COSC344
Database Theory and Applications

Lecture 6
SQL – Data Manipulation Language (1)

Learning Objectives of Lecture 6

- You should
  - be able to use INSERT, DELETE and UPDATE statement to insert, delete and change data in a table.
  - be able to write the SELECT statement for a given query.
  - understand the use of aggregate functions with GROUP BY clause.

- Source
  - Textbook: Chapter 6.3 – 6.4
  - Oracle documentation

INSERT

- Insert a tuple with values for all attributes
  - The values should be listed in the same order as specified in the CREATE TABLE command

```
INSERT INTO employee

VALUES ('John','B', 'Smith', '123456789', TO_DATE('09-01-1965',
'dd-mm-yyyy'), '731 Fondren, Houston,TX', 'M', 30000,
'333445555', 5);
```

- Insert a tuple with values for a specified list of attributes
  - Must include all attributes with NOT NULL specification and no default value

```
INSERT INTO employee (Fname, Lname, Dno, Ird)

VALUES ('John', 'Smith', 5, '123456789');
```

Must satisfy all integrity constraints specified in CREATE TABLE statement

```
CREATE TABLE employee
  (fname    VARCHAR2(15) NOT NULL,
   minit    CHAR,
   lname    VARCHAR2(15) NOT NULL,
   ird      CHAR(9)      PRIMARY KEY,
   bdate    DATE,
   address  VARCHAR2(30),
   sex      CHAR,
   salary   NUMBER(6),
   superird CHAR(9)
     CONSTRAINT superird_cnst REFERENCES employee(ird) DISABLE,
   dno      INT          NOT NULL   DEFAULT 1
     CONSTRAINT dno_cnst REFERENCES department(dnumber) DISABLE);

ALTER TABLE employee ENABLE CONSTRAINT dno_cnst;
ALTER TABLE employee ENABLE CONSTRAINT superird_cnst;

SQL> INSERT INTO employee (fname, lname, ird, superird, dno)
          VALUES('haibo', 'zhang', '222222222', '111111111', 1);
```

ERROR at line 1:
ORA-02291: integrity constraint (HAIBO.SUPERIRD_CNST) violated - parent key not found

DELETE and UPDATE

- DELETE command removes tuples from a relation

```
DELETE FROM employee
WHERE lname = 'Brown';
```

```
DELETE FROM employee;
```

- UPDATE command modifies attribute values of one or more tuple in a relation

```
UPDATE project
SET plocation='Bellaire', dnum=5
WHERE pnumber=10;
```

```
UPDATE employee
SET salary = salary * 1.1
WHERE Dno=5;
```

Referential Integrity Constraints (Revisit)

- FOREIGN KEY clause: specifies referencing integrity
  - Can be defined by following the attribute directly

```
dnum   INT    NOT NULL    REFERENCES   department(dnumber),
```

  - Can be defined using the FOREIGN KEY clause

```
FOREIGN KEY  (dnum)    REFERENCES   department(dnumber),
```

- Specify actions to deal with integrity violations
  - SET NULL, CASCADE, SET DEFAULT

```
FOREIGN KEY  (superird)    REFERENCES   employee(ird)
                    ON DELETE  SET NULL,
```

  - Oracle does not support cascade update, which can be implemented in other ways.

DELETE and UPDATE may propagate to tuples in other relations if referential triggered actions are specified on the referential integrity constraints.

SELECT

- The basic statement for retrieving data
- The SELECT statement is **not the same as** the SELECT operation in relational algebra
- Basic form of SELECT

```
SELECT <attribute list>
FROM <table list>
WHERE <condition>;
```

- A simple example:

```
SELECT bdate, address
FROM employee
WHERE fname='John' AND minit='B'
AND lname='Smith';
```

$$\Pi_{\text{bdate, address}}(\sigma_{\text{fname='John' AND minit='B' AND lname='smith'}}(\text{EMPLOYEE}))$$

SELECT With Join

- Example: Retrieve the name and address of all employees who work for the 'Research' department.

$\Pi_{\text{fname, lname, address}} (\sigma_{\text{dname='Research'}} (\text{EMPLOYEE} \bowtie_{\textit{dnumber=dno}} \text{DEPARTMENT}))$

  - SQL statement

```
SELECT fname, lname, address
FROM employee, department
WHERE dname='Research' AND dnumber=dno;
```

- Ambiguous Attribute Names
  - Example: Retrieve the name and sex for all employees and the name of their dependents
  - Both employee and dependent have an attribute named 'Sex'.

