Basic SQL

Database Systems (CSCI 440)

Fall 2014

Patrick Donnelly

Montana State University

MONTANA STATE UNIVERSITY

Administrivia

Reading:

Chapter 4

Office Hours:

MWF 10-10:50a in EPS 362

SQL

SQL $\rightarrow$ **S**tructured **Q**uery **L**anguage

Originally called SEQUEL $\rightarrow$ **S**tructured **E**nglish **QU**ery **L**anguage

SQL

SQL $\rightarrow$ **S**tructured **Q**uery **L**anguage

Originally called SEQUEL $\rightarrow$ **S**tructured **E**nglish **QU**ery **L**anguage

Factoid

Name changed to SQL because SEQUEL was copyrighted by UK-based Hawker Siddeley aircraft company.

The SQL language is considered of the major reasons for the commercial success of relational databases.

SQL gives statements for data definitions, queries, and updates (both DDL and DML).

SQL has a core specification (standaridized) plus specialized extensions.

A Brief History of SQL (1 / 3)

1970	Ted Codd first proposes a relational model in an influential paper in the ACM journal.
Early **1970's**	SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce for System R.
1974	Ingres is first RDMS to be released (UC Berkeley).
1979	Relational Software (now Oracle) released the first commerical first RDBS.
	IBM (finally) releases their System R only weeks later.
	Britton Lee founded by team that created Ingres.

A Brief History of SQL (2 / 3)

1981 IBM releases SQL/DS.

1983 IBM releases DB2 (still around today).

1986 SQL first standardized by ANSI.

1987 SQL standardized by ISO.

Sybase (#2 product of the 90's) releases first high-performance RDBMS.

1989 SQL89 (SQL1) is revised standard; adopted by US via Federal Information Processing Standard (FIPS).

1992 SQL92 (SQL2) is published.

Microsoft licensed the Sybase technology to develop MS SQL Server.

1995 Initial MySQL release, by Swedish company MySQL AB.

1999 SQL99 (SQL3) is published.

A Brief History of SQL (3 / 3)

2003 SQL:2003 standardized.

2006 SQL:2006 standardized.

2008 Sun Microsystems bought MySQL for $1 billion.

SQL:2008 standardized.

2009 Oracle bought Sun Microsystems for $5.6 billion.
50,000+ developers protest the European Commission.

MariaDB is a community-developed fork of the MySQL, by Monty Widenius, a founder of MySQL.

2011 SQL:2011 standardized.

2047 Quantum relational database technology acquired from invading alien army. Unfortunately, humanity is destroyed.

SQL Data Definition

Table, **row**, and **column** used for relational model terms *relation*, *tuple*, and *attribute*.

`CREATE` statement is SQL's command for data definition.

The **SQL schema** is identified by a **schema name** and includes an **authorization identifier** and **descriptors** for each element.

Schema elements include:

- tables,
- constraints,
- views,
- domains,
- and other constructs...

CREATE SCHEMA statement

Example

```
CREATE SCHEMA COMPANY AUTHORIZATION 'Jsmith';
```

Important

Notice that each statement in SQL ends with a semicolon!!;

Definition

A **Catalog** is a named collection of schemas in an SQL environment.

CREATE TABLE command

CREATE TABLE

The CREATE TABLE command

1. specifies a new relation
2. provide name
3. specify attributes and initial constraints

Can optionally specify schema:

- `CREATE TABLE COMPANY.EMPLOYEE...`

or

- `CREATE TABLE EMPLOYEE...`

`CREATE TABLE` command

Base table (base relation) is a relation and its tuples are actually created and stored as a file by the DBMS.

This is not to be confused with a **virtual relation** created through the `CREATE VIEW` statement.

