Relational Model

- Structure of Relational Databases
- Relational Algebra
  - Fundamental Relational Algebra Operations
  - Additional Relational Algebra Operations
  - Extended Relational Algebra Operations

A Relation: A Table

attributes
(or columns)

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

tuples
(or rows)

The *Instructor* Relation

Relational Database

Database is a collection of data

- A relational database consists of a collection of tables.
- Each table is assigned a unique name.
- A row in a table represents a relationship among a set of values.
  - A table is a collection of such relationship.
- Relational data model is the primary data model for commercial data-processing applications.
  - Because of its simplicity
    - Easy use for programmer
    - Hard job for DBMS software.

What is Data Model:
To describe: Data, Data relalationship, Data Semantices, Data Constraints

Basic Structure

- Formally, given sets D_1, D_2, …. D_n , A *relation r* is a subset of
  D_1 x D_2 x … x D_n

  Thus a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ *where* $a_i \in D_i$

- Example: if

  Dept-name = { Biology, Comp.Sci, Finance, History }
  Building = {Watson, Taylor, Painter}
  Budget = {90000, 120000, 50000}

  Then *r* = { *(Biology, Watson, 90000),*
  (Comp.Sci, Taylor, 120000),
  (Finance, Painter, 120000),
  (History, Painter, 50000)
  }

  is a relation over *Dept-name x Building x Budget*

Attribute Types

- A_i : Each attribute of a relation has a name
- D_i : The set of allowed values for each attribute is called the domain of the attribute
- Attribute values in a relation are (normally) required to be atomic , that is, indivisible
  - Can NOT assuming part of a value has specific meaning, E.g. the student ID only represents the unique id of a student, but not other information, major/year of registration, etc.
  - E.g. Phone number(s) of a instructor
- The special value *null* is a member of every domain
  - The null value causes complications in the definition of many operations

Relation Schema and Instance

- $R = (A_1, A_2, \ldots, A_n)$ is a **relation schema**, *while*
  - $A_1, A_2, \ldots, A_n$ are **attributes**
  - *Example:*
    Instructor-Schema = (ID, name, dept_name, salary)

- r(*R*) is a **relation** on the *relation schema R*
  - *E.g. instructor (Instructor-Schema)*
  - The values of a relation may be different time to time
  - The current values (relation instance) of a relation are specified by a table
  - An element t of r is a tuple, represented by a row in a table

- *People often use the same name for the schema and relation.*

Relations are Unordered

- Order of tuples (row) is irrelevant (may be stored in an arbitrary order)
  - *E.g. account relation with unordered tuples*

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

- Logically, Order of attributes (column) is irrelevant

Database Schema

- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
  - schema: *instructor* (*ID, name, dept_name, salary*)
  - Instance:

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

The concepts of Keys

- A database consists of multiple relations
  - Database = Set of relations
- How to specify a tuple within a given relation is distinguished?
- How to represents a relationship among the relations?
- The concepts of Keys
  - Super key (or Key)
  - Candidate key
  - Primary Key
  - Foreign Key

