3.0646515533165407220822837951301427372
 11/21/1996 00:00:00      3.1040268456375838926174496644295302013 4
 11/22/1996 00:00:00      3.1446540880503144654088050314465408805
 11/25/1996 00:00:00      3.1919361612767744645107097858042839143 2
 11/26/1996 00:00:00      3.2380151387720773759461732548359966358 3
```

```
11/27/1996 00:00:00      3.289751159848165331083930830873049346 27
11/29/1996 00:00:00      3.326315789473684210526315789473684210 53
12/02/1996 00:00:00      3.350083752093802345058626465661641541 04
12/03/1996 00:00:00      3.370786516853932584269662921348314606 74
12/04/1996 00:00:00      3.413821815154038301415487094088259783 51
12/05/1996 00:00:00      3.439701616245337753833402403646912556 98
12/06/1996 00:00:00      3.536842105263157894736842105263157894 74
12/09/1996 00:00:00      3.575936053849389987379049221708035338 66
12/10/1996 00:00:00      3.613445378151260504201680672268907563 03
12/11/1996 00:00:00      3.702127659574468085106382978723404255 32
12/12/1996 00:00:00      3.759077317385732592909013242204186245 19
12/13/1996 00:00:00      3.803418803418803418803418803418803418 8
---------------------------
```

TSMax

Format

```
ORDSYS.TimeSeries.TSMax (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the highest (maximum) of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the maximum is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the maximum is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the highest closing price for stock AONE for the month of January 1990:

```
SELECT ORDSYS.TimeSeries.TSMax(close,
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSMax
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSMAX
----------
   16.3914
1 row selected.
```

TSMaxN

Format

```
ORDSYS.TimeSeries.TSMaxN (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    NumValues INTEGER,
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumTab;
```

Description

Given an input ORDTNumSeries, a number of values to return, and optionally starting and ending dates, returns an ORDTNumTab with the specified number (*NumValues*) of the top (highest) values.

Parameters

ts
The input time series.

NumValues
Number of values to return.

startDate
Starting date within the time series for which the top values are to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the top values are to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.

- *endDate* is earlier than *startDate*.
- *NumValues* is zero (0) or negative.

Example

Return the 10 highest closing prices for stock AONE for the month of January 1996:

```
SELECT * FROM THE( SELECT CAST(
                  ORDSYS.TimeSeries.TSMaxN(close, 10,
                        to_date('01011996','MMDDYYYY'),
                        to_date('01311996','MMDDYYYY'))
                  as ORDSYS.ORDTNumTab)
             FROM ORDTDEV.stocks_ts
             WHERE ticker ='AONE');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
24-JAN-96    43.9138
25-JAN-96    42.9925
31-JAN-96    42.9925
26-JAN-96    42.7413
30-JAN-96    42.7413
29-JAN-96    42.5738
23-JAN-96    41.9875
22-JAN-96      41.82
19-JAN-96     41.485
18-JAN-96     40.815
10 rows selected.
```

TSMedian

Format

```
ORDSYS.TimeSeries.TSMedian (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the median of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the median is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the median is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the median closing price for stock AONE for the month of January 1990:

```
SELECT ORDSYS.TimeSeries.TSMedian(close,
```

```
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSMedian
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSMEDIAN
----------
   15.4649
1 row selected.
```

TSMin

Format

```
ORDSYS.TimeSeries.TSMin (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the lowest (minimum) of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the minimum is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the minimum is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the lowest closing price for stock AONE for the month of January 1990:

```
SELECT ORDSYS.TimeSeries.TSMin(close,
```

```
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSMin
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSMIN
----------
   15.1038
1 row selected.
```

TSMinN

Format

```
ORDSYS.TimeSeries.TSMinN (
    ts ORDSYS.ORDTNumSeriesIOTRef,
    NumValues INTEGER,
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumTab;
```

Description

Given an input ORDTNumSeries, a number of values to return, and optionally starting and ending dates, returns an ORDTNumTab with the specified number (*NumValues*) of the bottom (lowest) values.

Parameters

ts
The input time series.

NumValues
Number of values to return.

startDate
Starting date within the time series for which the bottom values are to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the bottom values are to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.

- *endDate* is earlier than *startDate*.
- *NumValues* is zero (0) or negative.

Example

Return the 10 lowest closing prices for stock AONE for the month of January 1996:

```
SELECT * FROM THE( SELECT CAST(
                  ORDSYS.TimeSeries.TSMinN(close, 10,
                        to_date('01011996','MMDDYYYY'),
                        to_date('01311996','MMDDYYYY'))
                  as ORDSYS.ORDTNumTab)
             FROM ORDTDEV.stocks_ts
             WHERE ticker ='AONE');
```

This example might produce the following output:

```
TSTAMP    VALUE
--------- ----------
15-JAN-96       37.8
09-JAN-96    37.9675
04-JAN-96    38.3025
10-JAN-96      38.47
03-JAN-96    38.6375
16-JAN-96    38.9725
11-JAN-96    39.0563
08-JAN-96    39.3075
12-JAN-96    39.5588
17-JAN-96    39.6425
10 rows selected.
```

