CS430: Database Systems

Dr. Chetan Jaiswal,
Department of Computer Science,
Truman State University
cjaiswal@truman.edu

Chapter 5: Relational Data Model and Constraints

- The Relational Data Model
- Relational Model Constraints and Relational Database Schemas
- Update Operations, Transactions, and Dealing with Constraint Violations

The Relational Data Model

- **Relational model**
  - **First commercial implementations available in early 1980s**
  - **Has been implemented in a large number of commercial system**
- **Hierarchical and network models**
  - **Preceded the relational model**

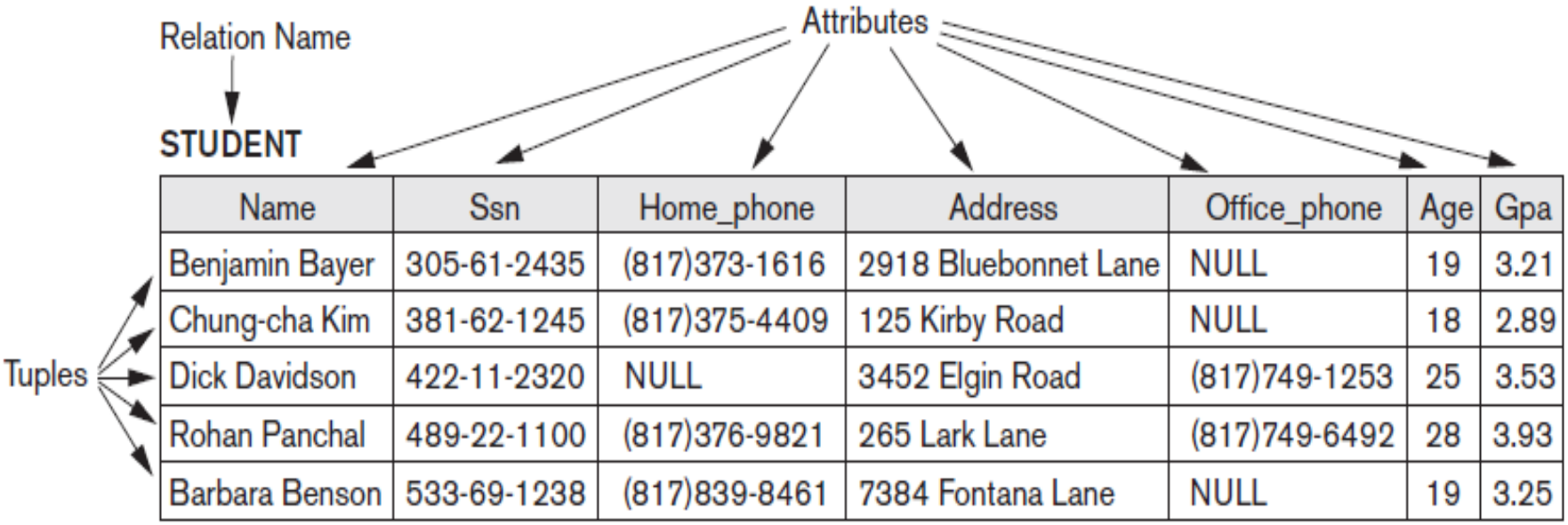
http://history-computer.com/ModernComputer/Software/Codd.html

http://math.hws.edu/vaughn/cpsc/343/2003/history.html

The Relational Data Model

- Represents data as a collection of relations
- Table of values
  - Row
    - Represents a collection of related data values
    - Fact that typically corresponds to a real-world entity or relationship
    - *Tuple*
- Table name and column names
  - Interpret the meaning of the values in each row *attribute*
  - Column name/attribute is the name of a role played by some domain

The Relational Data Model



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 3.1
The attributes and tuples of a relation STUDENT.

Domains, Attributes, Tuples, and Relations

- Domain D
  - Set of atomic values
- Atomic
  - Each value indivisible
- Specifying a domain
  - Name, datatype and format

Domains, Attributes, Tuples, and Relations

- **Relation schema *R***
  - Denoted by $R(A_1, A_2, ..., A_n)$
  - Made up of a relation name *R* and a list of attributes, $A_1, A_2, ..., A_n$
  - Example of a schema:

E-name	E-age	SSN	Salary	D-name

- **Attribute A_i**
  - Name of a role played by some domain *D* in the relation schema *R*
- **Degree (or arity) of a relation**
  - Number of attributes n of its relation schema

Example

- A relation of degree seven, stores info about students, has seven attributes:

STUDENT(Name:string, Ssn:string, Home_Phone:string, Address:string, Office_phone:string, Age:integer, Gpa:real)

Relational Model Notation

- A relation on schema R is represented as r(R)

(Where schema is $R(A_1, A_2, ..., A_n)$)

- *n-tuple t* in a relation *r(R)*
  - Denoted by $t = <v_1, v_2, ..., v_n>$
  - v_i is the value corresponding to attribute A_i

