Query

Query is a question or requesting information. Query language is a language which is used to retrieve information from a database.

Query language is divided into two types −
Procedural language
Non-procedural language

Procedural language: Information is retrieved from the database by specifying the sequence of operations to be performed.

For Example − Relational algebra.
Structure Query language (SQL) is based on relational algebra.

- Structured query language(SQL)

DDL Statements

SQL statements are divided into two major categories: data definition language (DDL) and data manipulation language (DML).

- **Data Definition Language (DDL)** statements are used to define the database structure or schema. Some examples:
- * CREATE - to create objects in the database
  * ALTER - alters the structure of the database
  * DROP - delete objects from the database

DML Statements

- **Data Manipulation Language (DML)** statements are used for managing data within schema objects. Some examples:

  * SELECT - retrieve data from the a database
  * INSERT - insert data into a table
  * UPDATE - updates existing data within a table
  * DELETE - deletes all records from a table, the space for the records remain(schema is not removed)

Other DDL Statements

- **Data Control Language (DCL)** statements. Some examples:

  * GRANT - gives user's access privileges to database
  * REVOKE - withdraw access privileges given with the GRANT command

- **Transaction Control (TCL)** used to control transactional processing in a database. A transaction is logical unit of work that comprises one or more SQL statements, usually a group of Data Manipulation Language (DML) statements.

  * COMMIT - save work done
  * SAVEPOINT - identify a point in a transaction to which you can later roll back
  * ROLLBACK - restore database to original since the last COMMIT
  * SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

DDL Statements

- CREATE – database, table, function, trigger
- Create a new table named account that has the following columns with the corresponding constraints:

  user_id – primary key

  username – unique and not null

  password – not null

  email – unique and not null

  created_on – not null

  last_login – null

Create the COMPANY Database

- To create

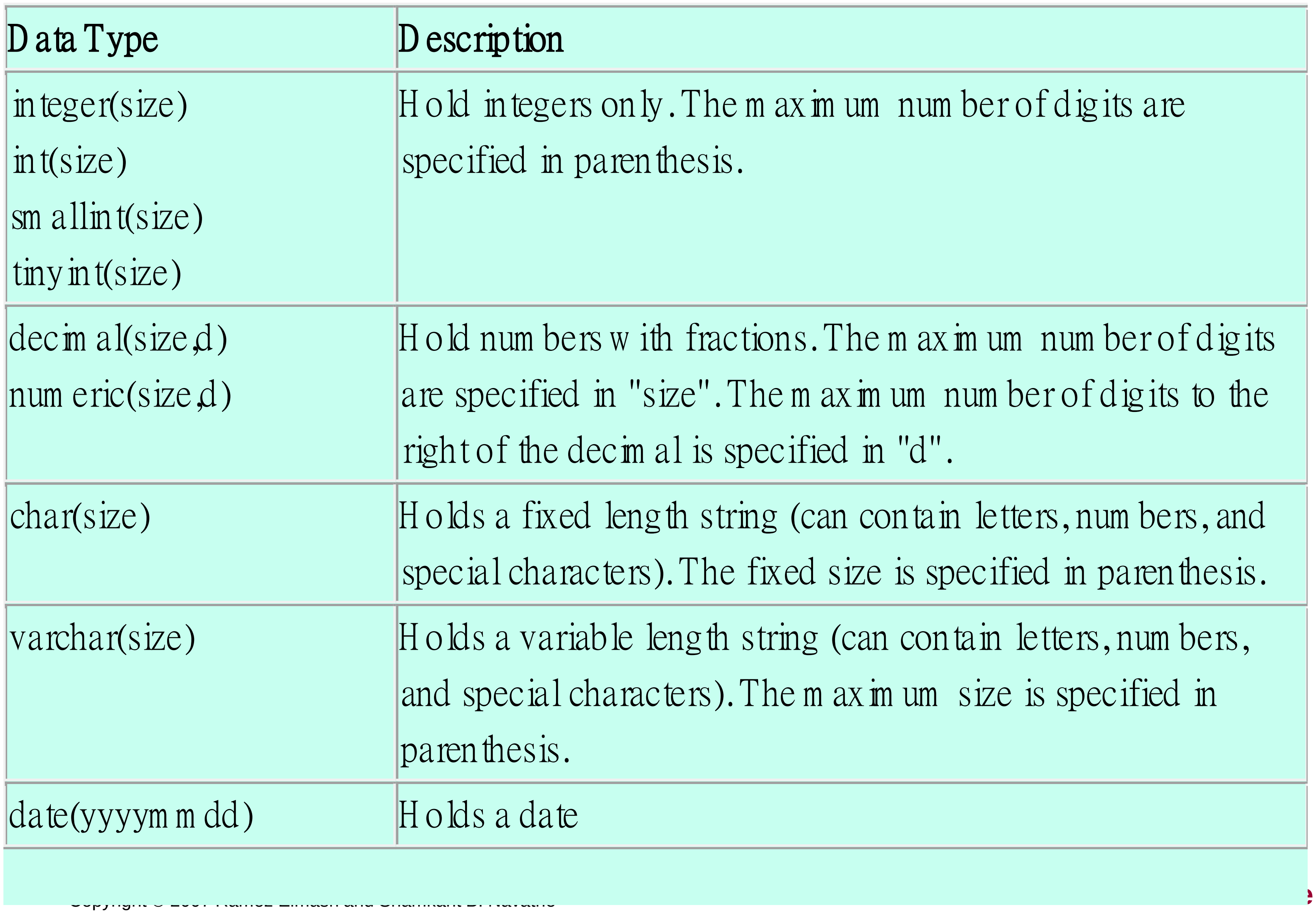
```
create datatbase COMPANY;
```

- Subsequent commands will operate on the COMPANY database by default.

DDL Statements

```
 CREATE TABLE account(
user_id  int PRIMARY KEY,
username VARCHAR (50) UNIQUE NOT NULL,
password VARCHAR (50) NOT NULL,
email VARCHAR (355) UNIQUE NOT NULL,
created_on TIMESTAMP NOT NULL,
last_login TIMESTAMP
);
```

CREATE TABLE: Data Type



Data Type	Description

integer(size)
int(size)
smallint(size)
tinyint(size)
	Hold integers only. The maximum number of digits are specified in parenthesis.

decimal(size,d)
numeric(size,d)
	Hold numbers with fractions. The maximum number of digits are specified in "size". The maximum number of digits to the right of the decimal is specified in "d".

char(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis.
varchar(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis.
date(yyyymmdd)	Holds a date

Additional Data Types

- **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
- **TIMESTAMP:**
  - Has both DATE and TIME components
- Others: Boolean, Float, Double Precision

Primary Keys

```
CREATE TABLE student (
    Number              INTEGER NOT NULL,
    Name                VARCHAR(50),
    Street              VARCHAR(20),
    City                VARCHAR(20),
    PostalCode          CHAR(7),
    Date_of_birth       DATE,
    PRIMARY KEY (Number));
```

- many attributes can have unique constraints but there is only one primary key.

Primary Keys (cont’d)

