SQL DESIGN AND IMPLEMENTATION

CONTENT SOURCES:

- ELAMSARI AND NAVATHE, FUNDAMENTALS OF DATABASE MANAGEMENT SYSTEMS
- BRAD LLOYD & MICHELLE ZUKOWSKI’S SLIDES
- -Silberschatz−Korth−Sudarshan • Database System Concepts, Fourth Edition

An Overview of SQL

- Structured Query Language, which is a computer language for storing, manipulating and retrieving data in relational database.
- SQL is an ANSI (American National Standards Institute) standard but there are many different versions of the SQL language.
- SQL is used to communicate with database.
- SQL are used to perform tasks such as create, alter, drop, insert, update and delete etc.
- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can create new tables, view, index, trigger and cursor etc

An Overview of SQL

- When a user wants to get some information from a database file, he can issue a **QUERY**.
- A query is a user request to retrieve data or information with a certain condition.
- SQL is a query language that allows user to specify the conditions.
- The user specifies a certain condition.
- The program will go through all the records in the database file and select those records that satisfy (searching) the condition.
- The result of the query will then be stored in form of a table.

SQL is used for:

- Data Definition Language(DDL)
- Data Manipulation Language(DML)
- Data QUERY Language(DQL)
- Data Control Language(DCL)
- Data Transaction Control Language(TCL)

OVERVIEW OF SQL

Data Definition Language (DDL)

DDL is the part of SQL that allows a database user to create and restructure database object.

- **CREATE :** Creates a database, new table, a view of a table, index, cursor and trigger or other object in database.
- **ALTER :** Modifies an existing database object, such as a table,view and index etc.
- **DROP :** Deletes an entire table, a view of a table or other object in the database.
- **RENAME:** Used to renaming table, view etc
- **TRUNCATE:** Used to delete record permanente of a table.

OVERVIEW OF SQL

Data Manipulation Language (DML)

DML is the part of SQL used to manipulate data within objects of a relational database.

- **INSERT:** used to insert a record on the table.
- **UPDATE:** modification of records in the table.
- **DELETE:** used to remove records in the table.

Data Query Language (DQL):

A query language is a language in which a user requests information from a table.

- **SELECT:** used to retrieve certain records from one or more table.

OVERVIEW OF SQL

Data Control Language (DCL):

SQL allow you to

control access to data within the database. DCL normally

used to create objects related to user access and also

control the distribution of privileges amoung users.

- **GRANT:** is used to grant both system-level and object-level privileges to an exting database user accounts.
- **REVOKE:** To removes privileges that have been granted to database users.

OVERVIEW OF SQL

Transaction Control Language (TCL):

commands are used to manage transactions in the database. These are used to manage the changes made by DML Statement.

- **COMMIT:** Used to permanently save any database transactions.
- **ROLLBACK:** Used restores the database to last commited state.
- **SAVEPOINT:** Used to temporarily save a transaction.

SQL is a Relational Database

- Represent all info in database as tables
- Keep logical representation of data independent from its physical storage characteristics
- Use one high-level language for structuring, querying, and changing info in the database
- Support the main relational operations
- Support alternate ways of looking at data in tables
- Provide a method for differentiating between unknown values and nulls (zero or blank)
- Support Mechanisms for integrity, authorization, transactions, and recovery

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL DML and DDL

- SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).
- The query and update commands form the DML part of SQL:

SELECT - extracts data from a database

UPDATE - updates data in a database

DELETE - deletes data from a database

INSERT INTO - inserts new data into a database

The most important DDL statements in SQL are:

- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

SQL CREATE TABLE Syntax

```
CREATE TABLE table_name
(
column_name1 data_type constraint if any,
column_name2 data_type constraint if any,
column_name3 data_type constraint if any,
....
);
```

```
CREATE TABLE Persons
(
P_Id int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

Specifying SQL Constraints

- Constraints are used to limit the type of data that can go into table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraints in SQL

NOT NULL : By default, a column can hold NULL. If you not want to allow NULL value in a column, you will want to place a constraint on this column.

```
CREATE TABLE Customer
(
SID integer  NOT NULL,
Last_Name varchar (30) NOT NULL,
First_Name varchar(30)
);
```

UNIQUE

- The UNIQUE constraint ensures that all values in a column are distinct.

