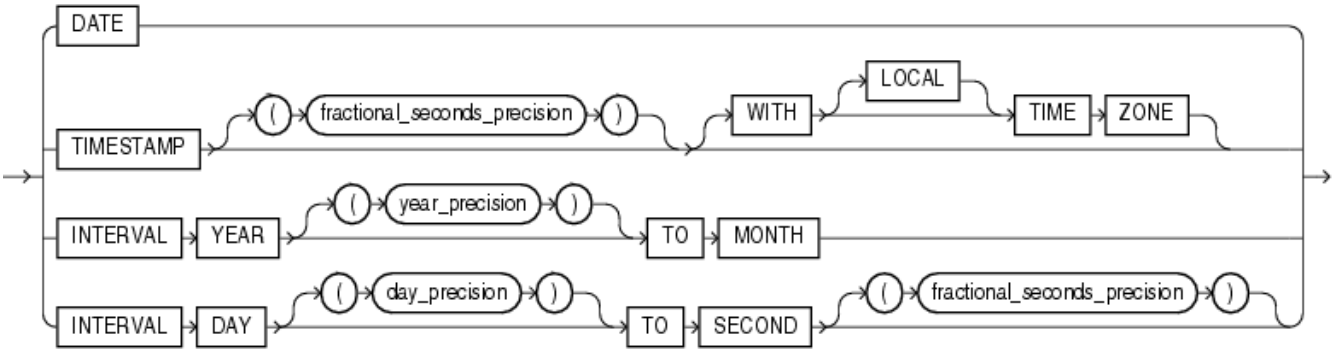
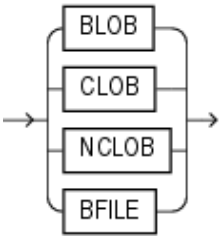
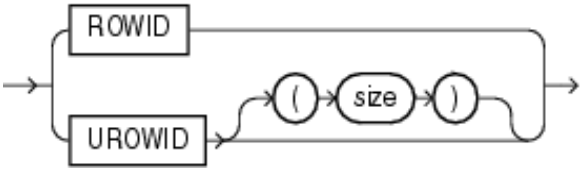


***large_object_datatypes*::=**

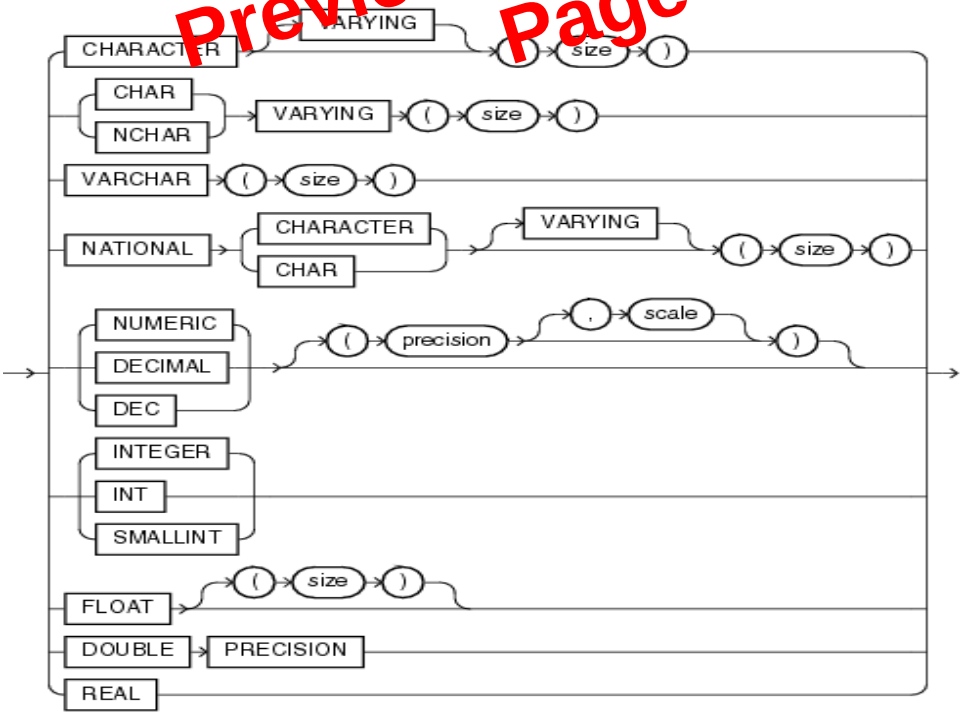


***rowid_datatypes*::=**



The ANSI-supported datatypes appear in the figure that follows. "ANSI, DB2, and SQL/DS Datatypes" discusses the mapping of ANSI-supported datatypes to Oracle built-in datatypes.

