DataTypes

Macneil Fernandes©2005

Introduction to Oracle DataTypes

When you create a table, you must specify a datatype for each of its columns.

Oracle provides the following built-in datatypes:

- Character Datatypes
  - CHAR Datatype
  - VARCHAR2 and VARCHAR Datatypes
  - NCHAR and NVARCHAR2 Datatypes
  - LONG Datatype
- NUMBER Datatype
- DATE Datatype
- LOB Datatypes
  - BLOB Datatype
  - CLOB and NCLOB Datatypes
  - BFILE Datatype

Macneil Fernandes©2005

Introduction to Oracle Datatypes(contd.)

- RAW and LONG RAW Datatypes
- ROWID and UROWID Datatypes
  - Physical Rowids
  - Logical Rowids

Macneil Fernandes©2005

Character Datatypes

- The character datatypes store character (alphanumeric) data in strings, with byte values corresponding to the character encoding scheme, generally called a character set or code page.
- The database's character set is established when you create the database. Examples of character sets are 7-bit ASCII (American Standard Code for Information Interchange), EBCDIC (Extended Binary Coded Decimal Interchange Code), Code Page 500, Japan Extended UNIX, and Unicode UTF-8.

Macneil Fernandes©2005

CHAR Datatype

- The `CHAR` datatype stores fixed-length character strings. When you create a table with a `CHAR` column, you must specify a string length (in bytes or characters) between 1 and 2000 bytes for the `CHAR` column width. The default is 1 byte.
- Oracle then guarantees that:
  - When you insert or update a row in the table, the value for the CHAR column has the fixed length.
  - If you give a shorter value, then the value is blank-padded to the fixed length.
  - If a value is too large, Oracle returns an error.

Macneil Fernandes©2005

VARCHAR2 and VARCHAR Datatypes

- The `VARCHAR2` datatype stores variable-length character strings. When you create a table with a `VARCHAR2` column, you specify a maximum string length (in bytes or characters) between 1 and 4000 bytes for the `VARCHAR2` column.
- For each row, Oracle stores each value in the column as a variable-length field unless a value exceeds the column's maximum length, in which case Oracle returns an error.
- Using `VARCHAR2` and `VARCHAR` saves on space used by the table.For example, assume you declare a column `VARCHAR2` with a maximum size of 50characters. In a single-byte character set, if only 10 characters are given for the `VARCHAR2` column value in a particular row, the column in the row's row piece stores only the 10 characters (10 bytes), not 50.
- The `VARCHAR` datatype is synonymous with the `VARCHAR2` datatype.

Macneil Fernandes©2005

NCHAR and NVARCHAR2 Datatypes

- `NCHAR` and `NVARCHAR2` are Unicode data types that store Unicode character data.
- The character set of `NCHAR` and `NVARCHAR2` datatypes can only be either `AL16UTF16` or `UTF8` and is specified at database creation time as the national character set.
- `AL16UTF16` and `UTF8` are both Unicode encoding.
- The `NCHAR` datatype stores fixed-length character strings that correspond to the national character set.
- The `NVARCHAR2` datatype stores variable length character strings.

Macneil Fernandes©2005

LONG Datatype

- Long Datatype is used to store large amounts of character data .
- The LONG datatype is provided for backward compatibility with existing applications.
- In new applications, we use `CLOB` and `NCLOB` datatypes.

Macneil Fernandes©2005

LOB Datatypes

- The `LOB` datatypes `BLOB, CLOB, NCLOB,` and `BFILE` enable you to store large blocks of unstructured data (such as text, graphic images, video clips, and sound waveforms) up to 4 gigabytes in size.
- **BLOB Datatype**

  The `BLOB` datatype stores unstructured binary data in the database. `BLOB`s can store up to 4 gigabytes of binary data.
- **CLOB and NCLOB Datatypes**

  The `CLOB` and `NCLOB` datatypes store up to 4 gigabytes of character data in the database. `CLOB`s store database character set data and `NCLOB`s store Unicode national character set data.

Macneil Fernandes©2005

LOB Datatypes(contd.)

BFILE Datatype

- The `BFILE` datatype stores unstructured binary data in operating-system files outside the database.
- A `BFILE` column or attribute stores a file locator that points to an external file containing the data.
- `BFILE`s can store up to 4 gigabytes of data.
- `BFILE`s are read-only; you cannot modify them.
- The database administrator must ensure that the file exists and that Oracle processes have operating-system read permissions on the file.