TSMultiply

Format

```
ORDSYS.TimeSeries.TSMultiply (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    ts2 ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.TSMultiply (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    k NUMBER
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given two input time series or a time series and a constant, and optionally starting and ending dates, returns a time series that reflects the multiplication of the first parameter by the second parameter.

Parameters

ts1
The time series (or first time series) whose elements are to be multiplied by either the corresponding elements in the second time series or a constant.

ts2
The time series whose elements are to be multiplied by corresponding elements in the first time series.

k
A constant to be multiplied by corresponding elements in the first time series.

startDate
Starting date within the time series for which the multiplication is to be performed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the multiplication is to be performed. If *endDate* is specified, *startDate* must also be specified.

Usage

The function performs a pairwise multiplication operation on each element of the time series (or first time series) by the corresponding element in the second time series or by a constant. This operation determines the value of each element of the returned time series. For example:

- If two time series contain daily trade volumes for two stocks, each element of the returned time series contains the result of multiplying the volume in the first time series by the volume in the second time series for that day.
- If a time series (*ts1*) contains closing prices for a stock and if a constant (*k*) of 2 is specified, each element of the returned time series contains the closing price of *ts1* multiplied by 2.

If *ts1* and *ts2* are specified, the function returns a time series whose calendar is the result of using the CombineCals function on the calendars associated with these two time series.

An exception is returned if any of the following conditions is true:

- An input time series is null.
- An input time series does not have an associated calendar.
- The calendars associated with *ts1* and *ts2* do not have the same frequency and aligned pattern.
- *endDate* is earlier than *startDate*.

Example

Multiply the high price for stock ACME by the low price for stock FUNCO for each trading day from 14-Nov-1996 through 14-Dec-1996:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
```

```
DECLARE
tstCal ORDSYS.ORDTCalendar;
startDate date;
endDate   date;
dummyval INTEGER;

BEGIN

 startDate := TO_DATE('11/14/1996');
 endDate   := TO_DATE('12/14/1996');
 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.TSMultiply(ts1.high, ts2.low, startDate, endDate),
     'TSMultiply Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts1, ORDTDEV.stockdemo_ts ts2
 WHERE ts1.ticker='ACME' and ts2.ticker='FUNCO';

END;
/
```

This example might produce the following output:

```
TSMultiply Results :

Calendar Data:
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/14/1996 00:00:00      1647.03
 11/15/1996 00:00:00      1668.8
 11/18/1996 00:00:00      1694.77
 11/19/1996 00:00:00      1717.2
 11/20/1996 00:00:00      1738.86
 11/21/1996 00:00:00      1764.16
 11/22/1996 00:00:00      1788.75
 11/25/1996 00:00:00      1809.56
 11/26/1996 00:00:00      1831.06
```

```
11/27/1996 00:00:00      1849.38
11/29/1996 00:00:00      1876.25
12/02/1996 00:00:00      1910.4
12/03/1996 00:00:00      1946.43
12/04/1996 00:00:00      1969.64
12/05/1996 00:00:00      2002.79
12/06/1996 00:00:00      1995
12/09/1996 00:00:00      2020.45
12/10/1996 00:00:00      2046.8
12/11/1996 00:00:00      2044.5
12/12/1996 00:00:00      2060.08
12/13/1996 00:00:00      2082.6
---------------------------
```

TSProd

Format

```
ORDSYS.TimeSeries.TSProd (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the product (result of multiplication) of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the product is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the product is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the product resulting from multiplying the daily closing prices for stock AONE for the month of January 1990. (This example is not very plausible, but is presented merely to illustrate the syntax.)

```
SELECT ORDSYS.TimeSeries.TSProd(close,
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSProd
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSPROD
----------
1.7126E+26
1 row selected.
```

TSStdDev

Format

```
ORDSYS.TimeSeries.TSStdDev (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the standard deviation of all non-null time series entries. (This function returns a value that is the square root of the value returned by the TSVar function.)

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the standard deviation is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the standard deviation is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

If the date range refers to a time series with fewer than two timestamps, a null is returned.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the average, variance, and standard deviation of the closing price of stock ACME:

```
--
-- Compute various aggregate statistics.
--
SELECT ORDSYS.TimeSeries.TSAvg(close), ORDSYS.TimeSeries.TSVariance(close),
ORDSYS.TimeSeries.TSStdDev(close)
  FROM ORDTDEV.stockdemo_ts
  WHERE ticker='ACME';
```

This example might produce the following output:

```
ORDSYS.ORD ORDSYS.ORD ORDSYS.ORD
---------- ---------- ----------
        79      143.5 11.9791486
1 row selected.
```

TSSubtract

Format

```
ORDSYS.TimeSeries.TSSubtract (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    ts2 ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

or

```
ORDSYS.TimeSeries.TSSubtract (
    ts1 ORDSYS.ORDTNumSeriesIOTRef,
    k NUMBER
    [,startDate DATE, endDate DATE]
    ) RETURN ORDSYS.ORDTNumSeries;
```

Description

Given two input time series or a time series and a constant, and optionally starting and ending dates, returns a time series that reflects the subtraction of the second parameter from the first parameter.

Parameters

ts1
The time series (or first time series) whose elements are to be decreased either by corresponding elements in the second time series or by a constant.

ts2
The time series whose elements are to be subtracted from corresponding elements in the first time series.

k
A constant to be subtracted from corresponding elements in the first time series.

startDate
Starting date within the time series for which the subtraction is to be performed. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the subtraction is to be performed. If *endDate* is specified, *startDate* must also be specified.

Usage

The function performs a pairwise subtraction operation on each element of *ts1*, decreasing it by either the corresponding element in *ts2* or by *k*. This operation determines the value of each element of the returned time series. For example:

- If two time series contain daily trade volumes for two stocks, each element of the returned time series contains the result of subtracting the *ts2* volume from the *ts1* volume for that day.
- If a time series (*ts1*) contains closing prices for a stock and if a constant (*k*) of 1 is specified, each element of the returned time series contains the closing price of *ts1* decreased by 1.

If *ts1* and *ts2* are specified, the function returns a time series whose calendar is the result of using the CombineCals function on the calendars associated with these two time series.

An exception is returned if any of the following conditions is true:

- An input time series is null.
- An input time series does not have an associated calendar.
- The calendars associated with *ts1* and *ts2* do not have the same frequency and aligned pattern.
- *endDate* is earlier than *startDate*.

Example

Subtract the low price for stock FUNCO from the high price for stock ACME for each trading day from 14-Nov-1996 through 14-Dec-1996:

```
CONNECT ORDTUSER/ORDTUSER
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';

DECLARE
```

```
tstCal ORDSYS.ORDTCalendar;
startDate date;
endDate   date;
dummyval INTEGER;

