SQL DML

Session 6 (CSCI-585)
Example Schema

Dept (DeptNo, Name, Location)

Emp (EmpNo, Name, Job, Sal, Comm, HireDate, Mgr, DeptNo)

Dept

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Number(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Varchar2(14)</td>
</tr>
<tr>
<td>Location</td>
<td>Varchar2(13)</td>
</tr>
</tbody>
</table>

Emp

<table>
<thead>
<tr>
<th>EmpNo</th>
<th>Number(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Varchar2(10)</td>
</tr>
<tr>
<td>Job</td>
<td>Varchar2(9)</td>
</tr>
<tr>
<td>Sal</td>
<td>Number(7,2)</td>
</tr>
<tr>
<td>Comm</td>
<td>Number(7,2)</td>
</tr>
<tr>
<td>HireDate</td>
<td>Date</td>
</tr>
<tr>
<td>Mgr</td>
<td>Number(4)</td>
</tr>
<tr>
<td>DeptNo</td>
<td>Number(2)</td>
</tr>
</tbody>
</table>
## Example Relations

### Emp

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>NAME</th>
<th>JOB</th>
<th>MGR</th>
<th>HIREDATE</th>
<th>SAL</th>
<th>COMM</th>
<th>DEPTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7369</td>
<td>SMITH</td>
<td>CLERK</td>
<td>7902</td>
<td>17-DEC-80</td>
<td>800</td>
<td>20</td>
<td>7499</td>
</tr>
<tr>
<td>7499</td>
<td>ALLEN</td>
<td>SALESMAN</td>
<td>7698</td>
<td>20-FEB-81</td>
<td>1600</td>
<td>300</td>
<td>30</td>
</tr>
<tr>
<td>7521</td>
<td>WARD</td>
<td>SALESMAN</td>
<td>7698</td>
<td>22-FEB-81</td>
<td>1250</td>
<td>500</td>
<td>30</td>
</tr>
<tr>
<td>7566</td>
<td>JONES</td>
<td>MANAGER</td>
<td>7839</td>
<td>02-APR-81</td>
<td>2975</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>7654</td>
<td>MARTIN</td>
<td>SALESMAN</td>
<td>7698</td>
<td>28-SEP-81</td>
<td>1250</td>
<td>1400</td>
<td>30</td>
</tr>
<tr>
<td>7698</td>
<td>BLAKE</td>
<td>MANAGER</td>
<td>7839</td>
<td>01-MAY-81</td>
<td>2850</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>7782</td>
<td>CLARK</td>
<td>MANAGER</td>
<td>7839</td>
<td>09-JUN-81</td>
<td>2450</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>7788</td>
<td>SCOTT</td>
<td>ANALYST</td>
<td>7566</td>
<td>19-APR-87</td>
<td>3000</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7839</td>
<td>KING</td>
<td>PRESIDENT</td>
<td></td>
<td>17-NOV-81</td>
<td>5000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7844</td>
<td>TURNER</td>
<td>SALESMAN</td>
<td>7698</td>
<td>08-SEP-81</td>
<td>1500</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>7876</td>
<td>ADAMS</td>
<td>CLERK</td>
<td>7788</td>
<td>23-MAY-87</td>
<td>1100</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7900</td>
<td>JAMES</td>
<td>CLERK</td>
<td>7698</td>
<td>03-DEC-81</td>
<td>950</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>7902</td>
<td>FORD</td>
<td>ANALYST</td>
<td>7566</td>
<td>03-DEC-81</td>
<td>3000</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7934</td>
<td>MILLER</td>
<td>CLERK</td>
<td>7782</td>
<td>23-JAN-82</td>
<td>1300</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

### Dept

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>NAME</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ACCOUNTING</td>
<td>NEW YORK</td>
</tr>
<tr>
<td>20</td>
<td>RESEARCH</td>
<td>DALLAS</td>
</tr>
<tr>
<td>30</td>
<td>SALES</td>
<td>CHICAGO</td>
</tr>
<tr>
<td>40</td>
<td>OPERATIONS</td>
<td>BOSTON</td>
</tr>
</tbody>
</table>
**INSERT Statement**

To create a tuple in SQL the following ‘Insert’ command is required:

\[
\text{insert into } R \ (attribute_1, attribute_2, \ldots \ attribute_n) \\
\text{values } (value_1, value_2, \ldots \ value_n)
\]

```sql
insert into Emp (empno, name, job, sal, comm, hiredate, mgr, deptno)
values (7839, 'King', 'President', 5000, NULL, '17-Nov-81', NULL, 10)
insert into Emp (empno, name, job, sal, comm, hiredate, mgr, deptno)
values (7698, 'Blake', 'Manager', 1600, NULL, '01-May-81', 7839, 30)
```