Anderson	21	123456789	50000	Research

- Component values of tuples
  - $t[A_i]$ and $t.A_i$ refer to the value v_i in t for attribute A_i
  - $t[A_u, A_w, ..., A_z]$ and $t.(A_u, A_w, ..., A_z)$ refer to the subtuple of values $<v_u, v_w, ..., v_z>$ from t corresponding to the attributes specified in the list

Domains, Attributes, Tuples, and Relations

- **Relation**
  - Set of n-tuples $r = \{t_1, t_2, ..., t_m\}$
  - Each n-tuple t
    - Ordered list of n values $t = <v_1, v_2, ..., v_n>$
    - Each value v_i, $1 \leq i \leq n$, is an element of dom(A_i) or is a special NULL value

Domains, Attributes, Tuples, and Relations

- **Relation $r(R)$**
  - **Mathematical relation of degree n on the domains $dom(A_1)$, $dom(A_2)$, ..., $dom(A_n)$**
  - **Subset of the Cartesian product of the domains that define R**
    - $r(R) \subseteq ((dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$
  - **Cardinality or total number of values? (assuming all domains are finite). Represents total number of possible instances or tuples.**

$|dom(A_1)| \times |dom(A_2)| \times ... \times |dom(A_n)|$

Domains, Attributes, Tuples, and Relations

- **Cardinality**
  - Total number of values in domain
- **Attribute names**
  - Indicate different roles, or interpretations, for the domain

Characteristics of Relations

- Ordering of tuples in a relation
  - Relation defined as a set of tuples
  - Elements have no order among them since it represents facts at a logical level
  - However, in a file records stored on disk so there exist order.
- Ordering of values within a tuple and an alternative definition of a relation
  - Order of attributes and values is not that important
  - As long as correspondence between attributes and values maintained

Characteristics of Relations

Figure 3.2
The relation STUDENT from Figure 3.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

- **Cardinality of Student (# of tuples) = 5**
- **Degree of student (# of attributes) = 7**

Characteristics of Relations

- **Values and NULLs in tuples**
  - Each value in a tuple is atomic
  - Flat relational model
    - Composite and multivalued attributes not allowed
    - First normal form assumption
- **Multivalued attributes**
  - Must be represented by separate relations
- **Composite attributes**
  - Represented only by simple component attributes in basic relational model

Characteristics of Relations

- **NULLs values**
  - Represent the values of attributes that may be unknown or may not apply to a tuple
  - Meanings for NULL values
    - *Value unknown*
    - *Value exists but is not available*
    - *Attribute does not apply to this tuple (also known as value undefined)*
  - Can NULL be a problem? Yes! Comparison leads to ambiguities… e.g. address of two customers.
  - Morale? It is best to avoid NULL values as much as possible.

Relational Model Notation

- Relation schema R of degree n
  - Denoted by $R(A_1, A_2, ..., A_n)$
- Uppercase letters Q, R, S
  - Denote relation names
- Lowercase letters q, r, s
  - Denote relation states
- Letters t, u, v
  - Denote tuples

Relational Model Notation

- Name of a relation schema: STUDENT
  - Indicates the current set of tuples in that relation
- Notation: STUDENT(Name, Ssn, ...)
  - Refers only to relation schema
- Lowercase letters *q*, *r*, *s*
  - Denote relation states
- Attribute *A* can be qualified with the relation name *R* to which it belongs
  - Using the dot notation *R.A*

Relational Model Constraints

- Constraints
  - Restrictions on the actual values in a database state
  - Derived from the rules in the miniworld that the database represents
- Inherent model-based constraints or implicit constraints
  - Inherent in the data model
- Schema based constraints or explicit constraints

  Can be directly expressed in schema using DDL
- Application based constraints/business rules
  - Cannot be directly expressed in schema
  - Must be enforced by application programs
  - Eg. this year's salary increase can be no more than last year's

Domain Constraints

- Typically include
  - Numeric data types for integers and real numbers
  - Characters
  - Booleans
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

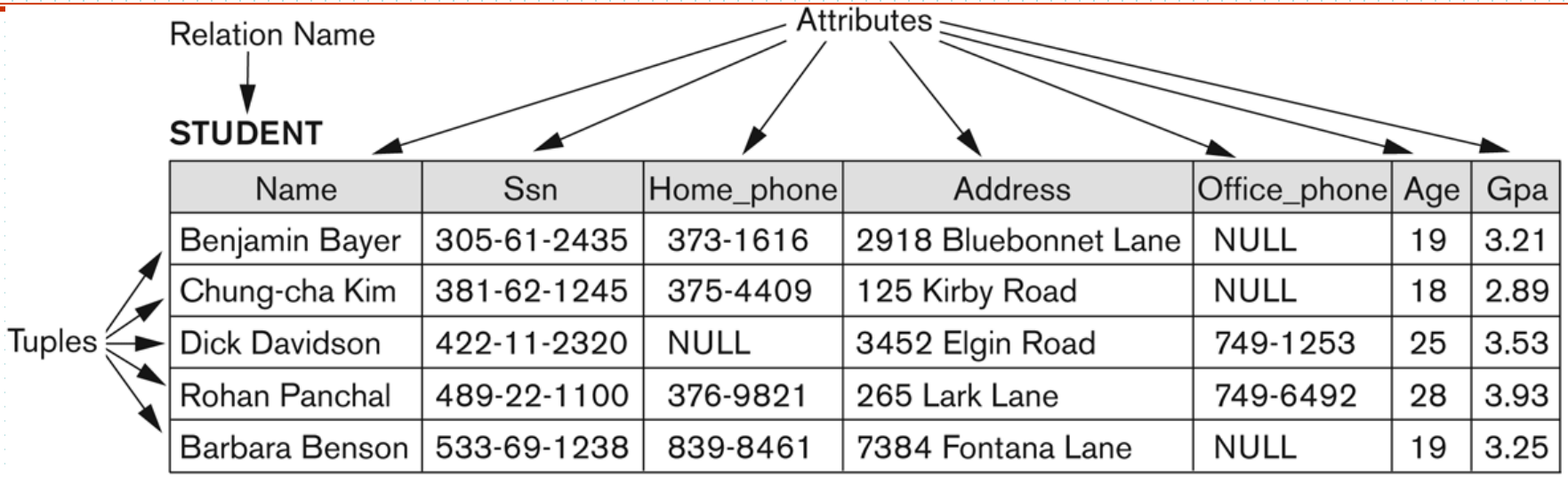
Key Constraints and Constraints on NULL Values