BEGIN

 startDate := TO_DATE('11/14/1996');
 endDate   := TO_DATE('12/14/1996');
 SELECT ORDSYS.TimeSeries.Display(
      ORDSYS.TimeSeries.TSSubtract(ts1.high, ts2.low, startDate, endDate),
     'TSSubtract Results') INTO dummyval
 FROM ORDTDEV.stockdemo_ts ts1, ORDTDEV.stockdemo_ts ts2
 WHERE ts1.ticker='ACME' and ts2.ticker='FUNCO';

END;
/
```

This example might produce the following output:

```
TSSubtract Results :

Calendar Data:
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/14/1996 00:00:00      45.13
 11/15/1996 00:00:00      46.16
 11/18/1996 00:00:00      47.13
 11/19/1996 00:00:00      48.15
 11/20/1996 00:00:00      49.18
 11/21/1996 00:00:00      50.16
 11/22/1996 00:00:00      51.15
 11/25/1996 00:00:00      52.19
 11/26/1996 00:00:00      53.22
 11/27/1996 00:00:00      54.29
```

```
11/29/1996 00:00:00      55.25
12/02/1996 00:00:00      56.12
12/03/1996 00:00:00      56.97
12/04/1996 00:00:00      57.98
12/05/1996 00:00:00      58.87
12/06/1996 00:00:00      60.25
12/09/1996 00:00:00      61.23
12/10/1996 00:00:00      62.2
12/11/1996 00:00:00      63.5
12/12/1996 00:00:00      64.59
12/13/1996 00:00:00      65.6
----------------------------
```

TSSum

Format

```
ORDSYS.TimeSeries.TSSum (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the sum of all non-null time series entries.

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the sum is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the sum is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the sum of the daily trading volumes for stock AONE for the month of January 1990 (that is, the total AONE volume for the month):

```
SELECT ORDSYS.TimeSeries.TSSum(volume,
                        to_date('01/01/1990 00:00:00',
                                'MM/DD/YYYY HH24:MI:SS'),
                        to_date('01/31/1990 23:59:59',
                                'MM/DD/YYYY HH24:MI:SS')) TSSum
  FROM ORDTDEV.Stocks_TS
  WHERE ticker='AONE';
```

This example might produce the following output:

```
TSSUM
----------
 104434900
1 row selected.
```

TSVariance

Format

```
ORDSYS.TimeSeries.TSVariance (
    ts ORDSYS.ORDTNumSeriesIOTRef
    [,startDate DATE, endDate DATE]
    ) RETURN NUMBER;
```

Description

Given an input ORDTNumSeries and optionally starting and ending dates, returns a number corresponding to the variance of all non-null time series entries. (This function is analogous to the SQL group function VAR.)

Parameters

ts
The input time series.

startDate
Starting date within the time series for which the variance is to be calculated. If *startDate* is specified, *endDate* must also be specified.

endDate
Ending date within the time series for which the variance is to be calculated. If *endDate* is specified, *startDate* must also be specified.

Usage

If the date range refers to a time series with fewer than two timestamps, a null is returned.

An exception is returned if any of the following conditions is true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- *endDate* is earlier than *startDate*.

Example

Return the average, variance, and standard deviation of the closing price of stock ACME:

```
--
-- Compute various aggregate statistics.
--
SELECT ORDSYS.TimeSeries.TSAvg(close), ORDSYS.TimeSeries.TSVariance(close),
ORDSYS.TimeSeries.TSStdDev(close)
  FROM ORDTDEV.stockdemo_ts
  WHERE ticker='ACME';
```

This example might produce the following output:

```
ORDSYS.ORD ORDSYS.ORD ORDSYS.ORD
---------- ---------- ----------
        79      143.5 11.9791486
1 row selected.
```

ValidateTS

Format

```
ORDSYS.TimeSeries.ValidateTS(
    ts IN ORDSYS.ORDTNumSeriesIOTRef,
    outMesg OUT VARCHAR2,
    loDateTab OUT ORDSYS.ORDTDateTab,
    hiDateTab  OUT ORDSYS.ORDTDateTab,
    impreciseDateTab OUT ORDSYS.ORDTDateTab,
    duplicateDateTab OUT ORDSYS.ORDTDateTab,
    extraDateTab  OUT ORDSYS.ORDTDateTab,
    missingDateTab  OUT ORDSYS.ORDTDateTab
    ) RETURN INTEGER;
```

or

```
ORDSYS.TimeSeries.ValidateTS(
    ts IN ORDSYS.ORDTVarchar2SeriesIOTRef,
    outMesg OUT VARCHAR2,
    loDateTab OUT ORDSYS.ORDTDateTab,
    hiDateTab  OUT ORDSYS.ORDTDateTab,
    impreciseDateTab OUT ORDSYS.ORDTDateTab,
    duplicateDateTab OUT ORDSYS.ORDTDateTab,
    extraDateTab OUT ORDSYS.ORDTDateTab,
    missingDateTab OUT ORDSYS.ORDTDateTab
    ) RETURN INTEGER;
```

Description

Checks whether a time series is valid, and if the time series is not valid, outputs a diagnostic message and tables with timestamps that are causing the time series to be invalid.

Parameters

ts
The time series to be checked for validity.

outMesg
If the time series is invalid (if the return value = 0), contains a diagnostic message describing any problems.

loDateTab
A table of dates before the starting date of the calendar associated with the time series.

hiDateTab
A table of dates after the ending date of the calendar associated with the calendar.

impreciseDateTab
A table of the imprecise timestamps found in the time series.

duplicateDateTab
A table of the duplicate timestamps found in the time series.

extraDateTab
A table of dates that are included in the time series but that should be excluded based on the calendar definition (for example, a Saturday timestamp that is in a Monday-Friday calendar and that is not an on-exception).

missingDateTab
A table of dates that are excluded from the time series but that should be included based on the calendar definition (for example, a Wednesday date that is not a holiday in a Monday-Friday calendar and for which there is no data). Such dates can be considered as "holes" in the time series.

