Lecture 3: Structured Query Language (SQL)

CS3402 Database Systems

Relational Query Languages

- Query languages
  - Data Definition Language (DDL): standard commands for defining the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.
  - Data Manipulation Language (DML): standard commands for dealing with the manipulation of data present in database. Common DDL statements SELECT, INSERT, UPDATE, and DELETE.
- Each statement in SQL ends with a semicolon (;)

CREATE SCHEMA Statement

- A schema is a way to logically group objects in a single collection and provide a unique namespace for objects.
- The CREATE SCHEMA statement is used to create a schema. A schema name cannot exceed 128 characters. Schema names must be unique within the database.
- Syntax
  - CREATE SCHEMA schemaName AUTHORIZATION user-name
- Example
  - CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

CREATE TABLE Statement (1/3)

- A CREATE TABLE statement creates a table. Tables contain columns and constraints, rules to which data must conform. Table-level constraints specify a column or columns. Columns have a data type and can specify column constraints (column-level constraints).

```
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) ,
          SUPERSSN       CHAR(9) ,
          DNO            INT                NOT NULL ,
    PRIMARY KEY (SSN) ,
    FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ,
    FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER) ) ;
```

CREATE TABLE Statement (2/3)

```
CREATE TABLE DEPARTMENT
        ( DNAME                 VARCHAR(15)         NOT NULL ,
          DNUMBER               INT                 NOT NULL ,
          MGRSSN                CHAR(9)             NOT NULL ,
          MGRSTARTDATE          DATE ,
        PRIMARY KEY (DNUMBER) ,
        UNIQUE (DNAME) ,
        FOREIGN KEY (MGRSSN) 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 Statement (3/3)

- Create 6 tables
  - EMPLOYEE
  - DEPARTMENT
  - DEPT_LOCATIONS
  - PROJECT
  - WORKS_ON
  - DEPENDENT

```
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) ) ;
```

Table Manipulation

- The ALTER TABLE statement allows you to:
  - Add a column to a table
  - Add a constraint to a table
  - Drop a column from a table
  - Drop an existing constraint from a table
  - Increase the width of a VARCHAR or VARCHAR FOR BIT DATA column
  - change the default value for a column
  - …
- DROP TABLE statement removes the specified table.
- The TRUNCATE TABLE statement allows you to quickly remove all content from the specified table and return it to its initial empty state.

One Possible Database State for the COMPANY Relational Database Schema (1/2)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

One Possible Database State for the COMPANY Relational Database Schema (2/2)

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

SELECT Statement (1/4)

- The basic statement for retrieving information from a database

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

where

- <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.

SELECT Statement (2/4)

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname='John' AND Minit='B' AND Lname='Smith';
```

Projection attributes

Selection conditions

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

Result:

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

SELECT Statement (3/4)

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;
```

Join conditions

Result:

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

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Administration	4	987654321	1995-01-01
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

SELECT Statement (4/4)

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 birth date.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

Project joins Department

Department joins Employee

Result:

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

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

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19
Administration	4	987654321	1995-01-01

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Supe
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	8886
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	8886
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	8886
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	8886
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	8886

Ambiguous Attribute Names

- Same name can be used for two (or more) attributes in different relations
  - As long as the attributes are in different relations
  - Must qualify the attribute name with the relation name to prevent ambiguity

```
Q1A:    SELECT    Fname, EMPLOYEE.Name, Address
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DEPARTMENT.Name='Research' AND
                  DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Aliasing, Renaming and Tuple Variables (1/2)

- Aliases or tuple variables
  - Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:
- For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
SELECT  E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn=S.Ssn;
```

Alias

One-level recursive query

- Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables

Aliasing, Renaming and Tuple Variables (2/2)

- The attribute names can also be renamed

  EMPLOYEE AS E (Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)

- Note that the relation EMPLOYEE now has a variable name E which corresponds to a tuple variable
- The “AS” may be dropped in most SQL implementations

  EMPLOYEE E (Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)

Unspecified WHERE Clause (1/2)

- Missing WHERE clause
  - Indicates no condition on tuple selection (select ALL)
- The resultant effect is a CROSS PRODUCT (JOIN n x m)
  - Result is all possible tuple combinations between the participating relations

Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