Some foreign keys may cause errors because of

- circular references
- reference to a table that has not yet been created

CREATE TABLE command

```
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 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) );
```

Attribute Data Types

Numeric data types:

- Integer numbers: INTEGER, INT, and SMALLINT
- Floating-point (real) numbers: FLOAT or REAL, and DOUBLE PRECISION

Character-string data types:

- Fixed length: CHAR(n), CHARACTER(n)
- Varying length: VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n)

Bit-string data types:

- Fixed length: BIT(n)
- Varying length: BIT VARYING(n)

Boolean data type:

- Values of TRUE or FALSE or NULL

Attribute Data Types

DATE data type:

- Ten positions
- Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

Timestamp data type (TIMESTAMP):

- Includes the DATE and TIME fields
- Plus a minimum of six positions for decimal fractions of seconds
- Optional WITH TIME ZONE qualifier

INTERVAL data type:

- Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp

Attribute Data Types

Custom **Domain**:

- Name used with the attribute specification
- Makes it easier to change the data type for a domain that is used by numerous attributes
- Improves schema readability

Example

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

Specifying Constraints in SQL

Basic constraints:

- Key and referential integrity constraints
- Restrictions on attribute domains and NULLs
- Constraints on individual tuples within a relation

Attribute Constraints:

- NOT NULL: NULL is not permitted for a particular attribute
- Default value: `DEFAULT <value>`
- CHECK clause:

```
Dnumber INT NOT NULL CHECK
(Dnumber > 0 AND Dnumber < 21);
```

Default Attribute Values

```
CREATE TABLE EMPLOYEE
    (  . . . ,
       Dno          INT              NOT NULL          DEFAULT 1,
    CONSTRAINT EMPPK
       PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
       FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                       ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
       FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                       ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    (  . . . ,
       Mgr_ssn     CHAR(9)          NOT NULL          DEFAULT '888665555',
       . . . ,
    CONSTRAINT DEPTPK
       PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
       UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
       FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                       ON DELETE SET DEFAULT   ON UPDATE CASCADE);
```

Key Constraints

PRIMARY KEY clause:

- Specifies one or more attributes that make up the primary key of a relation
- `Dnumber INT PRIMARY KEY;`

UNIQUE clause:

- Specifies alternate (secondary) keys
- `Dname VARCHAR(15) UNIQUE;`

Keyword CONSTRAINT

- Name a constraint
- Useful for later altering

Referential Integrity Constraints

FOREIGN KEY clause:

- Default operation: reject update on violation
- Attach referential triggered action clause
- Options include SET NULL, CASCADE, and SET DEFAULT
- Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE
- CASCADE option suitable for "relationship" relations

Specify constraints on tuples Using CHECK clauses at the end of a CREATE TABLE statement:

- Apply to each tuple individually

```
CHECK (Dept_create_date <= Mgr_start_date);
```

Basic Retrieval Queries in SQL

SELECT statement is one basic statement for retrieving information from a database.

SQL allows a table to have two or more tuples that are identical in all their attribute values

- Unlike relational model
- Multiset or bag behavior
- Use DISTINCT option with SELECT to constrain to be sets

SELECT-FROM-WHERE Structure

Basic form of the SELECT statement:

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-FROM-WHERE Structure

Logical comparison operators

- $=, <, <=, >, >=$, and $<>$

Projection attributes are attributes whose values are to be retrieved.

Selection condition are boolean condition that must be true for any retrieved tuple.

SELECT-FROM-WHERE Example - Query 0

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

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';
```

SELECT-FROM-WHERE Example - Query 0

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

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';
```

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

SELECT-FROM-WHERE Example - Query 1

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

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;
```

SELECT-FROM-WHERE Example - Query 1

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

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;
```

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

SELECT-FROM-WHERE Example - Query 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

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

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';
```

SELECT-FROM-WHERE Example - Query 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

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

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';
```

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

Ambiguous Attribute Names

Recall

The same name can be used for two (or more) attributes so long as the attributes are in different relations.

Therefore you must **qualify** the attribute name with the relation name to prevent ambiguity.

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname='Research' AND Dnumber=Dno;

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

Aliases Variables

We can declare alternative relation names, known as aliases.

Example

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

In this case E becomes an alias (also known as tuple variable) for the EMPLOYEE relation.

Also, `Fn` becomes an alias for `Fname`; `Mi` for `Minit`, etc. . . .

Unspecified WHERE Clause

A missing WHERE clause indicates no condition on tuple selection.

This returns the CROSS PRODUCT of all possible tuple combinations.