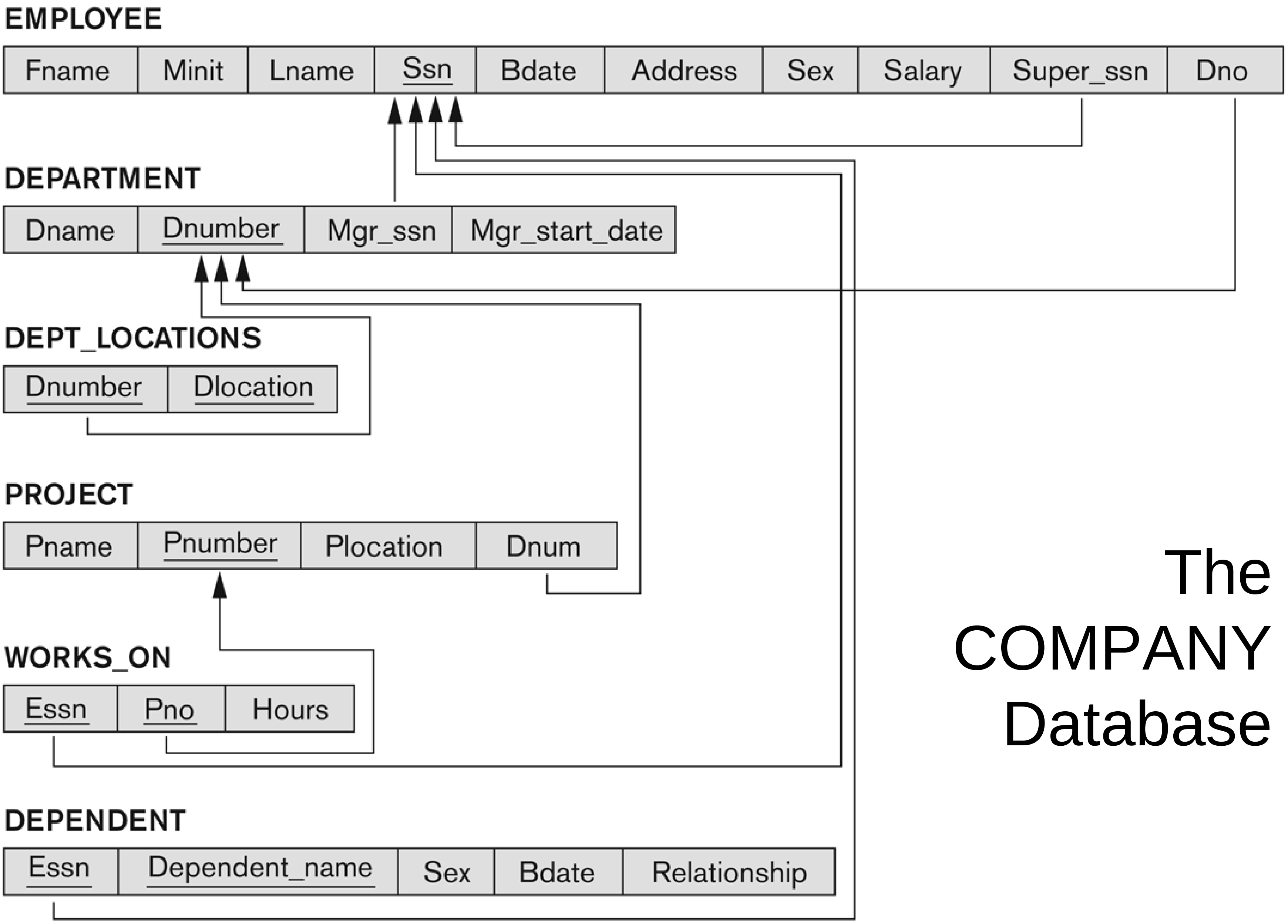
```
CREATE TABLE student (
Number               INTEGER PRIMARY KEY ,
Name                 VARCHAR(50) NOT NULL,
Street               VARCHAR(20),
City                 VARCHAR(20) NOT NULL,
Postal Code          CHAR(7) NOT NULL
Date_of_birth        DATE NOT NULL);
```

- The primary key can be defined immediately after the attribute if it is a single attribute primary key.

Foreign Keys

```
CREATE TABLE Projects(
   loc_code       CHAR(4) ~~primary key~~,
   pname          VARCHAR(30) NOT NULL ~~primary key~~,
   Start_date     Date,
   End_Date       Date,
   dnum(Dnumber)               INTEGER,
   PRIMARY KEY(loc_code,pname), pk with multiple attributes
   FOREIGN KEY (dnum) REFERENCES Department(Dnumber);
   FOREIGN KEY Dnumber REFERENCES Department
```

- The table referenced by a foreign key must have already been created
- If it hasn't been, you can alter the table later to include the foreign key. This solves the "circular reference" problem.



The COMPANY Database

CREATE TABLE

```
CREATE TABLE DEPARTMENT (
    Dname            VARCHAR(10) NOT NULL,
    Dnumber          INTEGER     Default 0,
    Mgr_ssn          CHAR(9),
    Mgr_Sartdate     CHAR(9),
    PRIMARY KEY      (Dnumber),
    UNIQUE           (Dname),
    FOREIGN KEY      (Mgr_ssn)
          REFERENCES EMPLOYEE (Ssn));
```

