Outline of Lecture

- Review of Relational Data Model
- Relational Algebra Operations
  - SELECT σ and PROJECT Π
  - Set Operations
  - JOIN Operations
  - Additional Relational Operations

Review of Relational Data Model

Relational Model

- A particular way of structuring data (relations)
- Simple
- Mathematically based
  - Expressions (queries) can be analyzed by DBMS
  - Transformed to equivalent expressions automatically (query optimization)
    - Optimizers have limits (=> programmer needs to know how queries are evaluated and optimized)

Definition Summary

Informal Terms	**Formal Term**
Table	Relation
Column	Attribute/Domain
Values in a column	Domain
Row	Tuple/Instance
Table Definition	Schema of Relation
Populated Table	Extension

Relation Instance

- Relation is a set of tuples
  - Tuple ordering immaterial
  - No duplicates
- All tuples in a relation have the same structure; constructed from the same set of attributes
  - Attributes named (=> ordering immaterial)
  - Value of an attribute drawn from the attribute's domain
  - Arity (degree) = number of attributes

Relation Instance (Example)

Id	Name	Address	Status
1111111	John	123 main	freshman
2345678	Mary	456 cedar	sophmore
4433322	Art	77 so. 3rd	senior
7654321	Pat	88 no. 4th	sophmore

Student

Data Structures of Relation

- an attribute name refers to a position in a tuple by **name** rather than position
- an attribute name indicate the **role of a domain** in a relation
- attribute names must be **unique** within relations
- by using attribute names we can forget the ordering of field values in tuples
- a relation definition includes the following $R(A_1{:}D_1, A_2 {:}D_2 , ..., A_n {:}D_n)$
  - Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)

Relation Schema

- Relation name
- Attribute names and domains
- Integrity constraints - e.g.,
  - The values of a particular attribute in all tuples are unique
- Default values

Relational Database

- Finite set of relations
- Each relation consists of a schema and an instance
- Database schema = set of relation schemas (and integrity constrains IC)
- Database state = set of (corresponding) relation states and each r satisfies constrains

Integrity Constraints

- Constraints: ***conditions*** that must hold on ***all*** valid relation instances.
- Tree main constrains:
  - **Key** constraints (single relation)
  - **entity integrity** constraints (single relation)
  - **referential integrity** constraints (two relations)

Key Constrains

- **Superkey**: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK
  - for any distinct tuples t1 and t2 in r(R),
    t1[SK] <> t2[SK].
- **Key:** A "**minimal**" superkey.
  - a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

Example:

CAR(<u>State</u>, <u>Reg#</u>, SerialNo, Make, Model, Year)

- Two **candidate keys**:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
  - Are Key1 and Key2 superkeys?
- {SerialNo, Make}, key or superkey?
- **Primary key**: If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are *underlined*.

Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database.
  - S is the *name* of the **database**.
    S = {R1, R2, ..., Rn}
- **Entity Integrity:**The ***primary key*** *attributes* PK of each relation schema R in S **cannot** have **null** values in any tuple of r(R).
  - primary key values are used to *identify* the individual tuples.
    - t[PK] <> null for any tuple t in r(R)

Referential Integrity
(Foreign Key Constrains)

- A constraint involving *two* relations
- specify a *relationship* among tuples in two relations:
  - the **referencing relation(R1)** and the **referenced relation (R2)**
  - **FK(foreign key** attributes) of R1 reference **PK** of R2
- displayed in a relational database schema as a directed arc from R1.FK to R2.PK

Foreign Key Constrain

- Satisfy 2 conditions:
  1. $Dom(R_1.FK) = Dom(R_2.PK)$
  2. $t_1[FK] = t_2\{PK]$ or
     $t_1[FK] = NULL$

Figure 7.6 One possible relational database state corresponding to the COMPANY schema.