Usage

The function returns one of the following values:

Value	Meaning
1	The time series is valid. No errors were found.
0	The time series in invalid.

A time series is invalid if one or more of the following conditions are true:

- The time series (*ts*) is null.
- The time series (*ts*) does not have an associated calendar.
- The calendar associated with the time series is invalid.
- The timestamps are not sorted.
- One or more timestamps are null, imprecise, or outside the date range of the calendar.
- One or more timestamps are included in the time series but should be excluded based on the calendar definition (for example, a Saturday timestamp that is in a Monday-Friday calendar and that is not an on-exception).
- One or more timestamps are excluded from the time series but should be included based on the calendar definition (for example, a Wednesday date that is not a holiday in a Monday-Friday calendar and for which there is no data). Such dates can be considered as "holes" in the time series.

Contrast this function with IsValidTS, which simply checks whether a time series is valid.

You can use the DisplayValTS procedure (documented in this chapter) to display the information returned by the ValidateTS function.

The ValidateTS function cannot be called from SQL. It must be called from PL/SQL because of the OUT parameters.

Example

Use the IsValidTS and ValidateTS functions and the DisplayValTS procedure with an invalid time series:

```
SET SERVEROUTPUT ON
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY HH24:MI:SS';
```

```
DECLARE
 numTS  ORDSYS.ORDTNumSeries;
 tempVal integer;
 retIsValid  integer;
 retValTS    integer;
 loDateTab  ORDSYS.ORDTDateTab := NULL;
 hiDateTab  ORDSYS.ORDTDateTab := NULL;
 impDateTab ORDSYS.ORDTDateTab := NULL;
 dupDateTab ORDSYS.ORDTDateTab := NULL;
 extraDateTab ORDSYS.ORDTDateTab := NULL;
 missingDateTab ORDSYS.ORDTDateTab := NULL;
 outMesg varchar2(2000);

BEGIN

   --  Set the buffer size
   DBMS_OUTPUT.ENABLE(100000);

   --
   -- NOTE: Here an instance of the time series is materialized
   -- so that it could be modified to generate an invalid time series.
   --
   SELECT ORDSYS.TIMESERIES.GetSeries(ts.open) INTO numTS
   FROM ordtdev.stockdemo_ts ts
   WHERE ts.ticker = 'ACME';

   -- Example of validating a valid time series.
   SELECT ordsys.timeseries.display(numTS, 'A VALID TIME SERIES') INTO tempVal
   FROM dual;
   retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
   retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg, loDateTab,
                           hiDateTab, impDateTab, dupDateTab,
                           extraDateTab, missingDateTab);
   DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
   DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
   ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                         impDateTab, dupDateTab, extraDateTab, missingDateTab,
                         'Testing DisplayValTS');
   DBMS_OUTPUT.NEW_LINE;

   -- For illustration let us first create an invalid timeseries.
   --
   -- Here we are adjusting the calendar's minDate and maxDate to avoid
   -- getting a huge list of missing dates.
   --
```

```
   numTS.cal.minDate := TO_DATE('10/28/1996');
   numTS.cal.maxDate := TO_DATE('01/05/1997');

   -- Add Dates Before numTS.cal.minDate
   numTS.series(10).tstamp := numTS.cal.minDate - 1;
   numTS.series(11).tstamp := numTS.cal.minDate - 2;

   -- Add Dates Beyond numTS.cal.maxDate
   numTS.series(12).tstamp := numTS.cal.maxDate + 1;
   numTS.series(13).tstamp := numTS.cal.maxDate + 2;

   -- Add some null timestamps
   numTS.series(14).tstamp := NULL;
   numTS.series(15).tstamp := NULL;

   -- Add some imprecise dates (some are duplicated)
   numTS.series(17).tstamp := numTS.series(16).tstamp + 1/24;
   numTS.series(18).tstamp := numTS.series(16).tstamp + 15/24;

   -- Add some duplicate timestamps
   numTS.series(19).tstamp := numTS.series(18).tstamp;
   numTS.series(21).tstamp := numTS.series(20).tstamp;

   -- Add some extra dates in the middle
   numTS.series(37).tstamp := TO_DATE('12/28/1996');
   numTS.series(36).tstamp := TO_DATE('12/29/1996');

   -- Add some holes at the end
   numTS.series(numTS.series.count).tstamp := TO_DATE('01/04/1997');

   -- Example of validating an invalid time series.
   SELECT ordsys.timeseries.display(numTS, 'AN INVALID TIME SERIES')
   INTO tempVal FROM dual;
   retIsValid := ORDSYS.TIMESERIES.IsValidTS(numTS);
   retValTS := ORDSYS.TIMESERIES.ValidateTS(numTS, outMesg,
                        loDateTab, hiDateTab, impDateTab,
                        dupDateTab, extraDateTab, missingDateTab);
   DBMS_OUTPUT.PUT_LINE('Value returned by IsValid  = ' || retIsValid);
   DBMS_OUTPUT.PUT_LINE('Value returned by ValidateTS  = ' || retValTS);
   ORDSYS.TIMESERIES.DisplayValTS(retValTS, outMesg, loDateTab, hiDateTab,
                     impDateTab, dupDateTab, extraDateTab, missingDateTab,
                     'Testing DisplayValTS');
END;
/
```