- No two tuples can have the same combination of values for all their attributes
  - Formal definition of a key:
    - *For any two disjoint tuples t1 and t2 of r, t1[K] ≠ t2[K]*
- Superkey
  - No two distinct tuples in any state *r* of *R* can have the same value for SK
  - Subset of attributes with the property that no two tuples will have the same combination of values.
  - It can have redundant attributes.
- Key
  - Superkey of R
  - Removing any attribute *A* from *K* leaves a set of attributes *K* that is not a key of *R* any more
  - Hence, it doesn't have redundant attributes.

Key Constraints and Constraints on NULL Values

- **Key satisfies two properties**
  - **Two distinct tuples in any state of relation cannot have identical values for (all) attributes in key**
  - **Minimal superkey**
    - **Key is a minimal super key**
    - **Time invariant**
    - **Cannot remove any attributes and still have uniqueness constraint in above condition hold**

Key Constraints and Constraints on NULL Values



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

- **superkey = {Ssn, Name, Age}, remove Name attr., still it's a superkey**
- **Key = {Ssn}, cannot remove any attr.**

Key Constraints and Constraints on NULL Values

DEPT_LOCATIONS

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

- superkey = {Dnumber, Dlocation, LocManPower (assumed)}, can remove LocManPower???
- Key = ??
- Will key be a minimal superkey?

Key Constraints and Constraints on NULL Values

- Candidate key
  - Relation schema may have more than one key
- Primary key of the relation
  - Designated among candidate keys
  - Underline attribute
  - Cannot be null
- Other candidate keys are designated as Alternate keys or Unique Keys (can contain nulls)

Key Constraints and Constraints on NULL Values

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 3.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

- **Candidate key = License_number, Engine_Serial_number.**

Relational Database and its Schema

- **Invalid state**
  - Does not obey all the integrity constraints
- **Valid state**
  - Satisfies all the constraints in the defined set of integrity constraints IC

Integrity, Referential Integrity, and Foreign Keys

- Entity integrity constraint
  - No primary key value can be NULL
- Referential integrity constraint
  - Specified between two relations
  - Maintains consistency among tuples in two relations

Integrity, Referential Integrity, and Foreign Keys

- **Foreign key rules**
  - The attributes in FK have the same domain(s) as the primary key (PK) or candidate key (CK) attribute(s)
  - Value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK or CK for some tuple t_2 in the current state $r_2(R_2)$ or is NULL

Update Operations

- Operations of the relational model can be categorized into retrievals and updates
- Basic operations that change the states of relations in the database
  - Insert
  - Delete
  - Update

Update Operations

Figure 3.6
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

The Insert Operation

- Provides a list of attribute values for a new tuple t that is to be inserted into a relation R
- Can violate any of the four types of constraints

(Domain, key, entity and referential integrity)

- If an insertion violates one or more constraints
  - Default option is to reject the insertion

The Insert Operation

Operation:
Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Operation:
Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Operation:
Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

- *Operation*:
  Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

The Delete Operation

- Can violate only referential integrity
  - If tuple being deleted is referenced by foreign keys from other tuples
- Restrict
  - Reject the deletion
- Cascade
  - Propagate the deletion by deleting tuples that reference the tuple that is being deleted
- Set null or set default
  - Modify the referencing attribute values that cause the violation

The Delete Operation

Operation:
Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

Operation:
Delete the EMPLOYEE tuple with Ssn = '999887777'.

Operation:
Delete the EMPLOYEE tuple with Ssn = '333445555'.

Example

CREATE TABLE `ping_details_file_new` (`pingidfile` bigint(20) unsigned NOT NULL AUTO_INCREMENT, `sourceid` varchar(255) NOT NULL, `destinationid` varchar(255) NOT NULL, `timestamp` datetime DEFAULT NULL, `delay` float DEFAULT NULL, `distanceinmiles` float DEFAULT NULL, PRIMARY KEY (`pingidfile)

CONSTRAINT FOREIGN KEY (`destinationid`) REFERENCES `node` (`nodename`) ON UPDATE CASCADE,

CONSTRAINT FOREIGN KEY (`sourceid`) REFERENCES `node` (`nodename`) ON UPDATE CASCADE

ENGINE=InnoDB AUTO_INCREMENT=356916 DEFAULT CHARSET=utf8;

Example

- Delete * from Employee where emp_id=12345
- Delete SALARY from Employee where emp_id=123
- Delete SALARY, PHONE from Employee where emp_id=111
- Delete from Employee where emp_id>111 AND emp_id<222

The Update Operation

- Necessary to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified
- If attribute not part of a primary key nor of a foreign key
  - Usually causes no problems
- Updating a primary/foreign key
  - Similar issues as with Insert/Delete

The Update Operation

Operation:
Update the salary of the EMPLOYEE tuple with Ssn = '999887777' to 28000.

Operation:
Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 1.

Operation:
Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7.

Operation:
Update the Ssn of the EMPLOYEE tuple with Ssn = '999887777' to '987654321'.

Relational Algebra (DML)

- **Relational Algebra**
  - Relational algebra is a non-procedural language that provides a set of operations to manipulate a relation.
- **Relational Algebra Components**
  - U = A set of attributes, called the Universe.
  - D = a set of domains
  - dom = a total function from U to D.
  - *R = {R1, R2, ..., Rp}* = a set of distinct relational schemes where $Ri \subseteq U$ for $1 \leq i \leq p$.
  - *r = {r1, r2, ..., rp}* = a set of relations, such that *ri* is relation on *Ri*, $1 \leq i \leq p$.
  - Θ = a set of comparators over D. It compares the values of two attributes with the same domain or with absolute value.
  - O = is the set of operators $\cup$, $\cap$, -, Π, Join, $\div$, and σ and theta-join using comparator in Q.
  - The relational algebra over *U, D, dom* and Θ is a 7-tuple *RA = <U, D, dom, R, d, Q, O>.*

Relational Algebra Operations

- Relational Algebra Operations
  - Select: Denoted as σ
  - Project: Denoted as Π
  - Join: Denoted as $\bowtie$
  - Cross product: Denoted as $\times$
- Relational Algebra Boolean Operations
  - Union: Denoted as $\cup$
  - Intersection: Denoted as $\cap$
- Relational Algebra Arithmetic Operations
  - Division: Denoted as $\div$
  - Difference (Minus): Denoted as -

Relational Algebra Operations

- We will use the following relation to show the use of relational algebra operations
  - EMP

E-name	E-age	SSN	Salary	Dept-name
Anderson	21	123456789	50000	Research
Decker	22	113456789	45000	Research
Jackson	21	122456789	45000	Admin

Relational Algebra Operations

- **Select σ**

  A monadic (unary) operator that takes only one operand (i.e., a relation). The result of a select is a new relation, which is a horizontal subset of the operand relation.

  Format: $\sigma_{<p>}$ (r) $\Rightarrow$ r' (a new relation)

  Predicate (p): a condition expressed using attributes and operators *AND, OR, NOR,* =, $\neq$, <, and >. Example: (*Salary = 50000*), (*Salary = 50000 AND (E_age < 50 OR E_age = 40)*), etc.

Relational Algebra Operations

- **Example of σ**

  $\sigma_{Salary = 50000}(EMP)$. *Get all tuples of EMP that satisfies predicate (Salary = 50000)*

  Result: A new relation

Anderson	21	123456789	50000	Research

Set theoretic representation: r' = {t ∈ r | t (A) = a}. It says the set of all tuples that satisfy predicate (A = a)

Relational Algebra Operations

- **Properties σ**
  - **Select operator commute under composition, that is,**

    $\sigma_{(A= a)}(\sigma_{(B = b)}(r)) = \sigma_{(B = b)}(\sigma_{(A= a)}(r))$
  - **Proof**

    $\sigma_{(A = a)}(\sigma_{(B = b)}(r))$

    $\sigma_{(A = a)}(\{t \in r | t(B) = b\})$

    $= \{t' \in \{t \in r | t(B) = b\} | t'(A) = a\}$

    $= \{t \in r | t(B) = b \text{ and } t(A) = a\}$

    $= \{t \in r | t(A) = a \text{ and } t(B) = b\}$

    $= \{t' \in \{t \in r | t(A) = a\} | t'(B) = b\}$

    $= \{t' \in \{\sigma_{(A= a)}(r)\} | t'(B) = b\}$

    $= \sigma_{(B = b)}(\sigma_{(A= a)}(r))$

Relational Algebra Operations

- **Properties σ**
  - **Select is distributive over the binary Boolean operations, that is,**

    $\sigma_{(A = a)}(r \gamma s) \equiv \sigma_{(A = a)}(r) \gamma \sigma_{(A = a)}(s)$; **where** $\gamma = \cup, \cap$ **and** ***r*** **and** ***s*** **are relations over** ***R***
  - **Proof: We want to prove** $\sigma_{(A = a)}(r \cap s) \equiv \sigma_{(A = a)}(r) \cap \sigma_{(A = a)}(s)$

    $\sigma_{(A = a)}(r \cap s)$

    $= \sigma_{(A = a)}(\{t|t \in r \text{ and } t \in s\})$

    $= \{t' \in (\{t|t \in r \text{ and } t \in s\}) \; t'(A) = a\}$

    $= \{t''|t'' \in r \text{ and } t''(A) = a\} \cap \{t|t \in s \text{ and } t(A) = a\}$

    $= \sigma_{(A= a)}(\{t''|t'' \in r\}) \cap \sigma_{(A= a)}(\{t|t \in s\})$

    $= \sigma_{(A= a)}(r) \cap \sigma_{(A= a)}(s)$

Relational Algebra Operations

- **Example:**

Student

FN	LN
Susan	Yao
Ramesh	Shah
Barbara	Jones
Amy	Ford
Jimmy	Wang

Prof

FN	LN
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

$\sigma_{(FN='Susan')}(\text{Student} \cap \text{Prof}) = ?$

$\sigma_{(FN='Susan')}(\text{Student}) \cap \sigma_{(FN='Susan')}(\text{Prof}) = ?$

Relational Algebra Operations

- **Projection Π**
  - **A unary (monadic) operator on a relation *r* that creates a new relation r' that is a *vertical* subset of the r**
  - **Let *X* be a subset of the attributes of *R(A1, A2, …, An)*. Let *X* be *(A1, A3, A5).* The result of the projection of *r(R)* onto *X* is a new relation *r'(X),* which obtained by extracting the columns (attributes) from *R* which are in *X, i.e., A1, A2, and A5*. At the end of this process the relation *r'(X)* may contain duplicate tuples, which are removed by PROJECT operation. Also known as duplicate elimination.**

Relational Algebra Operations

- **Format of Π**
  - Π *<list of attributes of the operand relation separated by ","> (r)*
  - **Example**

$\Pi_{Age}(Emp) \Rightarrow$ *Result:*

Age
21
22
19

$\Pi_{Age,SSN}(Emp) \Rightarrow$ *Result:*

Age	SSN
21	123456789
22	113456789
19	122456789

Relational Algebra Operations

- The result of a projection is a relation ⇒ projection involves duplicate removal
- Example: $\pi_{\text{DNo}}(\text{Employee})$

DNo
5
4
1

DNo
5
5
4
4
5
5
4
1

Relational Algebra Operations

- Example: $\pi_{\text{LName,FName,Salary}}(\text{Employee})$

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

Relational Algebra Operations

- **Properties of Π**
  - **If two Π s are applied on a relation in a row, then the latter subsumes the former**
  - **Proof**

    If $Y \in X \in R$ then $\Pi_Y(\Pi_X(r)) = \Pi_Y(r)$. Similarly, for a string of Π s only the outermost Π *counts*. If $X1 \in X2 \in ... \in Xm \in R$ then $\Pi_{x1}(\Pi_{x2}(. . . (\Pi_{xm}(r)) . . .)) = \Pi_{x1}(r)$.

    Point to remember: $Y \in X \in R$ or $X1 \in X2 \in ... \in Xm \in R$

Relational Algebra Operations

- **Properties of Π**
  - Π commutes with σ when the attribute(s) in the predicate(s) of σ are among the attributes in the set onto which the Π is being taken. That is if $A \in X$, $X \subseteq R$ and r is a relation on R, then $\Pi_X(\sigma_{(A = a)}(r)) = \sigma_{(A = a)}(\Pi_X(r))$
  - **Proof**

$$
\begin{aligned}
\Pi_X(\sigma_{(A = a)}(r)) &= \Pi_X(\{t \in r | t(A) = a\}) \\
&= \{t'(X) | t' \in \{t \hat{I} r | t(A) = a\}\} \\
&= \{t(X) | t \in r \text{ and } t(A) = a\} \\
&= \{t(X)\ t \in r \mid t(A) = a\} \\
&= \{t \hat{I} r \mid t(A) = a \text{ and } t(X) \mid t \in r\} \\
&= \sigma_{(A = a)}(\{t(X) | t \in r\}) \\
&= \sigma_{(A = a)}(\Pi_X(r))
\end{aligned}
$$

SELECT with PROJECTION and RENAME

List the name and salary of employees in department 5

(1) Nested form: $\pi_{\text{FName,LName,Salary}}(\boxed{\sigma_{\text{DNo}=5}(\text{Employee})})$

(2) Sequential form: $\text{Temp} \leftarrow \sigma_{\text{DNo}=5}(\text{Employee})$

Temp

SSN	FName	MInit	LName	BDate	Address	Sex	Salary	SuperSSN	DNo
123456789	John	B	Smith	09-Jan-55	...	M	30000	333445555	5
333445555	Franklin	T	Wong	08-Dec-45	...	M	40000	888665555	5
666884444	Ramesh	K	Narayan	15-Sep-52	...	M	38000	333445555	5
453453453	Joyce	A	English	31-Jul-62	...	F	25000	333445555	5

$$R(\text{FirstName}, \text{LastName}, \text{Salary}) \leftarrow \pi_{\text{FName,LName,Salary}}(\text{Temp})$$

R

FirstName	LastName	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Boolean Operations

- We will use the following relations in the discussion of these operators.

r(A	B	C)
a1	b1	c1
a1	b2	c1
a2	b1	c2

s(A	B	C)
a1	b2	c1
a2	b2	c1
a2	b2	c2

- Union compatibility rule: two relations are union compatible iff they have the same degree and the domain of corresponding attributes are the same, but their names may be different (synonyms)

2 relations R(A1, A2…An) and S(B1, B2…Bn) are union compatible if they have same degree n and if dom(Ai) = dom (Bi) for 1<=i<=n.