***ANSI_supported_datatypes*::=**



***Oracle_supplied_types*::=**

Restricted Rowids

Beginning with Oracle8, Oracle SQL incorporated an extended format for rowids to efficiently support partitioned tables and indexes and tablespace-relative data block addresses (DBAs) without ambiguity.

Character values representing rowids in Oracle7 and earlier releases are called **restricted** rowids. Their format is as follows:

```
block.row.file
```

where:

- ***block*** is a hexadecimal string identifying the data block of the datafile containing the row. The length of this string depends on your operating system.
- ***row*** is a four-digit hexadecimal string identifying the row in the data block. The first row of the block has a digit of 0.
- ***file*** is a hexadecimal string identifying the database file containing the row. The first datafile has the number 1. The length of this string depends on your operating system.

Extended Rowids

The **extended ROWID** datatype stored in a user column includes the data in the restricted rowid plus a **data object number**. The data object number is an identification number assigned to every database segment. You can retrieve the data object number from the data dictionary views **USER_OBJECTS**, **DBA_OBJECTS**, and **ALL_OBJECTS**. Objects that share the same segment (clustered tables in the same cluster, for example) have the same object number.

Extended rowids are stored as base 64 values that can contain the characters A-Z, a-z, 0-9, and the plus sign (+) and forward slash (/). Extended rowids are not available directly. You can use a supplied package, **DBMS_ROWID**, to interpret extended rowid contents. The package functions extract and provide information that would be available directly from a restricted rowid as well as information specific to extended rowids.

See Also:

PL/SQL Packages and Types Reference for information on the functions available with the **DBMS_ROWID** package and how to use them

Compatibility and Migration

The restricted form of a rowid is still supported in this release for backward compatibility, but all tables return rowids in the extended format.

See Also:

Oracle Database Upgrade Guide for information regarding compatibility and migration issues

UROWID Datatype

Each row in a database has an address. However, the rows of some tables have addresses that are not physical or permanent or were not generated by Oracle Database. For example, the row addresses of index-organized tables are stored in index leaves, which can move. Rowids of foreign tables (such as DB2 tables accessed through a gateway) are not standard Oracle rowids.

The **SDO_GEOMETRY** object type has the following definition:

```
CREATE TYPE SDO_GEOMETRY AS OBJECT ( sgo_gtype NUMBER, sdo_srid NUMBER, sdo_point SDO_POINT_TYPE, sdo_elem_info SDO_ELEM_INFO_ARRAY, sdo_ordinates SDO_ORDINATE_ARRAY);
```

SDO_TOPO_GEOMETRY

This type describes a topology geometry, which is stored in a single row, in a single column of object type **SDO_TOPO_GEOMETRY** in a user-defined table.

The **SDO_TOPO_GEOMETRY** object type has the following definition:

```
CREATE TYPE SDO_TOPO_GEOMETRY AS OBJECT ( tg_type NUMBER, tg_id NUMBER, tg_layer_id NUMBER, topology_id NUMBER);
```

SDO_GEORASTER

In the GeoRaster object-relational model, a raster grid or image object is stored in a single row, in a single column of object type **SDO_GEORASTER** in a user-defined table. Tables of this sort are called GeoRaster tables.

The **SDO_GEORASTER** object type has the following definition:

```
CREATE TYPE SDO_GEORASTER AS OBJECT ( rasterType NUMBER, spatialExtent SDO_GEOMETRY, rasterDataTable VARCHAR2(32), rasterID NUMBER, metadata XMLType);
```


See Also:

Oracle Spatial User's Guide and Reference, *Oracle Spatial Topology and Network Data Models*, and *Oracle Spatial GeoRaster* for information on the full implementation of the spatial datatypes and guidelines for using them

Media Types

Oracle *inter*Media uses object types, similar to Java or C++ classes, to describe multimedia data. An instance of these object types consists of attributes, including metadata and the media data, and methods. The *inter*Media datatypes are created in the **ORDSYS** schema. Public synonyms exist for all the datatypes, so you can access them without specifying the schema name.

