UNIT 3 -SQL

SQL CONTENTS :

- Data Definition and Data Types specifying basic constraints
- Basic retrieval queries in SQL: Insert, Delete and Update
- Additional features of SQL,
- More complex SQL Queries,
- Specifying Constraints as Assertion and Trigger, Views.

SQL Introduction

Standard language for querying and manipulating data

Structured **Q**uery **L**anguage

Many standards out there:

- ANSI SQL
- SQL92 (a.k.a. SQL2)
- SQL99 (a.k.a. SQL3)
- Vendors support various subsets of these
- What we discuss is common to all of them

History:

- 1970–E. F. Codd develops relational database concept
- 1974-1979–System R with Sequel (later SQL) created at IBM Research Lab
- 1979–Oracle markets first relational DB with SQL
- 1981 – SQL/DS first available RDBMS system on DOS/VSE
- Others followed: INGRES (1981), IDM (1982), DG/SGL (1984), Sybase (1986)
- 1986–ANSI SQL standard released
- 1989, 1992, 1999, 2003, 2006, 2008–Major ANSI standard updates
- Current–SQL is supported by most major database vendors

SQL

Data Definition Language (DDL)

Create/alter/delete tables and their attributes

Data Manipulation Language (DML)

Query one or more tables – discussed next !

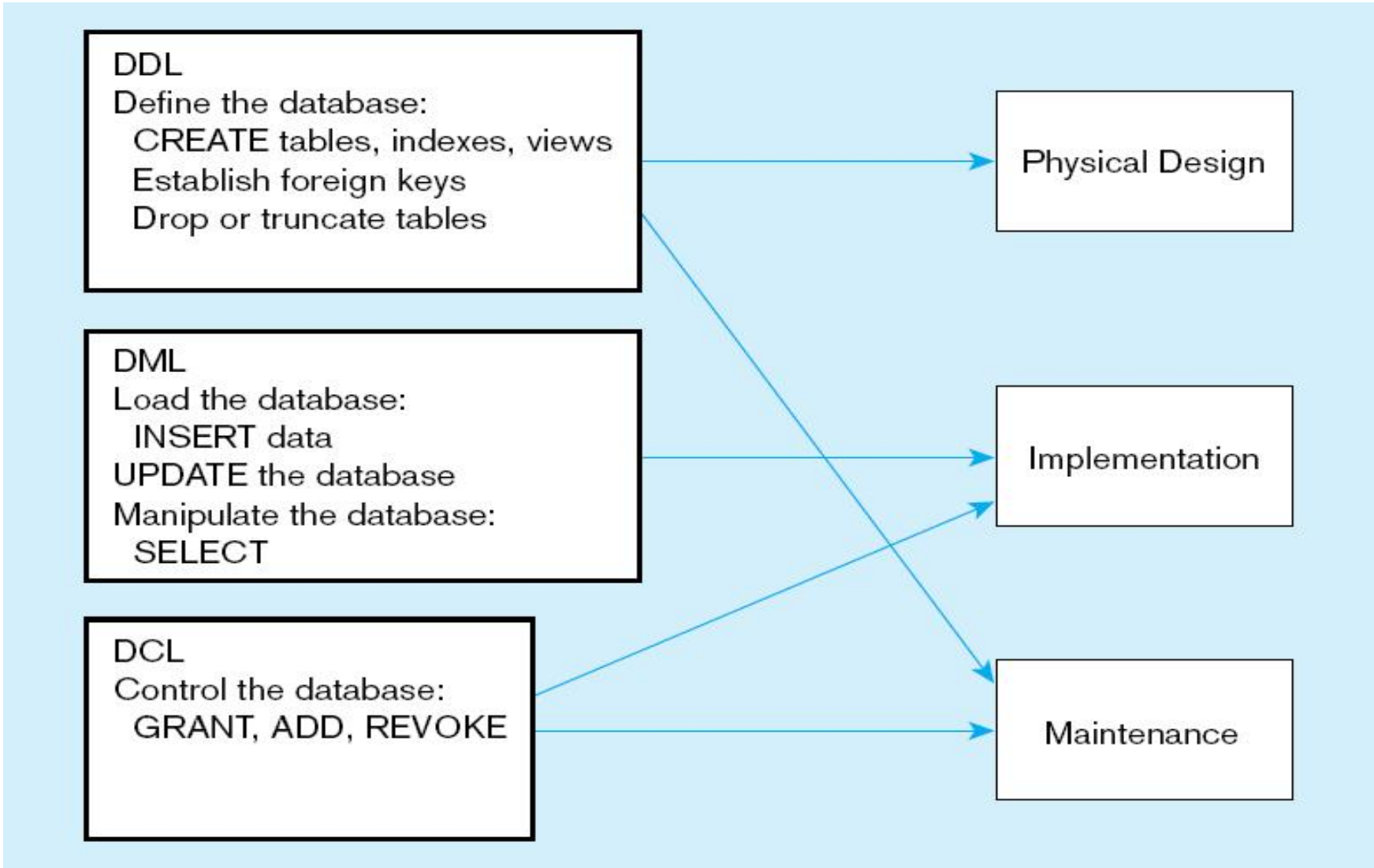
Insert/delete/modify tuples in tables

Transact-SQL

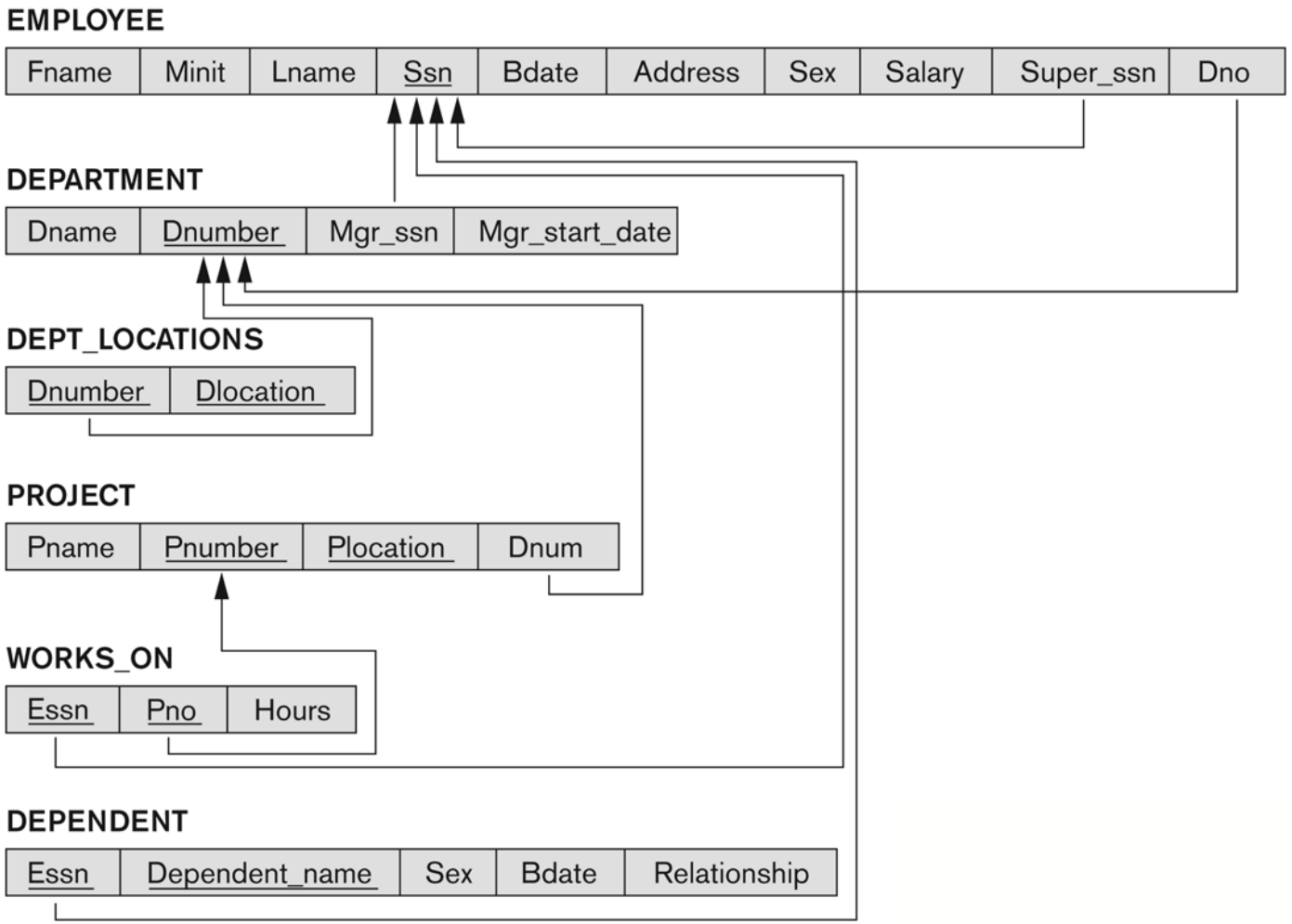
Idea: package a sequence of SQL statements → server