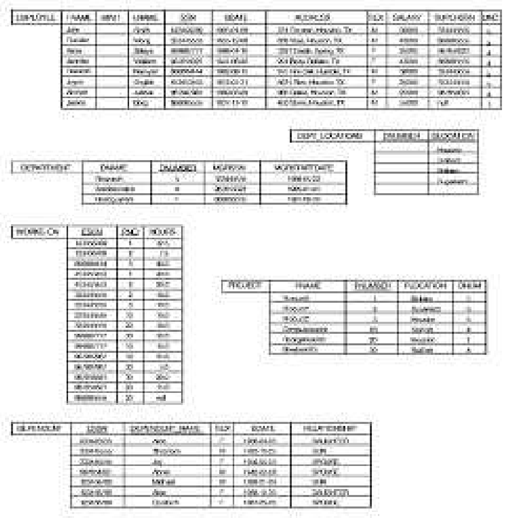
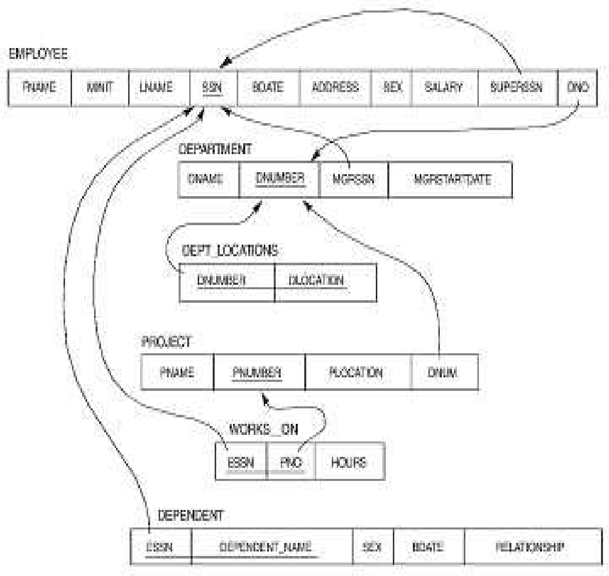


Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



AIRPORT

airportcode	name	city	state

FLT-SCHEDULE

flt#	airline	dtime	from-airportcode	atime	to-airportcode	miles	price

FLT-WEEKDAY

flt#	weekday

FLT-INSTANCE

flt#	date	plane#	#avail-seats

AIRPLANE

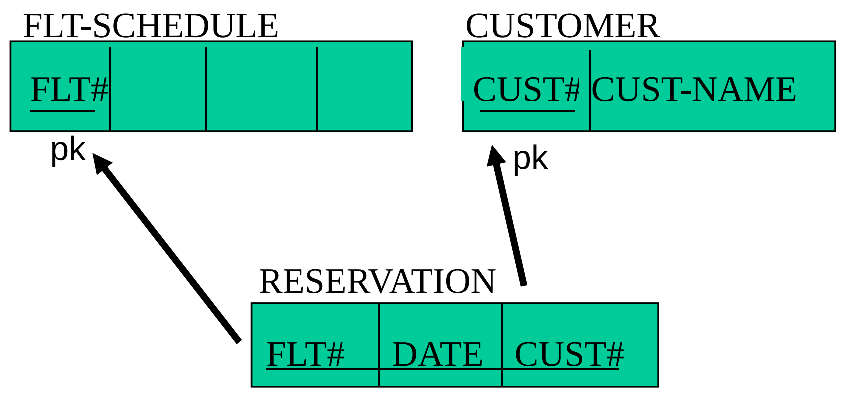
plane#	plane-type	total-#seats

CUSTOMER

cust#	first	middle	last	phone#	street	city	state	zip

RESERVATION

flt#	date	cust#	seat#	check-in-status	ticket#



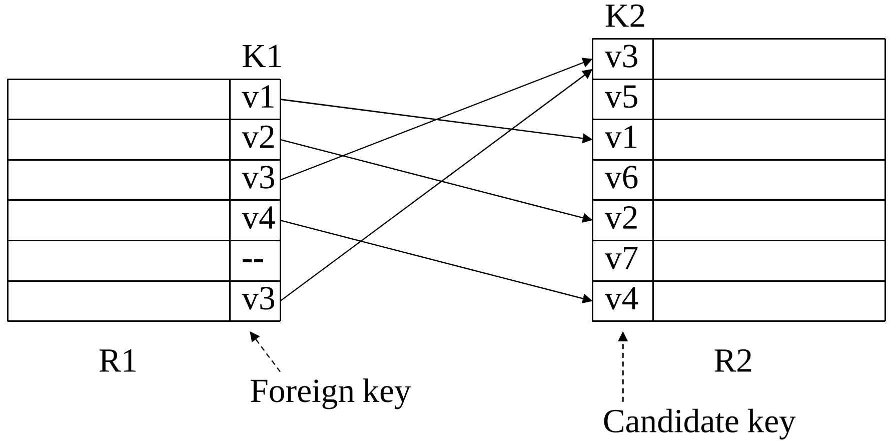
Database Schema (Example)

- Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)
- Professor (Id: INT, Name: STRING, DeptId: DEPTS)
- Course (DeptId: DEPTS, CrsName: STRING, CrsCode: COURSES)
- Transcript (CrsCode: COURSES, StudId: INT, Grade: GRADES, Semester: SEMESTERS)
- Department(DeptId: DEPTS, Name: STRING)

Foreign Key Constraint(Cont.)

- **Referential integrity -** attribute named in one relation must correspond to tuple(s) in another that describes the item
  - Transcript (CrsCode) references Course(CrsCode)
  - Professor(DeptId) references Department(DeptId)
- K1 is a **foreign key** of R1 referring to K2 in R2
  - if v is a value of K1, there is a *unique* tuple of R2 in which K2 has value v
  - This is a special case of referential integrity: K2 must be a candidate key of R2 (CrsCode is a key of Course)
  - If no row exists in R2 -- violation of referential integrity
  - Not all rows of R2 need to be referenced. Relationship is not symmetric (some course might not be taught)
  - Value of a foreign key might not be specified (DeptId column of some professor might be null)

Foreign Key Constraint (Example)



Foreign Key (con't)

- Names of K1 and K2 need not be the same.
  - With tables:

    Teaching(CrsCode: COURSES, Sem: SEMESTERS, ProfId: INT)
    Professor(Id: INT, Name: STRING, DeptId: DEPTS)

  ProfId attribute of Teaching references Id attribute of Professor
- R1 and R2 need not be distinct.
  - Employee(Id:INT, MgrId:INT, ….)
    - Employee(MgrId) references Employee(Id)
  - Every manager is also an employee and hence has a unique row in Employee

Foreign Key (con't)

- Foreign key might consist of several columns
  - (CrsCode, Semester) of Transcript references (CrsCode, Sem) of Teaching
- R1(A1, …An) references R2(B1, …Bn)
  - There exists a 1:1 relationship between A1,…An and B1,…Bn
  - Ai and Bi have same domains (although not necessarily the same names)
  - B1,…Bn is a candidate key of R2

Relational Algebra

- the Relational Algebra is procedural; you tell it how to construct the result
- it consists of a set of **operators** which, when applied to relations, yield relations
- Operations:
  - SELECT σ and PROJECT Π operations
  - Set operations: UNION U, INTERSECTION ∩, DIFFERENCE -, CARTESIAN PRODUCT X.
  - JOIN operations X
  - Others: DIVISION, OUTER JOIN, AGGREGATE FUNCTIONS.

SELECT

- Selects the tuples (rows) from a relation R that satisfy a certain *selection condition* c
- Form of the operation: $\sigma_c(R)$
- The condition c is an arbitrary Boolean expression on the attributes of R
- Resulting relation has the *same attributes* as R
- Resulting relation includes each tuple in r(R) whose attribute values satisfy the condition c

Selection

FLT-WEEKDAY

flt#	weekday

- "Retrieve (flt#, weekday) for all flights scheduled for Mondays"

  $\sigma_{weekday=MO}$ (FLT-WEEKDAY)
- the expression in $\sigma_{expression}$ (R) involves:
- operands: constants or attribute names of R
- comparison operators: $<$ $>$ $=$
- logical operators: $\vee$ $\wedge$ $\neg$
- nesting: ( )

Projection ($\boldsymbol{\Pi}$)

- Keeps only certain attributes (columns) from a relation R specified in an *attribute list* L
- Form of operation: $\boldsymbol{\Pi}_{L}(R)$
- Resulting relation has only those attributes of R specified in L

Examples of Projection

- e.g.1, $\boldsymbol{\Pi}_{\text{FNAME,LNAME,SALARY}}$(EMPLOYEE)
  - It **eliminates duplicate** tuples in the resulting relation (remains a mathematical set)
    - no duplicate elements
- E.g. 2 : $\boldsymbol{\Pi}_{\text{SEX,SALARY}}$(EMPLOYEE)
  - Guess what's the output relation? Any duplicate elements in result?