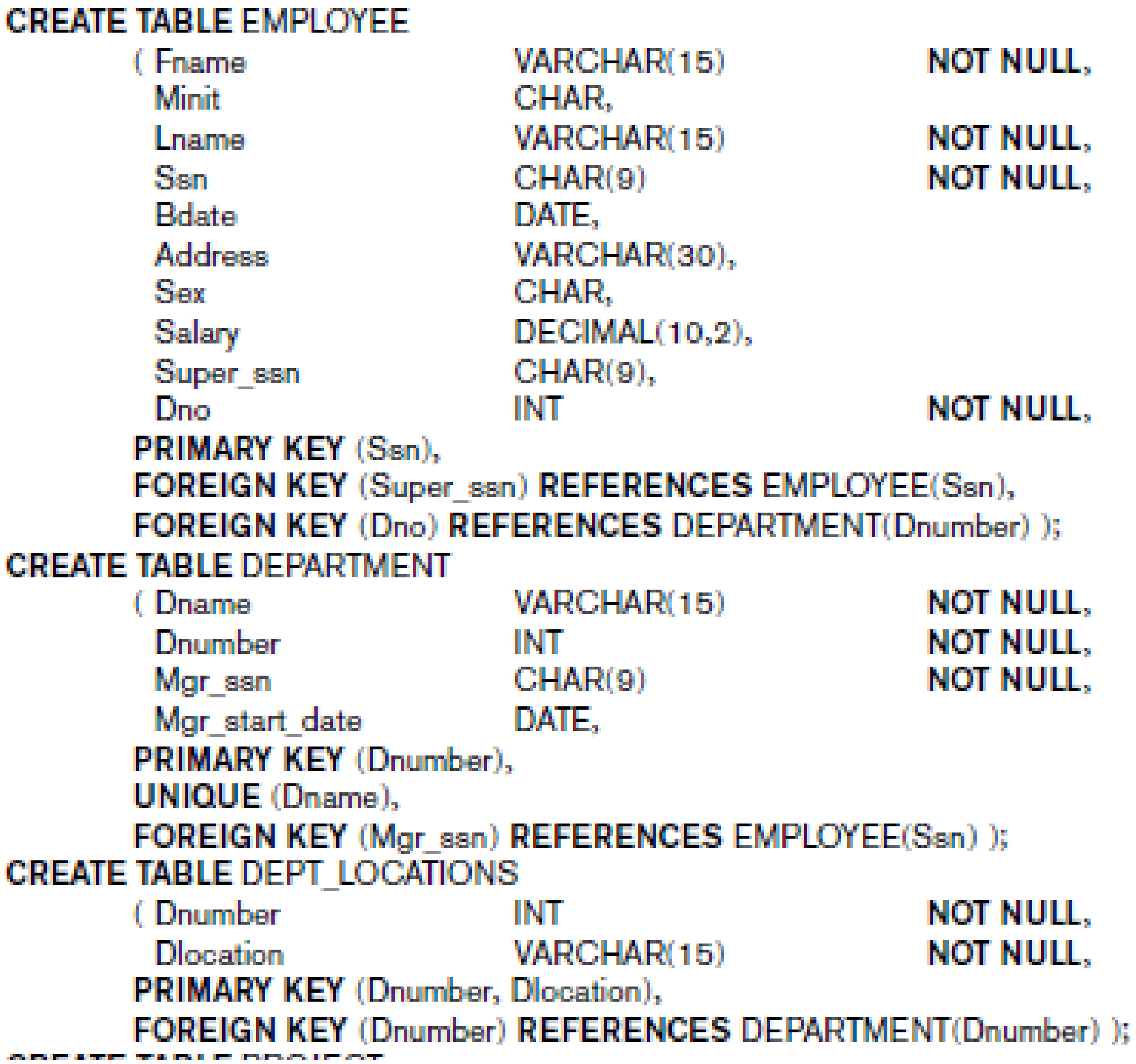
- The “UNIQUE” clause specifies secondary keys.
- `EMPLOYE` has to be created first for the FK `Mgr_ssn` to refer to it.
- How could we have defined the `Dno` FK in `EMPLOYEE`?

Adding the Dno FK to EMPLOYEE

- If "`create table EMPLOYEE`" is issued first, we cannot specify `Dno` as a FK in that `create` command.
- An ALTER command must be used to change the schema of EMPLOYEE, after the "`create table DEPARTMENT`," to add a FK.

```
alter table EMPLOYEE
    add constraint
          foreign key (Dno)
                references DEPARTMENT (Dnumber);
```

CREATE TABLE



```
CREATE TABLE EMPLOYEE
        ( Fname           VARCHAR(15)          NOT NULL,
          Minit           CHAR,
          Lname           VARCHAR(15)          NOT NULL,
          Ssn             CHAR(9)              NOT NULL,
          Bdate           DATE,
          Address         VARCHAR(30),
          Sex             CHAR,
          Salary          DECIMAL(10,2),
          Super_ssn       CHAR(9),
          Dno             INT                  NOT NULL,
        PRIMARY KEY (Ssn),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE DEPARTMENT
        ( Dname           VARCHAR(15)          NOT NULL,
          Dnumber         INT                  NOT NULL,
          Mgr_ssn         CHAR(9)              NOT NULL,
          Mgr_start_date  DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber         INT                  NOT NULL,
          Dlocation       VARCHAR(15)          NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

CREATE TABLE