Keys

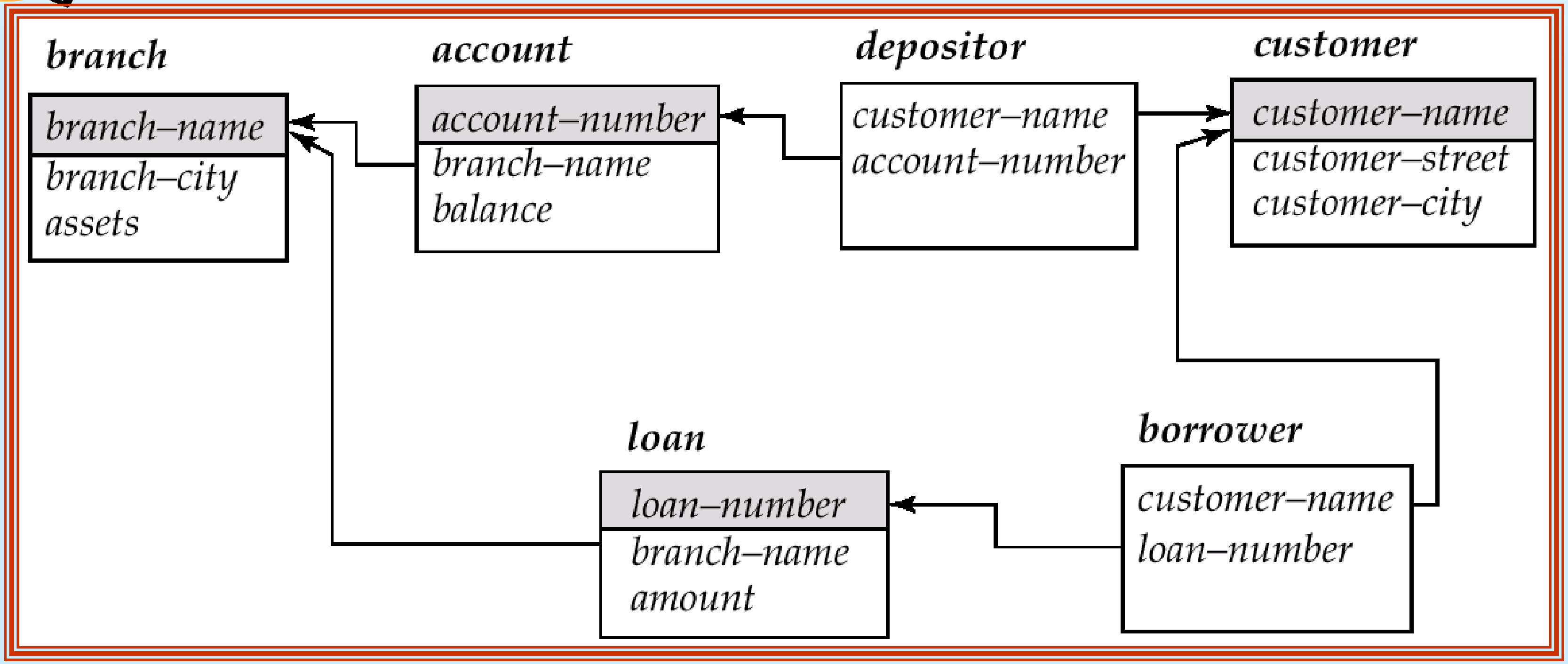
- Let K $\subseteq$ R
- *K* is a ***superkey*** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - By "possible *r*" we mean a relation *r* that could exist in the enterprise we are modeling.
  - Example:
    - {*ID} and* {*ID, name*} are both superkeys of *instructor* , if no two instructors can possibly have the same ID.
    - {*name*} is not a superkey of *Instructor*
- *K* is a ***candidate key*** if *K* is minimal
  - Formal definition: if K is a superkey of R, and any subset of K is not a superkey of R, then K is a candidate K of R
    - Example: {*ID* } is a candidate key for *Instructor*.

Primary Key

- **It is possible to have more than one candidate key.**
  - E.g. {ID} and {email-address} are both unique, can serve as candidate key for Instructor.
- **Primary key:** a candidate key chosen as the principal means of identifying tuples within a relation
  - Should choose an attribute whose value never, or very rarely, changes.
    - E.g. email address is unique, but may change