The insert order matters in terms of referential integrity constraints!
Group INSERT Statement

To create a set of tuples in SQL the following ‘Insert’ command can be used:

\[
\text{insert into } R (\text{attribute}_1, \text{attribute}_2, \ldots \text{attribute}_n) \\
\text{select } (\text{attribute}_1, \text{attribute}_2, \ldots \text{attribute}_n) \\
\text{from } \text{relation}_1, \text{relation}_2, \ldots \text{relation}_n \\
[\text{where condition-expression}] \\
[\text{group by } \text{attribute}_1, \text{attribute}_2, \ldots \text{attribute}_n] \\
[\text{having condition-expression}] \\
[\text{order by } \text{attribute}_1, \text{attribute}_2, \ldots \text{attribute}_n]
\]

Example: copy details of all employees that work in department 10 from the Emp relation into the DepA relation.

\[
\text{insert into DepA (staffno, name, job, hiredate)} \\
\text{select empno, name, job, hiredate} \\
\text{from Emp} \\
\text{where deptno = 10;}
\]

corresponding attributes have to be of the same type

Each tuple to be inserted has to be unique!
DELETE Statement

To delete a set of tuples in SQL the following ‘Delete’ command is used:

```
delete from R
[where condition-expression]
```

Example: remove details of all employees that work in department 10 from the Emp relation.

```
Delete from Emp
where deptno = 10;
```

If the where clause is omitted then all tuples in the relation will be removed!
UPDATE Statement

To alter a set of tuples in SQL the following ‘Update’ command is used:

```
update R
set attribute_1 = datavalue_1,
    attribute_2 = datavalue_2, ...
    attribute_n = datavalue_n
[where condition-expression]
```

Example: increase the salary of all employees that work in department 10 by 10%.

```
update Emp
set sal = sal * 1.1
where deptno = 10;
```

If the where clause is omitted then all tuples in the relation will be altered!
Modification of the Database – Updates

- Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.
  - Write two update statements:
    ```sql
    update account
    set balance = balance * 1.06
    where balance > 10000
    
    update account
    set balance = balance * 1.05
    where balance <= 10000
    ```
  - The order is important
  - Can be done better using the case statement (next slide)
Case Statement for Conditional Updates

- Same query as before: Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

```sql
update account
set balance = case
    when balance <= 10000 then balance * 1.05
    else balance * 1.06
end
```
ORDER BY

```sql
select emp.name, dept.name 
from Emp, Dept 
where (emp.deptno = dept.deptno) 
    and (emp.deptno = 10 or emp.deptno = 30) 
order by emp.name asc;
```

```sql
select emp.name, dept.name 
from Emp, Dept 
where (emp.deptno = dept.deptno) 
    and (emp.deptno = 10 or emp.deptno = 30) 
order by dept.name desc;
```

```sql
select name 
from Dept 
order by name;
```

Remember in relations neither tuples nor attributes have any intrinsic order!
Views in SQL

- A SQL view is a virtual table that is derived from other base or virtual tables.
- Base tables are defined by CREATE TABLE command and are permanently stored in a database.
- Virtual tables are defined by the CREATE VIEW command to avoid defining complex SQL retrieval expressions repeatedly.
- The definition of a view is stored in the Catalog, but it is not stored in the database itself, so it is computed every time it is used in a query.
Example

- A possible view definition
  ```sql
  CREATE VIEW StudOccupied AS
  SELECT g.StudId, SUM(Hours) AS Occupied
  FROM Grades g, Course p
  WHERE g.CourId = p.CourId AND Grade IS NULL
  GROUP BY StudId;
  ```

- Deleting a view
  ```sql
  DROP VIEW StudOccupied;
  ```
Update of a View

- Create a view of all loan data in `loan` relation, hiding the `amount` attribute
  ```sql
  create view branch-loan as
  select branch-name, loan-number
  from loan
  ```
- Add a new tuple to `branch-loan`
  ```sql
  insert into branch-loan
  values ('Perryridge', 'L-307')
  ```
  This insertion must be represented by the insertion of the tuple
  ```sql
  ('L-307', 'Perryridge', null)
  ```
  into the `loan` relation
- Updates on more complex views are difficult or impossible to translate, and hence are disallowed.
- Most SQL implementations allow updates only on simple views (without aggregates) defined on a single relation