```
Q9:   SELECT   Ssn
      FROM     EMPLOYEE;

Q10:  SELECT   Ssn, Dname
      FROM     EMPLOYEE, DEPARTMENT;
```

Ssn
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

8 rows in the result

Unspecified WHERE Clause (2/2)

Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

```
Q9:   SELECT   Ssn
      FROM     EMPLOYEE;

Q10:  SELECT   Ssn, Dname
      FROM     EMPLOYEE, DEPARTMENT;
```

Ssn
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

CROSS PRODUCT

Dname
Research
Administration
Headquarters

Ssn	Dname
123456789	Research
123456789	Administration
123456789	Headquarters
333445555	Research
333445555	Administration
333445555	Headquarters
999887777	Research
999887777	Administration
999887777	Headquarters
.....	
987987987	Research
987987987	Administration
987987987	Headquarters
888665555	Research
888665555	Administration
888665555	Headquarters

24 rows in the result

Use of the Asterisk for SELECT Clause (1/2)

- Specify an asterisk *
  - Retrieve all the attribute values of the selected tuples

```
Q1C:   SELECT   *
       FROM     EMPLOYEE
       WHERE    Dno=5;

Q1D:   SELECT   *
       FROM     EMPLOYEE, DEPARTMENT
       WHERE    Dname='Research' AND Dno=Dnumber;

Q10A:  SELECT   *
       FROM     EMPLOYEE, DEPARTMENT;
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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
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

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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
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

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22
Research	5	333445555	1988-05-22

Use of the Asterisk for SELECT Clause (2/2)

```
Q10A:  SELECT   *
       FROM     EMPLOYEE, DEPARTMENT;
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dname	Dnumber	Mgr_ssn	Mgr_start_date
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	Research	5	333445555	1988-05-22
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	Administration	4	987654321	1995-01-01
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	Headquarters	1	888665555	1981-06-19
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	Research	5	333445555	1988-05-22
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	Administration	4	987654321	1995-01-01
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	Headquarters	1	888665555	1981-06-19
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	Research	5	333445555	1988-05-22
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	Administration	4	987654321	1995-01-01
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	Headquarters	1	888665555	1981-06-19
.....													
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	Research	5	333445555	1988-05-22
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	Administration	4	987654321	1995-01-01
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	Headquarters	1	888665555	1981-06-19
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	Research	5	333445555	1988-05-22
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	Administration	4	987654321	1995-01-01
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	Headquarters	1	888665555	1981-06-19

Tables as Sets in SQL (1/3)

- The ALL and DISTINCT keywords determine whether duplicates are eliminated from the result of the operation.
  - If you specify the DISTINCT keyword, then the result will have no duplicate rows.
  - If you specify the ALL keyword, then there may be duplicates in the result, depending on whether there were duplicates in the input.
  - DISTINCT is the default, so if you don't specify ALL or DISTINCT, the duplicates will be eliminated.
- For example,

Retrieve the salary of every employee

SELECT ALL Salary

FROM EMPLOYEE;

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Retrieve all distinct salary values

SELECT DISTINCT Salary

FROM EMPLOYEE;

Salary
30000
40000
25000
43000
38000
55000

Tables as Sets in SQL (2/3)

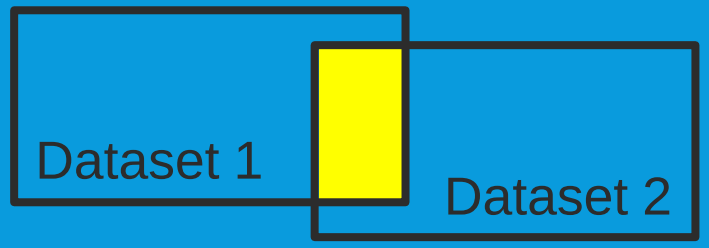
- You can combine SELECT statement using the set operators UNION, INTERSECT, and MINUS. All set operators have equal precedence.
- Each SELECT statement within the operator must have the same number of fields in the result sets with similar data types.
- UNION operator
  - The UNION operator is used to combine the result sets of 2 or more SELECT statements. It removes duplicate rows between the various SELECT statements.
- UNION ALL operator
  - The UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It returns all rows from the query and it does not remove duplicate rows between the various SELECT statements.

Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

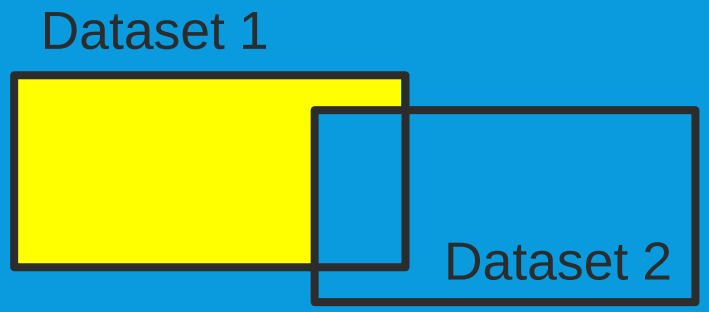
```
Q4A:  (SELECT    DISTINCT Pnumber
       FROM      PROJECT, DEPARTMENT, EMPLOYEE
       WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn
                 AND Lname='Smith' )
       UNION
      ( SELECT   DISTINCT Pnumber
       FROM      PROJECT, WORKS_ON, EMPLOYEE
       WHERE     Pnumber=Pno AND Essn=Ssn
                 AND Lname='Smith' );
```

Tables as Sets in SQL (3/3)

- INTERSECT operator
  - The INTERSECT operator is used to return the results of 2 or more SELECT statements.It only returns the rows selected by all queries or data sets. In other words, if a record exists in one query and not in the other, it will be omitted from the INTERSECT results.



- MINUS operator
  - The MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement. Each SELECT statement will define a dataset. The MINUS operator will retrieve all records from the first dataset and then remove from the results all records from the second dataset.



LIKE Conditions

- The LIKE condition allows wildcards to be used in the WHERE clause of a SELECT, INSERT, UPDATE, or DELETE statement. This allows you to perform pattern matching.

Wildcar	Explanation
%	Allows you to match any string of any length (including zero length)
_	Allows you to match on a single character

- For example, we want to find all of the customers whose first_name begins with ‘P’.

