

## **STRUCTURED QUERY LANGUAGE (SQL) (05/05/08)**

**RELATIONAL DATABASE MODEL – DR. E. F. “TED” CODD 1970**

**ORACLE, INGRES, SYSTEM/R USE(D) SQL**

**THE ANSI-SQL GROUP HAS PUBLISHED THREE STANDARDS OVER THE YEARS:**

- **SQL89 (SQL1)**
- **SQL92 (SQL2)**
- **SQL99 (SQL3)**

**SQL IS A DECLARATIVE (VS. PROCEDURAL) LANGUAGE**

- **ISSUE THE QUERY TO THE DATABASE**
- **PARSED FOR SYNTAX**
- **OPTIMIZED**
- **EXECUTED**
- **ANSWER SET IS RETURNED**

**SQL CAN BE AD HOC OR EMBEDDED IN A PROGRAM**

- **AD HOC – QUERY IS ISSUED AND RETURNS AN ANSWER SET**
- **EMBEDDED – QUERY REPLACES FILE I/O – REQUIRES TRANSITION FROM SET TO RECORD PROCESSING**

**DOCUMENTATION FOR MYSQL** <http://dev.mysql.com/doc/refman/5.0/en/>  
<http://dev.mysql.com/doc/refman/5.0/en/>

## DDL & DML STATEMENTS

### DROP TABLE

```
DROP TABLE [IF EXISTS] <tablename>
           [RESTRICT | CASCADE];
```

REMOVES THE TABLE FROM THE DATABASE

RESTRICT & CASCADE ARE CURRENTLY INOPERABLE  
BUT REFER TO THE HANDLING OF REFERENTIAL  
INTEGRITY CONSTRAINTS AT THE TABLE LEVEL

### CREATE TABLE

THE CREATE STATEMENT CREATES A TABLE IN THE  
DATABASE. IT DEFINES THE ATTRIBUTES OF THE TABLE AND  
ESTABLISHES THE PRIMARY AND FOREIGN KEYS.

```
CREATE TABLE [IF NOT EXISTS] <tablename>
           (COL-DEF1,
            COL-DEF2,
            .....,
            [TABLE-CONSTRAINT1], [TABLE-CONSTRAINT2, ...]
           );
```

```
COL-DEF      <COLUMN-NAME>
              <DATA-TYPE>
              [COLUMN-CONSTRAINTS]
```

## DATA-TYPE

- **CHAR(M) – FIXED LENGTH STRING RIGHT PADDED W/SPACES, RANGE 0 - 255**
- **VARCHAR(M) – VARIABLE LENGTH STRING RANGE 0 – 65,532**
- **TEXT[(M)] – A TEXT COLUMN\***
- **BLOB[(M)] – A BINARY LARGE OBJECT\***

**\*THESE ALSO COME IN TINY,  
MEDIUM, AND LONG**

- **ENUM('V1','V2', ...) – ENUMERATED TYPE, ONLY ONE VALUE ALLOWED**
- **SET('V1','V2', ...) – SET TYPE, 0 OR MORE VALUES FROM THE SET**
- **DATE, TIMESTAMP, DATETIME**
- **INT[(M)] – INTEGER  
(TYPES – TINYINT, SMALLINT, MEDIUM, BIGINT)**
- **BOOL, BOOLEAN – TINYINT(1) 0=FALSE**
- **FLOAT(M[, D]) – FLOATING POINT**
- **DOUBLE[(M,D)], DOUBLE PRECISION[(M,D)]**
- **DECIMAL(M[, D]) – FIXED POINT  
M = DECIMAL DIGITS,  
D = NUMBER OF DIGITS AFTER THE DECIMAL  
(SYNONYMS - DEC, NUMERIC, FIXED)**

## COLUMN-CONSTRAINTS

- [NOT] NULL
- UNIQUE [KEY]
- [PRIMARY] KEY
- [CONSTRAINT <NAME>] FOREIGN KEY  
REFERENCES <TABLE>[<COLUMN>]
- CHECK (<CONDITION>)
- DEFAULT <VALUE>

## TABLE-CONSTRAINTS

### MULTIPART PRIMARY KEYS

**PRIMARY KEY** (column-name1, column-name2, ...)

### MULTIPART FOREIGN KEYS

**FOREIGN KEY** (col\_name, ...)  
REFERENCES tbl\_name (col\_name, ...)  
[ON {DELETE | UPDATE}  
{RESTRICT | CASCADE | SET NULL | NO  
ACTION}]

