SQL-3

Joining multiple tables

Selecting data from multiple tables

- when data from more than one table in the database is required, a *join*
  condition is used
- rows in one table can be joined to rows in another table according to common values existing in corresponding columns, usually primary and foreign key columns

  table1.column1 = table2.column2

  - is the condition that joins (or relates) the tables together
  - this is know as a *Natural Join* – these are the most common join type in SQl
  - Extension reading: see text "Section 3.4 Relational Set Operators" for an explanation of the underlying relational operations in joins

Selecting data from multiple tables

SElECT table1.column, table2.column

FROM table1, table2

WHERE table1.column1 = table2.column2;

- suggest precede all column names with the table name for clarity
- if the same column name appears in more than one table, the column name **must** be prefixed with the table name
  - If not the column name would be regarded as "ambiguous" (having more than one meaning)

Table Aliases

- qualifying column names with table names can be very time consuming particularly if table names are lengthy;

  can use table *aliases* instead of table names

```
SElECT      d.deptno, d.deptname, e.empname, e.empjob
FROM        department d, employee e
WHERE       d.deptno = e.deptno
ORDER BY d.deptno, e.empname;
```

Table Aliases

- Oracle Guidelines:
  - can be up to 30 characters in length, but the shorter they are the better
  - should be meaningful
  - is valid only for the current SElECT statement

Selecting data from multiple tables

SElECT deptno, deptname, empname, empjob

From department d, employee e
WHERE d.deptno = e.deptno;

- What is the problem with this statement?

- when a join condition is invalid or omitted completely, the result is *Cartesian product* (or cross product) in which **all** combinations of rows will be displayed
  - all rows in the first table are joined to all rows in the second table

SElECT d.deptno, d.deptname, e.empname, e.empjob

FROM department d, employee e;

- Results in 4*14 = 56 rows (DO NOT 'solve' with
DISTINCT)

Outer Joins

- If a row does not satisfy a join condition, the row will not appear in the query result. For example, in the equijoin condition of EMPlOYEE and DEPARTMENT tables, department HR does not appear because no one works in that department.
- The missing row(s) can be returned if an *outer join* operator is used in the join condition.
- In Oracle the operator is a plus sign enclosed in parentheses (+), and is *placed on the* "*side*" *of the join that is deficient in information*. This operator has the effect of creating one or more null rows, to which one or more rows from the non-deficient table can be joined.

Outer Joins

SEIECT	*table1.column, table2.column*
FROM	*table1, table2*
WHERE	*table1.column = table2.column(+)*;

table1.column = table2.column

is the condition that joins the tables together.

*(+)*is the outer join symbol, which can be placed on either side of the WHERE clause condition, but not on both sides

- To write a query that performs an outer join of tables A and B and returns all rows from A (a left outer join) apply the outer join operator (+) to table B in the join condition of the WHERE clause.
  - For all rows in A that have no matching rows in B, Oracle returns null for any select list expressions containing columns of B.

```
SELECT   d.deptno, d.deptname, e.empname,
FROM     e.empjob department d, employee e
WHERE    d.deptno =
ORDER BY d.deptno
         e.deptno(+)
```

SELF JOINS

- Sometimes you need to join a table to itself. To find the name of each employee's manager, you need to join the EMPlOYEE table to itself, or
- perform a self join.

  example, to find the name of Blake's manager, you need to:

  - Find Blake in the EMPlOYEE table by looking at the EMPNAME column.
  - Find the manager number for Blake by looking at the MGRNO column. Blake's manager number is 7839.
  - Find the name of the manager with EMPNO 7839 by looking at the EMPNAME column. King's employee number is 7839, so King is Blake's manager.

  In this process, you look in the table twice. The first time you look in the table to find Blake in the EMPNAME column and MGRNO value of 7839. The second time you look in the EMPNO column to find 7839 and the EMPNAME column to find King.

```
SELECT worker.empname as "Worker",
       manager.empname as "Manager"
FROM   employee worker, employee
WHERE  manager worker.mgrno =
       manager.empno;
```

JOINS OF THREE OR MORE TABLES

```
SELECT   e.empname, h.deptno, d.deptname, h.histbegindate,
         h.histenddate, h.histmsal
FROM     department d, employee e, history h
WHERE    d.deptno = e.deptno
AND       h.deptno = d.deptno
AND       h.empno = e.empno
ORDER BY e.empname, h.histbegindate DESC, h.histenddate;
```

```
SELECT d.deptname as "Department", manager.empname as"Manager",
       worker.empname  as "Worker"

FROM  employee worker, employee manager, department d
WHERE worker.mgrno = manager.empno
AND   d.deptno = manager.deptno
ORDER BY d.deptname, manager.empname, worker.empname;
```

Alternative ANSI/ISO JOIN syntax

- JOIN ON

```
SELECT e.empname, m.empname AS
manager,
  e.empbdate
FROM employee m
  JOIN
  employee e
  ON e.mgrno = m.empno
WHERE e.empbdate >= '01-JAN-1960'
ORDER BY e.empname;
```

Alternative ANSI/ISO JOIN syntax

- JOIN USING

```
SELECT e.empname, deptno,
d.deptname
FROM employee
  e
  JOIN
  department d
  USING (deptno)
ORDER BY d.deptname, e.empname;
```