```
SELECT first_name
FROM customers
WHERE last_name LIKE ‘P%’;
```

ORDER BY Clause (1/2)

- The ORDER BY clause is used to sort the records in your result set. The ORDER BY clause can only be used in SELECT statements.
- Syntax: ORDER BY expression [ASC | DESC]
  - expressions: The columns or calculations that you wish to retrieve.
  - ASC: Optional. It sorts the result set in ascending order by expression (default, if no modifier is provider).
  - DESC: Optional. It sorts the result set in descending order by expression.

ORDER BY Clause (2/2)

- Example 1: List in alphabetic order all customers having a loan at Kowloon branch:

```
SELECT DISTINCT cname
FROM Borrow
WHERE bname = “Kowloon”
ORDER BY cname;
```

- Example 2: List the entire borrow table in descending order of amount, and if several loans have the same amount, order them in ascending order by loan#:

```
SELECT *
FROM Borrow
ORDER BY amount DESC,
loan# ASC;
```

INSERT STATEMENT (1/2)

- The INSERT statement is used to insert a single record or multiple records into a table.
- Syntax:
  - Insert a single record using the VALUES keyword

```
INSERT INTO table (column1, column2, ... column_n)
VALUES (expression1, expression2, ... expression_n);
```

  - Insert multiple records using a SELECT statement

```
INSERT INTO table (column1, column2, ... column_n)
SELECT expression1, expression2, ... expression_n
FROM source_table
[WHERE conditions];
```

INSERT STATEMENT (2/2)

- Example:
  - Insert a single record using the VALUES keyword

```
INSERT INTO suppliers
(supplier_id, supplier_name)
VALUES
(5000, 'Apple');
```

  - Insert multiple records using a SELECT statement

```
INSERT INTO suppliers (supplier_id, supplier_name)
SELECT account_no, name
FROM customers
WHERE customer_id > 5000;
```

DELETE Statement

- The DELETE statement is used to delete a single record or multiple records from a table.
- Syntax:

```
DELETE FROM table
[WHERE conditions];
```

  - table: The table that you wish to delete records from.
  - WHERE conditions: Optional. The conditions that must be met for the records to be deleted. If no conditions are provided, then all records from the table will be deleted.

- Example: Delete all records from the employee table where the first_name is Bob

```
DELETE FROM employee
WHERE first_name = 'Bob';
```

UPDATE Statement (1/2)

- The UPDATE statement is used to update existing records in a table.
- Syntax:

```
UPDATE table
SET column1 = expression1,
    column2 = expression2,
    ...
    column_n = expression_n
[WHERE conditions];
```

- Example 1: Update the last_name to ‘Bob’ in the employee table where the employee_id is 123.

