COMP163

Database Management Systems

Lecture 8 – Sections 4.1, 4.2, 4.4

SQL: Data Definition and Data Manipulation

SELECT: Syntax Summary

SELECT <attribute and function list>
FROM <table list>
WHERE <condition>
GROUP BY <grouping attributes>
HAVING <group condition>
ORDER BY <attribute list>

required

optional

SELECT: conceptual execution

1. FROM: cross product of tables
2. WHERE: select tuples
3. GROUP BY: group tuples
4. HAVING: filter groups
5. SELECT: project attributes and apply aggregates
6. ORDER BY: sort the tuples

This is not an efficient way to execute the query,
simply a way to define the meaning of the query conceptually.

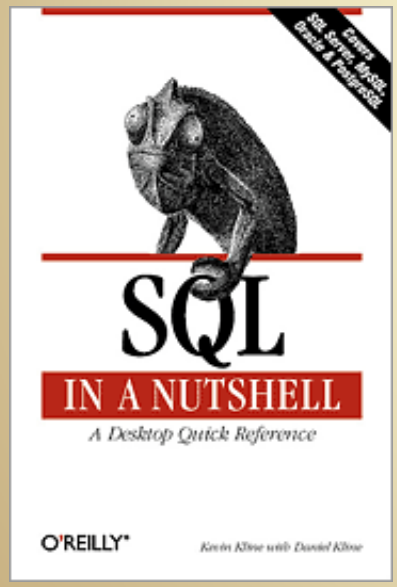
SQL Data Definition and Admin

- Administration:
  - **CREATE DATABASE**
  - CREATE SCHEMA
  - SET ROLE
  - GRANT PRIVILEGES
- Data Definition:
  - **CREATE TABLE**
  - **ALTER TABLE**
  - **DROP TABLE**
  - CREATE VIEW
- Data Modification:
  - INSERT
  - DELETE
  - UPDATE
- Queries:
  - SELECT

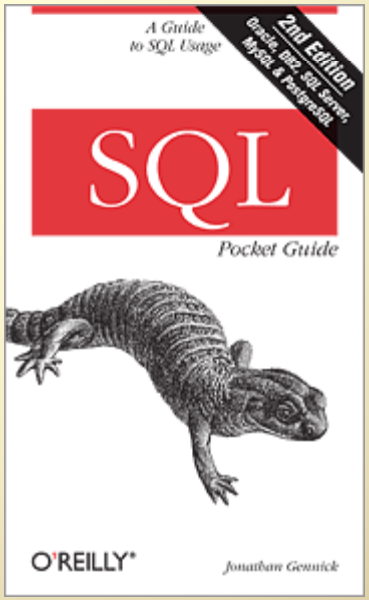
SQL: Suggested References

MySQL reference manual, section 12:

- http://dev.mysql.com/doc/refman/5.6/en/sql-syntax.html



"Nutshell" details syntax variations among Oracle, MS SQL Server and MySQL.



Good for quick syntax checks.

Create Database/Schema

- Specifies a new database schema by giving it a name
- MySQL uses CREATE DATABASE
- Some systems allow you to create multiple schema within a single database
- Often requires root access
  - then grant privileges on the new database/schema to individual users/developers

CREATE TABLE

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their domain
  - INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE DEPARTMENT (
      DNAME            VARCHAR(10)     NOT NULL,
      DNUMBER          INTEGER         NOT NULL,
      MGRSSN           CHAR(9),
      MGRSTARTDATE     CHAR(9)   );
```

CREATE TABLE

- Key attributes can be specified
  via the PRIMARY KEY and UNIQUE phrases
- Entity integrity constraints are specified
  in the FOREIGN KEY phrase.

```
CREATE TABLE DEPT (
  DNAME         VARCHAR(10)      NOT NULL,
  DNUMBER       INTEGER          NOT NULL,
  MGRSSN        CHAR(9),
  MGRSTARTDATE CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP  );
```

DROP TABLE

- Used to remove a relation
  and its definition (schema)

- The relation can no longer be used in queries,
  updates, or any other commands
  since its description no longer exists

- Example:

```
DROP TABLE DEPENDENT;
```

ALTER TABLE

- Used to add an attribute or keys to existing relations
- Examples:

```
ALTER TABLE EMPLOYEE
ADD JOB VARCHAR(12);

alter table StoreStock
add (foreign key (store)
     references Store(storeID));
```

Foreign Keys

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE DEPT (
   DNAME                VARCHAR(10)      NOT NULL,
   DNUMBER              INTEGER          NOT NULL,
   MGRSSN               CHAR(9),
   MGRSTARTDATE         CHAR(9),
   PRIMARY KEY (DNUMBER),
   UNIQUE (DNAME),
   FOREIGN KEY (MGRSSN) REFERENCES EMP
   ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Foreign Keys