### MULTIPART INDEXES

[CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]  
[index\_name] [index\_type] (index\_col\_name,...)  
[index\_type]

{FULLTEXT|SPATIAL} [INDEX|KEY]  
[index\_name] (index\_col\_name,...) [index\_type]

**index\_col\_name:**  
col\_name [(length)] [ASC | DESC]

**index\_type:**  
USING {BTREE | HASH | RTREE}

## **INSERTING ROWS**

```
INSERT INTO <table> [(<column> {, <column>})]  
VALUES(<expr> {, <expr>});
```

**ADDS A NEW ROW TO THE TABLE**

## **DELETING ROWS**

```
DELETE FROM <table>  
[where <where-condition>];
```

**DELETES ALL ROWS THAT SATISFY THE WHERE  
CONDITION. NO CONDITION DELETES ALL ROWS.**

## **UPDATING ROWS**

```
UPDATE <table>  
SET <COLUMN> = <EXPR>  
    {, <COLUMN> = <EXPR>}  
[where <where-condition>];
```

**SETS THE INDICATED COLUMNS TO THE VALUE OF THE  
EXPRESSION FOR ALL ROWS THAT SATISFY THE  
WHERE CONDITION**

## UPDATE EXAMPLE:

Script started on Mon Apr 03 09:24:47 2006

ucs-jne> cat updateDemo.sql

```
select * from class_offered
  limit 10;
alter table class_offered
  add (enrolled int);
update class_offered co
  set co.enrolled = (select count(*)
                    from class_taken ct
                    where co.crs_id = ct.crs_id
                      and co.sec_no = ct.sec_no
                      and co.term   = ct.term);
select * from class_offered
  limit 10;
ucs-jne> mysql
Enter password:
```

mysql> source updateDemo.sql

```
+-----+-----+-----+-----+
| crs_id | sec_no | term  | fac_id  |
+-----+-----+-----+-----+
| CMPS415 |      1 | 20001 | 583426154 |
| CMPS561 |      3 | 19992 | 525252521 |
| CMPS455 |      1 | 19992 | 512548693 |
| CMPS460 |      1 | 20001 | 505050501 |
| CMPS516 |      1 | 19982 | 555555551 |
| CMPS455 |      2 | 20001 | 512548693 |
| CMPS561 |      2 | 19992 | 525252521 |
| CMPS261 |      1 | 19991 | 535353531 |
| CMPS550 |      1 | 19971 | 545454541 |
| CMPS405 |      1 | 19992 | 596525105 |
+-----+-----+-----+-----+
```

10 rows in set (0.00 sec)

Query OK, 93 rows affected (0.01 sec)

Records: 93 Duplicates: 0 Warnings: 0

Query OK, 93 rows affected (0.01 sec)

Rows matched: 93 Changed: 93 Warnings: 0

```
+-----+-----+-----+-----+-----+
| crs_id | sec_no | term  | fac_id  | enrolled |
+-----+-----+-----+-----+-----+
| CMPS415 |      1 | 20001 | 583426154 |      9 |
| CMPS561 |      3 | 19992 | 525252521 |      1 |
| CMPS455 |      1 | 19992 | 512548693 |      1 |
| CMPS460 |      1 | 20001 | 505050501 |      2 |
| CMPS516 |      1 | 19982 | 555555551 |      2 |
| CMPS455 |      2 | 20001 | 512548693 |      3 |
| CMPS561 |      2 | 19992 | 525252521 |      1 |
| CMPS261 |      1 | 19991 | 535353531 |     10 |
| CMPS550 |      1 | 19971 | 545454541 |      1 |
| CMPS405 |      1 | 19992 | 596525105 |      1 |
```

```
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

```
mysql> exit
Bye
ucs-jne> ^D
script done on Mon Apr 03 09:28:04 2006
```

## CREATE EXAMPLE:

```
Script started on Wed Apr 05 07:49:54 2006
ucs-jne> cat createDemo.sql
drop table if exists employee;
#
create table employee
(  SSN          varchar(9) primary key,
   lname       varchar(12) not null,
   fname      varchar(9)  not null,
   age        int(3),
   supervisor_SSN varchar(9),
   start_date  date default '0000-00-00',
   constraint good_age
       check(age >= 0 and age <=120)
)
engine=InnoDB;
#
describe employee;
#
insert into employee
    values('111111111','Etheredge','Jim',35,'222222222','2006-04-03');
#
insert into employee
    values('222222222','Jones','Janet',61,NULL,'1983-02-01');
#
alter table employee add
    constraint supvsr
        foreign key (supervisor_SSN) references employee(SSN);
#
insert into employee
    values('333333333','Kid','New',16,'444444444',NULL);
#
insert into employee
    values('444444444','Smith','Ralph',161,NULL,NULL);
#
insert into employee (SSN, lname, fname, age, supervisor_SSN)
    values('555555555','Landry','Alice',23,NULL);
#
select * from employee;
```