```
UPDATE employee
SET last_name = ‘Bob’
WHERE employee_id = 123;
```

UPDATE Statement (2/2)

- Example 2: Increase the payment by 5% to all accounts; it is applied to each tuple exactly once.

```
UPDATE Deposit
SET balance = balance * 1.05;
```

- Example 3: Increase the payment by 6% to all accounts with balance over $10000; all others receive 5% increase

```
UPDATE Deposit
SET balance = balance * 1.06 WHERE balance > 10000;
UPDATE Deposit
SET balance = balance * 1.05 WHERE balance <= 10000;
```

Nested Queries and Set Comparisons (1/6)

- Nested queries
  - SELECT-FROM-WHERE blocks within WHERE clause of another query
  - For example, some queries require that existing values in the database be fetched and then used in a comparison condition
- Comparison operator IN
  - Compares value v with a set (or multiset) of values V
  - Evaluate to TRUE if v is one of the elements in V

Nested Queries and Set Comparisons (2/6)

```
Q4A:  SELECT   DISTINCT Pnumber
      FROM     PROJECT
      WHERE    Pnumber IN
               ( SELECT     Pnumber
                 FROM       PROJECT, DEPARTMENT, EMPLOYEE
                 WHERE      Dnum=Dnumber AND
                            Mgr_ssn=Ssn AND Lname='Smith' )
               OR
               Pnumber IN
               ( SELECT     Pno
                 FROM       WORKS_ON, EMPLOYEE
                 WHERE      Essn=Ssn AND Lname='Smith' );
```

List all project numbers for project that involves an employee whose last name is 'Smith' as a manager of the department that controls the project.

List all project numbers for project that involves an employee whose last name is 'Smith' as a worker.

Nested Queries and Set Comparisons (3/6)

- Place multiple values within parentheses for comparisons
- For example, select the Essn of all employees who work the same (project, hours) as the employee Essn =“123456789”

```
SELECT    DISTINCT Essn
FROM      WORKS_ON
WHERE     (Pno, Hours) IN ( SELECT    Pno, Hours
                            FROM      WORKS_ON
                            WHERE     Essn='123456789' );
```

Nested Queries and Set Comparisons (4/6)

- Use other comparison operators to compare a single value v
- = ANY (or = SOME) operator returns TRUE if the value v is equal to some value in the set V and is hence equivalent to IN
- = ALL returns TRUE if the value v is equal to all the values in the set V
- Other operators that can be combined: >, >=, <, <=, and <>
- Example 1: Find the last name and first name of the employees with salary higher than all the employees in the department with Dno=5

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL   ( SELECT    Salary
                           FROM      EMPLOYEE
                           WHERE     Dno=5 );
```

Nested Queries and Set Comparisons (5/6)

- Example 2: Find names of all branches that have greater assets than some branch located in Central

```
SELECT bname
FROM Branch
WHERE assets > SOME (SELECT assets
                     FROM Branch
                     WHERE b-city = “Central”);
```

or

```
SELECT X.bname
FROM Branch X, Branch Y
WHERE X. assets > Y.assets AND Y.b-city= “Central”;
```

Nested Queries and Set Comparisons (6/6)

- Example 3: Find all customers who have an account at some branch in which Jones has an account

```
SELECT DISTINCT T.cname
FROM Deposit T
WHERE T.cname != “Jones”
        AND T.bname IN (SELECT S.bname
                        FROM Deposit S
                        WHERE S.cname = “Jones”);
```

A set of branch name with “Jones” as cname.

or

```
SELECT DISTINCT T.cname
FROM Deposit S, Deposit T
WHERE S.cname = “Jones”  AND S.bname = T.bname
        AND T.cname != S.cname;
```