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

Ssn	Dname
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

Use of the Asterisk

Specifing an asterisk (*) retrieves all the attribute values of the selected tuples.

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

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

Tables as Sets

DISTINCT

SQL does not automatically eliminate duplicate tuples in query results. Use the keyword **DISTINCT** in the SELECT clause if only distinct tuples should remain in the result.

Q11: **SELECT** **ALL** Salary
FROM EMPLOYEE;

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Tables as Sets

DISTINCT

SQL does not automatically eliminate duplicate tuples in query results. Use the keyword **DISTINCT** in the SELECT clause if only distinct tuples should remain in the result.

Q11:
```
SELECT ALL Salary
FROM EMPLOYEE;
```

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Q11A:
```
SELECT DISTINCT Salary
FROM EMPLOYEE;
```

Salary
30000
40000
25000
43000
38000
55000

Tables as Sets

SQL has incorporated set operations:

- UNION, EXCEPT (difference), INTERSECT
- Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)

Tables as Sets

SQL has incorporated set operations:

- UNION, EXCEPT (difference), INTERSECT
- Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)

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' );
```

Substring Pattern Matching

LIKE

`LIKE` Comparison Operator

- used for string pattern matching
- % replaces an arbitrary number of zero or more characters
- underscore (_) replaces a single character

Substring Pattern Matching

LIKE

`LIKE` Comparison Operator

- used for string pattern matching
- % replaces an arbitrary number of zero or more characters
- underscore (_) replaces a single character

Standard arithmetic operators:

- addition (+)
- subtraction ()
- multiplication (*)
- and division (/)

Substring Pattern Matching

LIKE

`LIKE` Comparison Operator

- used for string pattern matching
- % replaces an arbitrary number of zero or more characters
- underscore (_) replaces a single character

Standard arithmetic operators:

- addition (+)
- subtraction ()
- multiplication (*)
- and division (/)

BETWEEN

`BETWEEN` comparison operator is provided for convenience.

LIKE Examples

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

```
SELECT * FROM Employee
WHERE Address LIKE '%TX';
```

```
SELECT * FROM Employee
WHERE SSN LIKE '123_56789';
```

Ordering of Query Results

ORDER BY

The ORDER BY clause:

- Keyword **DESC** to see result in a descending order of values
- Keyword **ASC** to specify ascending order explicitly

Example

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

SELECT <attribute list>
FROM <table list>
[**WHERE** <condition>]
[**ORDER BY** <attribute list>];

INSERT Command

INSERT

The INSERT command specifies the relation name and a list of values for the tuple.

INSERT Command

INSERT

The INSERT command specifies the relation name and a list of values for the tuple.

```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

INSERT Command

INSERT

The INSERT command specifies the relation name and a list of values for the tuple.

```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

```
U3B:  INSERT INTO   WORKS_ON_INFO ( Emp_name, Proj_name,
                    Hours_per_week )
      SELECT        E.Lname, P.Pname, W.Hours
      FROM          PROJECT P, WORKS_ON W, EMPLOYEE E
      WHERE         P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

DELETE Command

DELETE

The DELETE command removes tuples from a relation and can include a WHERE clause to select the tuples to be deleted.

DELETE Command

DELETE

The DELETE command removes tuples from a relation and can include a WHERE clause to select the tuples to be deleted.

```
U4A:  DELETE FROM   EMPLOYEE
      WHERE         Lname='Brown';

U4B:  DELETE FROM   EMPLOYEE
      WHERE         Ssn='123456789';

U4C:  DELETE FROM   EMPLOYEE
      WHERE         Dno=5;

U4D:  DELETE FROM   EMPLOYEE;
```

UPDATE Command

UPDATE

The UPDATE command modifies attribute values of one or more selected tuple.

The additional SET clause in the UPDATE command specifies attributes to be modified and new values.

UPDATE Command

UPDATE

The UPDATE command modifies attribute values of one or more selected tuple.

The additional SET clause in the UPDATE command specifies attributes to be modified and new values.

```
U5:   UPDATE   PROJECT
      SET      Plocation = 'Bellaire', Dnum = 5
      WHERE    Pnumber=10;
```