Won't discuss in class

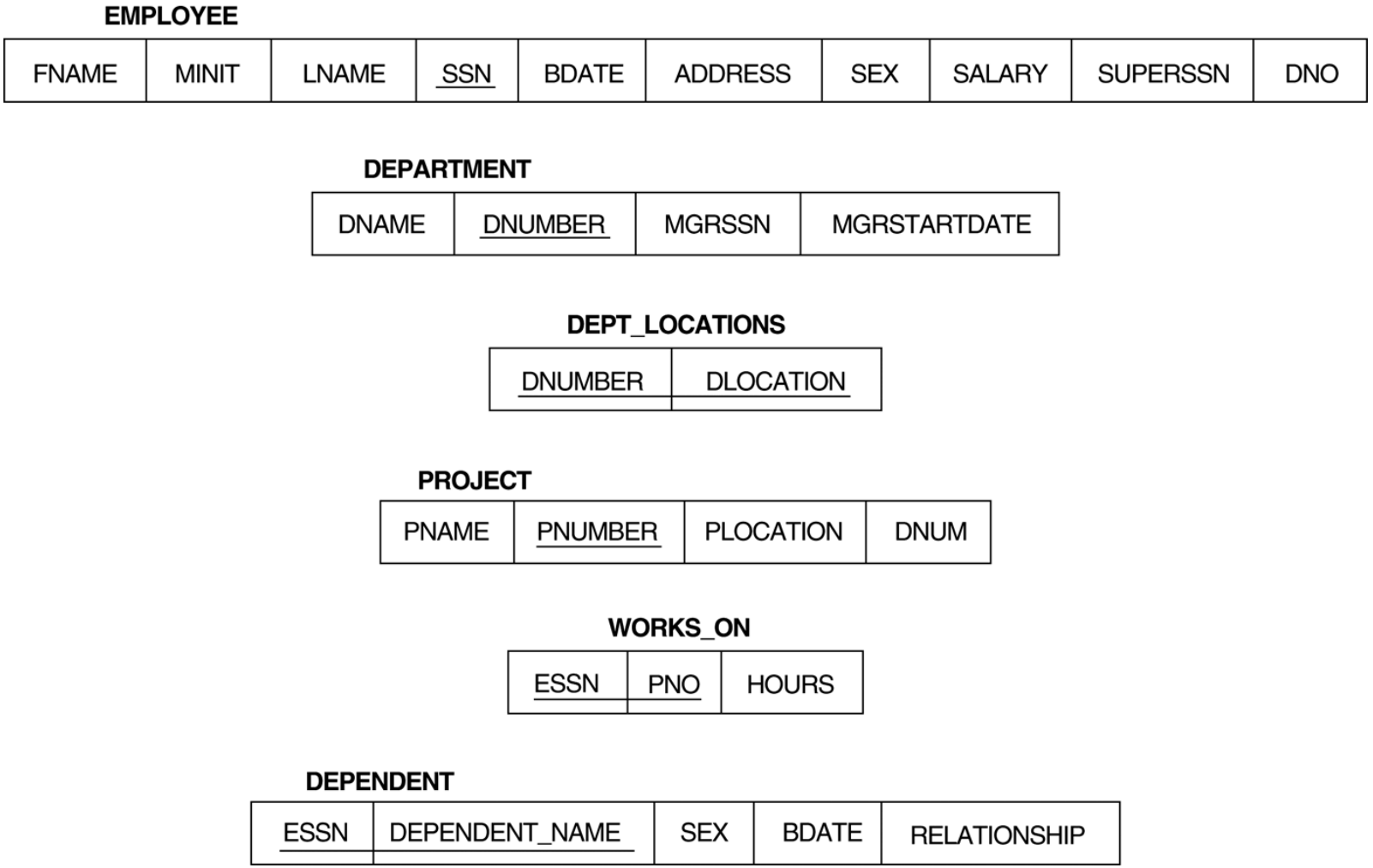
DDL, DML, DCL, and the database development process



COMPANY relational schema.



Relational Database Schema



SQL uses the terms:

- table,
- row, and
- column

for the formal relational model terms

- Relation,
- tuple, and
- attribute, respectively.

Tables in SQL

Table name

Attribute names

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Tuples or rows

Tables Explained

A tuple = a record

Restriction: all attributes are of atomic type

A table = a set of tuples

Like a list…

…but it is unordered: no **first()**, no **next()**, no **last()**.

Tables Explained

The *schema* of a table is the table name and its attributes:

Product(PName, Price, Category, Manfacturer)

A *key* is an attribute whose values are unique; we underline a key

Product(PName, Price, Category, Manfacturer)

Data Definition, Constraints, and Schema Changes

DDL Used to

- CREATE,
- DROP, and
- ALTER

the descriptions of the tables (relations) of a db.

CREATE

CREATE statement, can be used to create

- schemas,
- tables (relations), and
- domains

as well as other constructs such as:

- views,
- assertions, and
- triggers

CREATE SCHEMA

- Specifies a new database schema by giving it a name

CREATE SCHEMA SCHEMA-NAME;

CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

Attribute Data Types and Domains in SQL

The basic data types available for attributes include
numeric, character string, bit string, Boolean, date, and time.