```
CREATE TABLE Customer
(
SID integer Unique,
Last_Name varchar (30),
First_Name varchar(30)
);
```

SQL UNIQUE Constraint

- The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.
- You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

CHECK

- The CHECK constraint ensures that all values in a column satisfy certain conditions.

```
CREATE TABLE Customer
(
SID integer CHECK (SID > 0),
Last_Name varchar (30),
First_Name varchar(30)
);
```

Creating Primary key constraint

```
CREATE  TABLE Persons (
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
)
```

To DROP a PRIMARY KEY Constraint

```
ALTER TABLE Persons DROP PRIMARY KEY
```

Creating Foreign Key

- A foreign key is a field (or fields) that points to the primary key of another table.
- The purpose of the foreign key is to ensure referential integrity of the data.

```
CREATE TABLE ORDERS
(
Order_ID integer primary key,
Order_Date date,
Customer_SID integer references CUSTOMER(SID)
);
```

Naming the constraints

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
)
```

Using Alter

Altering Table

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

Altering Constraints

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY
   (P_Id,LastName)
```

SELECT… FROM… WHERE…

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

- <attribute list> is a list of attribute names
- <table list> is a list of the relation names
- <condition> is a conditional (Boolean) expression

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

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

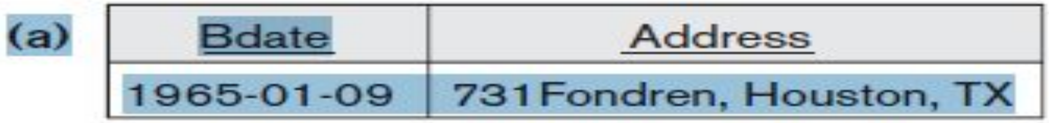
PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

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
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Simple SQL Queries

- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
Q0:   SELECT  BDATE, ADDRESS
      FROM           EMPLOYEE
      WHERE  FNAME='John' AND MINIT='B’
   AND         LNAME='Smith’
```



(a)

Bdate	Address
1965-01-09	731Fondren, Houston, TX

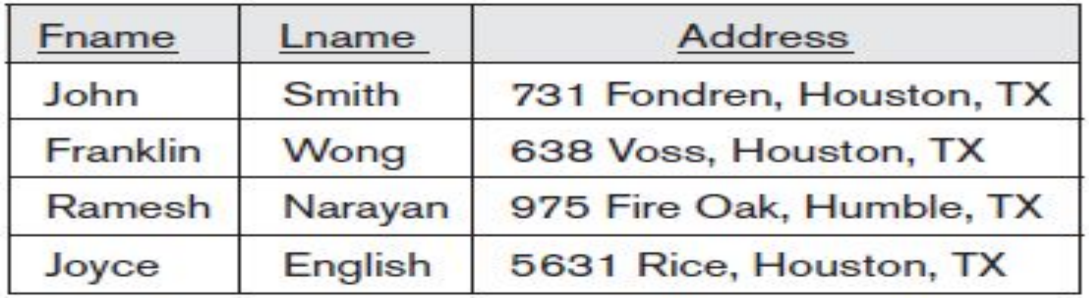
Simple SQL Queries (cont.)

- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:  SELECT      FNAME, LNAME, ADDRESS
     FROM   EMPLOYEE, DEPARTMENT
     WHERE  DNAME='Research' AND DNUMBER=DNO
```

  - (DNAME='Research') is a *selection condition* (corresponds to a SELECT operation in relational algebra)
  - (DNUMBER=DNO) is a *join condition* (corresponds to a JOIN operation in relational algebra)

(b)



Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

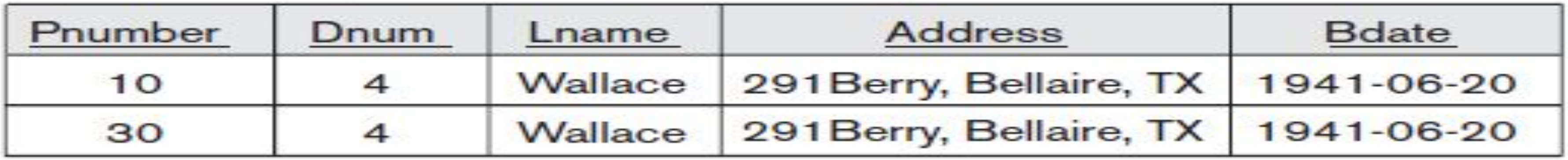
Simple SQL Queries (cont.)

- Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
Q2:  SELECT      PNUMBER, DNUM, LNAME, BDATE, ADDRESS
     FROM        PROJECT, DEPARTMENT, EMPLOYEE
     WHERE       DNUM=DNUMBER AND MGRSSN=SSN              AND
          PLOCATION='Stafford'
```