- Boolean operators require that operand relations must be union compatible.

Boolean Operations

- **Intersection ∩**
  - A binary operator. It creates a new relation from the two (operand) relations. The new relation contains the tuples, which are common in both the operand relations.
  - Example

$r \cap s = r(A\ B\ C) \cap s(A\ B\ C)$

r				s		
a1	b1	c1		a1	b2	c1
a1	b2	c1	∩	a2	b2	c1
a2	b1	c2		a2	b2	c2

$= q(A\ B\ C)$

q		
a1	b2	c1

Class task: Prove that $r \cap s = r - (r - s)$

Boolean Operations

- **Union ∪**
  - A binary operator. It creates a new relation from the two (operand) relations that contains the common and non-common tuples of the operand relations.
  - Example

$r \cup s = r(A\ B\ C)\ \ s(A\ B\ C) = q(A\ B\ C)$

r				s				q		
a1	b1	c1		a1	b2	c1		a1	b1	c1
a1	b2	c1	∪	a2	b2	c1		a1	b2	c1
a2	b1	c2		a2	b2	c2		a2	b1	c2
								a2	b2	c1
								a2	b2	c2

Boolean Operations

- **Difference -**
  - A binary operator. It creates a new relation from the two (operand) relations. The new relation contains the tuples, which are in the left hand relation but not in the right hand relation. Note that $r - s \neq s - r$
  - Example

$r - s = r(A\ B\ C)\ \ s(A\ B\ C) = q(A\ B\ C)$

	a1	b1	c1		a1	b2	c1	a1	b1	c1
	a1	b2	c1	–	a2	b2	c1	a2	b1	c2
	a2	b1	c2		a2	b2	c2			

Class task. Prove that $r - s \neq s - r$

Boolean Operations

- 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)

Boolean Operations

- We can use the UNION operation as follows:

$$DEP5\_EMPS \leftarrow \sigma_{DNO=5}(EMPLOYEE)$$

$$RESULT1 \leftarrow \pi_{SSN}(DEP5\_EMPS)$$

$$RESULT2(SSN) \leftarrow \pi_{SUPERSSN}(DEP5\_EMPS)$$

$$RESULT \leftarrow RESULT1 \cup RESULT2$$

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

Boolean Operations

- 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

Boolean Operations

(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.

Boolean Operations

- Notice that both union and intersection are *commutative* operations; that is
  - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

Relational Algebra Operations-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-Set Theory
CARTESIAN PRODUCT

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

Relational Algebra Operations-Set Theory
CARTESIAN PRODUCT

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

Relational Algebra Operations-Set Theory
CARTESIAN PRODUCT

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

Join Operations

- **Join**
  - **A Join combines two relations on a common attribute. In order to combine two relations there must at least be one attribute with common domain between both the relations but their names may or may not be the same. For example, one database may call income as *salary* and other as *monthly income*, however, their domains are the same. The result of a join is a new relation, which contains those tuples that satisfy the *join condition***

Join Operations

- 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*.

Join Operations

- 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 ⋈ 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

Join Operations

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

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{MGRSSN=SSN} EMPLOYEE$$

Is this what we wanted? Name of Manager? How to get?

Join Operations

- Consider the following JOIN operation:
  - R(A1, A2, . . ., An) $\bowtie_{R.Ai=S.Bj}$ 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—r from R and s from S, but *only if they satisfy the join condition* r[Ai]=s[Bj]
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
  - Only related tuples (based on the join condition) will appear in the result

Join Operations

- **Join types**
  - **There are three types of join**
    - **Equijoin ($\bowtie$)**
    - **Natural join (*)**
    - **Theta join (Θ)**

Join Operations

- 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

Join Operations

- 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.

Join Operations

- **Equijoin steps** ($\bowtie$)
  - Get cross product of two relations
  - Apply the join condition on the result of the cross product
  - Example
    - Get cross product.

r (A	*B*	*C)*	*s (M*	*B)*
a1	*b1*	*c1*	*m1*	*b1*
a2	*b2*	*c2*	*m2*	*b2*

Common domain attribute = B

r × s =	*A*	*B*	*C*	*M*	*B*
	a1	*b1*	*c1*	*m1*	*b1*
	a1	*b1*	*c1*	*m2*	*b2*
	a2	*b2*	*c2*	*m1*	*b1*
	a2	*b2*	*c2*	*m2*	*b2*

Join Operations

- **Equijoin steps ($\bowtie$)**
  - Get cross product of two relations
  - Apply the join condition on the result of the cross product
  - Example
    - Apply join condition (predicate), i.e., r.B = s.B

$r \bowtie s$ =

A	B	C	M	B
a1	b1	c1	m1	b1
a2	b2	c2	m2	b2

Join Operations

- 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.