Numeric:

- INTEGER or INT ,
- SMALLINT
- FLOAT or REAL ,
- DOUBLE PRECISION .
- Formatted numbers:
- DECIMAL (i,j) or DEC (i,j)
- NUMERIC (i,j)

—where i, the precision, is the total number of decimal digits
j, the scale, is the number of digits after the decimal point.

- The default for scale is zero, and the default for precision is implementation-defined.

Character-string data types are either fixed length

- CHAR (n)
- CHARACTER (n), where n is the number of characters

varying length—

- VARCHAR (n) or
- CHAR VARYING (n) or
- CHARACTER VARYING (n),

where n is the maximum number of characters.

Bit-string data types :

— BIT (n) fixed length n

— BIT VARYING (n) varying length

Boolean data type: TRUE or FALSE

Additional Data Types in SQL2 and SQL-99

Has DATE, TIME, and TIMESTAMP 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
- **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 in SQL2 and SQL-99 (contd.)

- **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

DOMAIN

It is possible to specify the data type of each attribute directly, ;
Alternatively, a domain can be declared, and the domain name used with the attribute specification.
This makes it easier to change the data type for a domain that is used by numerous attributes in a schema, and improves schema readability. We can use SSN_TYPE in place of CHAR (9)
For example, we can create a domain SSN_TYPE by the following statement:

CREATE DOMAIN SSN_TYPE AS CHAR(9) ;

We can use SSN_TYPE in place of CHAR (9)

CREATE TABLE

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types
  - (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

- In SQL2, can use the CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys).
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
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  );
```

SQL. 1:

```
CREATE TABLE EMP(
ENAME VARCHAR(30) NOT NULL,
ESSN CHAR(9) PRIMARY KEY ,
BDATE DATE,
DNO INT(3) REFERENCES DEPT,
SUPERSSN CHAR(9) REFERENCES EMP);
```

SQL. 2:

```
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(ESSN) );
```

DROP TABLE

- Used to remove a relation (base table) and its definition
- 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 to one of the base relations
  - The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute
- Example:

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

- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple.
  - This can be done using the UPDATE command.

Specifying Attribute Constraints and Attribute Defaults

Because SQL allows NULL s as attribute values, a constraint NOT NULL may be specified if NULL is not permitted for a particular attribute.

It is also possible to define a default value for an attribute by appending the clause
DEFAULT <value> to an attribute definition.

CHECK clause: can restrict attribute or domain values
EX: Dnumber INT NOT NULL
CHECK (Dnumber > 0 AND Dnumber < 21);
suppose that department numbers are restricted to integer numbers between 1 and 20; then, we can change the attribute declaration of Dnumber in the DEPARTMENT table

Specifying Key and Referential Integrity Constraints

Dnumber INT,
...,
....

PRIMARY KEY (Ssn),

OR

Dnumber INT PRIMARY KEY ;

The UNIQUE clause specifies alternate (secondary) keys

Dname VARCHAR(15) UNIQUE ;

Referential integrity is specified via the FOREIGN KEY clause

REFERENTIAL INTEGRITY OPTIONS

A referential integrity constraint can be violated when tuples are inserted or deleted, or when a foreign key or primary key attribute value is modified.

We can specify

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

ON DELETE SET NULL and
ON UPDATE CASCADE

ON DELETE SET NULL :

if the tuple for a supervising employee is deleted, the value of Super_ssn is automatically set to NULL for all employee tuples that were referencing the deleted employee tuple.

ON UPDATE CASCADE :

if the Ssn value for a supervising employee is updated (say, because it was entered incorrectly), the new value is cascaded to Super_ssn for all employee tuples referencing the updated employee tuple

CASCADE ON DELETE:

is to delete all the referencing tuples,

CASCADE ON UPDATE :

is to change the value of the referencing foreign key attribute(s) to the updated (new) primary key value for all the referencing tuples.

REFERENTIAL INTEGRITY OPTIONS (continued)

```
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);
```

Specifying Updates in SQL

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

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER

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 (contd.)

- 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 (contd.)

- 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 (contd.)

- 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 BYDNAME ;
```

INSERT (contd.)

- 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.

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

DELETE (contd.)

- 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 (contd.)

- 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 (contd.)

- 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

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the **SELECT** statement
  - This is *not the same as* the SELECT operation of the relational algebra
- Important distinction between SQL and the formal relational model:
  - SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
  - Hence, an SQL relation (table) is a **multi-set** (sometimes called a **bag**) of tuples; it is *not* a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL (contd.)

- Basic form of the SQL SELECT statement is called a *mapping* or a SELECT-FROM-WHERE *block*

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

  - <attribute list> is a list of attribute names whose values are to be retrieved by the query
  - <table list> is a list of the relation names required to process the query
  - <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL (contd.)

??