See Also:

Oracle interMedia Reference for information on the implementation of these types and guidelines for using them

ORDAudio

The **ORDAUDIO** object type supports the storage and management of audio data.

ORDImage

The **ORDIMAGE** object type supports the storage and management of image data.

SI_FeatureList

The **SI_FeatureList** object type is a list containing up to four of the image features represented by the preceding object types (**SI_AverageColor**,**SI_ColorHistogram**, **SI_PositionalColor**, and **SI_Texture**), where each feature is associated with a feature weight.

Expression Filter Type

The Oracle Expression Filter allows application developers to manage and evaluate conditional expressions that describe users' interests in data. The Expression Filter includes the following datatype:

Expression

Expression Filter uses a virtual datatype called **Expression** to manage and evaluate conditional expressions as data in database tables. The Expression Filter creates a column of **Expression** datatype from a **VARCHAR2** column by assigning an attribute set to the column. This assignment enables a data constraint that ensures the validity of expressions stored in the column.

You can define conditions using the **EVALUATE** operator on an **Expression** datatype to evaluate the expressions stored in a column for some data. If you are using Enterprise Edition, then you can also define an Expression Filter index on a column of **Expression** datatype to process queries using the **EVALUATE** operator.

1.

    a. The **DECIMAL** datatype can specify only fixed-point numbers. For this datatype, *b* defaults to 0..
    b. The **FLOAT** datatype is a floating-point number with a binary precision *b*. The default precision for this datatype is 126 binary or 38 decimal.

Do not define columns with the following SQL/DS and DB2 datatypes, because they have no corresponding Oracle datatype:

- **GRAPHIC**
- **LONG VARGRAPHIC**
- **VARGRAPHIC**
- **TIME**

Note that data of type **TIME** can also be expressed as Oracle datetime data.

See Also:

Datatypes in *Oracle Database SQL Reference*

User-Defined Types

User-defined datatypes use Oracle built-in datatypes and other user-defined datatypes as the building blocks of object types that model the structure and behavior of data in applications. The sections that follow describe the various categories of user-defined types.

See Also:

- *Oracle Database Concepts* for information about Oracle built-in datatypes
- CREATE TYPE and the CREATE TYPE BODY for information about creating user-defined types

XML Types

Extensible Markup Language (XML) is a standard format developed by the World Wide Web Consortium (W3C) for representing structured and unstructured data on the World Wide Web. Universal resource identifiers (URIs) identify resources such as Web pages anywhere on the Web. Oracle provides types to handle XML and URI data, as well as a class of URIs called **DBURIRef** types to access data stored within the database itself. It also provides a new set of types to store and access both external and internal URIs from within the database.

XMLType

This Oracle-supplied type can be used to store and query XML data in the database. **XMLType** has member functions you can use to access, extract, and query the XML data using XPath expressions. XPath is another standard developed by the W3C committee to traverse XML documents. Oracle**XMLType** functions support many W3C XPath expressions. Oracle also provides a set of SQL functions and PL/SQL packages to create **XMLType**values from existing relational or object-relational data.

XMLType is a system-defined type, so you can use it as an argument of a function or as the datatype of a table or view column. You can also create tables and views of **XMLType**. When you create an **XMLType** column in a table, you can choose to store the XML data in a **CLOB** column or object relationally.

You can also register the schema (using the **DBMS_XMLSCHEMA** package) and create a table or column conforming to the registered schema. In this case Oracle stores the XML data in underlying object-relational columns by default, but you can specify storage in a **CLOB** column even for schema-based data.

Queries and DML on **XMLType** columns operate the same regardless of the storage mechanism.

URI Datatypes

Oracle supplies a family of URI types—**URIType**, **DBURIType**, **XDBURIType**, and **HTTPURIType**—which are related by an inheritance hierarchy.**URIType** is an object type and the others are subtypes of **URIType**. Since **URIType** is the supertype, you can create columns of this type and store**DBURIType** or **HTTPURIType** type instances in this column.