```
CREATE TABLE EMP(
  ENAME          VARCHAR(30)     NOT NULL,
  ESSN           CHAR(9),
  BDATE          DATE,
  DNO            INTEGER  DEFAULT 1,
  SUPERSSN       CHAR(9),
  PRIMARY KEY (ESSN),
  FOREIGN KEY (DNO) REFERENCES DEPT
    ON DELETE SET DEFAULT ON UPDATE CASCADE,
  FOREIGN KEY (SUPERSSN) REFERENCES EMP
    ON DELETE SET NULL ON UPDATE CASCADE);
```

Additional Data Types: SQL-99

- **DATE:**
  - Made up of year-month-day in the format yyyy-mm-dd
- **TIME:**
  - Made up of hour:minute:second in the format hh:mm:ss
- **TIME(i):**
  - Made up of hour:minute:second plus i additional digits specifying fractions of a second
  - format is hh:mm:ss:ii...i

Additional Data Types: SQL-99

- **TIMESTAMP:**
  - Has both DATE and TIME components
- **INTERVAL:**
  - Specifies a relative value rather than an absolute value
  - Can be DAY/TIME intervals or YEAR/MONTH intervals
  - Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

Practical Schema Definition

- Develop your schema commands in a text file
  - most DMBSs provide some way to execute commands from a file
  - MySQL: use the SOURCE command
- For initial development,
  simply wipe the tables and recreate them
  - include DROP TABLE commands
  - prevents need for extensive use of ALTER TABLE
- CREATE TABLE order is important
  when there are foreign keys.
  - referenced table must be defined before referencing table
  - DROP order is reversed
  - Circular FKs are more easily handled with ALTER TABLE

SQL Data Modification

- Administration:
  - CREATE DATABASE
  - CREATE SCHEMA
  - SET ROLE
  - GRANT
- Data Definition:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
  - CREATE VIEW

- Modifications:
  - **INSERT**
  - **DELETE**
  - **UPDATE**
- Queries:
  - SELECT

INSERT

- In its simplest form,
  it is used to add one or more tuples to a relation

- Attribute values should be listed in the same order
  that the attributes were specified
  in the **CREATE TABLE** command

INSERT

insert into <tablename> [(column {, column})]
values (expression {, expression})
{,(expression {, expression})}

insert into <tablename> [(column {, column})]
<select-statement>

INSERT

```
insert into STORE (ID, PHONE, ZIPCODE)
        values (95, '456-1221', '95209')

insert into STOCK_ITEM
        values
        (1234, 'cat food', $3.49),
        (9876, ‘dog food', $8.69)
```

INSERT

- To reorder attributes or omit attributes, list attribute names explicitly

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
        VALUES ('Richard', 'Marini', '653298653')
```

INSERT

```
create table cust95204 (
        name varchar(50),
        phone char(10),
        constraint c_pk primary key (name, phone))

insert into cust95204
        select name, phone
        from customer
        where zipcode = '95204'
```

inserts result
of a query
into a table

DELETE

delete from <tablename>
[where <search_condition>]

DELETE

```
delete from STOCK_ITEM
   where DESCRIPTION = 'Toothpaste'

delete from PURCHASE        (integrity constraint violation!)
   where DATE < '01/01/00'

delete from PURCHASE_ITEM          (deletes all tuples)

delete from Customer
   where card_no not in
   (select card_no from Purchase)
```

DELETE

- Removes tuples from a relation
  - Includes a WHERE-clause to select the tuples to be deleted
  - Referential integrity should be enforced
  - Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
  - A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
  - The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

UPDATE

update <tablename>
  set <column = <expression>
    { , <column = <expression> }
  [where <search_condition>]

UPDATE

```
update STOCK_ITEM
   set base_price = base_price*1.1

update STOCK_ITEM
   set base_price = '$3.99‘
   where description = 'ice cream'

update STORE
   set MANAGER='Julliet', PHONE='7877-9870‘
   where ID = 22
```

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

UPDATE

- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
UPDATE PROJECT
   SET         PLOCATION = 'Bellaire',
               DNUM = 5
   WHERE       PNUMBER=10
```

UPDATE

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
UPDATE      EMPLOYEE
   SET      SALARY = SALARY *1.1
   WHERE    DNO  IN (SELECT    DNUMBER
                 FROM  DEPARTMENT
                 WHERE   DNAME='Research')
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
  - The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
  - The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification