Database Schema and Relational Algebra

Chapters 1, 2, 6.1

Schema

- Logical design of a database

- Example:

  student (banner_id, random_id, first_name, last_name, year_of_attendance)

  course (course_code, section_code, semester, year, building, room_number)

- Types of keys:
  - **Superkey** - a set of attributes that can be uniquely used to identify a database record
  - **Candidate key** - a ‘minimal’ superkey
  - **Primary key** - a DBA-chosen candidate key

Relational Algebra

- A family of algebras with a well-founded semantics used for modelling the data stored in relational databases, and defining queries on it
  - **Unary operators (ex., select)**: *op* <relation1> -> <relation2>
  - **Binary operators (ex., union):** <relation1> *op* <relation2> -> <relation3>

- Theoretical foundation for query languages for relational databases

- **Note: A relation is a set, does not contain duplicate values**

Basic Operators

- Select (σ)
- Project (π)
- Union (∪)
- Set Difference (–)
- Cartesian Product (×)
- Rename (ρ)

Select (σ)

- Notation: **σ** ***<predicate>*** **(<relation>)**
- Returns rows of the input relation satisfying the predicate
- Example:

$$\sigma_{\ dept\_name = \text{"Comp. Sci."} \wedge salary > 70000}\ \textbf{(instructor)}$$

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

→

ID	name	dept_name	salary
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

Select (σ) (cont.)

- Predicate may include comparisons between two attributes
- Example:

$$\sigma_{\,width\,=\,height}\ (\textbf{device})$$

Id	name	width	height
1	iphone	3.07	6.24
2	samsung	2.70	5.21
3	blackberry	5.0	5.0
4	lg	3.0	5.86

→

Id	name	width	height
3	blackberry	5.0	5.0

Project (π)

- Notation: **π** ***<attribute1>, <attribute2>, …*** **(<relation>)**
- Returns the given relation by leaving out the unspecified attributes
- Example:

π ***dept_name, salary*** **(instructor)**

ID	name	dept_name	salary
1	Srinivasan	Mathematics	65000
2	Wu	Finance	70000
3	Einstein	Physics	62000
4	Singh	Finance	70000

dept_name	salary
Mathematics	65000
Finance	70000
Physics	62000

Generalized Project (π)

- Instead of attributes, we can use arithmetic operations and/or string operations
- Example:

$$\pi_{\ name,\ salary\ \div\ 12}\ (\textbf{instructor})$$

ID	name	dept_name	salary
1	Srinivasan	Mathematics	65000
2	Wu	Finance	70000
3	Einstein	Physics	62000
4	Singh	Finance	70000

→

name	salary
Srinivasan	5416.7
Wu	5833.3
Einstein	5166.7
Singh	5833.3

Union (U)

- Notation: **<relation1> U <relation2>**
- <u>Two conditions:</u>
  - The relations must have the same number of attributes
  - Respective attributes must have the same domain
- Example:

Find the ids of all courses taught in Fall 2009, Spring 2010, or both.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Union (∪) (Cont.)

$$\pi_{course\_id}(\sigma_{semester = \text{“Fall”} \wedge year = 2009}(\text{section})) \cup$$
$$\pi_{course\_id}(\sigma_{semester = \text{“Spring”} \wedge year = 2010}(\text{section}))$$

course_id
CS-101
CS-347
PHY-101

∪

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

→

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Set Difference (-)

- Notation: **<relation1> - <relation2>**
- Returns tuples that are in <relation1> but not in <relation2>
- Example:

Find the courses offered in Fall 2009 but not in Spring 2010.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Set Difference (-) (Cont.)

$$\pi_{course\_id}(\sigma_{semester = \text{"Fall"} \wedge year = 2009}(section)) -$$
$$\pi_{course\_id}(\sigma_{semester = \text{"Spring"} \wedge year = 2010}(section))$$

course_id
CS-101
CS-347
PHY-101

-

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

→

course_id
CS-347
PHY-101

Cartesian Product (×)

- Notation: <relation1> x <relation2>
- Returns every possible tuple between <relation1> and <relation2>
- Example:

  brand x device

id	brand_name
1	Apple
2	LG

×

id	device_name
1	Phone
2	TV

→

brand.id	brand_name	device.id	device_name
1	Apple	1	Phone
1	Apple	2	TV
2	LG	1	Phone
2	LG	2	TV

Rename (ρ)

- Notation: **ρ** ***NewRelationName*** **(*NewAttr1Name, NewAttr2Name, ...*) (relation)**
- Renames the given relation along with its attributes (if specified)
- Useful for set operations

Additional Operators

- Set intersection (∩)
- Joins (⋈, ⟗, ⟕, ⟖)
- Assignment (⟵)
- Aggregation (G)
- Division (÷)

Set intersection (∩)

- Notation: **<relation1> ∩ <relation2>**
- Example:

  Find the courses offered in both Fall 2009 and Spring 2010

$$\pi_{course\_id}\,(\sigma_{semester = \text{“Fall”} \wedge year = 2009}\,(\text{section})) \cap$$
$$\pi_{course\_id}\,(\sigma_{semester = \text{“Spring”} \wedge year = 2010}\,(\text{section}))$$

course_id
CS-101
CS-347
PHY-101

∩

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

→

course_id
CS-101

Joins: Natural Join (⋈)