Examples of Projection

- e.g.1, $\boldsymbol{\Pi}_{\text{FNAME,LNAME,SALARY}}$(EMPLOYEE)
  - It **eliminates duplicate** tuples in the resulting relation (remains a mathematical set)
    - no duplicate elements
- E.g. 2 : $\boldsymbol{\Pi}_{\text{SEX,SALARY}}$(EMPLOYEE)
  - If several male employees have salary 30000, only a single tuple <M, 30000> is kept in the resulting relation.
- **Duplicate tuples are eliminated by the $\boldsymbol{\Pi}$ operation**.

Relational Algebra

- Combine a set of operations to form a *relational algebra expression* (query)
- E.g., Retrieve the names and salaries of employees who work in department 4.

Relational Algebra

- Combine a set of operations to form a *relational algebra expression* (query)
- E.g., Retrieve the names and salaries of employees who work in department 4.

$\Pi_{\text{FNAME,LNAME,SALARY}} (\textit{б}_{\text{DNO=4}}(\text{EMPLOYEE}))$

- we specify explicit intermediate relations for each step:
  - DEPT4_EMPS ← $\boldsymbol{б}_{\text{DNO=4}}$(EMPLOYEE)
  - R ← $\boldsymbol{\Pi}_{\text{FNAME,LNAME,SALARY}}$(DEPT4_EMPS)
- Attributes can optionally be *renamed* in the resulting left-hand-side relation
  - DEPT4_EMPS ← $\boldsymbol{б}_{\text{DNO=4}}$(EMPLOYEE)
  - R(FIRSTNAME,LASTNAME,SALARY) ←
    $\boldsymbol{\Pi}_{\text{FNAME,LNAME,SALARY}}$(DEPT4_EMPS)

FLT-WEEKDAY

flt#	weekday

- "find flt# for all flights scheduled for Mondays

  $\pi_{\text{flt\#}}(\sigma_{\text{weekday=MO}}(\text{FLT-WEEKDAY}))$

- the attributes in the attribute list of $\pi_{A1, A2, \ldots, An}(R)$ must be attributes of the operand R

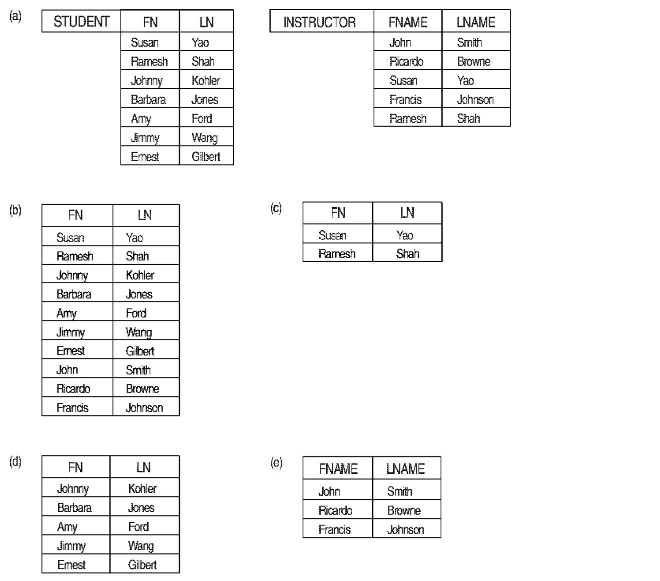
Set Operations

- Binary operations from mathematical set theory:
  - **UNION**: R1 U R2
  - **INTERSECTION**: R1 ∩ R2
  - **SET DIFFERENCE**: R1 **-** R2
  - **CARTESIAN PRODUCT**: R1 X R2

Union compatibility

- For U, ∩, -, the operand relations R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) must have the same number of attributes, and the domains of corresponding attributes must be compatible
- dom(Ai)=dom(Bi) for i=1, 2, ..., n.
- The resulting relation for U, ∩, or - has the same attribute names as the *first* operand relation R1 (by convention).

