Module 6

Introduction to SQL

Objectives

- **Learn the basic operations and structure of the SQL database query language**
- **Reference:**
  Elmasri & Navathe, Chapter 7

95.305

Topics

- SQL -background
- Data Definition in SQL
- Queries in SQL

SQL Background

- Commercial Relational Databases don’t use Relational Algebra directly -it is of theoretical interest
- In relational algebra all operations, and the order of operations are specified -leaves no room for optimization
- Commercial query languages are more declarative, specifying what is desired and not completely how.

SQL

- SQL - Structured Query Language (aka SEQUEL)
- ANSI standard SQL 1986 (SQL1)
  ANSI standard SQL 1992 (SQL2, SQL'92)
- SQL serves as
  - DDL -data definition language
  - DML -data manipulation
  - view definition
- Imbedded SQL -calls from within general purpose programming language

SQL

- SQL is as expressive as Relational Algebra
- More programmer oriented
- Widely accepted in commercial databases
- The relational database language
- First used as primary query language, now an interface for higher level query applications

95.305

Topics

- SQL -background
- Data Definition in SQL
- Queries in SQL

Terminology

- table = relation
- row = tuple
- column = attribute
- These terms are used interchangeable in general and in text book

Data Definition in SQL

- DDL operations
  - -CREATE
  - -ALTER
  - -DROP
- Schema’s and Catalogues

Schema Declaration in SQL

SQL 2 Concept which groups together tables belonging to the same application

CREATE SCHEMA COMPANY **AUTHORIZATION JSMITH**

Catalog in SQL2 is a collection of named schemas.

Schema within the same catalog can share domain descriptors or have referential constraints

Creating Tables in SQL

```
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) );
```

SQL DATA TYPES

- **Numeric**
  INTEGER, INT, SMALLINT
  FLOAT, REAL, DOUBLE PRECISION
  DECIMAL(I,J)
  precision I = number of digits
  scale J = number of digits after decimal point
- **Character**

CHAR(N)	**fixed N character string**
VARCHAR(N)	**variable length string**
BIT(N)	**bit-string of length N**

- **Date and Time**

DATE	**YYYY-MM-DD**
TIME	**HH:MM:SS**
TIMESTAMP	**Date and time**
INTERVAL	**offset for Date, Time, Timestamp**

Types can be named or assigned directly

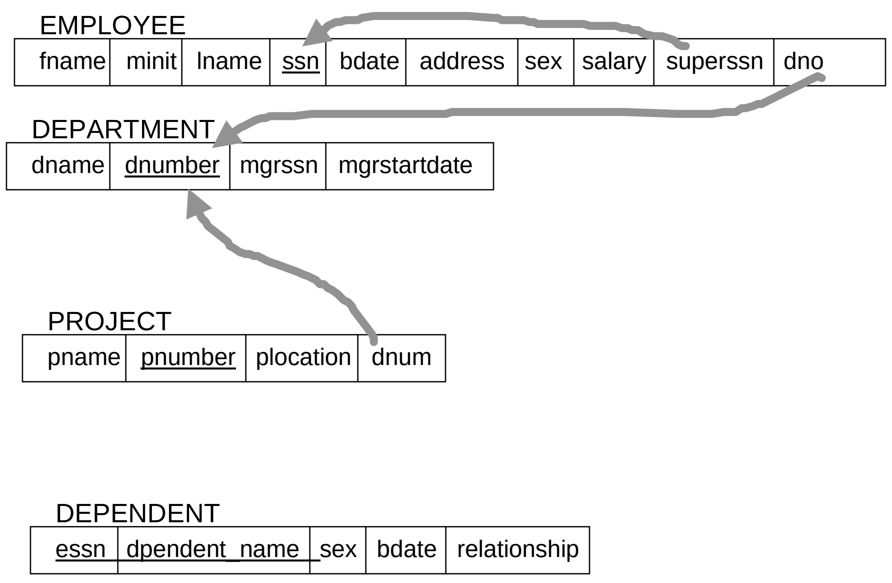
```
CREATE TABLE EMPLOYEE
(
FNAME                VARCHAR(15)          NOT NULL
...
SSN                  CHAR(9)              NOT NULL
...
```

```
CREATE DOMAIN SSN_TYPE AS CHAR(9)
 CREATE TABLE EMPLOYEE
 (
 FNAME                VARCHAR(15)          NOT NULL
 ...
 SSN                  SSN_TYPE             NOT NULL
 ...
```

Defaults, named constraints and referential triggers

```
CREATE TABLE EMPLOYEE
(
FNAME          VARCHAR(15)          NOT NULL,
...
DNO            CHAR(9)              NOT NULL      DEFAULT 1,
...
CONSTRAINT EMPPK
   PRIMARY KEY (SSN),
CONSTRAINT EMPSUPERFK
   FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
          ON DELETE SET NULL         ON UPDATE CASCADE,
CONSTRAINT EMPDEPFK
   FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER)
          ON DELETE SET DEFAULT      ON UPDATE CASCADE );
```

Foreign Key Constraints


EMPLOYEE
fname | minit | lname | ssn | bdate | address | sex | salary | superssn | dno
DEPARTMENT
dname | dnumber | mgrssn | mgrstartdate
PROJECT
pname | pnumber | plocation | dnum
DEPENDENT
essn | dpendent_name | sex | bdate | relationship

Deleting a tuple with referential constraint

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

FOREIGN KEY (SUPERSSN) **REFERENCES** EMPLOYEE(SSN)

ON DELETE SET NULL **ON UPDATE** CASCADE,

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	NULL	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	NULL	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4

Updating a tuple with referential constraint

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	111111111	10-Nov-27	450 Stone	M	55000	NULL	1

FOREIGN KEY (SUPERSSN) **REFERENCES** EMPLOYEE(SSN)
ON DELETE SET NULL **ON UPDATE** CASCADE,

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	111111111	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	111111111	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	111111111	10-Nov-27	450 Stone	M	55000	NULL	1

Deleting Schemas and Tables

DROP TABLE DEPENDENT **CASCADE**

- **Deletes specified table**
  CASCADE option also deletes any constraints that refer to the table
  RESTRICT option deletes only if table is not referenced

DROP SCHEMA COMPANY **CASCADE**

- **Deletes specified schema**
  CASCADE option also deletes all schema elements
  RESTRICT option deletes only if is empty

Changing Table Descriptions

- **ALTER TABLE command can be used to**
  -add, delete an attribute (column)
  -change attribute definition
  -adding, deleting constraints

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

ALTER TABLE EMPLOYEE **DROP** ADDRESS **CASCADE**;

ALTER TABLE DEPARTMENT **ALTER** MGRSSN **DROP DEFAULT;**

ALTER TABLE DEPARTMENT **ALTER** MGRSSN **SET DEFAULT** “111222333”**;**

ALTER TABLE EMPLOYEE **DROP CONSTRAINT** EMPSUPERFK **CASCADE;**

95.305

Topics

- **SQL -background**
- **Data Definition in SQL**
- **Queries in SQL**

Basic Queries in SQL

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

- **SQL queries are done using the SELECT statement**
- **not the same as Relational Algebra SELECT**
- **SQL tables allow duplicated tuples, unlike relational algebra's relations as sets of tuples.**

Basic SELECT-FROM-WHERE

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

- <attribute list> **attribute names to be retrieved**
- <table list> **relations required to process the query**
- <condition> **Boolean expression that identifies the tuples to be retrieved.**

example: Single table query

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

SELECT BDATE, ADDRESS
FROM EMPLOYEE
WHERE FNAME = ‘John’ **AND** INIT = ‘B’ **AND** LNAME = ‘Smith’

BDATE	ADDRESS
9-Jan-55	731 Fondren

Note: the single table case is like the relational algebra select-project pair

example: SQL “Join” Query

Find the name and address of everyone who works for in ‘Research’ dept

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	9998						
Jennifer	S	Wallace	9876						
Ramesh	K	Narayan	6668						
Joyce	A	English	4534						
Ahmad	V	Jabber	9879						
James	E	Borg	8886						

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-May-78
Administration	4	987654321	1-Jan-85
Headquarters	1	888665555	19-Jun-71

SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME = ‘Research’ **AND** DNUMBER = DNO

Note: This is like a select-project-join combination in relational algebra

...example: SQL “Join” Query

EMPLOYEE

FNAME	INIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya							4
Jennifer	S	Wallace							4
Ramesh	K	Naraya							5
Joyce	A	English							5
Ahmad	V	Jabber							4
James	E	Borg							1

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-May-78
Administration	4	987654321	1-Jan-85
Headquarters	1	888665555	19-Jun-71

SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME = ‘Research’ **AND** DNUMBER = DNO

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren
Franklin	Wong	638 Voss
Ramesh	Narayan	975 Fire Oak
Joyce	English	5631 Rice

example: SQL “Double Join” Query

List project number, controlling dept. no., managers last name, address and birthdate for all projects located at ‘Stafford’

EMPLOYEE

fname	minit	lname	<u>ssn</u>	bdate	address	sex	salary	superssn	dno

DEPARTMENT

dname	<u>dnumber</u>	mgrssn	mgrstartdate

PROJECT

pname	<u>pnumber</u>	plocation	dnum

SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM PROJECT,DEPARTMENT, EMPLOYEE
WHERE DNUM = DNUMBER **AND** MGRSSN = SSN **AND**
PLOCATION = ‘Stafford’

...example: SQL “Double Join” Query

List project number, controlling dept. no., managers last name, address and birthdate for all projects located at ‘Stafford’

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

PNUMBER	DNUM	LNAME	ADDRESS	BDATE
10	4	Wallace	291 Berry	20-Jun-31
30	4	Wallace	291 Berry	20-Jun-31

When attribute names are ambiguous

Find the name and address of everyone who works in the ‘Research’ dept

EMPLOYEE

FNAME	INIT	NAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALAR	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	9998						
Jennifer	S	Wallace	9876						
Ramesh	K	Narayan	6668						
Joyce	A	English	4534						
Ahmad	V	Jabber	9879						
James	E	Borg	8886						

DEPARTMENT

NAME	<u>DNO</u>	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-May-78
Administration	4	987654321	1-Jan-85
Headquarters	1	888665555	19-Jun-71

```
SELECT FNAME, EMPLOYEE.NAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DEPARTMENT.NAME = ‘Research’ AND
          DEPARTMENT.DNO  = EMPLOYEE.DNO
```

Recursive Query (one-level)

Retrieve for each employee their first name and last name, and their supervisor's first name and last name

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN = S.SSN

Renaming the Relations

What Happened to "James Borg"

Retrieve for each employee their first name and last name, and their supervisors first name and last name

SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN = S.SSN

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong						555	5
Alicia	J	Zelay						987	4
Jennifer	S	Walla						555	4
Ramesh	K	Naray						555	5
Joyce	A	Engli						555	5
Ahmad	V	Jabbe						321	4
James	E	Borg						NULL	1

E.FNAME	E.LNAME	S.FNAME	S.LNAME
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

Omitting the WHERE Clause

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

SELECT SSN
FROM EMPLOYEE

SSN
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

...Omitting the WHERE Clause

EMPLOYEE

FNAME	INIT	NAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777						4
Jennifer	S	Wallace	987654321						4
Ramesh	K	Narayan	666884444						5
Joyce	A	English	453453453						5
Ahmad	V	Jabber	987987987						4
James	E	Borg	888665555						1

DEPARTMENT

DNAME	DNO	MGRSSN	MGRSTARTD
Research	5	333445555	22-May-78
Administration	4	987654321	1-Jan-85
Headquarters	1	888665555	19-Jun-71

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT

...Omitting the WHERE Clause

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT

Like a Cartesian Product, Project combination

How would you specify an actual Cartesian product?

Note: Very large, and incorrect relations can result if the WHERE clause is not completely or properly specified

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

Retrieving all Attributes with "*"

Retrieve all attributes for those employees who work in department 5

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

SELECT *
FROM EMPLOYEE
WHERE DNO = 5

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5

Removing Duplicate Tuples

- SQL does not automatically remove duplicate tuples
- Duplicates must be removed explicitly using the DISTINCT clause
- Duplicate removal takes time, so you should only do it if you need to
- Some queries are easier to formulate if duplicates are not removed

...Removing Duplicates

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

SELECT SALARY
FROM EMPLOYEE

SALARY
30000
40000
25000
43000
38000
25000
25000
55000

SELECT DISTINCT SALARY
FROM EMPLOYEE

SALARY
30000
40000
25000
43000
38000
55000

SET Based operations in SQL

- **SQL supports the set based operations of UNION, INTERSECTION, and DIFFERENCE using the**

  UNION, INTERSECT, and EXCEPT operations respectively

- **Duplicates <u>are</u> eliminated for the set based operations**
- **Duplicate elimination can be suppressed using the ALL keyword after the operation**

UNION example in SQL

List 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 who controls the project

```
(SELECT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MSGSSN=SSN AND LNAME =
   'Smith')
UNION
(SELECT PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER = PNO AND ESSN = SSN AND LNAME = 'Smith')
```

List 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 who controls the project

```
SELECT DISTINCT PNUMBER
FROM PROJECT
WHERE PNUMBER IN
   (SELECT PNUMBER
   FROM PROJECT, DEPARTMENT, EMPLOYEE
   WHERE DNUM=DNUMBER AND
                  MGRSSN=SSN AND
                 LNAME = ‘Smith’)
   OR
   PNUMBER IN
   (SELECT PNO
   FROM WORKS_ON, EMPLOYEE
   WHERE ESSN=SSN AND LNAME = ‘Smith’)
```

Using the IN operator

List the social insurance numbers of all employees who work the same (hours, project) combination as ‘John Smith’ does

```
SELECT DISTINCT ESSN
FROM WORKS_ON
WHERE (PNO, HOURS) IN
   (SELECT PNO, HOURS
   FROM WORKS_ON
   WHERE SSN = ‘123456789’);
```

...Other set comparison operations

List the names of all employees whose salary is greater than that of all the employees in dept. 5

```
SELECT DISTINCT FNAME, LNAME
FROM EMPLOYEE
WHERE SALARY > ALL
   (SELECT SALARY
   FROM EMPLOYEE
   WHERE DNO = 5);
```

...Other set comparison operations

- =, >, <, >=, <=, <> **combined with** SOME, ANY, ALL

- SOME **and** ANY **have the same meaning**
- ANY **was used in earlier versions of SQL but it's ambiguous**
- **e.g. Find all the bank branches with assets greater than any branch in "Ottawa" -what does this mean?**

Testing for Empty Relations -EXISTS clause

List the name of each employee who has a dependent with the same first name and same sex as the employee

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE E
WHERE EXISTS
   (SELECT *
   FROM DEPENDENT
   WHERE E.SSN=ESSN AND SEX=E.SEX AND
              E.FNAME = DEPENDENT_NAME );
```

Testing for Empty Relations - NOT EXISTS clause

List the names of employees who have no dependents

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE E
WHERE NOT EXISTS
   (SELECT *
   FROM DEPENDENT
   WHERE SSN = ESSN);
```

UNIQUE operator

Testing for duplicates

```
UNIQUE (SELECT ...
        FROM ...
        WHERE ... )
```

Returns true if query results contains no duplicates, and false otherwise

Testing for Subsets

List the name of each employee who works on all projects controlled by department 5

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE
   ((SELECT PNO
   FROM WORKS_ON
   WHERE SSN=ESSN)
   CONTAINS
   (SELECT PNUMBER
   FROM PROJECT
   WHERE DNUM = 5));
```

UNFORTUNATELY many commercial databases do not support the CONTAINS operator

...Testing for Subsets without using CONTAINS

List the name of each employee who works on all projects controlled by department 5

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS
   (SELECT *
   FROM WORKS_ON B
   WHERE (B.PNO IN (SELECT PNUMBER
                     FROM PROJECT
                     WHERE DNUM = 5 ))
            AND
            NOT EXISTS (SELECT *
                        FROM WORKS_ON C
                        WHERE C.ESSN=SSN AND C.PNO = B.PNO))
```