```
ucs-jne> mysql
Enter password:
mysql> source creatDemo.sql
Query OK, 0 rows affected (0.01 sec)
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| SSN            | varchar(9)    | NO   | PRI |          |       |
| lname         | varchar(12)   | NO   |     |          |       |
| fname        | varchar(9)    | NO   |     |          |       |
| age           | int(3)        | YES  |     |          |       |
| supervisor_SSN | varchar(9)    | YES  |     |          |       |
| start_date    | date          | YES  |     | 0000-00-00 |       |
+-----+-----+-----+-----+-----+-----+
```

```
6 rows in set (0.01 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 2 rows affected (0.02 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
constraint fails (`460_jne/employee`, CONSTRAINT `supvsr` FOREIGN KEY
(`supervisor_SSN`) REFERENCES `employee` (`SSN`))
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
+-----+-----+-----+-----+-----+-----+
| SSN          | lname        | fname | age | supervisor_SSN | start_date |
+-----+-----+-----+-----+-----+-----+
| 1111111111 | Etheredge   | Jim   | 35  | 2222222222     | 2006-04-03 |
| 2222222222 | Jones       | Janet | 61  | NULL           | 1983-02-01 |
| 4444444444 | Smith       | Ralph | 161 | NULL           | NULL        |
| 5555555555 | Landry      | Alice | 23  | NULL           | 0000-00-00 |
+-----+-----+-----+-----+-----+-----+
```

```
4 rows in set (0.00 sec)
```

```
mysql> exit
script done on Wed Apr 05 07:50:40 2006
```

## CREATE [OR REPLACE] VIEW

<viewname> (column name, ...)

AS subquery

;

### EXAMPLE:

```
Script started on Wed Apr 05 08:39:01 2006
ucs-jne> cat viewDemo.sql
drop view if exists jd_classes;
#
create view jd_classes (jd_crs, jd_sect,
                        jd_term, jd_fid, jd_fname)
  as (select co.crs_id,
            co.sec_no,
            co.term,
            f.fac_id,
            f.fac_name
      from faculty f, class_offered co
      where f.fac_id = co.fac_id
            and (co.crs_id like '____1%'
                 or
                 co.crs_id like '____2%')
    );
#
select * from jd_classes;
# Add a new faculty member
insert into faculty values('000123456', 'Magdy Bayoumi');
# Add a new class
insert into class_offered
      values('CMPS150',1,20071,'000123456',NULL);
# Show the added jd class
select * from jd_classes where jd_crs = 'CMPS150';
# Clean up
delete from faculty where fac_id = '000123456';
delete from class_offered where term = 20071;
```

```
ucs-jne> mysql
Enter password:
mysql> source viewDemo.sql
Query OK, 0 rows affected (0.00 sec)
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
+-----+-----+-----+-----+-----+
| jd_crs | jd_sect | jd_term | jd_fid   | jd_fname   |
+-----+-----+-----+-----+-----+
| CMPS261 |      1 |  19991 | 535353531 | Bill Manaris |
| CMPS260 |      1 |  19991 | 515151511 | Ursula Jackson |
| CMPS150 |      1 |  19991 | 515151511 | Ursula Jackson |
| CMPS260 |      3 |  19982 | 515151511 | Ursula Jackson |
| CMPS261 |      3 |  19983 | 535353531 | Bill Manaris |
| CMPS150 |      3 |  19973 | 515151511 | Ursula Jackson |
| CMPS260 |      1 |  19992 | 515151511 | Ursula Jackson |
| CMPS260 |      2 |  19993 | 515151511 | Ursula Jackson |
| CMPS260 |      2 |  20001 | 515151511 | Ursula Jackson |
| CMPS150 |      2 |  20001 | 515151511 | Ursula Jackson |
| CMPS261 |      2 |  20001 | 535353531 | Bill Manaris |
+-----+-----+-----+-----+-----+
```

```
11 rows in set (0.01 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
+-----+-----+-----+-----+-----+
| jd_crs | jd_sect | jd_term | jd_fid   | jd_fname   |
+-----+-----+-----+-----+-----+
| CMPS150 |      1 |  19991 | 515151511 | Ursula Jackson |
| CMPS150 |      1 |  20071 | 000123456 | Magdy Bayoumi |
| CMPS150 |      2 |  20001 | 515151511 | Ursula Jackson |
| CMPS150 |      3 |  19973 | 515151511 | Ursula Jackson |
+-----+-----+-----+-----+-----+
```