Nested Queries and EXISTS Condition (1/3)

- The EXISTS condition is used in combination with a nested query and is considered “to be met” if the nested query returns at least one row.
- The NOT EXISTS condition is used in combination with a nested query and is considered “to be met” if the nested query returns empty result.
- Example 1: Find all customers of Central branch who have an account there but no loan there

```
SELECT C.cname
FROM Customer C
WHERE EXISTS
                    (SELECT *
                     FROM Deposit D
                     WHERE D.cname = C.cname
                          AND D.bname = “Central”)
AND NOT EXISTS
                    (SELECT *
                     FROM Borrow B
                     WHERE B.cname = C.cname
                          AND B.bname = “Central”);
```

Nested Queries and EXISTS Condition (2/3)

- Example 2: Find branches having greater assets than all branches in N.T.

```
SELECT X.bname
FROM Branch X
WHERE NOT EXISTS (SELECT *
                  FROM Branch Y
                  WHERE Y.b-city = "N.T."
                        AND Y.assets >= X.assets);
```

The branches in N.T. with assets greater than or equal to branch X

or

```
SELECT bname
FROM Branch
WHERE assets > ALL (SELECT assets
                    FROM Branch
                    WHERE b-city = "N.T.");
```

Nested Queries and EXISTS Condition (3/3)

- Example 3: Find all customers who have a deposit account at ALL branches located in Kowloon

```
SELECT DISTINCT S.cname
FROM Deposit S
WHERE NOT EXISTS ((SELECT bname
                   FROM Branch
                   WHERE b-city = “Kowloon”)
MINUS
                  (SELECT T.bname
                   FROM Deposit T
                   WHERE S.cname = T.cname));
```

No branches in “Kowloon” S does not has account

Branches in “Kowloon”

Branches where S has an account (cname)

Aggregate Functions (1/5)

- Built-in aggregate functions
  - COUNT, SUM, MAX, MIN, and AVG
- Used to summarize information from multiple tuples into a single tuple
- Example 1: Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.

Salary
30000
40000
25000
43000
38000
25000
25000
55000

```
SELECT   SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM     EMPLOYEE;
```

SUM(Salary)	MAX(Salary)	MIN(Salary)	AVG(Salary)
281000	55000	25000	35125

Aggregate Functions (2/5)

- Example 2: Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT   SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM     EMPLOYEE, DEPARTMENT
WHERE    Dno=Dnumber AND Dname=‘Research’;
```

	Sex	Salary	Super_ssn	Dno
ouston, TX	M	30000	333445555	5
ton, TX	M	40000	888665555	5
ring, TX	F	25000	987654321	4
ire, TX	F	43000	888665555	4
umble, TX	M	38000	333445555	5
ston, TX	F	25000	333445555	5
ston, TX	M	25000	987654321	4
ston, TX	M	55000	NULL	1

Dname	Dn
Research	
Research	
Administration	
Administration	
Research	
Research	
Administration	
Headquarters	

SUM(Salary)	MAX(Salary)	MIN(Salary)	AVG(Salary)
133000	40000	25000	33250

Aggregate Functions (3/5)

- Example 3: Retrieve the total number of employees in the company.

```
SELECT    COUNT(*)
FROM      EMPLOYEE;
```

COUNT(*) returns the number of rows

COUNT(*)
8

- Example 4: Retrieve the number of employees in the ‘Research’ department.

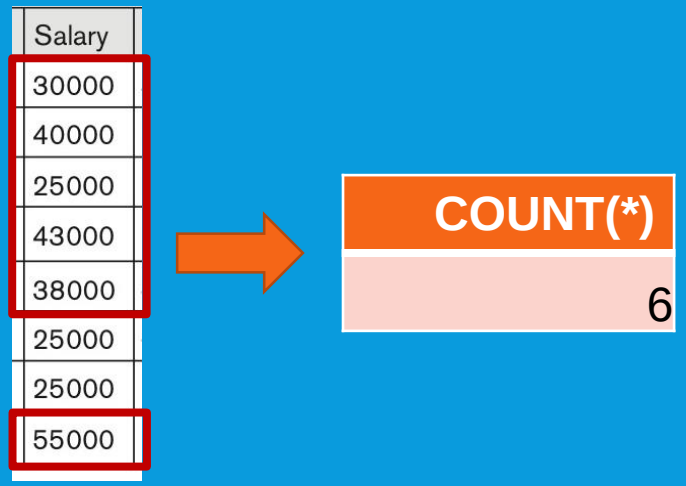
```
SELECT    COUNT(*)
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dno=Dnumber AND Dname=‘Research’
```

COUNT(*)
4

Aggregate Functions (4/5)

- Example 5: Count the number of distinct salary values in the database

