TDDD12 Databasteknik
Föreläsning 2: Relationsalgebra och sql

av Juha Takkinen 2008-04-03
Institutionen för datavetenskap (IDA)
Linköpings universitet

Ljusbilderna baserade på Elmasri & Navathes original

Chapter 6

The Relational Algebra and Calculus

Chapter Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations From Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
  - Examples of Queries in Relational Algebra

Relational Algebra Overview

- Relational algebra is the basic set of operations for the relational model
- These operations enable a user to specify **basic retrieval requests** (or **queries**)
- The result of an operation is a *new relation*, which may have been formed from one or more *input* relations
  - This property makes the algebra "closed" (all objects in relational algebra are relations)

Relational Algebra Overview (continued)

- The **algebra operations** thus produce new relations
  - These can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a **relational algebra expression**
  - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)

Brief History of Origins of Algebra

- Muhammad ibn Musa al-Khwarizmi (800-847 CE) wrote a book titled al-jabr about arithmetic of variables
  - Book was translated into Latin.
  - Its title (al-jabr) gave Algebra its name.
- Al-Khwarizmi called variables "shay"
  - "Shay" is Arabic for "thing".
  - Spanish transliterated "shay" as "xay" ("x" was "sh" in Spain).
  - In time this word was abbreviated as x.
- Where does the word Algorithm come from?
  - Algorithm originates from "al-Khwarizmi"
  - Reference: PBS (http://www.pbs.org/empires/islam/innoalgebra.html)

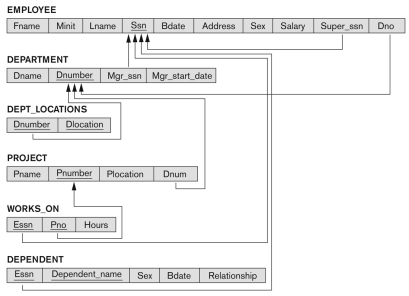
Relational Algebra Overview

- Relational Algebra consists of several groups of operations
  - Unary Relational Operations
    - SELECT (symbol: σ (sigma))
    - PROJECT (symbol: π (pi))
    - RENAME (symbol: ρ (rho))
  - Relational Algebra Operations From Set Theory
    - UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, –)
    - CARTESIAN PRODUCT ($\times$)
  - Binary Relational Operations
    - JOIN (several variations of JOIN exist)
    - DIVISION
  - Additional Relational Operations
    - OUTER JOINS, OUTER UNION
    - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

Database Schema for COMPANY

All examples discussed below refer to the COMPANY database shown here.

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Unary Relational Operations: SELECT

- The SELECT operation (denoted by σ (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
  - The selection condition acts as a **filter**
  - Keeps only those tuples that satisfy the qualifying condition
  - Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)
- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{DNO = 4} \text{ (EMPLOYEE)}$$

  - Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{SALARY > 30,000} \text{ (EMPLOYEE)}$$

Unary Relational Operations: SELECT

- In general, the *select* operation is denoted by $\sigma_{\text{<selection condition>}}(R)$ where
  - the symbol σ (sigma) is used to denote the *select* operator
  - the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
  - tuples that make the condition **true** are selected
    - appear in the result of the operation
  - tuples that make the condition **false** are filtered out
    - discarded from the result of the operation

The following query results refer to this database state

Figure 5.6
One possible database state for the COMPANY relational database schema.

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

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

Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by π (pi)
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
  - PROJECT creates a vertical partitioning
    - The list of specified columns (attributes) is kept in each tuple
    - The other attributes in each tuple are discarded
- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{LNAME, FNAME, SALARY}(\text{EMPLOYEE})$$

Unary Relational Operations: PROJECT (cont'd)

- The general form of the *project* operation is:

$$\pi_{<\text{attribute list}>}(R)$$

  - π (pi) is the symbol used to represent the *project* operation
  - <attribute list> is the desired list of attributes from relation R.
- The project operation *removes any duplicate tuples*
  - This is because the result of the *project* operation must be a *set of tuples*
    - Mathematical sets *do not allow* duplicate elements.

Unary Relational Operations: PROJECT (cont'd)

- PROJECT Operation Properties
  - The number of tuples in the result of projection $\pi_{<\text{list}>}(R)$ is always less or equal to the number of tuples in R
    - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
  - PROJECT is *not* commutative
    - $\pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R)) = \pi_{<\text{list1}>}(R)$ as long as <list2> contains the attributes in <list1>

Examples of applying SELECT and PROJECT operations

Figure 6.1
Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \text{ AND } Salary>25000) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}$ (EMPLOYEE). (b) $\pi_{Lname, Fname, Salary}$(EMPLOYEE). (c) $\pi_{Sex, Salary}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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

(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - DEP5_EMPS $\leftarrow \sigma_{\text{DNO=5}}$(EMPLOYEE)
  - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)

Unary Relational Operations: RENAME

- The general RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_{S(B1, B2, ..., Bn)}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1, B2, ..., Bn)}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Unary Relational Operations: RENAME (cont'd)

- For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:
  - If we write:
    - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)
    - RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)
- If we write:
  - RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)$\leftarrow \rho_{\text{RESULT (F,M,L,S,B,A,SX,SAL,SU, DNO)}}$ (DEP5_EMPS)
  - The 10 attributes of DEP5_EMPS are *renamed* to F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

Relational Algebra Operations from Set Theory: UNION

- UNION Operation
  - Binary operation, denoted by $\cup$
  - The result of R $\cup$ S, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be "type compatible" (or UNION compatible)
    - R and S must have same number of attributes
    - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Relational Algebra Operations from Set Theory: UNION

- Example:
  - To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)
  - We can use the UNION operation as follows:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO}=5}\,(\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{SSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT2(SSN)} \leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

  - The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

Example of the result of a UNION operation

- UNION Example

Figure 6.3
Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION $\cup$, (also for INTERSECTION $\cap$, and SET DIFFERENCE –, see next slides)
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- The resulting relation for R1$\cup$R2 (also for R1$\cap$R2, or R1–R2, see next slides) has the same attribute names as the *first* operand relation R1 (by convention)

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by $\cap$
- The result of the operation R $\cap$ S, is a relation that includes all tuples that are in both R and S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont'd)

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes all tuples that are in R but not in S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Example result of UNION, INTERSECT, and DIFFERENCE

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Figure 6.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT − INSTRUCTOR. (e) INSTRUCTOR − STUDENT.

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

- CARTESIAN (or CROSS) PRODUCT Operation
  - This operation is used to combine tuples from two relations in a combinatorial fashion.
  - Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—one from R and one from S.
  - Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R x S will have $n_R * n_S$ tuples.
  - The two operands do NOT have to be "type compatible"

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont'd)

- Generally, CROSS PRODUCT is not a meaningful operation
  - Can become meaningful when followed by other operations
- Example (not meaningful):
  - FEMALE_EMPS ← $\sigma_{SEX=F}$(EMPLOYEE)
  - EMPNAMES ← $\pi_{FNAME, LNAME, SSN}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS ← EMPNAMES x DEPENDENT
- EMP_DEPENDENTS will contain every combination of EMPNAMES and DEPENDENT
  - whether or not they are actually related

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont'd)

- To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows
- Example (meaningful):
  - FEMALE_EMPS ← $\sigma_{SEX=F}$(EMPLOYEE)
  - EMPNAMES ← $\pi_{FNAME, LNAME, SSN}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS ← EMPNAMES x DEPENDENT
  - ACTUAL_DEPS ← $\sigma_{SSN=ESSN}$(EMP_DEPENDENTS)
  - RESULT ← $\pi_{FNAME, LNAME, DEPENDENT\_NAME}$ (ACTUAL_DEPS)
- RESULT will now contain the name of female employees and their dependents

Example of applying CARTE-SIAN PRODUCT

Figure 6.5
The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

Binary Relational Operations: JOIN

- JOIN Operation (denoted by ⋈)
  - The sequence of CARTESIAN PRODECT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  - A special operation, called JOIN combines this sequence into a single operation
  - This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
  - The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:
    $R \bowtie_{\text{<join condition>}} S$
  - where R and S can be any relations that result from general *relational algebra expressions.*

Binary Relational Operations: JOIN (cont'd)

- Example: Suppose that we want to retrieve the name of the manager of each department.
  - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
  - We do this by using the join $\bowtie$ operation.
  - DEPT_MGR ← DEPARTMENT $\bowtie_{MGRSSN=SSN}$ EMPLOYEE
- MGRSSN=SSN is the join condition
  - Combines each department record with the employee who manages the department
  - The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

Example of applying the JOIN operation

DEPT_MGR

Dname	Dnumber	Mgr_ssn	···	Fname	Minit	Lname	Ssn	···
Research	5	333445555	···	Franklin	T	Wong	333445555	···
Administration	4	987654321	···	Jennifer	S	Wallace	987654321	···
Headquarters	1	888665555	···	James	E	Borg	888665555	···

Figure 6.6
Result of the JOIN operation

DEPT_MGR ← DEPARTMENT $\bowtie_{MGRSSN=SSN}$ EMPLOYEE

Some properties of JOIN

- The general case of JOIN operation is called a Theta-join:
  R $\bowtie_{theta}$ S
- The join condition is called *theta*
- *Theta* can be any general boolean expression on the attributes of R and S; for example:
  - R.Ai<S.Bj AND (R.Ak=S.Bl OR R.Ap<S.Bq)
- Most join conditions involve one or more equality conditions "AND"ed together; for example:
  - R.Ai=S.Bj AND R.Ak=S.Bl AND R.Ap=S.Bq

Binary Relational Operations: EQUIJOIN

- EQUIJOIN Operation
- The most common use of join involves join conditions with *equality comparisons* only
- Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple.
  - The JOIN seen in the previous example was an EQUIJOIN.

Binary Relational Operations: NATURAL JOIN Operation

- NATURAL JOIN Operation
  - Another variation of JOIN called NATURAL JOIN — denoted by * — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
    - because one of each pair of attributes with identical values is superfluous
  - The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the same name* in both relations
  - If this is not the case, a renaming operation is applied first.

Binary Relational Operations NATURAL JOIN (cont'd)

- Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:
  - DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS
- Only attribute with the same name is DNUMBER
- An implicit join condition is created based on this attribute: DEPARTMENT.DNUMBER=DEPT_LOCATIONS.DNUMBER
- Another example: Q ← R(A,B,C,D) * S(C,D,E)
  - The implicit join condition includes *each pair* of attributes with the same name, "AND"ed together:
    - R.C=S.C AND R.D.S.D
  - Result keeps only one attribute of each such pair:
    - Q(A,B,C,D,E)

Example of NATURAL JOIN operation

(a)

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

DEPT_LOCS

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

Figure 6.7
Results of two NATURAL JOIN operations.
(a) PROJ_DEPT ← PROJECT * DEPT.
(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

Binary Relational Operations: DIVISION

- DIVISION Operation
  - The division operation is applied to two relations
  - R(Z) ÷ S(X), where X subset Z. Let Y = Z - X (and hence Z = X ∪ Y); that is, let Y be the set of attributes of R that are not attributes of S.
  - The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with
    - t_R [X] = t_s *for every tuple* t_s in S.
  - For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S.