```
CREATE TABLE PROJECT
        ( Pname             VARCHAR(15)            NOT NULL,
          Pnumber           INT                    NOT NULL,
          Plocation         VARCHAR(15),
          Dnum              INT                    NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn              CHAR(9)                NOT NULL,
          Pno               INT                    NOT NULL,
          Hours             DECIMAL(3,1)           NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn              CHAR(9)                NOT NULL,
          Dependent_name    VARCHAR(15)            NOT NULL,
          Sex               CHAR,
          Bdate             DATE,
          Relationship      VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

The Check Clause

- Used to specify user-defined constraints
- Assume that dept. numbers are from 0 to 99.

```
create table DEPARTMENT (
    …
    Dnumber INTEGER     Default 0
        check (Dnumber>=0 AND Dumber<=99),
    …);


create table DEPARTMENT (
    …
    Dept_create_date date,
    Mgr_start_date   date,
    check (Dept_create_date <= Mgr_start_date)
  );
```

Referential Integrity Options

- **Causes** of referential integrity violation for a foreign key FK (consider the `Mgr_ssn` of `DEPARTMENT`).
  - **On Delete**: when deleting the foreign tuple
    - What to do when deleting the manager tuple in `EMPLOYEE` ?
  - **On Update:** when updating the foreign tuple
    - What to do when updating/changing the SSN of the manager tuple in `EMPLOYEE` is changed ?
- **Actions** when the above two causes occur.
  - **Set Null**: the `Mgr_ssn` is set to null.
  - **Set Default**: the `Mgr_ssn` is set to the default value.
  - **Cascade**: the `Mgr_ssn` is updated accordingly

The Mgr_ssn Example

```
CREATE TABLE DEPARTMENT (
    …
    Mgr_ssn          CHAR(9),
    …
    FOREIGN KEY (Mgr_ssn)
       REFERENCES EMPLOYEE (Ssn)
            ON DELETE SET Null
           ON UPDATE  CASCADE);
```

Another Example

```
Create table EMP(
   …
   ESSN            CHAR(9),
   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);
```

DDL Statements

- Create table
- Alter table
- Add, remove, or rename column.
- Set default value for the column.
- Add CHECK constraint to a column.
- Rename table

DDL Statements

- **ALTER TABLE** emp **ADD COLUMN** active **boolean**;
- **ALTER TABLE** emp **DROP COLUMN** active;
- **ALTER TABLE** emp **RENAME COLUMN** mg_ssn **TO** m_ssn;
- **ALTER TABLE** emp **ALTER COLUMN** m_ssn **SET DEFAULT** 'blank';
- **ALTER TABLE** emp **RENAME TO** employ;

Add Columns to Existing Tables

- To add spouse SSN (`S_ssn`) to `EMPLOYEE`

```
alter table EMPLOYEE add column S_ssn char(9);
```

  - The new attribute will have NULLs in all the tuples of the relation right after the command is executed

- Alternatively, we can set a default value.

```
alter table EMPLOYEE add column Superssn
  char(9)
      default "000000000";
```

Delete Columns from Existing Tables

- To delete column `S_ssn`

```
alter table EMPLOYEE drop column S_ssn;
```

- **Reminder:** changing relation schemas typically indicates ill-executed design phase of the database.

DDL Statements

Drop:

- **DROP TABLE** [**IF EXISTS**] table_name [**CASCADE** | **RESTRICT**];

```
CREATE TABLE author (
author_id INT NOT NULL PRIMARY KEY,
firstname VARCHAR (50),
lastname VARCHAR (50)
);

CREATE TABLE page (
page_id serial PRIMARY KEY,
title VARCHAR (255) NOT NULL,
author_id INT NOT NULL,
FOREIGN KEY (author_id) REFERENCES author (author_id)
);
```

DDL Statements

DROP TABLE [**IF EXISTS**] table_name [**CASCADE** | **RESTRICT**];

- **DROP TABLE IF EXISTS** author;

---Because the constraint on the page table depends on the author table, SQL issues an error message.

- **DROP TABLE** author **CASCADE**;

--SQL removes the author table as well as the constraint in the page table.

--RESTRICT refuses to drop table if there is any object depends on it. SQL uses RESTRICT by default

Miscellaneous Commands

- `show databases;`
  - Show all the databases on the server
- `show tables;`
  - Show all the tables of the present database
- `show columns from table EMPLOYEE;`
- `drop table` *`t_name`*`;`
  - Delete the entire table *t_name*
- `drop database` *`db_name`*`;`
  - Delete the entire database *db_name*

Question?

