SQL Approach to Data Definition & Control

- common data sublanguage employed by most relational database management systems (Oracle, Sybase, DB-2, MySQL, SQLServer, Informix, …)

DDL	**DML**	**DCL**
CREATE TABLE/DATABASE	SELECT	GRANT
DROP TABLE/DATABASE	DELETE	REVOKE
ALTER TABLE	UPDATE	
	INSERT	

+
CREATE VIEW
DROP VIEW

Data Definition Language (DDL)

(i) *CREATE / DROP DATABASE*

```
CREATE DATABASE companydb;

DROP DATABASE companydb;
```

```
CREATE DATABASE companydb on DbDsk01
  INITIAL=20, EXTENT=5;
```

(ii) *CREATE / ALTER / DROP TABLE*

```
CREATE TABLE PROJECT
    (   PNAME       VARCHAR(15)     NOT NULL,
        PNUMBER     INT             NOT NULL,
        PLOCATION   VARCHAR(15)     DEFAULT 'Houston',
        DNUM        INT             NOT NULL,
      PRIMARY KEY (PNUMBER),
      UNIQUE (PNAME),
      FOREIGN KEY (DNUM) REFERENCES DEPARTMENT(DNUMBER)  );
```

```
CREATE TABLE PROJECT
    (   PNAME       VARCHAR(15)     NOT NULL,
        PNUMBER     INT             NOT NULL,
        PLOCATION   VARCHAR(15)     DEFAULT 'Houston',
        DNUM        INT             NOT NULL,
       PRIMARY KEY (PNUMBER),
      CONSTRAINT  OneName  UNIQUE (PNAME),
      CONSTRAINT  Controller  FOREIGN KEY (DNUM) REFERENCES
                DEPARTMENT(DNUMBER) ON DELETE CASCADE );
```

- observations:
  - the effect of the CREATE TABLE command is to store a descriptor of the table in the system catalog [the effect of ALTER TABLE is to modify that descriptor and DROP TABLE is to remove the descriptor]
  - MySQL: constraint names are ignored (cannot be dropped in ALTER TABLE statements)
  - MySQL: a single variable-length column, explicit or implicit, causes all columns to be silently converted to variable-length data types

Data Types in MySQL

Numeric	**String**	**Date/Time**
TINYINT	CHAR	DATE ('CCYY-MM-DD')
SMALLINT	VARCHAR	TIME ('hh:mm:ss')
MEDIUMINT	TINYBLOB	DATETIME
INT	BLOB	TIMESTAMP
BIGINT	MEDIUMBLOB	YEAR ('CCYY')
FLOAT	LONGBLOB	
DOUBLE	TINYTEXT	
DECIMAL	TEXT	
	MEDIUMTEXT	
	LONGTEXT	
	ENUM	
	SET	

- variants of the same basic type (e.g. INT family) differ in their storage requirements – and upper/lower limits (e.g. 1, 2, 3, 4, 8 bytes, with values up to 127, 32767, 8388607, 2147483647, …); [actually, these values are for signed numbers, unsigned (positive) numbers can double in size]
- these types may be parameterized for storage and/or display purposes:
  - INT (3) - displays in column width 3
  - DECIMAL (7,2) - displays in "7.2" format
  - CHAR (15) - storage 15 characters, padded
  - VARCHAR (15) - max storage 15 characters, unpadded
- the TEXT & BLOB families are implicitly of variable length
- observations:
  - variable-length types are designed for saving of storage space (!)
  - TEXT/BLOB types are relatively uncommon, as they don't fit easily into a table structure; they are rarely used as conditions in queries; when retrieved within queries – not necessarily as part of the condition – they can prove inefficient; to overcome this, these types are often extracted into a separate table

ALTER TABLE PROJECT

```
ADD  START_DATE   DATE;

ALTER TABLE EMPLOYEE
ADD  POSITION  CHAR(12)  NOT NULL;           { legal? }

ALTER TABLE  PROJECT
DROP  PLOCATION;

ALTER TABLE  EMPLOYEE
MODIFY  ADDRESS  CHAR(50);

DROP TABLE DEPT_LOCATIONS;
```