```
SELECT    COUNT(DISTINCT Salary)
FROM      EMPLOYEE
```



Aggregate Functions (5/5)

- Example 6: Retrieve the names of all employees who have two or more dependents

```
SELECT   Lname, Fname
FROM     EMPLOYEE
WHERE    (SELECT COUNT(*)
          FROM DEPENDENT
          WHERE SSN=ESSN) >= 2
```

EMPLOYEE

Fname	Minit	Lname	Ssn	
John	B	Smith	123456789	3
Franklin	T	Wong	333445555	3
Alicia	J	Zelaya	999887777	19
Jennifer	S	Wallace	987654321	1
Ramesh	K	Narayan	666884444	19
Joyce	A	English	453453453	19
Ahmad	V	Jabbar	987987987	19
James	E	Borg	888665555	19

DEPENDENT

Essn	Dependent_na
333445555	Alice
333445555	Theodore
333445555	Joy
987654321	Abner
123456789	Michael
123456789	Alice
123456789	Elizabeth

Lname	Fname
Smith	John
Wong	Franklin

GROUP BY Clause (1/4)

- We can apply the aggregate functions to subgroups of tuples in a relation based on some attribute values. For example, find the average salary of employees in each department.
- Grouping the tuples that have same value of some attribute(s), called the grouping attribute(s), and the aggregate function is applied to each subgroup independently.
- SQL has the GROUP BY clause for this purpose
- The GROUP BY clause specifies the grouping attributes, which should also appear in the SELECT clause, so that the value resulting from applying each function to a group of tuples appears along with the value of the grouping attributes.

GROUP BY Clause (2/4)

- Example 1: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT    Dno, COUNT(*), AVG(Salary)
FROM      EMPLOYEE
GROUP BY Dno
```

- The EMPLOYEE tuples are divided into groups , i.e., each group having the same value for the grouping attribute Dno.

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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
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
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
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

GROUP BY Clause (3/4)

- The COUNT and AVG functions are applied to each group of tuples.
- Note that the SELECT clause includes only the grouping attribute and the functions applied on each group of tuples.

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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
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
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
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

33250

31000

55000

Dno	COUNT(*)	AVG (Salary)
5	4	33250
4	3	31000
1	1	55000

```
SELECT    Dno, COUNT(*), AVG(Salary)
FROM      EMPLOYEE
GROUP BY Dno
```

GROUP BY Clause (4/4)

- Example 2: For each project, retrieve the project number, the project name, and the number of employees who work on that project

```
SELECT    Pnumber, Pname, COUNT(*)
FROM      PROJECT, WORKS_ON
WHERE     Pnumber=Pno
GROUP BY Pnumber, Pname
```

HAVING Clause (1/4)

- Retrieve the values of the aggregate functions only for groups that satisfying certain conditions
- SQL provides the HAVING clause which is used in conjunction with the GROUP BY clause for this purpose.
- HAVING clause specifies a condition on the group of tuples associated with each value of the grouping attributes; and only the groups that satisfy the condition are retrieved in the query result.

HAVING Clause (2/4)

- Example 1: For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT      Pnumber, Pname, COUNT (*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber, Pname
HAVING      COUNT (*) > 2;
```

- The selection conditions in the WHERE clause limit the tuples to which functions are applied, and then the HAVING clause serves to choose groups.

HAVING Clause (3/4)

- Example 2: For each department that has more than five employees, retrieve the department number and the number of its employees who are marking more than $40,000.

```
SELECT      Dname, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       Dnumber=Dno AND Salary>40000
GROUP BY    Dname
HAVING      COUNT (*) > 5;
```

- This is incorrect because it will select only departments that have more than five employees who earn more than $40,000. This is because the WHERE clause is executed first to select individual tuples, and then the HAVING clause is applied later to select individual groups of tuples.

HAVING Clause (4/4)

- Example 2: For each department that has more than five employees, retrieve the department number and the number of its employees who are marking more than $40,000.