Macneil Fernandes©2005

Number DataType

- The `NUMBER` datatype stores fixed and floating-point numbers. Numbers of virtually any magnitude can be stored and are guaranteed portable among different systems operating Oracle, up to 38 digits of precision.
- For numeric columns, you can specify the column as:

```
column_name NUMBER
```

- Optionally, you can also specify a **precision** (total number of digits) and **scale** (number of digits to the right of the decimal point):

```
column_name NUMBER (precision, scale)
```

- If a precision is not specified, the column stores values as given. If no scale is specified, the scale is zero.

Macneil Fernandes©2005

Number Datatype(contd.)

- Oracle guarantees portability of numbers with a precision equal to or less than 38 digits. You can specify a scale and no precision:

```
column_name NUMBER (*, scale)
```

In this case, the precision is 38, and the specified scale is maintained.

Input Data	Specified As	Stored As
7,456,123.89	NUMBER	7456123.89
7,456,123.89	NUMBER(*,1)	7456123.9
7,456,123.89	NUMBER(9)	7456124
7,456,123.89	NUMBER(9,2)	7456123.89
7,456,123.89	NUMBER(9,1)	7456123.9
7,456,123.89	NUMBER(6)	(not accepted, exceeds precision)
7,456,123.89	NUMBER(7,-2)	7456100

Macneil Fernandes©2005

DATE DataType

- For input and output of dates, the standard Oracle default date format is `DD-MON-YY`

  e.g. `'13-NOV-92'`
- You can change this default date format for an instance with the parameter `NLS_DATE_FORMAT.`
- To enter dates that are not in standard Oracle date format, use the TO_DATE function with a format mask:

```
TO_DATE ('November 13, 1992', 'MONTH DD, YYYY')
```

- Oracle stores time in 24-hour format—`HH:MI:SS.`
- To enter the time portion of a date, use the `TO_DATE` function with a format mask indicating the time portion, as in:

```
INSERT INTO birthdays (bname, bday) VALUES
('ANDY',TO_DATE('13-AUG-66 12:56 A.M.','DD-MON-YY
HH:MI A.M.'));
```

Macneil Fernandes©2005

RAW and LONG RAW Datatypes

- The RAW and LONG RAW datatypes are used for data that is not to be interpreted (not converted when moving data between different systems) by Oracle.
- These datatypes are intended for binary data or byte strings. For example, `LONG RAW` can be used to store graphics, sound, documents, or arrays of binary data.
- `RAW` is a variable-length datatype like the `VARCHAR2` character datatype, except Oracle Net Services (which connects user sessions to the instance) and the Import and Export utilities do not perform character conversion when transmitting `RAW` or `LONG RAW` data.
- In contrast, Oracle Net Services and Import/Export automatically convert `CHAR, VARCHAR2,` and `LONG` data between the database character set and the user session character set (set by the `NLS_LANGUAGE` parameter of the `ALTER SESSION` statement), if the two character sets are different.

Macneil Fernandes©2005

ROWID and UROWID Datatypes

Oracle uses a `ROWID` datatype to store the address *(***rowid***)* of every row in the database.

- **Physical rowids** store the addresses of rows in ordinary tables (excluding index-organized tables), clustered tables, table partitions and subpartitions, indexes, and index partitions and subpartitions.
- **Logical rowids** store the addresses of rows in index-organized tables.

A single datatype called the **universal rowid**, or `UROWID,` supports both logical
and physical rowids, as well as rowids of foreign tables such as non-Oracle tables accessed through a gateway.

Macneil Fernandes©2005

Physical Rowids

- Physical rowids provide the fastest possible access to a row of a given table. They contain the physical address of a row (down to the specific block) and allow you to retrieve the row in a single block access. Oracle guarantees that as long as the row exists, its rowid does not change.
- Every row in a nonclustered table is assigned a unique rowid that corresponds to the physical address of a row's row piece (or the initial row piece if the row is chained among multiple row pieces). In the case of clustered tables, rows in different tables that are in the same data block can have the same rowid.
- A row's assigned rowid remains unchanged unless the row is exported and imported using the Import and Export utilities. When you delete a row from a table and then commit the encompassing transaction, the deleted row's associated rowid can be assigned to a row inserted in a subsequent transaction.