...Testing for Subsets without using CONTAINS

Select each employee such that there does not exist a project controlled by dept. 5 that the employee does not work for

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS
   (SELECT *
   FROM WORKS_ON B
   WHERE (B.PNO IN (SELECT PNUMBER
                     FROM PROJECT
                     WHERE DNUM = 5 ))
            AND
            NOT EXISTS (SELECT *
                        FROM WORKS_ON C
                        WHERE C.ESSN=SSN AND C.PNO = B.PNO))
```

Explicitly refering to sets and NULLS

List the social security number of all employees who work on projects 1, 2, or 3

```
SELECT DISTINCT ESSN
FROM WORKS_ON
WHERE PNO IN (1, 2, 3)
```

List the names of all employees who do not have supervisors

```
SELECT DISTINCT FNAME LNAME
FROM EMPLOYEE
WHERE SUPERSSN IS NULL
```

Renaming attributes

List the last name of each employee and their supervisor, while renaming the attributes to EMPLOYEE_NAME and SUPERVISOR_NAME

```
SELECT E.LNAME AS EMPLOYEE_NAME,
   S.LNAME AS SUPERVISOR_NAME
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.SUPERSSN = S.SSN
```

JOINing tables in the FROM clause

List the name and address of each employee of the ‘Research department’

```
SELECT FNAME, LNAME, ADDRESS
FROM (EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER)
WHERE DNAME = ‘Research’
```

May be easier than listing all conditions in where clause
Also supports NATURAL JOIN, OUTER JOIN

Aggregate Functions in SQL

Aggregate functions include:

COUNT	**-count the tuples in query result**
SUM	**-sum tuples of numeric field in table**
MIN	**-answer minimum tuple value of a numeric field**
MAX	**-answer maximum tuple value of a numeric field**
AVG	**-answer the mean of tuple values of numeric field**

Aggregate Functions

Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and average salary

SELECT SUM(SALARY), **MAX**(SALARY), **MIN**
(SALARY), **AVG**(SALARY)
FROM EMPLOYEE

...Aggregate Functions

Find the sum of the salaries of all employees of the 'Research' dept., as well as the maximum salary, the minimum salary, and average salary in this dept.

SELECT SUM(SALARY), **MAX**(SALARY), **MIN**
(SALARY), **AVG**(SALARY)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER **AND** DNAME='Research'

...Aggregate Functions

Count the employees of the 'Research' dept.

```
SELECT  COUNT(*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'
```

Count the number of distinct salary values of employees

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

...Aggregate Functions

List the name of all employees who have two or more dependents

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

...Applying Aggregates to Subgroups

For each dept. list dept. number, number of employees, and their average salary

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

NOTE: group by attributes must appear in the select arguments

...Applying Aggregates to Subgroups

For each dept. list dept. number, number of employees, and their average salary

EMPLOYEE

FNAME	INIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-55	731 Fondern	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-45	638 Voss	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-58	3321 Castle	F	25000	987987987	4
Jennifer	S	Wallace	987654321	20-Jun-31	291 Berry	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-52	975 Fire Oak	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-62	5631 Rice	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-Mar-59	980 Dallas	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-27	450 Stone	M	55000	NULL	1

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

DNO	COUNT(*)	AVG(SALARY)
5	4	33250
4	3	31000
1	1	55000

...Applying Aggregates to Subgroups

For each project which have at least two employees, list the project number, project name and number of employees who work on the project

Limits tuples

Limits entire groups

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT(*) > 2
```

...Applying Aggregates to Subgroups

For each dept. having more than five employees, retrieve the department name and number of employees making more than $40,000

```
SELECT DNAME, COUNT(*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNUMBER=DNO AND SALARY >40000
   AND
   DNO IN (SELECT DNO
             FROM EMPLOYEE
             GROUP BY DNO
             HAVING COUNT(*) > 5)
GROUP BY DNAME
```

Matching partial substrings

List all employees whose address includes “Houston TX”

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE ADDRESS LIKE ‘%Houston,TX%’
```

Find all employees born during the 50’s

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE BDATE LIKE ‘__5_______’
```