This example might produce the following output:

```
A VALID TIME SERIES :

Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 01/01/1990 00:00:00
 MaxDate = 01/01/2001 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 11/14/1996 00:00:00      68
 11/15/1996 00:00:00      69
 11/18/1996 00:00:00      70
 11/19/1996 00:00:00      71
 11/20/1996 00:00:00      72
 11/21/1996 00:00:00      73
 11/22/1996 00:00:00      74
 11/25/1996 00:00:00      75
 11/26/1996 00:00:00      76
 11/27/1996 00:00:00      77
 11/29/1996 00:00:00      78
 12/02/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
```

```
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/23/1996 00:00:00      94
 12/24/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 12/31/1996 00:00:00      99
 ----------------------------

Value returned by IsValid  = 1
Value returned by ValidateTS  = 1

DisplayValTS: Testing DisplayValTS:

TS-SUC: the input time series is a valid time series


AN INVALID TIME SERIES :

Name = ACME open NumSeries
Calendar Data:
Calendar Name = BUSINESS-96
 Frequency = 4
 MinDate = 10/28/1996 00:00:00
 MaxDate = 01/05/1997 00:00:00
 patBits:
          0111110
 patAnchor = 01/07/1996 00:00:00
 onExceptions  :
 offExceptions :
     11/28/1996 00:00:00     12/25/1996 00:00:00
Series Data:
 ----------------------------
 Date                    Value
 11/01/1996 00:00:00      59
 11/04/1996 00:00:00      60
 11/05/1996 00:00:00      61
```

```
 11/06/1996 00:00:00      62
 11/07/1996 00:00:00      63
 11/08/1996 00:00:00      64
 11/11/1996 00:00:00      65
 11/12/1996 00:00:00      66
 11/13/1996 00:00:00      67
 10/27/1996 00:00:00      68
 10/26/1996 00:00:00      69
 01/06/1997 00:00:00      70
 01/07/1997 00:00:00      71
       72
       73
 11/22/1996 00:00:00      74
 11/22/1996 01:00:00      75
 11/22/1996 15:00:00      76
 11/22/1996 15:00:00      77
 11/29/1996 00:00:00      78
 11/29/1996 00:00:00      79
 12/03/1996 00:00:00      80
 12/04/1996 00:00:00      81
 12/05/1996 00:00:00      82
 12/06/1996 00:00:00      83
 12/09/1996 00:00:00      84
 12/10/1996 00:00:00      85
 12/11/1996 00:00:00      86
 12/12/1996 00:00:00      87
 12/13/1996 00:00:00      88
 12/16/1996 00:00:00      89
 12/17/1996 00:00:00      90
 12/18/1996 00:00:00      91
 12/19/1996 00:00:00      92
 12/20/1996 00:00:00      93
 12/29/1996 00:00:00      94
 12/28/1996 00:00:00      95
 12/26/1996 00:00:00      96
 12/27/1996 00:00:00      97
 12/30/1996 00:00:00      98
 01/04/1997 00:00:00      99
 ----------------------------

Value returned by IsValid  = 0
Value returned by ValidateTS  = 0

DisplayValTS: Testing DisplayValTS:
```

```
TS-WRN: the input time series has errors. See the message for details

message output by validateTS:

TS-ERR: the input time series is unsorted
TS-ERR: the time series has null timestamps
TS-ERR: the time series has timestamps < calendar minDate (refer LoDateTab)
TS-ERR: the time series has timestamps > calendar maxDate (refer HiDateTab)
TS-ERR: the time series has imprecise timestamps (refer impreciseDateTab)
TS-ERR: the time series has duplicate timestamps (refer DuplicateDateTab)

list of dates < calendar minDate - lowDateTab :

     10/26/1996 00:00:00     10/27/1996 00:00:00


list of dates > calendar maxDate - hiDateTab :
     01/06/1997 00:00:00     01/07/1997 00:00:00

list of imprecise dates - impreciseDateTab :

     11/22/1996 01:00:00     11/22/1996 15:00:00

list of duplicate dates - duplicateDateTab :

     11/22/1996 15:00:00     11/29/1996 00:00:00

ExtraDateTab :

     12/28/1996 00:00:00     12/29/1996 00:00:00     01/04/1997 00:00:00

MissingDateTab :

     10/28/1996 00:00:00     10/29/1996 00:00:00     10/30/1996 00:00:00
     10/31/1996 00:00:00     11/14/1996 00:00:00     11/15/1996 00:00:00
     11/18/1996 00:00:00     11/19/1996 00:00:00     11/20/1996 00:00:00
     11/21/1996 00:00:00     11/25/1996 00:00:00     11/26/1996 00:00:00
     11/27/1996 00:00:00     12/02/1996 00:00:00     12/23/1996 00:00:00
     12/24/1996 00:00:00     12/31/1996 00:00:00     01/01/1997 00:00:00
     01/02/1997 00:00:00   01/03/1997 00:00:00
```

A

Error Messages

This appendix lists the Time Series cartridge error messages, including the cause and recommended user action for each.

TS-00500, "internal error"

Cause: This is the generic internal error number for Time Series exceptions. This indicates that a process has encountered an exceptional condition.

Action: Report as a bug.

TS-00501, "the input patterns are not of the same length"

Cause: The input calendars have patterns of different lengths. For example, '0,1,1,1,1,1,0' and '0,1,1,1,1,1,0' were specified.

Action: Use calendars with patterns of the same length.

TS-00503, "patanchor can be null only if patbits.count=1 or all patbits are the same"

Cause: Pattern anchor was null, and pattern was not acceptable for a null pat-anchor. The anchor can be null only when using all-zero or all-one pattern bits.

Action: Supply a pattern anchor date, or adjust the pattern bits.

TS-00504, "illegal validflag parameter was passed to DisplayValCal/Display-ValTS"

Cause: DisplayValCal or DisplayValTS was called with invalid parameters.

Action: Only call DisplayValCal and DisplayValTS with the output of Validate-Cal or ValidateTS, respectively.

TS-00505, "illegal outmessage parameter was passed to DisplayValCal/Display-ValTS"

Cause: DisplayValCal or DisplayValTS was called with invalid parameters.

Action: Only call DisplayValCal and DisplayValTS with the output of Validate-Cal or ValidateTS, respectively.

TS-00506, "the calendar pattern is null"

Cause: The Time Series Cartridge encountered a calendar having a null pattern.

Action: Ensure that all calendars have a non-null pattern.

TS-00507, "the calendar has an imprecise mindate or maxdate"

Cause: The Time Series Cartridge encountered a calendar having an imprecise mindate or maxdate.

Action: Ensure that all calendar mindates and maxdates are precise.