Macneil Fernandes©2005

A physical rowid datatype has one of two formats:

1. The **extended rowid** format supports tablespace-relative data block addresses and efficiently identifies rows in partitioned tables and indexes as well as nonpartitioned tables and indexes. Tables and indexes created by an Oracle8*i* (or higher) server always have extended rowids.
2. A **restricted rowid** format is also available for backward compatibility with applications developed with Oracle7 or earlier releases.

Macneil Fernandes©2005

Extended Rowids

```
SELECT ROWID, ename FROM emp WHERE deptno = 20;
```

can return the following row information:

```
       ROWID                     ENAME
------------------------------------------------------
AAAAaoAATAAABrXAAA       BORTINS
AAAAaoAATAAABrXAAE       RUGGLES
AAAAaoAATAAABrXAAG       CHEN
AAAAaoAATAAABrXAAN       BLUMBERG
```

An extended rowid has a four-piece format, `OOOOOOFFFBBBBBBRRR:`

`OOOOOO:` The **data object number** that identifies the database segment (`AAAAao` in the example). Schema objects in the same segment, such as a cluster of tables, have the same data object number.

`FFF:` The tablespace-relative **datafile number** of the datafile that contains the row (file `AAT` in the example).

Macneil Fernandes©2005

`BBBBBB:` The **data block** that contains the row (block `AAABrX` in the example). Block numbers are relative to their datafile, **not** tablespace. Therefore,two rows with identical block numbers could reside in two different datafiles of the same tablespace.

`RRR:` The **row** in the block.

Macneil Fernandes©2005

Restricted Rowids

- Restricted rowids use a binary representation of the physical address for each row selected. When queried using SQL*Plus, the binary representation is converted to a VARCHAR2/hexadecimal representation.
- The following query:

```
SELECT ROWID, ename FROM emp
WHERE deptno = 30;
```

can return the following row information:

```
   ROWID                                                ENAME
------------------------------------------------------------
00000DD5.0000.0001                          RAVI
00000DD5.0001.0001                          SANDEEP
00000DD5.0002.0001                          RAJ
```

Macneil Fernandes©2005

The table shows that a restricted rowid's `ARCHAR2` /hexadecimal representation is in a three-piece format, **block.row.file**:

1. The **data block** that contains the row (block DD5 in the example). Block numbers are relative to their datafile, **not** tablespace. Therefore, two rows with identical block numbers could reside in two different datafiles of the same tablespace.
2. The **row** in the block that contains the row (rows 0, 1, 2 in the example). Row numbers of a given block always start with 0.
3. The **datafile** that contains the row (file 1 in the example). The first datafile of every database is always 1, and file numbers are unique within a database.

Macneil Fernandes©2005

Logical Rowids

- Rows in index-organized tables do not have permanent physical addresses—they are stored in the index leaves and can move within the block or to a different block as a result of insertions. Therefore their row identifiers cannot be based on physical addresses.
- Instead, Oracle provides index-organized tables with logical row identifiers, called **logical rowids**, that are based on the table's primary key. Oracle uses these logical rowids for the construction of secondary indexes on index-organized tables.
- Each logical rowid used in a secondary index can include a **physical guess**, which identifies the block location of the row in the index-organized table at the time the guess was made; that is, when the secondary index was created or rebuilt.

Macneil Fernandes©2005

- Oracle can use guesses to probe into the leaf block directly, bypassing the full key search. This ensures that rowid access of nonvolatile index-organized tables gives comparable performance to the physical rowid access of ordinary tables.
- In a volatile table, however, if the guess becomes stale the probe can fail, in which case a primary key search must be performed.

  The values of two logical rowids are considered equal if they have the same primary key values but different guesses.

Guesses in Logical Rowids

When a row's physical location changes, the logical rowid remains valid even if it contains a guess, although the guess could become stale and slow down access to the row. Guess information cannot be updated dynamically. For secondary indexes on index-organized tables, however, you can rebuild the index to obtain fresh guesses. Note that rebuilding a secondary index on an index-organized table involves reading the base table, unlike rebuilding an index on an ordinary table.

Macneil Fernandes©2005

- When you collect index statistics with the `DBMS_STATS` package or `ANALYZE` statement, Oracle checks whether the existing guesses are still valid and records the percentage of stale/valid guesses in the data dictionary. After you rebuild a secondary index (recomputing the guesses), collect index statistics again.
- In general, logical rowids without guesses provide the fastest possible access for a highly volatile table. If a table is static or if the time between getting a rowid and using it is sufficiently short to make row movement unlikely, logical rowids with guesses provide the fastest access.

Macneil Fernandes©2005