Figure 7.11 Illustrating the set operations union, intersection, and difference. (a) Two union compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∪ INSTRUCTOR. (d) STUDENT − INSTRUCTOR. (e) INSTRUCTOR − STUDENT.



(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

Cartesian Product

- Also called **CROSS JOIN, X**
- **Binary** set operation
- Do not have to be *union compatible*
- *R(A1, A2, ..., Am, B1, B2, ..., Bn) ←*

  R1(A1, A2, ..., Am) X R2 (B1, B2, ..., Bn)

```
A  B    x    C  D      =     A  B  C  D
a1 b1        c1 d1           a1 b1 c1 d1
a2 b2        c2 d2           a1 b1 c2 d2
             c3 d3           a1 b1 c3 d3
                             a2 b2 c1 d1
                             a2 b2 c2 d2
                             a3 b3 c3 d3
```

- CARTESIAN PRODUCT is a *meaningless operation* on its own.
- It can *combine related tuples* from two relations *if followed by the appropriate SELECT operation*
- Example: Combine each DEPARTMENT tuple with the EMPLOYEE tuple of the manager.

- CARTESIAN PRODUCT is a *meaningless operation* on its own.
- It can *combine related tuples* from two relations *if followed by the appropriate SELECT operation*
- Example: Combine each DEPARTMENT tuple with the EMPLOYEE tuple of the manager.

- DEP_EMP ← DEPARTMENT X EMPLOYEE
- DEPT_MANAGER ← ***б***$_{\text{MGRSSN=SSN}}$(DEP_EMP)

JOIN Operation

- **THETA JOIN (∞)**: Similar to a CARTESIAN PRODUCT followed by a SELECT. The condition c is called a *join condition*.

$$R \infty_C S = \sigma_C (R \times S)$$

- A general JOIN condition: <c>AND <c>…AND<c>
  - Each c is $A_i \ \theta \ B_i$
  - θ is one of the comparison operators {=, <, ≤, >, ≥, <>}

EQUIJOIN

- The join condition c includes one or more *equality comparisons* involving attributes from R1 and R2.
  - $(A_i=B_j)$ AND ... AND $(A_h=B_k)$
  - A_i, ..., A_h are called the **join attributes** of R_1
  - B_j, ..., B_k are called the **join attributes** of R_2

Climbers (C2):

CId	CName	Skill	Age
123	Edmund	EXP	80
214	Arnold	BEG	25
313	Bridget	EXP	33
212	James	MED	27

Climbs (C1):

CId	RId	Date	Duration
123	1	10/10/88	5
123	3	11/08/87	1
313	1	12/08/89	5
214	2	08/07/92	2
313	1	06/07/94	3

$\sigma_{CId:1=CId:2}(\text{Climbs} \times \text{Climbers}):$

C1.CId	RId	Date	Duration	C2.CId	CName	Skill	Age
123	1	10/10/88	5	123	Edmund	EXP	80
123	3	11/08/87	1	123	Edmund	EXP	80
313	1	12/08/89	5	313	Bridget	EXP	33
214	2	08/07/92	2	214	Arnold	BEG	25
313	1	06/07/94	3	313	Bridget	EXP	33

Natural JOIN (*)

- In an EQUIJOIN $R \leftarrow R_1 \infty_c R_2$, the join attribute of R_2 appear *redundantly* in the result relation R.
- In a NATURAL JOIN, the *redundant join attributes* of R_2 are *eliminated* from R. The equality condition is *implied* and need not be specified.

$\text{Climbs} \infty \text{Climbers}:$

CId	RId	Date	Duration	CName	Skill	Age
123	1	10/10/88	5	Edmund	EXP	80
123	3	11/08/87	1	Edmund	EXP	80
313	1	12/08/89	5	Bridget	EXP	33
214	2	08/07/92	2	Arnold	BEG	25
313	1	06/07/94	3	Bridget	EXP	33

- Example: Retrieve each EMPLOYEE's name and the name of his/her SUPERVISOR:

SUPERVISOR(SUPERSSN,SFN,SLN)$\leftarrow$

$\mathbf{\Pi}_{\text{SSN,FNAME,LNAME}}$(EMPLOYEE)

T$\leftarrow$EMPLOYEE * SUPERVISOR

RESULT $\leftarrow \Pi_{\text{FNAME,LNAME,S\_FN,S\_LN}}$(T)