TS-00510, "datetab has dates outside the bounds of the calendar"

Cause: DeriveExceptions encountered dates outside of the input calendar's mindate/maxdate.

Action: Adjust mindate/maxdate or remove extraneous dates from the input DateTab.

TS-00511, "calendar pattern bits array is either empty or null"

Cause: The Time Series Cartridge encountered a calendar with an empty or null array of pattern bits.

Action: Update the calendar to include a valid array of pattern bits.

TS-00512, "invalid frequency value - valid frequencies are: 1,2,3,4,6,8"

Cause: The Time Series Cartridge has encountered a calendar with an unsupported frequency.

Action: Restrict all calendars to frequencies: 1,2,3,4,6,8.

TS-00513, "the input dates are in the wrong order"

Cause: The date range provided was in reverse order.

Action: When specifying a date range, always list the earlier date first.

TS-00514, "calendar pattern has an imprecise anchor date"

Cause: The Time Series Cartridge has encountered a calendar with an anchor having the wrong precision.

Action: Adjust the precision of the anchor to match the calendars frequency.

TS-00515, "input date is less than calendar mindate"

Cause: The Time Series Cartridge has encountered a date less than mindate.

Action: Ensure that all input dates are within the mindate-maxdate range of the calendar.

TS-00516, "input date is greater than calendar maxdate"

Cause: The Time Series Cartridge has encountered a date greater than maxdate.

Action: Ensure that all input dates are within the mindate-maxdate range of the calendar.

TS-00519, "the series attribute of the time series type is null"

Cause: The Time Series Cartridge has encountered a null series within a time series.

Action: Ensure that all time series have a non-null series component.

TS-00520, "the input calendar is null"

Cause: The Time Series Cartridge has encountered a null calendar.

Action: Ensure that all calendars are non-null.

TS-00522, "error scaling date to calendar"

Cause: Input date cannot be scaled to given calendar.

Action: Ensure that the given calendar is valid and that the calendar's mindate and maxdate encompass all potential timestamp values.

TS-00523, "the input date is null"

Cause: Scaleup has encountered a null date. No scaling semantics are defined for a null date.

Action: Ensure that all input to Scaleup is non-null.

TS-00525, "the input time series is null"

Cause: The Time Series Cartridge has encountered a null time series.

Action: Ensure that all time series are not atomically null.

TS-00526, "the input time series has a null calendar"

Cause: The Time Series Cartridge has encountered a null calendar within a time series.

Action: Ensure that all time series include valid (non-null) calendars.

TS-00527, "error scaling up to the target calendar frequency"

Cause: Scaleup encountered a target calendar of finer frequency than that of the input time series' calendar.

Action: Scaleup requires a target calendar of equal or coarser frequency.

TS-00528, "calendar has a null mindate or a null maxdate"

Cause: The Time Series Cartridge has encountered a calendar with a null mindate or maxdate.

Action: Ensure that all calendars have a valid mindate and maxdate.

TS-00529, "calendar mindate is greater than its maxdate"

Cause: The Time Series Cartridge has encountered a calendar with mindate > maxdate.

Action: Ensure that all calendars have a valid mindate <= maxdate.

TS-00530, "series indexes must be greater than 0"

Cause: GetNthElement encountered an index less than 1.

Action: Use indexes greater than 0.

TS-00531, "the input time series has a null calendar reference"

Cause: The Time Series Cartridge has encountered a time series with a null calendar reference.

Action: Ensure that all calendar references are valid.

TS-00532, "unable to DEREF calendar referenced by time series"

Cause: The Time Series Cartridge was unable to dereference a calendar reference.

Action: Verify that the user executing the query has select privileges for the calendar table storing the object, and that the correct calendar has been referenced by the time series ref.

TS-00533, "the time series has data beyond its calendar mindate/maxdate"

Cause: The Time Series Cartridge has encountered a time series with data beyond mindate / maxdate.

Action: Ensure that all timestamps in a time series are within the calendar's mindate / maxdate.

TS-00534, "the number of rows requested must be a positive integer"

Cause: The requested number of rows was less than 0.

Action: Use a positive number to specify the number of rows requested.

TS-00535, "the time series ref has a null table_name parameter"

Cause: The Time Series Cartridge has encountered a time series ref having a null table_name.

Action: Ensure that all time series refs include a valid table name.

TS-00536, "the time series ref has a null tstamp_colname parameter"

Cause: The Time Series Cartridge has encountered a time series ref having a null tstamp_colname.

Action: Ensure that all time series refs include a valid timestamp column name.

TS-00537, "the time series ref has a null value_colname parameter"

Cause: The Time Series Cartridge has encountered a time series ref having a null value_colname.

Action: Ensure that all time series refs include a valid value column name.

TS-00538, "the time series ref has a null qualifier_colname parameter"

Cause: The Time Series Cartridge has encountered a time series ref having a null qualifier_colname.

Action: Ensure that all time series refs include a valid qualifier column name.

TS-00539, "the time series ref has a null qualifier_value parameter"

Cause: The Time Series Cartridge has encountered a time series ref having a null qualifier_value.

Action: Ensure that all time series refs include a valid qualifier value.

TS-00540, "the projected lead timestamp is beyond the calendar mindate/maxdate"

Cause: The given parameters result in timestamps outside of mindate / maxdate.

Action: Adjust the lead timestamp or lead units to remain within calendar mindate / maxdate, or extend the mindate / maxdate.

TS-00541, "the projected lag timestamp is beyond the calendar mindate/maxdate"

Cause: The given parameters result in timestamps outside of mindate / maxdate.

Action: Adjust the lag timestamp or lag units to remain within calendar mindate / maxdate, or extend the mindate / maxdate.

TS-00542, "the window size for mavg/msum must be >= 1"

Cause: Window size parameter passed to moving average/sum was not greater than 0.

Action: Use a window size parameter greater than or equal to 1.

TS-00547, "the input fill type is invalid"

Cause: Fill has been called with a filltype less than 0 or greater than 2.