  - In Q2, there are *two* join conditions
  - The join condition DNUM=DNUMBER relates a project to its controlling department
  - The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

c)



Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

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

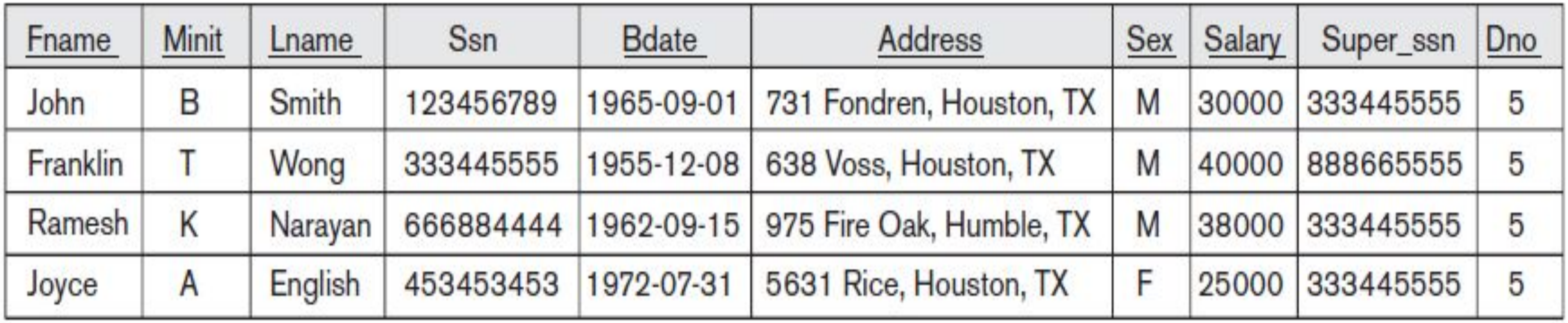
DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*
  <u>Examples:</u>

```
Q1C:    SELECT  *
        FROM EMPLOYEE
        WHERE   DNO=5
```

(g)

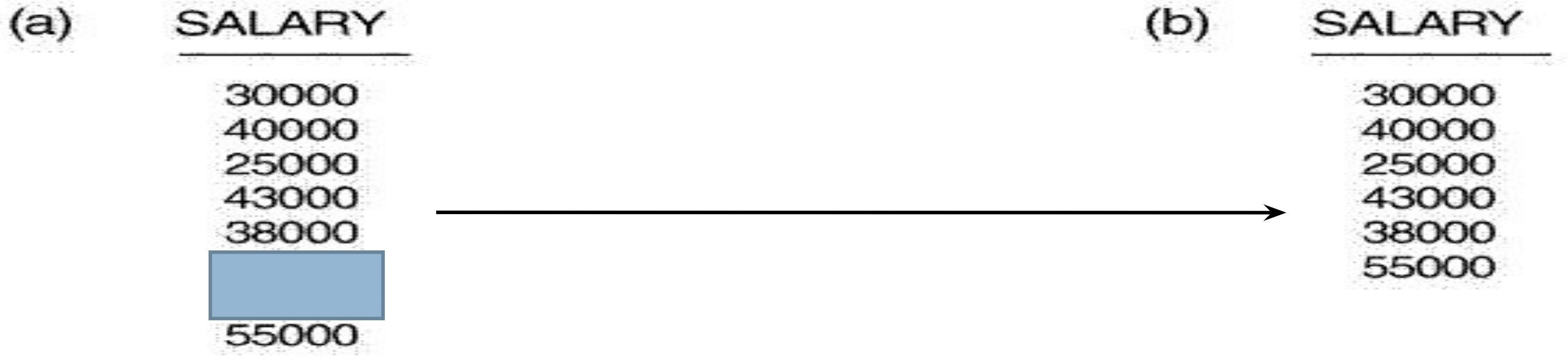


Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

DISTINCT

- SQL does not treat a relation as a set; *duplicate tuples can appear*
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used

```
Q11:     SELECT   SALARY
         FROM EMPLOYEE
Q11A:    SELECT   DISTINCT SALARY
         FROM EMPLOYEE
```



