Week 2 – Part II
Relational Algebra

Querying and Updating a Database

The Relational Algebra

Union, Intersection, Difference

Renaming, Selection and Projection

Join, Cartesian Product

Query Languages for Relational Databases

- Operations on databases:
  - ✓**Queries** — read data from the database;
  - ✓**Updates** — change the content of the database.
- In this lecture unit we discuss the relational algebra, a procedural language that defines database operations in terms of algebraic expressions.
- [The Relational Calculus is a declarative language for database operations based on Predicate Logic; we will not discuss it here.]

Relational Algebra

- A collection of **algebraic operators** that
  - Are defined on relations;
  - Produce relations as results,

    and therefore can be combined to form complex algebraic expressions.

Operators:

- Union, intersection, difference;
- Renaming;
- Selection and Projection;
- Join (natural join, Cartesian product, theta join).

Union, Intersection, Difference

- Relations are sets, so we can apply set-theoretic operators
- However, we want the results to be relations (that is, homogeneous sets of tuples)
- It is therefore meaningful to only apply union, intersection, difference to pairs of relations defined over the same attributes.

Union

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∪ Managers

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38
9297	O'Malley	56

Intersection

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∩ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

Difference

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

A Meaningful but Impossible Union

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity ∪ Maternity ???

- → The problem: `Father` and `Mother` are different names, but both represent a parent.
- → The solution: rename attributes!

Renaming

- This is a unary operator which changes attribute names for a relation without changing any values.
- Renaming removes the limitations associated with set operators.
- Notation: $\rho_{\text{OldName}\rightarrow\text{NewName}}(r)$
- For example, $\rho_{\text{Father}\rightarrow\text{Parent}}(\text{Paternity})$
- If there are two or more attributes involved in a renaming operation, then ordering is meaningful:

  e.g., $\rho_{\text{Branch,Salary}\rightarrow\text{Location,Pay}}(\text{Employees})$

Example of Renaming

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

$\rho_{\text{Father}\text{--}> \text{Parent}}(\text{Paternity})$

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

- The textbook allows positions rather than attribute names, e.g., **`1 → Parent`**
- Textbook also allows renaming of the relation itself,e.g.,**`Paternity,1→ Parenthood,Parent`**

Renaming and Union

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

$\rho_{\text{Father->Parent}}$(Paternity) $\cup$ $\rho_{\text{Mother->Parent}}$(Maternity)

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Renaming and Union, with Several Attributes

Employees

Surname	Branch	Salary
Patterson	Rome	45
Trumble	London	53

Staff

Surname	Factory	Wages
Patterson	Rome	45
Trumble	London	53

$\rho_{\text{Branch,Salary}\rightarrow\text{Location,Pay}}$(Employees) $\cup$ $\rho_{\text{Factory, Wages}\rightarrow\text{Location,Pay}}$(Staff)

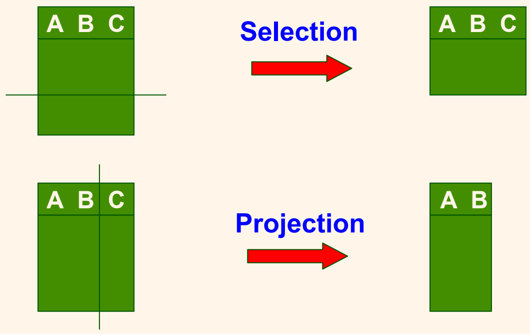
Surname	Location	Pay
Patterson	Rome	45
Trumble	London	53
Cooke	Chicago	33
Bush	Monza	32

Selection and Projection

→These are unary operators, in a sense orthogonal:

✓selection for "horizontal" decompositions;

✓projection for "vertical" decompositions.



Selection

→This is a unary operation which returns a relation

✓with the same schema as the operand;

✓but, with a **subset of the tuples** of the operand, i.e., only those that satisfy a condition.

→Notation: $\sigma_F(r)$

→Semantics: $\sigma_F(r) = \{ t \mid t \in r \text{ s.t. } t \text{ satisfies } F, \text{ I.e., } F(t)\}$

Selection Example

Employees

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Black	Lucy	40	3000
Verdi	Nico	36	4500
Smith	Mark	40	3900

$\sigma_{\text{Age<30 v Salary>4000}}$ **(Employees)**

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Verdi	Nico	36	4500

Selection, Another Example

Citizens

Surname	FirstName	PlaceOfBirth	Residence
Smith	Mary	Rome	Milan
Black	Lucy	Rome	Rome
Verdi	Nico	Florence	Florence
Smith	Mark	Naples	Florence

$\sigma_{\textbf{PlaceOfBirth=Residence}}$ **(Citizens)**

Surname	FirstName	PlaceOfBirth	Residence
Black	Lucy	Rome	Rome
Verdi	Nico	Florence	Florence

Projection

- Projection returns a relation which includes a **subset of the attributes** of the operand.
- Notation: Given a relation r(X) and a subset Y of X:

  $\pi_Y(r)$
- Semantics: $\pi_Y(r) = \{ t[Y] \mid t \in r \}$

Example of Projection

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{\text{Surname, FirstName}}$**(Employees)**

Surname	FirstName
Smith	Mary
Black	Lucy
Verdi	Mary
Smith	Mark

Another Example

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{\text{Department, Head}}$ **(Employees)**

Department	Head
Sales	De Rossi
Personnel	Fox

Cardinality of Projection Operations

- Note that the result of a projection contains at most as many tuples as the operand relation.
- However, it may contain fewer, if several tuples collapse, i.e., they are identical in all their values.
- **Theorem:** $\pi_Y(r)$ contains as many tuples as r if and only if Y is a superkey for r.
- This property holds even if Y is "by chance” a superkey, i.e., it is not defined as a superkey in the schema, but it is a superkey for the current database, see the example.

Tuples that Collapse

Students

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

$\pi_{\text{Surname, DegreeProg}}$ **(Students)**

Surname	DegreeProg
Smith	Computing
Smith	Engineering
Black	Fine Art

Tuples that do not Collapse, "by Chance"

Students

RegNum	Surname	FirstName	BirthDate	DegreeProg
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Engineering

$\pi_{\text{Surname, DegreeProg}}$ **(Students)**

Surname	DegreeProg
Smith	Computing
Smith	Engineering
Black	Fine Art
Black	Engineering

Join

- The most used operator in the relational algebra.
- Allows us to establish connections among data in different relations, taking advantage of the "value-based" nature of the relational model.
- Two main versions of the join:
  - "**natural**" join: takes attribute names into account;
  - "**theta**" join.
- Both join operations are denoted by the symbol $\bowtie$.

A Natural Join

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
sales	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Smith	sales	Brown
Black	production	Mori
White	production	Mori

Definition of Natural Join

- $r_1 (X_1)$, $r_2 (X_2)$
- $r_1 \bowtie r_2$ (natural join of r_1 and r_2) is a relation on X_1X_2 (the **union** of the two sets):

  $\{ t \text{ on } X_1X_2 \mid t[X_1] \in r_1 \text{ and } t[X_2] \in r_2 \}$

  or, equivalently

$$\{ t \text{ on } X_1X_2 \mid \text{exist } t_1 \in r_1 \text{ and } t_2 \in r_2 \text{ with } t[X_1] = t_1 \text{ and } t[X_2] = t_2 \}$$

Natural Join: Comments

- The tuples in the resulting relation are obtained by combining tuples in the operands with **equal values on the common attributes**
- The common attributes **often form a key of one of the operands** (remember: references are realized by means of foreign keys, and we join in order to follow references)
  - Not always! Consider Person(Name,Addr,PostalC) and let us define Neighbour(Name,Addr,Name1,Addr1,PostalC) by joining Person with $\rho_{Name,Addr \rightarrow Name1,Addr1}$(Person); What is criterion for neighbourhood here?

Another Example

Offences

Code	Date	Officer	Dept	Registartion
143256	25/10/1992	567	75	5694 FR
987554	26/10/1992	456	75	5694 FR
987557	26/10/1992	456	75	6544 XY
630876	15/10/1992	456	47	6544 XY
539856	12/10/1992	567	47	6544 XY

Cars

Registration	**Dept**	**Owner**	…
6544 XY	75	Cordon Edouard	…
7122 HT	75	Cordon Edouard	…
5694 FR	75	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Offences ⋈ Cars