% matches any substring, _ matches one character

Applying arithmetic operators to numeric values

List the resulting salaries if each employee working on ‘ProductX’ is given a 10% raise

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

Also allowed: +, -, *, /, || (concatenation for strings)

Ordering Tuples

List employees and the projects they are working on, ordered by department and, within department, alphabetically by last name, first name

SELECT DNAME, LNAME, FNAME, PNAME

FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT

WHERE DNUMBER=DNO **AND** SSN=ESSN **AND** PNO=PNUMBER

ORDER BY DNAME, LNAME, FNAME

Also possible:
ORDER BY DNAME **DESC,** LNAME **ASC,** FNAME **ASC**

SELECT-FROM-WHERE Summary

SELECT <attribute list>

FROM <table list>

[**WHERE** <condition>]

[**GROUP BY** < grouping attributes>]

[**HAVING** <group condition>]

[**ORDER BY** <attribute list>

There are usually several ways to form a query

Unfortunately, though they should, databases may not process all phrasings the same, thus there are more, and less efficient ways to pose a query which depends on the implementation

Updates in SQL

Three updating operations: INSERT, DELETE, UPDATE

```
INSERT INTO EMPLOYEE
VALUES ('Richard', 'K', 'Marini', '653298653', '30-dec-52',
        '98 Oak', 'M', 37000, 987654321', 4)
```

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

unmentioned attributes are set to DEFAULT or NULL

Creating a Temporary Table

```
CREATE TABLE DEPTS_INFO (DEPT_NAME VARCHAR(15)
                         NO_EMPS     INTEGER
                         TOTAL_SAL    INTEGER );

INSERT INTO DEPTS_INFO (DEPT_NAME, NO-EMPS,
   TOTAL_SAL)
   SELECT DNAME, COUNT(*), SUM(SALARY)
   FROM DEPARTMENT, EMPLOYEE
   WHERE DNUMBER = DNO
   GROUP BY DNAME;
```

This could be dangerous if table is not temporary -why?

Deleting in SQL

```
DELETE FROM EMPLOYEE
WHERE LNAME = ‘Brown’
```

```
DELETE FROM EMPLOYEE
WHERE DNO IN (SELECT DNUMBER
              FROM DEPARTMENT
              WHERE DNAME = ‘Research’)
```

UPDATE

```
UPDATE PROJECT
SET PLOCATION = ‘Bellaire’, DNUM = 5’
WHERE PNUMBER = 10
```

```
UPDATE EMPLOYEE
SET SALARY = SALARY*1.1
WHERE DNO IN (SELECT DNUMBER
              FROM DEPARTMENT
              WHERE DNAME = ‘Research’)
```

VIEWS in SQL

VIEWs are virtual tables -they are always up to date

```
CREATE VIEW DEPT_INFO (DEPT_NAME, NO_EMPS,
   TOTAL_SAL)
AS SELECT DNAME,  COUNT(*), SUM(SALARY)
   FROM DEPARTMENT, EMPLOYEE
   WHERE DNUMBER = DNO
   GROUP BY DNAME;
```

Updating of views table is tricky -research issue

More Constraints

Create arbitrary assertions to enforce business rules

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS (SELECT *
                    FROM EMPLOYEE E, EMPLOYEE M,
                         DEPARTMENT D
                    WHERE E.SALARY > M.SALARY AND
                         E.DNO=D.DNUMBER AND
                         D.MGRSSN =M.MSSN) );
```

e.g. an employee cannot make more than their manager

Some implementation also provide TRIGGERS and ACTIONS

Sample ingres SQL Queries using banking_dbs

Objectives

- Show some examples of queries done with ingres on the banking_dbs database
- Reference:
  none

banking_dbs

- person (name,dob,sin,sex,straddr,city,phone)
- deposit (brchname,deptno,accntno,name,dob)
- account (accntno,bal,ovrdftlmt)
- branch (brchname,assets,brchcity)
- borrow (brchname,deptno,loanno,name,dob,crdstatus)
- loans (loanno,amt,mthpymt,amtrem)
- works (empno,brchname,deptno,name,dob,pos,wsdate,wphone,sal)
- dept (deptno,brchname,deptname)
- manages (empno,brchname,deptno,mgrstatus,mgrsdate)
- dependent (empno,deptname,depdob,kinship)

- list the names and kinship, if any, of those employees who have no dependents, or who have a daughter