```
4 rows in set (0.00 sec)
```

```
Query OK, 1 row affected (0.01 sec)
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> exit
script done on Wed Apr 05 08:39:47 2006
```

## DATA RETRIEVAL (SELECT STATEMENTS)

<b>SELECT</b>	<b>{ attributes }</b>	$\pi$
<b>FROM</b>	<b>{ relations }</b>	<b>x</b>
<b>WHERE</b>	<b>{ condition }</b>	$\sigma$ $\cup$ $\cap$ $ $ $\div$

### EXAMPLES :-

$\pi_{A,B} (R) \Rightarrow$  **SELECT DISTINCT A, B  
FROM R;**

$\sigma_{C='x'} (R) \Rightarrow$  **SELECT \*  
FROM R  
WHERE C = 'x' ;**

$\pi_{A,B} (\sigma_{C='x'} (R)) \Rightarrow$  **SELECT DISTINCT A, B  
FROM R  
WHERE C = 'x' ;**

$R \times S \Rightarrow$  **SELECT \*  
FROM R, S ;**

$R \cup S \Rightarrow$  **(SELECT R.c, R.d  
FROM R )  
UNION  
(SELECT \*  
FROM S ) ;**

$R - S \Rightarrow$  **SELECT \*  
FROM R  
WHERE NOT EXISTS  
(SELECT \*  
FROM S  
WHERE R.KEY = S.KEY ) ;**

$R \cap S \Rightarrow$  **SELECT \*  
FROM R  
WHERE EXISTS  
( SELECT \*  
FROM S  
WHERE R.KEY = S.KEY  
);**

**$R - (R - S)$  IS TOUGH WITHOUT THE MINUS OPERATOR**

**SELECT \* FROM R R1 WHERE NOT EXISTS  
(SELECT \* FROM R R2 WHERE  
R1.KEY = R2.KEY AND  
NOT EXISTS  
(SELECT \* FROM S WHERE S.KEY = R2.KEY)  
);**

**SELECT** [ ALL | **DISTINCT** ] { **select\_list** | \* }

**SELECT LIST**

- 1) **ATTRIBUTE NAMES**
- 2) **CONSTANTS**
- 3) **EXPRESSIONS**
- 4) **AGGREGATE FUNCTIONS**

**3) EXPRESSIONS**

**NUMERIC TYPE** + | - | \* | / **NUMERIC TYPE**

**EXAMPLE :-**

**SELECT EMP\_NAME, PAY\_RATE, HRS\_WORKED,  
PAY\_RATE \* HRS\_WORKED**

•  
•  
•

#### 4) AGGREGATE FUNCTIONS

**SUM ( [ DISTINCT ] NUMERIC ITEM )**

**AVG ( [ DISTINCT ] NUMERIC ITEM )**

**MAX ( ATTRIBUTE )**

**MIN ( ATTRIBUTE )**

**COUNT ( \* | [ DISTINCT ] ATTRIBUTE )**

**EXAMPLES :-**

```
SELECT COUNT(DISTINCT STU_ID)
FROM CLASS_TAKEN
WHERE TERM = 20071 ;
```

```
SELECT STU_ID, MAX(CRS_ID),
MIN ( CRS_ID )
FROM CLASS_TAKEN
WHERE GRADE < 'F'
GROUP BY STU_ID ;
```

```

SELECT  { select_list }
        FROM    { table_list };

```

**TABLE\_LIST**

**TABLE NAME [ ALIAS ], ...**

**ALL TABLE NAMES ARE X' d.  
TABLE ALIASES ARE USED TO QUALIFY  
AMBIGUOUS ATTRIBUTE NAMES.**

**EXAMPLE :-**

```

SELECT  DISTINCT D.SSN, D.NAME
FROM    LIKES L1, LIKES L2, DRINKER D
WHERE   L1.SSN = D.SSN AND
        L1.SSN = L2.SSN AND
        L1.BEER <> L2.BEER ;

```