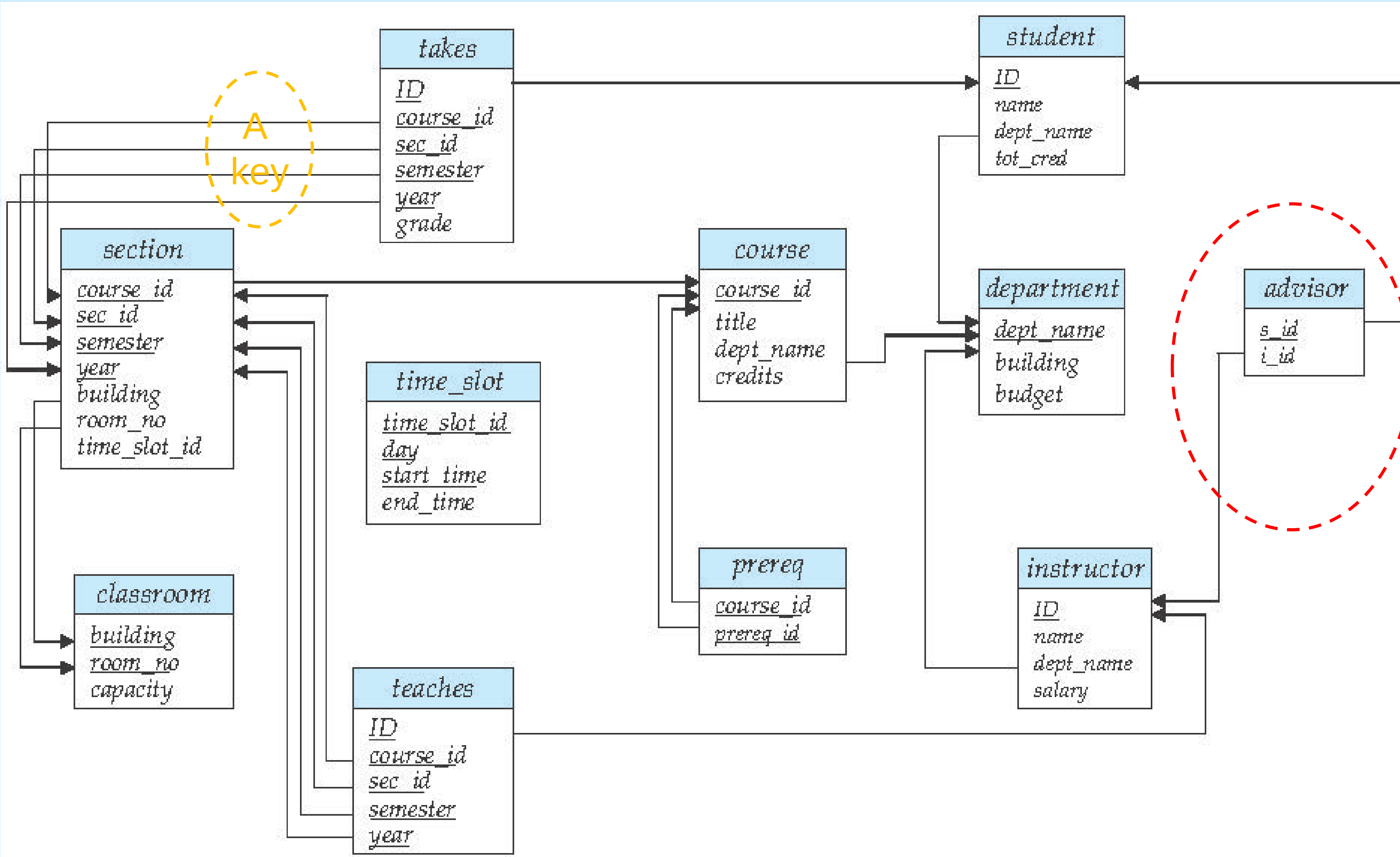
Primary key is chosen by me, which one do I like?

Schema Diagram for the Banking Enterprise



- **Foreign Key: The attributes of a relation schema r1 is the primary key of another relation schema r2. The attributes is called a foreign key from r1, referencing r2.**
  - The attribute branch-name in Account is a foreign key referencing Branch.
  - Only values occurring in the primary key attribute of the **Referenced relation** may occur in the foreign key attribute of the **Referencing relation**

Schema Diagram for University Database



Relational Query Languages

- Language in which user requests information from the database.
- Categories of languages
  - Procedural: The user instructs the system to perform a sequences of operations on the database.
  - non-procedural: The user describes the desired information without giving a specified procedure for obtaining that information.
- "Pure" languages: are equivalent in computing power
  - Relational Algebra *Procedural*
  - Tuple Relational Calculus *Non-Procedural*
  - Domain Relational Calculus *Non-Procedural*
- Pure languages form underlying basis of query languages that people use.

Relational Algebra

- Procedural language
- Six fundamental operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- The operators take two or more relations as inputs and give a new relation as a result.
  - The operators could be combined as needed to perform a sophisticated query to the database.

Select Operation – Example

- Relation *r*

A	*B*	*C*	*D*
α	α	*1*	*7*
α	β	*5*	*7*
β	β	*12*	*3*
β	β	*23*	*10*

- $\sigma_{A=B \wedge D > 5}(r)$

A	*B*	*C*	*D*
α	α	*1*	*7*
β	β	*23*	*10*

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq. <. \leq$

- Example of selection:

$$\sigma_{dept\_name=\text{“Physics”}}(instructor)$$

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\Pi_{A1,\ A2,\ \ldots,\ Ak}\ (r)$$

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query*:*

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

- Result:

ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

$$\Pi_{name}(\sigma_{dept\_name\ =\text{“Physics”}}\ (instructor))$$

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

Cartesian-Product Operation

- The Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- Notation *r* x *s*
  - Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

  - Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
  - If attributes of *r(R)* and *s(S*) are not disjoint, then renaming must be used.

Cartesian-Product Operation-Example

Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

The *instructor* x *teaches* table

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Set Union Operation

- The union operation allows us to combine two relations
- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  1. *r, s* must have the *same* *arity* (same number of attributes)
  2. The attribute domains must be *compatible* (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \ \land \ year=2017} (section)) \ \cup$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \ \land \ year=2018} (section))$$

Union Operation (Cont.)