Action: Use a valid filltype: 0, 1, or 2.

TS-00551, "error parsing the SQL statement with the time series ref"

Cause: The SQL statement constructed from the time series ref was invalid.

Action: Verify the validity of the time series ref:

- Verify the validity of all components of the time series ref.
- No spaces or invalid punctuation may appear in table or column names.
- The user must have select privileges on the table referenced.
- The table name must be qualified with its schema name.

TS-00552, "error executing the SQL statement with the time series ref"

Cause: The SQL statement constructed from the time series ref was invalid.

Action: Verify the validity of the time series ref:

- Verify the validity of all components of the time series ref.
- No spaces or invalid punctuation may appear in table or column names.
- The user must have select privileges on the table referenced.
- The table name must be qualified with its schema name.

TS-00553, "divide by zero error"

Cause: An attempt was made to divide by zero with TSDivide.

Action: When dividing by a constant, ensure that the constant is non-zero.

TS-00554, "the input calendar patterns are not equal"

Cause: DeriveExceptions requires the reference time series' calendar to have the same pattern as the time series being processed.

Action: Ensure that DeriveExceptions is called only with time series having the same pattern.

TS-00555, "the input calendar frequencies are not equal"

Cause: DeriveExceptions requires the reference time series' calendar to have the same frequency as the time series being processed.

Action: Ensure that DeriveExceptions is called only with time series having the same frequency.

TS-00556, "mindate of the ref calendar exceeds the mindate of the target calendar"

Cause: DeriveExceptions encountered a reference time series' calendar having a mindate greater than that of the target time series' calendar.

Action: Ensure that DeriveExceptions is called only with appropriate time series.

TS-00557, "maxdate of the target calendar exceeds the maxdate of the ref calendar"

Cause: DeriveExceptions encountered a reference time series' calendar having a maxdate less than that of the target time series' calendar.

Action: Ensure that DeriveExceptions is called only with appropriate time series.

TS-00558, "the target calendar should have empty on/off exception lists"

Cause: DeriveExceptions encountered a target time series' calendar having non-empty exception lists.

Action: Ensure that DeriveExceptions is called only with target time series having empty exception lists.

TS-00559, "the caltype field in the calendar has an illegal value"

Cause: The Time Series Cartridge encountered a calendar with an invalid calendar type.

Action: Ensure that all calendars have valid calendar type value. Valid calendar types are: (Exception-driven calendars = 0)

Index

A

addition
  TSAdd function, 5-93
  TSSum function, 5-127
advanced-developer demo, 1-4
aggregate functions, 2-28
architecture
  Time Series cartridge, 2-10
arithmetic functions, 2-28
average
  TSAvg function, 5-97
average, moving
  Mavg function, 2-29, 5-69

B

bottom values
  TSMinN function, 5-113
bulk loading of time series data, 3-10
  consistency, 2-20, 3-10

C

calendar, 2-5
  datatypes, 2-7
  defining, 2-8
  exceptions, 2-5
  frequency, 2-6
  pattern, 2-5
  precision, 2-7
  validating, 4-34, 4-69
calendar functions, 2-20, 4-1
Cavg function, 5-3
Cmax function, 5-5
Cmin function, 5-7
collection-based interface, 2-13
CombineCals function, 4-2
consistency of time series data
  approaches, 2-20, 3-10
conventional path (SQL*Loader), 3-13
conversion functions, 2-30
count
  TSCount function, 5-99
Cprod function, 5-9
cumulative sequence functions, 2-29

D

data cartridge
  definition, 1-1
datatypes
  calendar, 2-7
  time series, 2-23
defining
  calendar, 2-8
DeleteExceptions function, 4-7
demos of Time Series cartridge
  advanced-developer, 1-4
  Developer/2000, 1-4
  OCI, 1-4
  PRO*C/C++, 1-4
  usage, 1-4
DeriveExceptions function, 2-9, 5-13
Developer/2000 demo, 1-4
direct path (SQL*Loader), 3-13
Display function, 5-15
DisplayValCal procedure, 4-10

DisplayValTS procedure, 5-18
division
  TSDivide function, 5-101

E

EqualCals function, 4-17
error messages, A-1
exceptions, 2-5
  deriving, 2-9
ExtractCal function, 5-26
ExtractDate function, 5-28
extraction functions, 2-26
ExtractTable function, 5-30
ExtractValue function, 5-32

F

Fill function, 5-34
First function, 5-40
FirstN function, 5-42
frequency, 2-6
functions
  calendar, 2-20, 4-1
  Cavg, 5-3
  Cmax, 5-5
  Cmin, 5-7
  CombineCals, 4-2
  Cprod, 5-9
  DeleteExceptions, 4-7
  DeriveExceptions, 2-9, 5-13
  Display, 5-15
  DisplayValCal procedure, 4-10
  DisplayValTS procedure, 5-18
  EqualCals, 4-17
  ExtractCal, 5-26
  ExtractDate, 5-28
  ExtractTable, 5-30
  ExtractValue, 5-32
  Fill, 5-34
  First, 5-40
  FirstN, 5-42
  GetDatedElement, 5-44
  GetNthElement, 5-46
  GetOffset, 4-20
  GetSeries, 5-48
  InsertExceptions, 4-23
  IntersectCals, 4-27
  InvalidTimeStampsBetween, 4-31
  IsValidCal, 4-34
  IsValidDate, 4-40
  IsValidTS, 5-51
  Lag, 5-59
  Last, 5-62
  LastN, 5-64
  Lead, 5-66
  Mavg, 5-69
  Msum, 5-72
  NumInvalidTimeStampsBetween, 4-43
  NumOffExceptions, 4-46
  NumOnExceptions, 4-49
  NumTimeStampsBetween, 4-52
  OffsetDate, 4-55
  Scaleup, 5-74
  ScaleupAvg, 5-76
  ScaleupCount, 5-78
  ScaleupFirst, 5-80
  ScaleupLast, 5-82
  ScaleupMax, 5-84
  ScaleupMin, 5-86
  ScaleupSum, 5-88
  SetPrecision, 4-58
  time scaling, 2-31, 5-1
  time series, 2-23, 5-1
  TimeStampsBetween, 4-61
  TrimSeries, 5-90
  TSAdd, 5-93
  TSAvg, 5-97
  TSCount, 5-99
  TSDivide, 5-101
  TSMax, 5-105
  TSMaxN, 5-107
  TSMedian, 5-109
  TSMin, 5-111
  TSMinN, 5-113
  TSMultiply, 5-115
  TSProd, 5-119
  TSStdDev, 5-121
  TSSubtract, 5-123
  TSSum, 5-127