Example of DIVISION

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

Figure 6.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

Recap of Relational Algebra Operations

Table 6.1
Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 * _{\langle \text{join condition} \rangle} R_2$, OR $R_1 * _{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$, OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Additional Relational Operations: Aggregate Function Operation

- Use of the Aggregate Functional operation $\mathcal{F}$
  - $\mathcal{F}_{\text{MAX Salary}}$ (EMPLOYEE) retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE) retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{SUM Salary}}$ (EMPLOYEE) retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE) computes the count (number) of employees and their average salary
    - Note: count just counts the number of rows, without removing duplicates

Using Grouping with Aggregation

- The previous examples all summarized one or more attributes for a set of tuples
  - Maximum Salary or Count (number of) Ssn
- Grouping can be combined with Aggregate Functions
- Example: For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol
  - $_{\text{DNO}}\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE)
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department

Examples of applying aggregate functions and grouping

Figure 6.10
The aggregate function operation.
(a) $\rho_{R(Dno, No\_of\_employees, Average\_sal)}(_{Dno}\mathfrak{Z}_{COUNT\ Ssn,\ AVERAGE\ Salary}(EMPLOYEE))$.
(b) $_{Dno}\mathfrak{Z}_{COUNT\ Ssn,\ AVERAGE\ Salary}(EMPLOYEE)$.
(c) $\mathfrak{Z}_{COUNT\ Ssn,\ AVERAGE\ Salary}(EMPLOYEE)$.

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Illustrating aggregate functions and grouping

Figure 8.6
Results of GROUP BY and HAVING. (a) Q24. (b) Q26.

(a)

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Borg	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

Additional Relational Operations (cont'd)

- The OUTER JOIN Operation
  - In NATURAL JOIN and EQUIJOIN, tuples without a *matching* (or *related*) tuple are eliminated from the join result
    - Tuples with null in the join attributes are also eliminated
    - This amounts to loss of information.
- The left outer join operation keeps <u>every tuple</u> in the first or left relation R in R ⟕ S; if no matching tuple is found in S, then the attributes of S in the join result are filled or "padded" with null values.
- A similar operation, right outer join, keeps every tuple in the second or right relation S in the result of R ⟖ S.
- A third operation, full outer join, denoted by ⟗ keeps all tuples <u>in both the left and the right relations</u> when no matching tuples are found, padding them with null values as needed.

Additional Relational Operations (cont'd)

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Figure 6.12
The result of a LEFT OUTER JOIN operation.

Additional Relational Operations (cont'd)

- OUTER UNION Operations
  - The outer union operation was developed to take the union of tuples from two relations if the relations are *not type compatible*.
  - This operation will take the union of tuples in two relations R(X, Y) and S(X, Z) that are **partially compatible**, meaning that only some of their attributes, say X, are type compatible.
  - The attributes that are type compatible are represented only once in the result, and those attributes that are not type compatible from either relation are also kept in the result relation T(X, Y, Z).

Additional Relational Operations (cont'd)

- Example: An outer union can be applied to two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank).
  - Tuples from the two relations are matched based on having the same combination of values of the shared attributes— Name, SSN, Department.
  - If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.
  - The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

Examples of Queries in Relational Algebra: Procedural Form

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

  RESEARCH_DEPT ← $\sigma_{DNAME='Research'}$ (DEPARTMENT)

  RESEARCH_EMPS ← (RESEARCH_DEPT $\bowtie_{DNUMBER=DNO(SUPERVISOR)}$ EMPLOYEE)

  RESULT ← $\pi_{FNAME, LNAME, ADDRESS}$ (RESEARCH_EMPS)

- **Q6: Retrieve the names of employees who have no dependents.**

  ALL_EMPS ← π_{SSN}(EMPLOYEE)

  EMPS_WITH_DEPS(SSN) ← π_{ESSN}(DEPENDENT)

  EMPS_WITHOUT_DEPS ← (ALL_EMPS - EMPS_WITH_DEPS)

  RESULT ← $\pi_{LNAME, FNAME}$ (EMPS_WITHOUT_DEPS * EMPLOYEE)

Examples of Queries in Relational Algebra – Single expressions

As a single expression, these queries become:

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

  $\pi_{Fname, Lname, Address}$ ($\sigma_{Dname= 'Research'}$ (DEPARTMENT $\bowtie_{Dnumber=Dno}$(EMPLOYEE))

- **Q6: Retrieve the names of employees who have no dependents.**

  $\pi_{Lname, Fname}$((π_{Ssn} (EMPLOYEE) – ρ_{Ssn} (π_{Essn} (DEPENDENT))) * EMPLOYEE)

Chapter 8

SQL-99: SchemaDefinition, Constraints, and Queries and Views

Data Definition, Constraints, and Schema Changes

- Used to CREATE, DROP, and ALTER the descriptions of the tables (relations) of a database

CREATE TABLE

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE DEPARTMENT (
    DNAME            VARCHAR(10) NOT
NULL,
    DNUMBER      INTEGER      NOT NULL,
    MGRSSN       CHAR(9),
    MGRSTARTDATE     CHAR(9)   );
```


CREATE TABLE

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

```
CREATE TABLE DEPT (
  DNAME              VARCHAR(10)    NOT NULL,
  DNUMBER        INTEGER        NOT NULL,
  MGRSSN         CHAR(9),
  MGRSTARTDATE   CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP  );
```

DROP TABLE

- Used to remove a relation (base table) and its definition
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

```
DROP TABLE  DEPENDENT;
```


ALTER TABLE

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

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

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

Features Added in SQL2 and SQL-99

- Create schema
- Referential integrity options
  - RESTRICT, CASCADE, SET NULL, and SET DEFAULT on foreign keys

REFERENTIAL INTEGRITY OPTIONS

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

```
CREATE TABLE DEPT (
   DNAME          VARCHAR(10)    NOT NULL,
      DNUMBER        INTEGER        NOT NULL,
      MGRSSN         CHAR(9),
      MGRSTARTDATE   CHAR(9),
      PRIMARY KEY (DNUMBER),
      UNIQUE (DNAME),
      FOREIGN KEY (MGRSSN) REFERENCES EMP
   ON DELETE SET DEFAULT ON UPDATE CASCADE);
```


REFERENTIAL INTEGRITY OPTIONS (continued)

```
CREATE TABLE EMP(
  ENAME       VARCHAR(30)    NOT NULL,
  ESSN        CHAR(9),
  BDATE       DATE,
  DNO         INTEGER  DEFAULT 1,
  SUPERSSN    CHAR(9),
  PRIMARY KEY (ESSN),
  FOREIGN KEY (DNO) REFERENCES DEPT
   ON DELETE SET DEFAULT ON UPDATE
  CASCADE,
  FOREIGN KEY (SUPERSSN) REFERENCES
  EMP ON DELETE SET NULL ON UPDATE
  CASCADE);
```


Retrieval Queries in SQL

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

Retrieval Queries in SQL (cont'd)

- A **bag** or **multi-set** is like a set, but an element may appear more than once.
  - Example: {A, B, C, A} is a bag. {A, B, C} is also a bag that also is a set.
  - Bags also resemble lists, but the order is irrelevant in a bag.
- Example:
  - {A, B, A} = {B, A, A} as bags
  - However, [A, B, A] is not equal to [B, A, A] as lists

Retrieval Queries in SQL (cont'd)

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

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

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

Relational Database Schema--Figure 5.5

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Popu-lated Data-base--Fig.5.6

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

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

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

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

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

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

Simple SQL Queries

- Basic SQL queries correspond to using the following operations of the relational algebra:
  - SELECT
  - PROJECT
  - JOIN
- All subsequent examples use the COMPANY database

Simple SQL Queries (cont'd)

- Example of a simple query on one relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

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

- Similar to a SELECT-PROJECT pair of relational algebra operations:
- The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition
- However, the result of the query may contain duplicate tuples

Simple SQL Queries (cont'd)

•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
```

- Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations
- (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra)
- (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

Simple SQL Queries (cont'd)

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

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

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

Aliases, * and DISTINCT, Empty WHERE-clause

•In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations*

•A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name

•Example:

•**EMPLOYEE**.LNAME, **DEPARTMENT**.DNAME

ALIASES

•Some queries need to refer to the same relation twice

- In this case, *aliases* are given to the relation name

•Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
Q8:   SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM     EMPLOYEE E S
      WHERE    E.SUPERSSN=S.SSN
```

- In Q8, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
- We can think of E and S as two different *copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

ALIASES (cont'd)

•Aliasing can also be used in any SQL query for convenience

•Can also use the AS keyword to specify aliases

```
Q8:   SELECT   E.FNAME, E.LNAME,
               S.FNAME, S.LNAME
      FROM     EMPLOYEE AS E,
               EMPLOYEE AS S
      WHERE    E.SUPERSSN=S.SSN
```


USE OF *

•To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

Examples:

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

Q1D: SELECT   *
     FROM     EMPLOYEE,
DEPARTMENT
     WHERE    DNAME='Research' AND
              DNO=DNUMBER
```

USE OF DISTINCT

•SQL does not treat a relation as a set; duplicate tuples can appear

•To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used

•For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

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


SET OPERATIONS

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

```
Q4:           (SELECT     PNAME
              FROM        PROJECT, DEPARTMENT,
                                EMPLOYEE
              WHERE       DNUM=DNUMBER AND
                    MGRSSN=SSN AND LNAME='Smith')
              UNION
              (SELECT     PNAME
              FROM        PROJECT, WORKS_ON,
EMPLOYEE
              WHERE       PNUMBER=PNO AND
                          ESSN=SSN AND NAME='Smith')
```


NESTING OF QUERIES

•A complete SELECT query, called a *nested query*, can be specified within the WHERE-clause of another query, called the *outer query*

- Many of the previous queries can be specified in an alternative form using nesting

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

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


CORRELATED NESTED QUERIES

•If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query*, the two queries are said to be *correlated*

- The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

•Query 12: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q12: SELECT     E.FNAME, E.LNAME
     FROM       EMPLOYEE AS E
     WHERE      E.SSN IN
                (SELECT     ESSN
                FROM        DEPENDENT
                WHERE       ESSN=E.SSN AND
                E.FNAME=DEPENDENT_NAME)
```


CORRELATED NESTED QUERIES (cont'd)

•In Q12, the nested query has a different result in the outer query

•A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can ***always*** be expressed as a single block (non-nested) query. For example, Q12 may be written as in Q12A

```
Q12A:         SELECT     E.FNAME, E.LNAME
              FROM            EMPLOYEE E,
DEPENDENT D
              WHERE      E.SSN=D.ESSN AND
                    E.FNAME=D.DEPENDENT_NAME
```


CORRELATED NESTED QUERIES (cont'd)

•Query 3: Retrieve the name of each employee who works on all the projects controlled by department number 5.

```
Q3:   SELECT     FNAME, LNAME
              FROM            EMPLOYEE
              WHERE (   (SELECT    PNO
                        FROM            WORKS_ON
                        WHERE     SSN=ESSN)
                              CONTAINS
                        (SELECT   PNUMBER
                        FROM            PROJECT
                        WHERE     DNUM=5) )
```

CORRELATED NESTED QUERIES (cont'd)

- In Q3, the second nested query, which is *not correlated* with the outer query, retrieves the project numbers of all projects controlled by department 5
- The first nested query, which is correlated, retrieves the project numbers on which the employee works, which is *different for each employee tuple* because of the correlation

THE EXISTS FUNCTION (cont'd)

- Query 12: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q12B:   SELECT    FNAME, LNAME
        FROM      EMPLOYEE
        WHERE     EXISTS (SELECT *
                  FROM          DEPENDENT
                  WHERE   SSN=ESSN
                  AND
FNAME=DEPENDENT_NAME)
```


THE EXISTS FUNCTION (cont'd)

- Query 6: Retrieve the names of employees who have no dependents.

```
Q6:   SELECT    FNAME, LNAME
      FROM      EMPLOYEE
      WHERE     NOT EXISTS (SELECT
*
                FROM
DEPENDENT
                WHERE     SSN=ESSN)
```

- In Q6, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If *none exist*, the EMPLOYEE tuple is selected
  - EXISTS is necessary for the expressive power of SQL

EXPLICIT SETS

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- Query 13: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q13:   SELECT   DISTINCT ESSN
       FROM     WORKS_ON
       WHERE    PNO IN (1, 2, 3)
```


NULLS IN SQL QUERIES

- SQL allows queries that check if a value is **NULL** (missing or undefined or not applicable)
- SQL uses **IS** or **IS NOT** to compare NULLs because it considers each NULL value distinct from other NULL values, so *equality comparison is not appropriate*.
- Query 14: Retrieve the names of all employees who do not have supervisors.

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

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

Joined Relations Feature in SQL2

- Can specify a "joined relation" in the FROM-clause
  - Looks like any other relation but is the result of a join
  - Allows the user to specify different types of joins (regular "theta" JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, etc)

Joined Relations Feature in SQL2 (cont'd)

•Examples:

```
Q8:   SELECT   E.FNAME, E.LNAME, S.FNAME,
  S.LNAME
      FROM     EMPLOYEE E S
      WHERE    E.SUPERSSN=S.SSN
```

•can be written as:

```
Q8:   SELECT   E.FNAME, E.LNAME, S.FNAME,
  S.LNAME
      FROM     (EMPLOYEE E LEFT OUTER
  JOIN         EMPLOYEES ON
  E.SUPERSSN=S.SSN)
```


Joined Relations Feature in SQL2 (cont'd)

•Examples:

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

•could be written as:

```
Q1:   SELECT   FNAME, LNAME, ADDRESS
      FROM     (EMPLOYEE JOIN
  DEPARTMENT
               ON DNUMBER=DNO)
      WHERE    DNAME='Research'
```

•or as:

```
Q1:   SELECT   FNAME, LNAME, ADDRESS
      FROM     (EMPLOYEE NATURAL JOIN
  DEPARTMENT
               AS DEPT(DNAME, DNO, MSSN,
  MSDATE)
      WHERE    DNAME='Research'
```


Joined Relations Feature in SQL2 (cont'd)

•Another Example: Q2 could be written as follows; this illustrates multiple joins in the joined tables

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


AGGREGATE FUNCTIONS

•Include **COUNT, SUM, MAX, MIN, and AVG**

•Query 15: Find the maximum salary, the minimum salary, and the average salary among all employees.

```
Q15:  SELECT   MAX(SALARY),
               MIN(SALARY), AVG(SALARY)
      FROM     EMPLOYEE
```

•Some SQL implementations *may not allow more than one function* in the SELECT-clause

AGGREGATE FUNCTIONS (cont'd)

•Queries 17 and 18: Retrieve the total number of employees in the company (Q17), and the number of employees in the 'Research' department (Q18).

```
Q17:  SELECT   COUNT (*)
      FROM     EMPLOYEE

Q18:  SELECT   COUNT (*)
      FROM     EMPLOYEE,
  DEPARTMENT
      WHERE    DNO=DNUMBER AND
               DNAME='Research'
```


GROUPING (cont'd)

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

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

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

GROUPING (cont'd)

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

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

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

THE HAVING-CLAUSE

•Sometimes we want to retrieve the values of these functions for only those *groups that satisfy certain conditions*

•The **HAVING**-clause is used for specifying a selection condition on groups (rather than on individual tuples)

THE HAVING-CLAUSE (cont'd)

•Query 22: For each project *on which more than two employees work*, retrieve the project number, project name, and the number of employees who work on that project.

```
Q22:  SELECT   PNUMBER, PNAME,
               COUNT(*)
      FROM     PROJECT, WORKS_ON
      WHERE    PNUMBER=PNO
      GROUP BY PNUMBER, PNAME
      HAVING   COUNT (*) > 2
```


SUBSTRING COMPARISON

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

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


SUBSTRING COMPARISON (cont'd)

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

- Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_____5_', with each underscore as a place holder for a single arbitrary character.

