Database Design

Sorting Rows

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Construct a query to sort a result set in ascending or descending order
- State the order in which expressions are evaluated and calculated based on the rules of precedence
- Construct a query to order a result set using a column alias
- Construct a query to order a result set for single or multiple columns

Purpose

By nature, most of us need order in our lives. Imagine if each time you had dinner, you had to look in every kitchen drawer or cabinet to find a knife and a fork? Ordering, grouping, and sorting makes finding things easier.

Biologists group animals in phyla, astronomers order brightness of stars by magnitude, and Java programmers organize code in classes. For database design, business functions are ordered by entities and attributes; in database information, SQL uses the ORDER BY clause.

Purpose (cont.)

Being able to sort results is a convenient feature in SQL and enables programmers to display information in many different ways.

ORDER BY Clause

Information sorted in ascending order is familiar to most of us. It's what makes looking up a number in a phone book, finding a word in the dictionary, or locating a house by its street address relatively easy.

SQL uses the ORDER BY clause to order data. The ORDER BY clause can specify several ways in which to order rows returned in a query.

ORDER BY Clause (cont.)

The following DJs on Demand example uses the ORDER BY clause to order the years in ascending (default) order. Note: The ORDER BY clause must be the last clause of the SQL statement.

```
SELECT title, year
FROM  d_cds
ORDER BY year;
```

TITLE	YEAR
The Celebrants Live in Concert	1997
Graduation Songbook	1998
Songs from My Childhood	1999
Carpe Diem	2000
Party Music for All Occasions	2000
Here Comes the Bride	2001
Back to the Shire	2002
Whirled Peas	2004

ORDER BY Clause (cont.)

- The default sort order is ascending.
- Numeric values are displayed lowest to highest.
- Date values are displayed with the earliest value first.
- Character values are displayed in alphabetical order.
- Null values are displayed last in ascending order and first in descending order.
- NULLS FIRST Specifies that NULL values should be returned before non-NULL values. NULLS LAST Specifies that NULL values should be returned after non-NULL values.

Sorting in Descending Order

You can reverse the default order in the ORDER BY clause to descending order by specifying the DESC keyword after the column name in the ORDER BY clause.

```
SELECT title, year
FROM  d_cds
ORDER BY year DESC;
```

TITLE	YEAR
Whirled Peas	2004
Back to the Shire	2002
Here Comes the Bride	2001
Party Music for All Occasions	2000
Carpe Diem	2000
Songs from My Childhood	1999
Graduation Songbook	1998
The Celebrants Live in Concert	1997

How would you order the following dates in descending order? 22-MAY-1985, null, 10-JAN-2004, 17-NOV-1955, 21-DEC-1998

Using Column Aliases

You can order data by using a column alias. The alias used in the SELECT statement is referenced in the ORDER BY clause.

```
SELECT title, year AS "Recording
Date"
FROM  d_cds
ORDER BY "Recording Date";
```

TITLE	RECORDING DATE
The Celebrants Live in Concert	1997
Graduation Songbook	1998
Songs from My Childhood	1999
Carpe Diem	2000
Party Music for All Occasions	2000
Here Comes the Bride	2001
Back to the Shire	2002
Whirled Peas	2004

Sorting with Other Columns

It is also possible to use the ORDER BY clause to order output by a column that is not listed in the SELECT clause. In the following example, the data is sorted by the last_name column even though this column is not listed in the SELECT statement.

```
SELECT employee_id, first_name
FROM employees
WHERE employee_id < 105
ORDER BY last_name;
```

EMPLOYEE_ID	FIRST_NAME
102	Lex
104	Bruce
103	Alexander
100	Steven
101	Neena

Order of Execution

The order of execution of a SELECT statement is as follows:

1. FROM clause: locates the table that contains the data
2. WHERE clause: restricts the rows to be returned
3. SELECT clause: selects from the reduced data set the columns requested
4. ORDER BY clause: orders the result set

Sorting with Multiple Columns

It is also possible to sort query results by more than one column. In fact, there is no limit on how many columns you can add to the ORDER BY clause.

Sorting with Multiple Columns (cont.)

An example of sorting results by more than one column would be first sorting all the students in the school by teacher, then by class, then by grade level, and then by last name to generate class rosters for each teacher.

```
SELECT title, year
FROM   d_cds
ORDER BY title, year;
```

TITLE	YEAR
Back to the Shire	2002
Carpe Diem	2000
Graduation Songbook	1998
Here Comes the Bride	2001
Party Music for All Occasions	2000
Songs from My Childhood	1999
The Celebrants Live in Concert	1997
Whirled Peas	2004

Sorting with Multiple Columns (cont.)

To create an ORDER BY clause to sort by multiple columns, specify the columns to be returned and separate the column names using commas. If you want to reverse the sort order of a column, add DESC after its name.

```
SELECT title, year
FROM  d_cds
ORDER BY year DESC, title;
```

TITLE	YEAR
The Celebrants Live in Concert	1997
Graduation Songbook	1998
Songs from My Childhood	1999
Party Music for All Occasions	2000
Carpe Diem	2000
Here Comes the Bride	2001
Back to the Shire	2002

Summary

In this lesson, you should have learned how to:

- Construct a query to sort a result set in ascending or descending order
- State the order in which expressions are evaluated and calculated based on the rules of precedence
- Construct a query to order a result set using a column alias
- Construct a query to order a result set for single or multiple columns