Code	**Date**	**Officer**	**Dept**	**Registration**	**Owner**	…
143256	25/10/1992	567	75	5694 FR	Latour Hortense	…
987554	26/10/1992	456	75	5694 FR	Latour Hortense	…
987557	26/10/1992	456	75	6544 XY	Cordon Edouard	…
630876	15/10/1992	456	47	6544 XY	Cordon Edouard	…
539856	12/10/1992	567	47	6544 XY	Mimault Bernard	…

Yet Another Join

→ In this example, join gives very different results from union (see earlier example)

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity ⋈ Maternity

Father	Child	Mother
Adam	Cain	Eve
Abraham	Isaac	Sarah
Abraham	Ishmael	Hagar

Joins can be Incomplete

→If a tuple does not have a "counterpart" in the other relation, then it does not contribute to the join ("dangling" tuple)

r_2

Department	Head
production	Mori
purchasing	Brown

r_1

Employee	Department
Smith	sales
Black	production
White	production

$r_1 \bowtie r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori

Joins can be Empty

→As an extreme, we might have that no tuple has a counterpart, and all tuples are dangling

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
marketing	Mori
purchasing	Brown

$r_1 \bowtie r_2$

Employee	Department	Head

Another Extreme

- If each tuple of each operand can be combined with all the tuples of the other, then the join has a cardinality that is the product of the cardinalities of the operands

r_1

Employee	Project
Smith	A
Black	A
White	A

r_2

Project	Head
A	Mori
A	Brown

$r_1 \bowtie r_2$

Employee	Project	Head
Smith	A	Mori
Black	A	Brown
White	A	Mori
Smith	A	Brown
Black	A	Mori
White	A	Brown

How Many Tuples in a Join?

- Given $r_1\,(X_1)$, $r_2\,(X_2)$ the join has cardinality

$$0 \leq |\, r_1 \bowtie r_2 \,| \leq |\, r_1 \,| \times |\, r_2 |$$

where $|\, r \,|$ is the cardinality of relation r.

- Moreover:
  - ✓ if the join is complete, then its cardinality is at least the maximum of $|\, r_1 \,|$ and $|\, r_2 |$.
  - ✓ if $X_1 \cap X_2$ contains a key for r_2,

    then $|\, r_1 \bowtie r_2 \,| \leq |\, r_1 |$
  - ✓ if $X_1 \cap X_2$ is the primary key for r_2, and there is a referential constraint between $X_1 \cap X_2$ in r_1 and such a key, then $|\, r_1 \bowtie r_2 \,| = |\, r_1 |$.

Outer Join

- A variant of the join, to keep all pieces of information from the operands.
- An outer join operation "pads with nulls" the tuples in one operant relation that have no counterpart in the other relation.
- Three variants:
  - ✓**LEFT** — only tuples of left operand are padded;
  - ✓**RIGHT** — only tuples of right operand are padded;
  - ✓**FULL** — tuples of both operands are padded.

Outer Join Operations

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

$r_1 \bowtie_{LEFT} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori

$r_1 \bowtie_{RIGHT} r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

$r_1 \bowtie_{FULL} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

N-ary Join Operations

- The natural join is
  - commutative: $r_1 \bowtie r_2 = r_2 \bowtie r_1$
  - associative: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$
- Therefore, we can write n-ary joins without ambiguity:

$$r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

Example of N-ary Join Operation

r_1

Employee	Department
Smith	sales
Black	production
Brown	marketing
White	production

r_2

Department	Division
production	A
marketing	B
purchasing	B

r_3

Division	Head
A	Mori
B	Brown

$r_1 \bowtie r_2 \bowtie r_3$

Employee	Department	Division	Head
Black	production	A	Mori
Brown	marketing	B	Brown
White	production	A	Mori

Join and Intersection

- We have made no assumptions about the sets of attributes X_1 and X_2 on which the operands of a join operation are defined; the two sets could even be equal or disjoint.
- If $X_1 = X_2$ then $r_1 \bowtie r_2 = r_1 \cap r_2$ since, by definition, the result is a relation which includes tuples t such that $t[X_1] \in r_1$ and $t[X_2] \in r_2$, and $X_1 = X_2$.

Natural Join as Cartesian Product

- The natural join is defined also when the operands have no attributes in common.
- In this case no condition is imposed on tuples, and therefore the result contains tuples obtained by combining the tuples of the operands in all possible ways.