➢ Notation: **<relation1> ⋈ <relation2>**
➢ Joins two relation on their common attributes
➢ Example:

instructor ⋈ teaches

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

⋈

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

→

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Joins: Outer Joins (⟕, ⟖, ⟗)

- ➢ Left Outer Join (⟕), Right Outer Join (⟖), Full Outer Join (⟗)
- ➢ Includes the tuples from specified relation that did not have any matches in natural join
- ➢ Example:

  instructor ⟕ teaches

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
33456	Gold	Physics	87000	*null*	*null*	*null*	*null*
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
58583	Califieri	History	62000	*null*	*null*	*null*	*null*
76543	Singh	Finance	80000	*null*	*null*	*null*	*null*
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Assignment (←)

- Notation: **<variable>** ← **<temporary_relation>**
- Example:

fall2009Courses ← $\pi_{course\_id}(\sigma_{semester = \text{“Fall”} \wedge year = 2009}(section))$

spring2010Courses ← $\pi_{course\_id}(\sigma_{semester = \text{“Spring”} \wedge year = 2010}(section))$

fall2009Courses ∩ spring2010Courses

Aggregation (G)

- Notation: *<attribute1>, <attribute2>, ...* G *<Aggregate Function1>, <Aggregate Function2>, ...* (relation)
- Common aggregate functions:
  - sum(<attribute>)
  - avg(<attribute>)
  - count(<attribute>)
  - min/max(<attribute>)
- Example:

  Find the sum of the salaries of all instructors:

G ***sum(salary)*** **(instructor)** →

sum(salary)
898000

Aggregation (G) (Cont.)

➢ Example 2:

For each combination of brand and type, find the sum and average of the prices of the devices

$$_{brand,\ type}\,\mathcal{G}_{\,sum(price),\ avg(price)}\ (device)$$

id	name	brand	type	price
1	iPhone 6	Apple	Phone	400
2	MacBook Air	Apple	Laptop	700
3	LG G4	LG	Phone	300
4	iPhone X	Apple	Phone	1000
5	GPad F2	LG	Tablet	150
6	LG G7	LG	Phone	750
7	Macbook Pro	Apple	Laptop	1300
8	GPad X	LG	Tablet	130

→

brand	type	sum(price)	avg(price)
Apple	Phone	1400	700
Apple	Laptop	2000	1000
LG	Phone	1050	525
LG	Tablet	280	140

Aggregation (G) (Cont.)

- To apply aggregate function on distinct values of an attribute, append **distinct**
- Example:

$$\mathbf{G}_{\ count\text{-}distinct(type)}\ \mathbf{(device)}$$

id	name	brand	type	price
1	iPhone 6	Apple	Phone	400
2	MacBook Air	Apple	Laptop	700
3	LG G4	LG	Phone	300
4	iPhone X	Apple	Phone	1000
5	GPad F2	LG	Tablet	150
6	LG G7	LG	Phone	750
7	Macbook Pro	Apple	Laptop	1300
8	GPad X	LG	Tablet	130

→

count-distinct(type)
3

Division (÷)

- Notation: **<R> ÷ <S>**
- Returns a relation with attribute values from R that can be paired to all tuples in S
- Some queries are difficult to
- express in relational algebra

Find the Models that were produced in 2000, 2001 and 2002 (i.e. all years in Year relation)

Model	Year
A	2000
A	2001
A	2002
B	2000
B	2001
C	2002

Year
2000
2001
2002

Division (÷)

- Notation: **<R> ÷ <S>**
- Returns a relation with attribute values from R that can be paired to all tuples in S
- Some queries are difficult to express in relational algebra

Find the Models that were produced in 2000, 2001 and 2002 (i.e. all years in Year relation)

Model	Year
A	2000
A	2001
A	2002
B	2000
B	2001
C	2002

÷

Year
2000
2001
2002

=

Model
A

Division (÷) (is the opposite of ×)

$$\alpha \times \beta = \gamma$$

A
B

1
2
3

A	1
A	2
A	3
B	1
B	2
B	3

Division (÷) (is the opposite of ×)

$\gamma \div \beta = \alpha$

γ

A	1
A	2
A	3
B	1
B	2
B	3

β

1
2
3

α

A
B

$\gamma \div \alpha = \beta$

γ

A	1
A	2
A	3
B	1
B	2
B	3

α

A
B

β

1
2
3

Division (÷) (is the opposite of ×)

- Find all possible answers
- Find *universe* of pairings
- Find *impossible* pairings
- Remove them from possible values

$$\gamma \div \beta = \alpha$$

A	1
A	2
A	3
B	1
B	2
B	3
C	2
C	3

1
2
3

A
B

Division (÷)

- Find *possible* answers

A
B
C

Division (÷)

➢ Find *universe* of pairings

A
B
C

×

1
2
3

=

A	1
A	2
A	3
B	1
B	2
B	3
C	1
C	2
C	3

Division (÷)

➢ Find *impossible* pairings

A
B
C

×

1
2
3

=

A	1
A	2
A	3
B	1
B	2
B	3
C	1
C	2
C	3

-

A	1
A	2
A	3
B	1
B	2
B	3
C	2
C	3

=

C	1

Division (÷)

- ➢ Remove them from possible answers