Consider following schema and write SQL for the followings:

employee (<u>empid</u> , empname, salary, dno(fk));

deptartment(<u>dno</u>, dname);

- Create table for employee and department entity with primary key and foreign key constraints.
- Apply check and not null constraint on salary and empname column respectively.
- Insert at least 2 different rows in employee and department table.
- Add new column 'location' in department table (after creation of table department)

- DML operations in sql

Specifying Updates in SQL

- There are three SQL commands to modify the database; INSERT, DELETE, and UPDATE

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 as the attributes were specified in the CREATE TABLE command

INSERT (cont.)

- Example:

  U1: INSERT INTO EMPLOYEE
  VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
  '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4)

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple
- Attributes with NULL values can be left out
- Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

  U1A: INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
  VALUES ('Richard', 'Marini', '653298653')

INSERT (cont.)

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
- Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

INSERT (cont.)

- Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

```
U3A:    CREATE TABLE  DEPTS_INFO
                 (DEPT_NAME    VARCHAR(10),
                  NO_OF_EMPS   INTEGER,
                  TOTAL_SAL    INTEGER);

U3B:    INSERT INTO   DEPTS_INFO (DEPT_NAME,
                      NO_OF_EMPS, TOTAL_SAL)
        SELECT        DNAME, COUNT (*), SUM (SALARY)
        FROM          DEPARTMENT, EMPLOYEE
        WHERE         DNUMBER=DNO
        GROUP BY      DNAME ;
```

INSERT (cont.)

- Note: The DEPTS_INFO table may not be up-to-date if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations *after* issuing U3B. We have to create a view (see later) to keep such a table up to date.

Inserting values in table

```
create table courses (
  cid int(10)  NOT NULL ,
  cname varchar(30) NOT NULL,
  credit tinyint(2)  NOT NULL,
  croom varchar(10),
  primary key (cid)
);
insert into courses values (1, 'DBMS', 3, '307');
Insert into courses(cid,cname,credit,croom) values(1, 'DBMS',
    3, '307');
Insert into courses(cname,cid,croom,credit) values('DBMS', 1,
    '307‘, 3);
```

DELETE

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- 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
- Referential integrity should be enforced

DELETE (cont.)

- Examples:

```
U4A:     DELETE FROM    EMPLOYEE
         WHERE          LNAME='Brown’

U4B:     DELETE FROM    EMPLOYEE
         WHERE          SSN='123456789’

U4C:     DELETE FROM    EMPLOYEE
         WHERE          DNO  IN
               (SELECT  DNUMBER
               FROM     DEPARTMENT
               WHERE    DNAME='Research')

U4D:     DELETE FROM    EMPLOYEE
```

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 (cont.)

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

```
U5: UPDATE     PROJECT
    SET        PLOCATION = 'Bellaire', DNUM = 5
    WHERE      PNUMBER=10
```

UPDATE (cont.)

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

```
U6:  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

Simple Examples

```
create database courses_students;
use courses_students;
# Table definition
create table courses (
  cid int(10)  NOT NULL auto_increment,
  cname varchar(30) NOT NULL,
  credit tinyint(2)  NOT NULL,
  croom varchar(10),
  primary key (cid)
);
create table students (
  sid int(10) unsigned NOT NULL
      auto_increment,
  sname varchar(30) NOT NULL default '',
  syear tinyint(2) unsigned,
  primary key (sid)
);
create table selected (
  cid int(10) unsigned,
  sid int(10) unsigned,
primary key(cid, sid),
foreign key cid references courses,
foreign key sid references students
);
```

```
# Data
insert into courses values (1, 'DBMS', 3, '307');
insert into courses values (2, 'OS', 3, '406');
insert into courses values (3, 'Algorithm', 3, '307');

insert into students values (1, 'Sam', 4);
insert into students values (2, 'Joe', 3);
insert into students values (3, 'Mary', 3);
insert into students values (4, 'John', 3);

insert into selected values (1, 1);
insert into selected values (1, 2);
insert into selected values (1, 4);
insert into selected values (2, 1);
insert into selected values (2, 3);
insert into selected values (2, 4);
insert into selected values (2, 2);
insert into selected values (3, 2);
insert into selected values (3, 4);
insert into selected values (5, 4);
insert into selected values (6,8);
```