```

SELECT          { select_list }
FROM            { table_list }
WHERE           { condition };

```

### ARITHMETIC OPERATORS

**+, -, \*, /**

### COMPARISON OPERATORS

**=, (!=, <>), >, >=, <, <=**

**ANY**                    { WHERE x = ANY ( SELECT... ) }

**ALL**                    { WHERE x >= ALL ( SELECT... ) }

**IN**                      { (item1, ... , itemn) | (SELECT...) }

**NOT IN**                { != ALL }

**[ NOT | EXISTS**                {EMPTY CHECK}

**[ NOT | LIKE**                { PATTERN MATCH }  
 (CASE INSENSITIVE !)  
 { \_ = ANY 1 CHARACTER }  
 { % = ANY STRING (0 OR MORE) }

**IS [ NOT | NULL**                {NULL CHECK}\*

**BETWEEN**                    { INCLUSIVE }

## **LOGICAL OPERATORS**

**NOT, AND, OR**

## **SET OPERATORS**

**UNION**  
 (THERE IS NO INTERSECT IN MYSQL)  
 (THERE IS NO MINUS IN MYSQL)  
 (THERE IS NO DIVIDE IN MYSQL)

**IN PREDICATE****LIST**

```
SELECT NAME
FROM DRINKER
WHERE FAV_BEER IN ( 'Bud', 'Coors', 'Miller' );
```

**SUBQUERY**

```
SELECT D.NAME
FROM DRINKER D
WHERE D.FAV_BEER IN
      ( SELECT S.BEER
        FROM SERVES S, FREQ F
        WHERE S.BAR = F.BAR
        AND D.SSN = F.SSN );
```

**NULL PREDICATE**

```
SELECT NAME
FROM DRINKER
WHERE FAV_BEER IS NULL ;
```

**BETWEEN**

```
SELECT *
FROM BEER
WHERE PROOF BETWEEN 8 AND 10;
```

**LIKE PREDICATE**

```

SELECT NAME
  FROM CUST
 WHERE NAME LIKE 'Jim%'
        OR NAME LIKE '%Jim%'
        OR NAME LIKE '_a%'
        OR NAME LIKE 'J____%'
        OR NAME LIKE 'J% E%' ;

```

**GROUP BY**

SELECTED TUPLES ARE GROUPED BY LIKE VALUES OF THE ATTRIBUTES SPECIFIED IN THE GROUP BY OPERATOR.

ONLY ONE RESULT ROW IS RETURNED PER GROUP.

SELECTED ITEMS 'SHOULD' BE EITHER "GROUP BY" ITEMS OR AGGREGATE FUNCTIONS.

EXAMPLES :-

```

SELECT  D.SSN, D.NAME, COUNT (*)
  FROM  FREQ F, DRINKER D
 WHERE F.SSN = D.SSN
 GROUP BY D.SSN, D.NAME ;

```

```

SELECT  SSN, AVG(HOURS)
  FROM  EMPLOYEE, WORKS-ON
 WHERE  SSN = EMP-SSN
 AND   HOURS > 0
 GROUP BY SSN ;

```

## HAVING

### SEARCH CONDITION

**EXPRESSION THAT COMPARES GROUP PROPERTIES WITH OTHER GROUP PROPERTIES OR CONSTANTS**

### EXAMPLES :-

```
SELECT  NAME, SUM(HOURS )  
FROM EMP, WORKS-ON  
WHERE  SSN = EMP-SSN  
AND HOURS > 0  
GROUP BY NAME  
HAVING COUNT (PROJ-NAME ) >= 5 ;
```

```
SELECT  S.BAR  
FROM SERVES  
GROUP BY BAR  
HAVING COUNT ( DISTINCT BEER_NAME )  
      >= (SELECT COUNT ( * ) / 2  
          FROM BEER);
```



## **JOINS**

**SELECT \*  
FROM TABLE1, TABLE2;**

**SELECT \*  
FROM TABLE1 INNER JOIN TABLE2;**

**SELECT \*  
FROM TABLE1 LEFT OUTER JOIN TABLE2  
ON TABLE1.COL = TABLE2.COL;**

**UNMATCHED ROWS IN TABLE1 ARE PAIRED WITH NULL  
VALUES FOR TABLE2 COLUMNS.**

**SELECT \*  
FROM TABLE1 RIGHT OUTER JOIN TABLE2  
ON TABLE1.COL = TABLE2.COL;**

**UNMATCHED ROWS IN TABLE2 ARE PAIRED WITH NULL  
VALUES FOR TABLE1 COLUMNS.**

**SELECT \*  
FROM TABLE1 NATURAL JOIN TABLE2  
ON TABLE1.COL = TABLE2.COL;**

**JOIN IS PERFORMED ON MATCHING COLUMNS AND  
REDUNDANT COLUMNS ARE ELIMINATED.**