(iii) *CREATE / DROP INDEX*

```
CREATE INDEX EMPIX ON EMPLOYEE(SSN);

CREATE INDEX WEP ON WORKS_ON(ESSN, PNO);

CREATE UNIQUE INDEX DEPTX ON DEPARTMENT(DNUMBER);

DROP INDEX EMPIX;
```

- observations:
    - a user creates an index because he/she believes that it will be of assistance for commonly-occurring queries; it is the *query optimizer* that chooses whether or not to use the index when evaluating a specific query
    - creating an index effectively directs the storage engine [file system] to employ indexed sequential files for storing a table
    - indexes can be dropped – and subsequently recreated – at times of file volatility

Data Control Language (DCL)

(i) Database Privileges

```
GRANT CONNECT
TO violet
IDENTIFIED BY passwdX

GRANT RESOURCE
TO silver
IDENTIFIED BY TopSecretPsw;

GRANT DBA
TO ruby
IDENTIFIED BY PassWdXX;

GRANT RESOURCE
TO violet;

REVOKE RESOURCE
FROM violet;
```

(ii) Table-Level Permissions

```
GRANT  SELECT
ON  companydb.employee
TO  white;

GRANT SELECT
ON companydb.*
TO brown
WITH GRANT OPTION;

GRANT  SELECT, INSERT
ON companydb.department
TO gray
```

```
WITH GRANT OPTION;

GRANT SELECT, UPDATE (Salary, SuperSSN)
ON companydb.employee
To  ruby, green, black;

REVOKE INSERT
ON companydb.department
FROM ruby;

REVOKE  ALL
ON companydb.*
FROM violet;
```

- observations:
    - there are, in effect, two forms of GRANT & REVOKE: one for user management and one for table management
    - the effect of GRANT is to store privilege/permission information in the system catalog; the effect of REVOKE is to modify or remove that information
    - when a DBMS is first installed, it is usually configured for a single user with DBA privilege level

Views in SQL

- in SQL-based DBMS, it is possible to define *views* (also called *derived* or *virtual tables*) into existing (base) tables

(i) View Creation & Removal

```
CREATE VIEW D5EMPS AS
SELECT  Ssn, Fname, Lname, Address, Salary
FROM Employee
WHERE  Dn = 5;

CREATE VIEW  P5EMPS AS
SELECT Ssn, Fname, Lname, Address, Dno
FROM  Employee, Works_On
WHERE Ssn = Essn
  AND  Pno = 5;

CREATE VIEW D5NAMES AS
SELECT Fname, Lname
FROM D5EMPS;

CREATE VIEW STAFFCOUNTS (Dno, NumStaff) AS
SELECT Dno, COUNT (*)
FROM  EMPLOYEE
GROUP BY Dno;

DROP VIEW D5EMPS
```

- observations:

  - a view can be created by any user provided he/she has permission to execute the associated SELECT statement
  - views are useful both for convenience [they can reorganize the tables of a database into a more useful set of structures] and for security [they can hide information from users]
  - a user can be granted access [usually SELECT permission] on a view, but not on the underlying (base) tables
  - the effect of issuing a CREATE VIEW statement is to (i) have the associated SELECT statement validated, and (ii) record the view definition [the text of the CREATE VIEW statement] in the system catalog; the effect of the DROP VIEW statement is to remove the previously recorded information

(ii) View Use

```
SELECT Fname, Lname
FROM D5EMPS
WHERE Address LIKE '%Houston%'

SELECT NumStaff
FROM STAFFCOUNTS
WHERE Dno = 5;
```

- observation:
  - a user can query the contents of a view in the same way as he/she can query the contents of a base table – in fact, a user may not be aware of what type of table they are accessing

(iii) View Implementation

- observations:

- when a query is submitted against a view (assuming it passes the validity test!), the view definition text is retrieved from the system catalog and is merged with the text of the user query to produce an equivalent query that operates against base tables only – this process is known as *query modification*

User Query:

```
SELECT Fname, Lname
FROM D5EMPS
WHERE Address LIKE '%Houston%'
```

View Definition:

```
SELECT  Ssn, Fname, Lname, Address, Salary
FROM Employee
WHERE  Dn = 5;
```

Final Query:

```
SELECT Fname, Lname
FROM Employee
WHERE  Dn = 5
  AND  Address LIKE '%Houston%'
```