Cartesian Product: Example

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes $\bowtie$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

Theta-Join

→ In most cases, a Cartesian product is meaningful only if followed by a selection:

- ✓ **theta-join**: a derived operator

$$r_1 \bowtie_F r_2 = \sigma_F(r_1 \bowtie r_2)$$

- ✓ if F is a conjunction of equalities, then we have an **equi-join**

Equi-join: example

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes $\bowtie_{Project=Code}$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	B	Mars

Division

- Consider two relations A(x,y), B(y) and suppose we want to specify the query

  "Find all A's that are associated with all B's"
- This can be expressed as

  $$A/B = \pi_x(A) - \pi_x((\pi_x(A) \bowtie B) - A)$$
- This means that division does not extend the expressiveness of Relational Algebra, but it is a convenient operation to use in many situations.

Example of Division

- Assume
  - ✓ Take(x,y) - "student x has taken course y",
  - ✓ CS(y) - "y is a CS course"
- We want "All students who have taken all CS courses"
  - ✓ $\pi_x(\text{Take}) \bowtie \text{CS}$ -- ? Table of all students, CS courses ....
  - ✓ $(\pi_x(\text{Take}) \bowtie \text{CS}) - \text{Take}$ -- ?? Table of all students and the CS courses they have not taken
  - ✓ $\pi_x((\pi_x(\text{Take}) \bowtie \text{CS}) - \text{Take})$ -- ??? All students who have not taken a CS course
  - ✓ $\pi_x(\text{Take}) - \pi_x((\pi_x(\text{Take}) \bowtie \text{CS}) - \text{Take})$ -- ???? All students who have taken all CS courses

Queries

- A query is a function from database instances to relations.
- Queries are formulated in relational algebra by means of expressions over relations.

A Sample Database

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Example 1

"Find the numbers, names and ages of employees earning more than 40k."

Employees(<u>Number</u>,Name,Age,Salary)

Supervision(Head,<u>Emp</u>)

Try it!

Example 2

→ "Find the registration numbers of the supervisors of the employees earning more than 40M."

Employees(<u>Number</u>,Name,Age,Salary)

Supervision(Head,<u>Emp</u>)

Try it!

Example 3

→ "Find the names and salaries of the supervisors of the employees earning more than 40M."

Employees(<u>Number</u>,Name,Age,Salary)

Supervision(Head,<u>Emp</u>)

Try it! (this is a bit tougher)

Example 4

→ "Find the employees earning more than their respective supervisors, return registration numbers, names and salaries of the employees and their supervisors."

Employees(Number,Name,Age,Salary)

Supervision(Head,Emp)

Try it! Definitely challenging ☺

Example 5

→ "Find registration numbers and names of supervisors, ***all*** of whose employees earn more than 40M."

Employees(Number,Name,Age,Salary)

Supervision(Head,Emp)

Try it!

Another Series of Examples:

Films(<u>Film#</u>,Title,Director,Year,ProdCost)

Artists(<u>Actor#</u>,Surname,FirsName,Sex,Birthday, Nationality)

Roles(<u>Film#,Actor#,Character</u>)

→ Find "The titles of films starring Henry Fonda

Try it!

Example 2

Films(<u>Film#</u>,Title,Director,Year,ProdCost)

Artists(<u>Actor#</u>,Surname,FirsName,Sex,Birthday, Nationality)

Roles(<u>Film#,Actor#,Character</u>)

→ Find "The titles of all films in which the director is also an actor"

Try it!

Example 3

Films(<u>Film#</u>,Title,Director,Year,ProdCost)

Artists(<u>Actor#</u>,Surname,FirsName,Sex,Birthday, Nationality)

Roles(<u>Film#,Actor#,Character</u>)

→ Find "The actors who have played two characters in the same film; show the title of each such film, first name and surname of the actor and the two characters"

Try it!

Example 4

Films(<u>Film#</u>,Title,Director,Year,ProdCost)

Artists(<u>Actor#</u>,Surname,FirsName,Sex,Birthday, Nationality)

Roles(<u>Film#,Actor#,Character</u>)

→ "The titles of the films in which the actors are all of the same sex"

Try it!

Relational Algebra and Null Values

People

Name	Age	Salary
Aldo	35	15
Andrea	27	21
Maria	NULL	42

→ Consider $\sigma_{Age>30}$ (People)

→ Which tuples belong to the result?

→ The first yes, the second no, but the third??

Lecture Example (for blackboard)

Blackboard Example II