```
1> /* find employee number and name for all employees
2>    who have dependents  */
3> select w.empno, w.name
4> from works w, dependent d
5> where w.empno = d.empno

+-------------+--------------------+
|empno        |name                |
+-------------+--------------------+
|        45678|Hayes, B. B.        |
|        45678|Hayes, B. B.        |
|        45678|Hayes, B. B.        |
|        53099|Kopecky, S.         |
|        53099|Kopecky, S.         |
|        53882|Hutton, E. B.       |
|        66890|Witton, J.          |
|        66890|Witton, J.          |
|        67222|White. J.           |
```

```
1> /* find employee number and name and kinship for
 all employees
2>    who have dependents  */
3> select w.empno, w.name, d.kinship
4> from works w, dependent d
5> where w.empno = d.empno

+-------------+--------------------+---------------+
|empno        |name                |kinship        |
+-------------+--------------------+---------------+
|        45678|Hayes, B. B.        |Daughter       |
|        45678|Hayes, B. B.        |Son            |
|        45678|Hayes, B. B.        |Spouse         |
|        53099|Kopecky, S.         |Son            |
|        53099|Kopecky, S.         |Spouse         |
|        53882|Hutton, E. B.       |Daughter       |
|        66890|Witton, J.          |Daughter       |
|        66890|Witton, J.          |Spouse         |
|        67222|White. J.           |Daughter       |
|        67321|McGuire, P.         |Spouse         |
1> /* find employee number and name and kinship for
 all employees
2>    who have daughters*/
3> select w.empno, w.name, d.kinship
4> from works w, dependent d
5> where (w.empno = d.empno) and d.kinship like
 'Daughter'

+-------------+--------------------+---------------+
|empno        |name                |kinship        |
+-------------+--------------------+---------------+
|        45678|Hayes, B. B.        |Daughter       |
|        53882|Hutton, E. B.       |Daughter       |
|        66890|Witton, J.          |Daughter       |
|        67222|White. J.           |Daughter       |
|        69820|Landry, W.          |Daughter       |
|        99775|Blumberg, Z.        |Daughter       |
+-------------+--------------------+---------------+
```

```
1> /* find all employees who do have dependents */
2> select w.empno, w.name
3> from works w
4> where w.empno in (select d.empno
5>                   from dependent d)

+-------------+--------------------+
|empno        |name                |
+-------------+--------------------+
|        45678|Hayes, B. B.        |
|        53099|Kopecky, S.         |
|        53882|Hutton, E. B.       |
|        66890|Witton, J.          |
|        67222|White. J.           |
|        67321|McGuire, P.         |
|        69820|Landry, W.          |
1> /* find all employees who do NOT have dependents */
2> select w.empno, w.name
3> from works w
4> where w.empno not in (select d.empno
5>                       from dependent d)
```

```
+-------------+--------------------+
|empno        |name                |
+-------------+--------------------+
|        12305|Robinson, S. R.     |
|        12340|Brooks, C. P.       |
|        33399|Mandic, L.          |
|        33889|Huber, J.           |
|        41400|Green, C.           |
|        41411|Verducci, M.        |
|        45432|Kosher, P.          |
|        45454|Cameron, L.         |
```

```
1> /* select name and a kinship column with NULLs for
2>    all those employees who do not have dependents */
3> select w.name, kinship=NULL
4> from works w
5> where w.empno not in (select d.empno
6>                       from dependent d)
```

```
+--------------------+------+
|name                |kinshi|
+--------------------+------+
|Robinson, S. R.     |      |
|Brooks, C. P.       |      |
|Mandic, L.          |      |
|Huber, J.           |      |
|Green, C.           |      |
|Verducci, M.        |      |
|Kosher, P.          |      |
  1> /* select the name, and kinship if any of those employees
  2>    who have no dependents, or who have a daughter */
  3> (select w.name, kinship=NULL
  4> from works w
  5> where w.empno not in (select d.empno
  6>                       from dependent d))
  7> union
  8> (select dw.name, dd.kinship
  9>  from works dw, dependent dd
 10>  where (dw.empno = dd.empno) and dd.kinship like
  'Daughter')