Join Operations

- 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)

Join Operations

(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.

Join Operations

- **Natural join steps ($\bowtie$)**
  - Get cross product of two relations
  - Apply the join condition on the result of the cross product
  - Apply projection to remove duplicate column
  - Example
    - Apply join condition (predicate), i.e., r.B = s.B

$r \bowtie s$ =

A	B	C	M	B
a1	b1	c1	m1	b1
a2	b2	c2	m2	b2

$P_{(A,B,C,M)}(r \bowtie s)$ =

A	B	C	M
a1	b1	c1	m1
a2	b2	c2	m2

Division Operation

- **Division (÷)**
  - **Division is a binary operation that can be defined on two relations where the entire structure of one (the divisor relation) is a part of the structure of the other (the dividend relation). For example *R(A B C)* and *S(B C)*, and *B* and *C* have the same domains in *R* and *S*. *R* can be divided by *S* and the resultant relation, say *RR*, will have only column *A* of *R*. Only those tuples will be in *RR* that contains all tuple values of the divisor *S***

Division Operation

- DIVISION Operation
  - The division operation is applied to two relations
  - $R(Z) \div S(X)$, where X subset Z. Let $Y = Z - X$ (and hence $Z = X \cup Y$); 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.

Division Operation

- **Division example (÷)**

Enroll(C#	*FID*	*SID*	*SNAME)*
103A	*101*	*1001*	*A*
Set 1 *103A*	*101*	*1002*	*B*
103A	*101*	*1013*	*C*
103A	*101*	*1005*	*D*
Set 2 *201A*	*105*	*1001*	*A*
201A	*105*	*1005*	*D*
Set 3 *101B*	*110*	*1001*	*A*
101B	*110*	*1002*	*B*
101B	*110*	*1013*	*C*

Stu(SID	*SNAME)*
1001	*A*
1005	*D*

Enroll can be divided by Stu

Division Operation

- **Enroll ÷ Stu**

  Result: C# FID

  103A 101

  201A 105

  The quotient contains values of *C#* and *FID* that appear in Enroll with all combinations of *SID* and *SNAME* in the *Stu* table. Since *1001* and *1005* both appear with *103A (Set 1)* and with *201A (Set 2)*, these two values will appear in the result. *101B (Set 3)* will not appear in the result, since Stu (*1005 D)* does not appear in *Enroll* with *1005*.

Division Operation

- **More Division examples (÷)**

Enroll(C#	FID	SID	SNAME)
103A	101	1001	A
103A	101	1002	B
103A	101	1013	C
103A	101	1005	D
201A	105	1001	A
201A	105	1005	D
101B	110	1001	A
101B	110	1002	B
101B	110	1013	C

Stu(SID	SNAME)
1001	A
1002	B
1013	C

Enroll ÷ Stu (C#	FID)
103A	101
101B	110

Division Operation

- **More Division examples (÷)**

Enroll(C#	FID	SID	SNAME)
103A	101	1001	A
103A	101	1002	B
103A	101	1013	C
103A	101	1005	D
201A	105	1001	A
201A	105	1005	D
101B	110	1001	A
101B	110	1002	B
101B	110	1013	C