TSVariance, 5-129
UnionCals, 4-65
ValidateCal, 4-69
ValidateTS, 5-131

G

GetDatedElement function, 5-44
GetNthElement function, 5-46
GetOffset function, 4-20
GetSeries function, 5-48

H

highest values
TSMaxN, 5-107

I

incremental loading of time series data, 3-12
InsertExceptions function, 4-23
installation of the cartridge, 1-3
instance-based interface, 2-13
INSTEAD OF triggers, 3-4
IntersectCals function, 4-27
InvalidTimeStampsBetween function, 4-31
irregular time series, 2-2, 2-6
IsValidCal function, 4-34
IsValidDate function, 4-40
IsValidTS function, 5-51

K

kit installation, 1-3

L

Lag function, 5-59
Last function, 5-62
LastN function, 5-64
Lead function, 5-66
loading
time series data, 3-9
lowest values
TSMinN, 5-113

M

Mavg function, 2-29, 5-69
maximum
TSMax function, 5-105
median
TSMedian function, 5-109
messages
error, A-1
metadata for usage demo, 1-6, 2-16
minimum
TSMin function, 5-111
moving average
Mavg function, 2-29, 5-69
moving sum
Msum function, 2-29, 5-72
Msum function, 2-29, 5-72
multiplication
TSMultiply function, 5-115
TSProd function, 5-119

N

null operand semantics, 2-24
NumInvalidTimeStampsBetween function, 4-43
NumOffExceptions function, 4-46
NumOnExceptions function, 4-49
NumTimeStampsBetween function, 4-52

O

object relational technology, 1-2
OCI demo, 1-4
off-exception, 2-5
semantics, 2-25
OffsetDate function, 4-55
on-exception, 2-5
Oracle Forms demo (Developer/2000), 1-4
ORDSYS schema, 1-4
ORDTCalendar datatype, 2-7
ORDTDateTab datatype, 2-24
ORDTExceptions datatype, 2-7
ORDTNumCell datatype, 2-23
ORDTNumSeries datatype, 2-23
ORDTNumSeriesIOTRef datatype, 2-23
ORDTNumSeriesIOTRef type, 2-15

ORDTNumTab datatype, 2-23
ORDTPattern datatype, 2-7
ORDTPatternBits datatype, 2-7
ordtsyn.sql (public synonyms), 1-4
ORDTVarchar2Cell datatype, 2-23
ORDTVarchar2Series datatype, 2-23
ORDTVarchar2SeriesIOTRef datatype, 2-24
ORDTVarchar2Tab datatype, 2-23

P

package names
  public synonyms for, 1-4
pattern, 2-5
precision, 2-7
  purified timestamps, 2-19
PRO*C / C++ demo, 1-4
procedures
  DisplayValCal, 4-10
  DisplayValTS, 5-18
product
  TSAvg function, 5-119
public synonyms for package names, 1-4
purified timestamps, 2-19

R

reference-based interface, 2-15
reference-based view, 1-6, 2-16
regular time series, 2-1

S

Scaleup function, 5-74
ScaleupAvg function, 5-76
ScaleupCount function, 5-78
ScaleupFirst function, 5-80
ScaleupLast function, 5-82
ScaleupMax function, 5-84
ScaleupMin function, 5-86
ScaleupSum function, 5-88
security view, 1-6, 2-16, 3-4
semantics
  null operands, 2-24
  off-exception operands, 2-25
server output, setting, 5-16
SET SERVEROUTPUT ON statement, 5-16
SetPrecision function, 4-58
shift functions, 2-27
SQL formatting functions, 2-27
SQL*Loader utility, 3-9
  bulk loading, 3-10
  conventional and direct paths, 3-13
  incremental loading, 3-12
standard deviation
  TSStdDev function, 5-121
stockdemo_metadata table, 1-6, 2-16
stockdemo_sv security view, 1-6, 2-16
stockdemo_ts reference-based view, 1-6, 2-16
subtraction
  TSSubtract function, 5-123
sum
  TSSum function, 5-127
sum, moving
  Msum function, 2-29, 5-72
synonyms
  public (package names), 1-4

T

time scaling functions, 2-31, 5-1
time series
  cartridge architecture, 2-10
  data storage, 2-11
  datatypes, 2-23
  irregular, 2-2, 2-6
  regular, 2-1
  validating, 3-11, 5-51, 5-131
time series functions, 2-23, 5-1
TimeStampsBetween function, 4-61
top values
  TSMax function, 5-107
trim functions, 2-26
TrimSeries function, 5-90
TSAdd function, 5-93
TSAvg function, 5-97
TSCount function, 5-99
TSDivide function, 5-101
TSMax function, 5-105
TSMaxN function, 5-107

TSMedian function, 5-109
TSMin function, 5-111
TSMinN function, 5-113
TSMultiply function, 5-115
TSProd function, 5-119
TSStdDev function, 5-121
TSSubtract function, 5-123
TSSum function, 5-127
TSVariance function, 5-129

U

UnionCals function, 4-65
usage demo, 1-4
  files, 1-5
  tables and views, 1-6, 2-16

V

ValidateCal function, 4-69
ValidateTS function, 5-131
validating
  calendar, 4-34, 4-69
  time series, 3-11, 5-51, 5-131
variance
  TSVariance function, 5-129