+--------------------+---------------+
|name                |kinship        |
+--------------------+---------------+
|Adams, E.           |               |
|Aitken, J. A.       |               |
|Appelton, E.        |               |
|Blumberg, Z.        |Daughter       |
|Brooks, C. P.       |               |
|Brown, C.           |               |
|Cameron, L.         |               |
|Clerk, D.           |               |
|Curan, A.           |               |
```

- list the names and kinship of all employees who have children

```
1> /* select the name, and kinship of all employees who have
2>    children */
3> (select dw.name, dd.kinship
4>  from works dw, dependent dd
5>  where (dw.empno = dd.empno) and (dd.kinship like
 'Daughter')
6>                                  or
7>                                  (dd.kinship like 'Son') )

+--------------------+---------------+
|name                |kinship        |
+--------------------+---------------+
|Hayes, B. B.        |Daughter       |
|Brooks, C. P.       |Son            |
|Hayes, B. B.        |Son            |
|Johnston, K. M.     |Son            |
|Witton, J.          |Son            |
```

- list all customers who have a loan and an account

```
1> /* list all customers who have a loan and an account*/
2> select d.name
3> from deposit d, borrow l
4> where (d.name = l.name) and (d.dob = l.dob)
```

```
+-------------------------+
|name                     |
+-------------------------+
|Adams, E.                |
|Adibe, A.                |
|Adibe, B.                |
|Ahsan, F. G.             |
|Bohdarm, K. R.           |
|Bohdarm, K. R.           |
|Brooks, C. P.            |
|Brown, C.                |
```

```
1> /* list all customers who have a loan and an account*/
2> select d.name, d.accntno, l.loanno
3> from deposit d, borrow l
4> where (d.name = l.name) and (d.dob = l.dob)
```

```
+-------------------------+-------------+-------------+
|name                     |accntno      |loanno       |
+-------------------------+-------------+-------------+
|Adams, E.                |       820333|        28016|
|Adibe, A.                |      9514146|        91047|
|Adibe, B.                |      9514146|        91047|
|Ahsan, F. G.             |      1378785|        13038|
|Bohdarm, K. R.           |       485853|         4043|
|Bohdarm, K. R.           |       755561|         4043|
|Brooks, C. P.            |       520199|        52010|
|Brown, C.                |       720335|        12031|
1> /* list all customers who have a loan and an account
2>    also list their accntno with balance and loanno with
 amt */
3> (select d.name, d.accntno, a.bal
4> from deposit d, account a
5> where d.accntno = a.accntno)
6> union
7> (select b.name, b.loanno, l.amt
8> from borrow b, loans l
9> where b.loanno = l.loanno)

+-------------------------+-------------+--------------------+
|name                     |accntno      |bal                 |
+-------------------------+-------------+--------------------+
|Adams, E.                |        28016|            $1300.00|
|Adams, E.                |       820333|             $200.00|
|Adibe, A.                |        91047|            $5400.00|
|Adibe, A.                |      9514146|              $35.76|
|Adibe, B.                |        91047|            $5400.00|
```

```
 1> /* select the name acctno and balance, or the loanno and amt
 2>    for all account and loan owners, also indicate whether the
 3>    entry is an account or loan */
 4> (select d.name, d.accntno, a.bal, type='acct'
 5> from deposit d, account a
 6> where d.accntno = a.accntno)
 7> union
 8> (select b.name, b.loanno, l.amt, type='loan'
 9> from borrow b, loans l
10> where b.loanno = l.loanno)

+-------------------------+-------------+--------------------+------+
|name                     |accntno      |bal                 |type  |
+-------------------------+-------------+--------------------+------+
|Adams, E.                |        28016|            $1300.00|loan  |
|Adams, E.                |       820333|             $200.00|acct  |
|Adibe, A.                |        91047|            $5400.00|loan  |
|Adibe, A.                |      9514146|              $35.76|acct  |
```