Solution to Ambiguous Attribute Names

- Qualify the attribute name with the relation name

```
SELECT Fname, Lname, employee.sex, dependent_name
FROM employee, dependent
WHERE ird=eird;
```

- Define alias as alternatives for relation names

```
SELECT Fname, Lname, e.sex, dependent_name
FROM employee e, dependent
WHERE ird=eird;
```

A special Example: Retrieve the employee's first name and last name and the first and last name of his/her supervisor

```
SELECT e.fname, e.lname, s.fname, s.lname
  FROM employee e, employee s
  WHERE e.superird=s.ird;
```

The WHERE Clause

- A missing WHERE clause indicates no condition on tuple selection

```
SELECT ird
FROM employee;
```

- The WHERE clause specifies the Boolean condition which must be true for any retrieved tuple.
- The condition in the WHERE clause can be a complex expression
  - Comparison operators: =, >,>=,<,<=,<>
  - Logical operators: AND, OR, NOT
  - Special operators
    - IN
    - BETWEEN
    - LIKE
    - IS NULL

Special Operators – IN and BETWEEN

- IN operator syntax

<attr> IN (<list of values>)

```
SELECT pname FROM project
WHERE plocation IN ('Bellaire', 'Stafford');
```

```
SELECT pname FROM project
WHERE NOT plocation IN ('Bellaire', 'Stafford');
```

- BETWEEN operator syntax

<attr> BETWEEN <value1> AND <value2>

```
SELECT * FROM employee
WHERE salary BETWEEN 25000 AND 38000;
```

```
(salary>=25000) AND (salary <=38000)
```

Inclusive

Special Operators - LIKE

- Used for substring matches for CHAR or VARCHAR2 attributes
- LIKE syntax

  <attr> LIKE <match string>

  – Underscore (_) matches a single character
  – Percent (%) matches a sequence of any number of characters (including zero characters)
  – If an underscore or % is needed as a literal character in the match string, use the escape character (\) before them.

```
SELECT pname FROM project
WHERE pname LIKE 'P%';
SELECT fname FROM employee
WHERE fname LIKE 'J____';
```

4 underscores

Special Operators - IS NULL

- When NULL is compared to any value, even another NULL, the result is not true or false, but unknown, and Unknown behaves like false
- IS NULL syntax

  <attr> IS [NOT] NULL

```
SELECT fname FROM employee

WHERE superird IS NULL;
```

NEVER use

```
              WHERE superird = NULL
OR          WHERE superird <> NULL
```

Duplicates

- **An important distinction** between SQL and the formal relational model: SQL allows a table to have two or more tuples that are identical.
- An SQL table is
  - not a *set* of tuples
  - a multiset (or bag) of tuples
- SELECT can yield relations with duplicate tuples
- DISTINCT option to remove duplicates
  - SELECT DISTINCT . . .

```
SELECT DISTINCT dnum
FROM project;
```

What to Project

- Specify a list of attributes
- Using asterisk (*) to stand for all the attributes

```
SELECT *
FROM employee;
```

- Aggregate functions
  - Take groups of values from a field and reduce them to a single value.
  - Used like attribute names in the SELECT part
    - SUM(<attr>)
    - MIN(<attr>)
    - MAX(<attr>)
    - AVG(<attr>)
    - COUNT(*)
    - COUNT(DISTINCT <attr>)

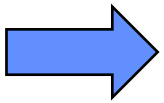
```
SELECT AVG(salary)
FROM employee;
```

  - Generally used with the GROUP BY clause

GROUP BY Clause

- Allows one to define a subset of the values of a particular field and to apply an aggregate function to the subsets
- GROUP BY applies the aggregate function independently to each subset
- Can have multiple attributes in the GROUP BY clause

```
SELECT pno, MAX(hours)
FROM works_on
GROUP BY pno;
```



PNO	MAX(HOURS)
1	32.5
30	30
2	20
20	15
3	40
10	35

HAVING Clause

- May follow a GROUP BY clause and is used to restrict the groups to satisfy certain conditions.

```
SELECT pno, MAX(hours)
FROM works_on
GROUP BY pno
HAVING MAX(hours) > 21;
```

PNO	MAX(HOURS)
1	32.5
30	30
3	40
10	35

- What is the difference between HAVING and WHERE?

ORDER BY Clause

- Orders the output according to the values in one or more selected columns
- The default order is ascending
  - ASC
  - DESC

```
SELECT * FROM works_on
ORDER BY pno, hours;
```

```
SELECT * FROM works_on
ORDER BY pno, hours DESC;
```

- Can also be used with GROUP BY

Summary of SQL Queries

SELECT [ALL | DISTINCT] <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>];

UNION Operator

- Puts multiple queries together and combines their results
- Example - Find all the salespeople and customers located in London

```
SELECT snum,sname
  FROM salespeople
  WHERE city='London'
UNION
SELECT cnum,cname
  FROM customer
  WHERE city='London';

Result:
  1001 Peel
  1004 Motika
  2001 Hoffman
  2006 Clemens
```

UNION Operator (continued)

- No column headings are in the output.
- Duplicates are automatically eliminated.
- UNION ALL does not eliminate duplicates.
- The queries must be *union compatible*
  - Same number of columns
  - Each column must be data type compatible with its corresponding column
- Can follow with an ORDER BY clause to order the results.

Lab for week 4

- The Company database and the Sales database. Needed next week
- Compare the .SQL files to create and populate these databases.
- The Company database has good examples of integrity constraints
- The Sales database is an example of how not to create tables. There are no constraints at all.