```
Q26:  SELECT   FNAME, LNAME
      FROM     EMPLOYEE
      WHERE    BDATE LIKE '_____5_'
```

•The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible

- Hence, in SQL, character string attribute values are not atomic

ARITHMETIC OPERATIONS

•The standard arithmetic operators **'+', '-'. '*', and '/'** (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result

•Query 27: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

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

ORDER BY

•The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)

•Query 28: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

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


ORDER BY (cont'd)

•The default order is in ascending order of values

•We can specify the keyword **DESC** if we want a descending order; the keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

Summary of SQL Queries

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

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


Summary of SQL Queries (cont'd)

•The SELECT-clause lists the attributes or functions to be retrieved

•The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries

•The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause

•GROUP BY specifies grouping attributes

•HAVING specifies a condition for selection of groups

•ORDER BY specifies an order for displaying the result of a query

- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

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

INSERT

•Example:

```
U1:     INSERT INTO     EMPLOYEE
        VALUES ('Richard','K','Marini', '653298653', '30-
DEC-52',
        '98 Oak Forest,Katy,TX', 'M', 37000,'987654321',
4 )
```

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

- Attributes with NULL values can be left out

•Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
U1A:    INSERT INTO     EMPLOYEE (FNAME,
LNAME,                                SSN)
        VALUES ('Richard', 'Marini', '653298653')
```


DELETE

•Removes tuples from a relation

- Includes a WHERE-clause to select the tuples to be deleted
- Referential integrity should be enforced
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

DELETE (cont'd)

•Examples:

```
U4A:  DELETE FROM  EMPLOYEE
      WHERE        LNAME='Brown'

U4B:  DELETE FROM  EMPLOYEE
      WHERE        SSN='123456789'

U4C:  DELETE FROM  EMPLOYEE
      WHERE        DNO IN
                   (SELECT DNUMBER
                           FROMDEPARTMENT
                           WHERE
                   DNAME='Research')

U4D:  DELETE FROM  EMPLOYEE
```


UPDATE

•Used to modify attribute values of one or more selected tuples

•A WHERE-clause selects the tuples to be modified

•An additional SET-clause specifies the attributes to be modified and their new values

•Each command modifies tuples *in the same relation*

•Referential integrity should be enforced

UPDATE (cont'd)

•Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
U5:  UPDATE  PROJECT
     SET     PLOCATION = 'Bellaire',
             DNUM = 5
     WHERE   PNUMBER=10
```


UPDATE (cont'd)

•Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6:  UPDATE   EMPLOYEE
     SET      SALARY = SALARY *1.1
     WHERE    DNO IN (SELECT     DNUMBER
                 FROM   DEPARTMENT
                 WHERE  DNAME='Research')
```

•In this request, the modified SALARY value depends on the original SALARY value in each tuple

- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Chapter 9

Introduction to SQL Programming Techniques:
Views (and Triggers; see lecture 6)

LiU expanding reality

Views in SQL

•A view is a "virtual" table that is derived from other tables

•Allows for limited update operations

- Since the table may not physically be stored

•Allows full query operations

•A convenience for expressing certain operations

Specification of Views

- SQL command: **CREATE VIEW**
  - a table (view) name
  - a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
  - a query to specify the table contents

SQL Views: An Example

- Specify a different WORKS_ON table

```
CREATE VIEW WORKS_ON_NEW AS
SELECT FNAME, LNAME, PNAME, HOURS
    FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND PNO=PNUMBER
    GROUP BY PNAME;
```


Using a Virtual Table

- We can specify SQL queries on a newly create table (view):

```
SELECT FNAME, LNAME
    FROM WORKS_ON_NEW
    WHERE PNAME='Seena';
```

- When no longer needed, a view can be dropped:

```
DROP WORKS_ON_NEW;
```