FNAME	LNAME	S_FN	S_LN
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong

JOIN two relations

EMPLOYEE EMP and DEPARTMENT DEPT

<u>JOIN ATTRIBUTES</u>	<u>RELATIONSHIP</u>
EMP.SSN=DEPT.MGRSSN	EMP *manage* DEPT
EMP.DNO=DEPT.DNUMBER	EMP *works for* DEPT

Example:

List each EMP's name and the name of the DEPT he/she works for

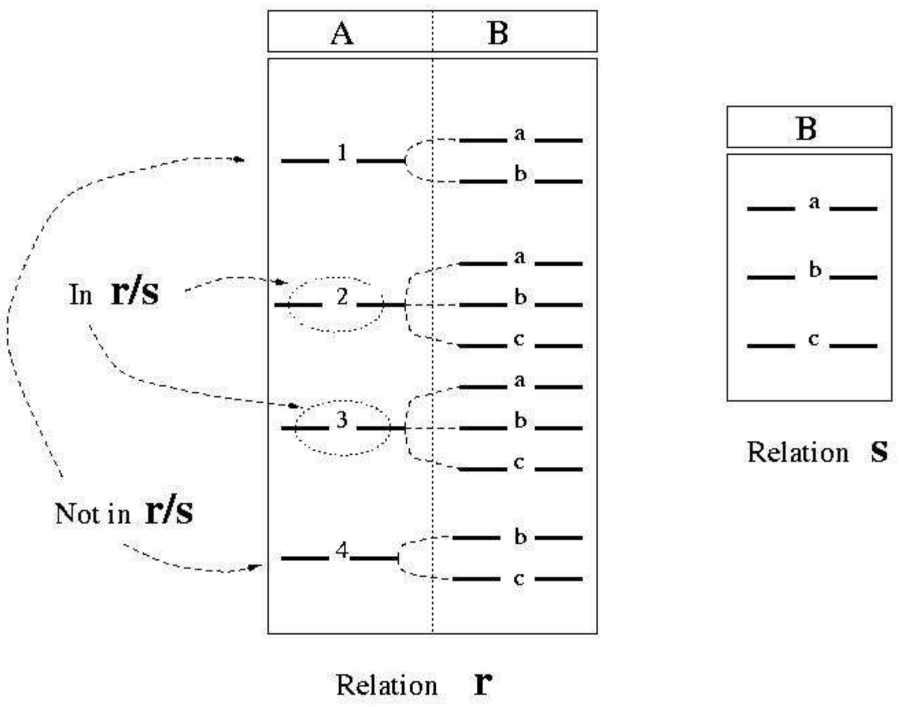
T$\leftarrow$EMPLOYEE $\infty_{\text{DNO=DNUMBER}}$DEPARTMENT

RESULT $\leftarrow \boldsymbol{\Pi}_{\text{FNAME,LNAME,DNAME}}$ (T)

Division

- Goal: Produce the tuples in one relation, r, that match *all* tuples in another relation, s
  - *r* $(A_1, \ldots A_n, B_1, \ldots B_m)$
  - *s* $(B_1 \ldots B_m)$
  - *r/s*, with attributes $A_1, \ldots A_n$, is the set of all tuples <*a*> such that for every tuple <*b*> in *s*, <*a,b*> is in *r*
- Can be expressed in terms of projection, set difference, and cross-product

Division (con't)



Division - Example

- List the Ids of students who have passed *all* courses that were taught in spring 2000

Division - Example

- List the Ids of students who have passed *all* courses that were taught in spring 2000
- *Numerator*: StudId and CrsCode for every course passed by every student
  - $\pi_{StudId,\ CrsCode}\ (\sigma_{Grade \neq \text{'F'}}\ (Transcript)\ )$
- *Denominator*: CrsCode of all courses taught in spring 2000
  - $\Pi_{CrsCode}\ (\sigma_{Semester = \text{'S2000'}}\ (Teaching)\ )$
- Result is *numerator/denominator*

Complete Set of Relational Algebra Operations

- the set {*б*, *П*, U, - , X } is called a *complete set* of relational algebra operations. Any query language *equivalent to* these operations is called **relationally complete**.
- All the basic operations discussed so far can be described as a sequence of *only* the above set
- Additional operations were not part of the *original* relational algebra
  - Aggregate functions (SUM, AVG), grouping
  - Outer Join, outer union