(a)

SALARY
30000
40000
25000
43000
38000
55000

(b)

SALARY
30000
40000
25000
43000
38000
55000

GROUPING

To apply the aggregate functions *to subgroups of tuples in a relation*

Each subgroup of tuples consists of the set of tuples that have *the same value* for the *grouping attribute(s)*

The function is applied to each subgroup independently

GROUPING (cont.)

- Query 20: For each department, retrieve the department number, the number of employees in the department, and their average salary.

-
  ```
  Q20: SELECT     DNO, COUNT (*), AVG (SALARY)
    FROM  EMPLOYEE
    GROUP BY DNO
  ```

  - In Q20, the EMPLOYEE tuples are divided into groups--each group having the same value for the grouping attribute DNO
  - The COUNT and AVG functions are applied to each such group of tuples separately
  - The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
  - A join condition can be used in conjunction with grouping

GROUPING (cont.)

- Query 21: For each project, retrieve the project number, project name, and the number of employees who work on that project

-
```
Q21:      SELECT   PNUMBER, PNAME, COUNT (*)
     FROM  PROJECT, WORKS_ON
     WHERE    PNUMBER=PNO
     GROUP BY    PNUMBER, PNAME
```

- In this case, the grouping and functions are applied *after* the joining of the two relations

SUBSTRING COMPARISON

- The **LIKE** comparison operator is used to compare partial strings
- Two reserved characters are used:

1. '%‘ or '*‘	replace an arbitrary number of characters
2. '_'	replaces a single arbitrary character

SUBSTRING COMPARISON (cont.)

Query 25: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX'.

```
Q25:      SELECT   FNAME, LNAME
       FROM      EMPLOYEE
       WHERE        ADDRESS LIKE  '%Houston,TX%’
```

SUBSTRING COMPARISON (cont.)

- □ Query 26: Retrieve all employees who were born during the 1950s.

```
Q26:     SELECT   FNAME, LNAME
    FROM       EMPLOYEE
    WHERE         BDATE LIKE '_______5_’
```

The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible; hence, in SQL, character string attribute values are not atomic

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_______5_', with each underscore as a place holder for a single arbitrary character.

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-'. '*', and '/' can be applied to numeric values in an SQL query result
- Query 27: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
  Q27:     SELECT          FNAME, LNAME, 1.1*SALARY
        FROM       EMPLOYEE, WORKS_ON, PROJECT
        WHERE      SSN=ESSN AND PNO=PNUMBER AND
PNAME='ProductX’
```

ORDER BY

- The ORDER BY clause is used to sort the tuples in a query result based on the values of some attribute(s)
- Query 28: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
Q28: SELECT          DNAME, LNAME, FNAME, PNAME
       FROM       DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
  WHERE      DNUMBER=DNO AND SSN=ESSN  AND  PNO=PNUMBER
  ORDER BY  DNAME, LNAME
```

ORDER BY (cont.)

- The default order is in ascending order of values
- We can specify the keyword DESC if we want a descending order;
- The keyword ASC can be used to explicitly specify ascending order, even though it is the default

NULLS IN SQL QUERIES

- SQL allows queries that check if a value is NULL

  - (missing / undefined /not applicable)

- SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate .
- Query 14: Retrieve the names of all employees who do not have supervisors.

```
Q14: SELECT     FNAME, LNAME
     FROM       EMPLOYEE
     WHERE           SUPERSSN  IS  NULL
```

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

- **SELECT <attribute list>**
  FROM <table list>
  [WHERE <condition>]
  [GROUP BY <grouping attribute(s)>]
  [HAVING <group condition>]
  [ORDER BY <attribute list>]

Summary of SQL Queries (cont.)

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

Specifying Updates in SQL

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

INSERT

- 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', 7000,'987654321', 4 )
```

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple

- 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.)

- 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.

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.)

```
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.)

- <u>Example:</u> 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

Thankyou