```
SELECT      Dnumber, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       Dnumber=Dno AND Salary>40000 AND Dno in
                        (SELECT Dno
                         FROM EMPLOYEE
                         GROUP BY Dno
                         HAVING COUNT(*) > 5)
GROUP BY    Dnumber;
```

Views (Virtual Tables) (1/2)

- A VIEW is a virtual table that does not physically exist.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
- A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.
- When you update record(s) in a VIEW, it updates the records in the underlying tables that make up the View. However, most SQL-based DBMSs restrict that a modification is permitted through a view ONLY IF the view is defined in terms of ONE underlying table.

Views (Virtual Tables) (2/2)

- CREATE VIEW statement

```
V1:   CREATE VIEW    WORKS_ON1
      AS SELECT      Fname, Lname, Pname, Hours
          FROM       EMPLOYEE, PROJECT, WORKS_ON
          WHERE      Ssn=Essn AND Pno=Pnumber;

V2:   CREATE VIEW    DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT      Dname, COUNT (*), SUM (Salary)
          FROM       DEPARTMENT, EMPLOYEE
          WHERE      Dnumber=Dno
          GROUP BY   Dname;
```

- Once a View is defined, SQL queries can use the View relation in the FROM clause
- DROP VIEW statement delete a view

NULL Values

- Information can be very often incomplete in the real world
- Unknown attributes are assigned a null value
- One proposal to deal with NULL values is by using 3-valued logic

NOT	
T	F
U	U
F	T

AND	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

OR	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

NULL Value Comparisons

- Syntax: expression IS NULL
  - Expression: The value to test whether it is a null value
  - If *expression* is a NULL value, the condition evaluates to TRUE.
  - If *expression* is not a NULL value, the condition evaluates to FALSE.
- Syntax: expression IS NOT NULL
  - Expression: The value to test whether it is a not null value

Condition	Value of a	Evaluation
a IS NULL	10	FALSE
a IS NOT NULL	10	TRUE
a IS NULL	NULL	TRUE
a IS NOT NULL	NULL	FALSE

Summary of SQL Syntax (1/2)

Table 7.2 Summary of SQL Syntax

```
CREATE TABLE <table name> ( <column name> <column type> [ <attribute constraint> ]
                                { , <column name> <column type> [ <attribute constraint> ] }
                                [ <table constraint> { , <table constraint> } ] )

DROP TABLE <table name>
ALTER TABLE <table name> ADD <column name> <column type>

SELECT [ DISTINCT ] <attribute list>
FROM ( <table name> { <alias> } | <joined table> ) { , ( <table name> { <alias> } | <joined table> ) }
[ WHERE <condition> ]
[ GROUP BY <grouping attributes> [ HAVING <group selection condition> ] ]
[ ORDER BY <column name> [ <order> ] { , <column name> [ <order> ] } ]

<attribute list> ::= ( * | ( <column name> | <function> ( ( [ DISTINCT ] <column name> | * ) ) )
                           { , ( <column name> | <function> ( ( [ DISTINCT] <column name> | * ) ) } ) )

<grouping attributes> ::= <column name> { , <column name> }

<order> ::= ( ASC | DESC )

INSERT INTO <table name> [ ( <column name> { , <column name> } ) ]
( VALUES ( <constant value> , { <constant value> } ) { , ( <constant value> { , <constant value> } ) }
| <select statement> )
```

Summary of SQL Syntax (2/2)

Table 7.2 Summary of SQL Syntax

DELETE FROM <table name> [WHERE <selection condition>]
UPDATE <table name> SET <column name> = <value expression> { , <column name> = <value expression> } [WHERE <selection condition>]
CREATE [UNIQUE] INDEX <index name> ON <table name> (<column name> [<order>] { , <column name> [<order>] }) [CLUSTER]
DROP INDEX <index name>
CREATE VIEW <view name> [(<column name> { , <column name> })] AS <select statement>
DROP VIEW <view name>

NOTE: The commands for creating and dropping indexes are not part of standard SQL.