Additional Relational Operations

- Aggregate Functions and Grouping
- Outer JOIN and Outer UNION

Aggregate functions

- Functions are often applied to sets of values or sets of tuples in DB applications
  - SUM, COUNT, AVERAGE, MIN, MAX
  - $_{\text{<grouping attributes>}}\,\boldsymbol{F}\,_{\text{<function list>}}$ (R)
  - Grouping attributes are optional
- Example:

List the average salary of **all** employees (no grouping needed):

For each department, retrieve the department number, the number of employees, and the average salary:

Aggregate functions

- Functions are often applied to sets of values or sets of tuples in DB applications
  - SUM, COUNT, AVERAGE, MIN, MAX
  - $_{\text{<grouping attributes>}}\,\boldsymbol{F}\,_{\text{<function list>}}$ (R)
  - Grouping attributes are optional
- Example:

List the average salary of **all** employees (no grouping needed):

$\text{R(AVGSAL)} \leftarrow \boldsymbol{F}_{\text{AVG SALARY}}\ \text{(EMPLOYEE)}$

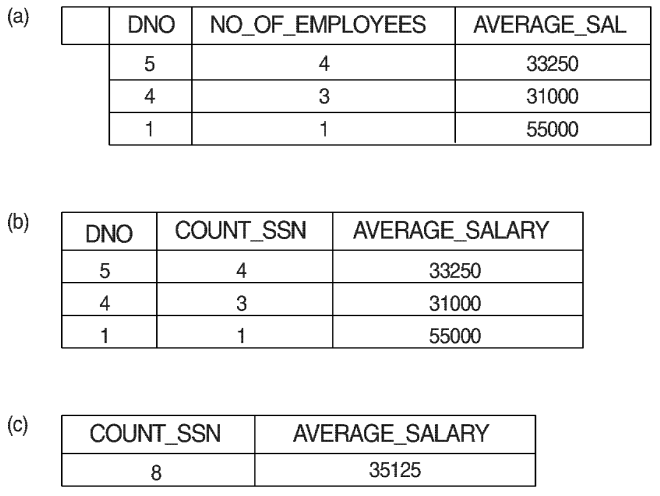
For each department, retrieve the department number, the number of employees, and the average salary:

$\text{R(DNO,NUMEMPS,AVGSAL)} \leftarrow$

$_{\text{DNO}}\,\boldsymbol{F}_{\text{COUNT SSN, AVERAGE SALARY}}\ \text{(EMPLOYEE)}$

– DNO is called the ***grouping attribute***

Figure 7.16 An illustration of the AGGREGATE FUNCTION operation. (a) $R(\text{DNO, NO\_OF\_EMPLOYEES, AVERAGE\_SAL}) \leftarrow {}_{\text{DNO}}\mathfrak{F}_{\text{COUNT SSN,AVERAGE SALARY}}(\text{EMPLOYEE})$. (b) ${}_{\text{DNO}}\mathfrak{F}_{\text{COUNT SSN,AVERAGE SALARY}}(\text{EMPLOYEE})$. (c) $\mathfrak{F}_{\text{COUNT SSN,AVERAGE SALARY}}(\text{EMPLOYEE})$.



(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

Outer join

- In a regular EQUIJOIN or NATURAL JOIN operation, tuples in R1 or R2 that do not have matching tuples in the other relation *do not appear in the result*
- Tuples with null in the join attributes are also eliminated.
- Some queries require all tuples in R1 (or R2 or both) to appear in the result
- When no matching tuples are found, **null**s are placed for the missing attributes

Outer join (Cont.)

- **LEFT OUTER JOIN**:
  - R1 left∞ R2 lets every tuple in R1 appear in the result
- **RIGHT OUTER JOIN**:
  - R1 ∞right R2 lets every tuple in R2 appear in the result
- **FULL OUTER JOIN**:
  - R1 left∞right R2 lets every tuple in R1 or R2 appear in the result

Figure 7.18 The LEFT OUTER JOIN operation.

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

Outer UNION

- If two relations are **not union compatible**
- Partial compatible
- example