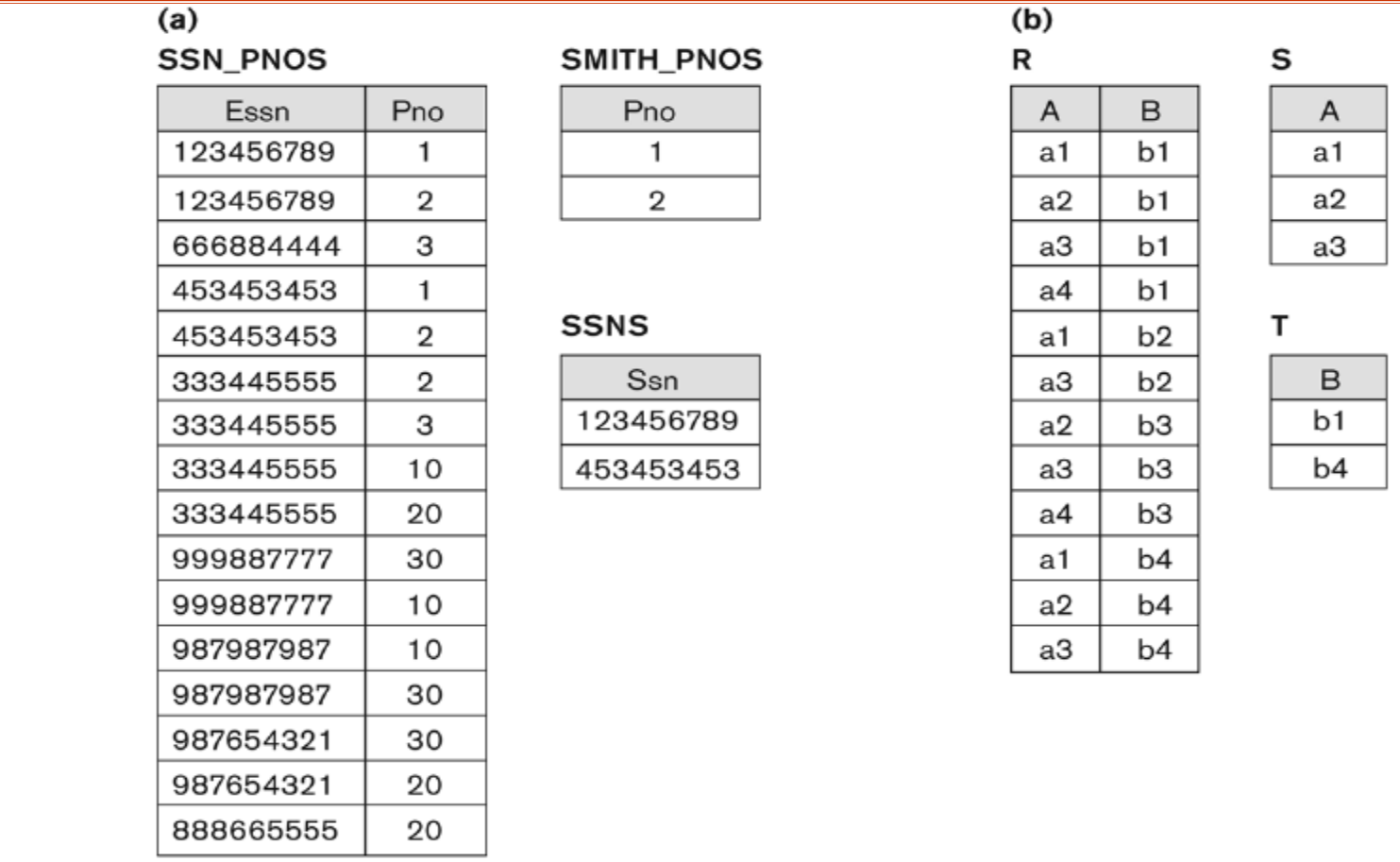
Stu(SID	SNAME)
1001	A
1002	B
1013	C
1005	D

This tuple does not appear in the result because there is no 101B 110 1005 D tuple in Enroll.

Enroll ÷ Stu (C# FID)

103A 101

Division Operation



(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) $T \leftarrow R \div S$.

Division Operation

- Class exercise on division.

Division Operation

- **Query that require division**

  Consider the following relations

P (Pilot,Equipment)		E (Equipment)	E' (Equipment)
Desmond	707	707	707
Desmond	727	727	
Desmond	747	747	
Doyle	707		
Doyle	727		
Davis	707		
Davis	727		
Davis	747		
Davis	1011		
Dow	727		

Division Operation

- **Query that require division**

  Query: Find those pilots who can fly all types of aircraft in some set.

Division Operation

- **Query that require division**

  Query: Find those pilots who can fly all types of aircraft in some set.

P ÷ E = P' (Pilot)

Desmond

Davis

P ÷ E' = P' (Pilot)

Desmond

Doyle

Davis

Division Operation

- **Precedence of Relational Algebra operations**

  Highest to Lowest:

  Project (Π), Select (σ), Cross Product ($\times$), Join ($\bowtie$), Division ($\div$), Difference (-), Union ($\cup$) and Intersection ($\cap$)

This Chapter

- **Class Exercise:**

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

This Chapter

- **Class Exercise:**

$$RESEARCH\_DEPT \leftarrow \sigma_{DNAME='Research'}(DEPARTMENT)$$

$$RESEARCH\_EMPS \leftarrow (RESEARCH\_DEPT \bowtie_{DNUMBER=DNOEMPLOYEE} EMPLOYEE)$$

$$RESULT \leftarrow \pi_{FNAME, LNAME, ADDRESS}(RESEARCH\_EMPS)$$

This Chapter

- **Class Exercise:**

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

This Chapter

- **Class Exercise:**

$$ALL\_EMPS \leftarrow \pi_{SSN}(EMPLOYEE)$$

$$EMPS\_WITH\_DEPS(SSN) \leftarrow \pi_{ESSN}(DEPENDENT)$$

$$EMPS\_WITHOUT\_DEPS \leftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS)$$

$$RESULT \leftarrow \pi_{LNAME, FNAME}(EMPS\_WITHOUT\_DEPS * EMPLOYEE)$$

This Chapter

- **Class Exercise:**

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

$$\pi_{\text{Lname, Fname}}((\pi_{\text{Ssn}} (\text{EMPLOYEE}) - \rho_{\text{Ssn}} (\pi_{\text{Essn}} (\text{DEPENDENT}))) * \text{EMPLOYEE})$$

Reference

- 1. Dr. Vijay Kumar
- 2. Elmasri/Navathe
- Google, ofcourse…