- Result of:

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \ \wedge \ year=2017} (section)) \ \cup$$

$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \ \wedge \ year=2018} (section))$$

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

Set Difference Operation

- Notation *r – s*
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - *r* and *s* must have the *same arity*
  - attribute domains of *r* and *s* must be compatible
- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \land year=2017}(section)) -$$
$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \land year=2018}(section))$$

course_id
CS-347
PHY-101

Set Difference Operation – Example

- Relations *r, s:*

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

r – s:

A	*B*
α	1
β	1

Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator, ρ, is provided for that purpose
  - The expression:

    $$\rho_x (E)$$

    returns the expression *E* under the name *X*

- To refer to a relation by more than one name
  - Relations r

A	B
α	1
β	2

  - r x ρ_s(r)

$r.A$	$r.B$	$s.A$	$s.B$
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Rename Operation

- If a relational-algebra expression E has arity n, then

$$\rho_{X\ (A1,\ A2,\ \ldots,\ An)}\ (E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to *A1, A2, …., An*.

Formal Definition of Relational Algebra

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural Join and Theta Join
- Outer Join
- Assignment
- Division

Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations.
- Notation: $r \cap s$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
  - Note: $r \cap s = r - (r - s)$
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \ \wedge \ year=2017} (section)) \cap$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \ \wedge \ year=2018} (section))$$

course_id
CS-101

Natural-Join Operation

- Notation: r $\bowtie$ s
- Let *r* and *s* be relations on schemas *R* and *S* respectively. The result is a relation on schema $R \cup S$ which is obtained by considering each pair of tuples t_r from *r* and t_s from *s*.
- If t_r and t_s have the same value on each of the attributes in $R \cap S$, a tuple *t* is added to the result, where
  - *t* has the same value as t_r on *r*
  - *t* has the same value as t_s on *s*
- Example:

  $R = (A, B, C, D)$

  $S = (E, B, D)$
- Result schema = (A, B, C, D, E)
- $r \bowtie s$ is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B\, =\, s.B \quad r.D\, =\, s.D}\ (r \ \times\ s))$$

Natural Join Operation – Example

- Relations r, s:

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Join Operation (Cont.)

- The university sample

$$\Pi_{name,\ course\_id}\ (instructor \bowtie \text{teaches})$$

- The *Theta join* operation is a variant of the natural join that combine the selection and Cartesian production into a single operation.

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

- Example

$$\sigma_{instructor.id\ =\ teaches.id}\ (instructor\ \text{x}\ teaches\ ))$$

Can equivalently be written as

$$instructor \bowtie_{Instructor.id\ =\ teaches.id} teaches.$$

Outer Join Operation

- The outer join operation is an extension of the join to deal with the missing information.
  - Example: if there is some instructors who teaches no course

$$instructor \bowtie \text{teaches}$$

- Outer join *instructor* ⟕ teaches makes sure all instructor's data appears at lease once in the result.

- Three outer join operations:
  - Left outer join: ⟕
  - Right outer join: ⟖
  - Full outer join: ⟗

- Outer join operation need deal with null values, will further discuss in SQL language.

The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables.
- The assignment operation is denoted by $\leftarrow$ and works like assignment in a programming language.
- Example: Find all instructor in the "Physics" and Music department.

$$Physics \leftarrow \sigma_{dept\_name=\text{“}Physics\text{”}}(instructor)$$

$$Music \leftarrow \sigma_{dept\_name=\text{“}Music\text{”}}(instructor)$$

$$Physics\ r \cup Music$$

- With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.
  - May use temporary variable in subsequent expressions.

Division Operation

$$r \div s$$

- Suited to queries that include the phrase "for all".
- Let *r* and *s* be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s \, (tu \in r) \}$$

Where *tu* means the concatenation of tuples *t* and *u* to produce a single tuple

Division Operation – Example

Relations *r*, *s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

$r \div s$:

A
α
β

Another Division Example

Relations *r, s*:

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

s

$r \div s$:

A	B	C
α	a	γ
γ	a	γ

Division Operation (Cont.)

- Property
  - Let $q = r \div s$
  - Then q is the largest relation satisfying $q \times s \subseteq r$
- Definition in terms of the basic algebra operation
  Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}\left((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r) \right)$$

To see why

- $\Pi_{R-S,S}(r)$ simply reorders attributes of r
- $\Pi_{R-S}(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ gives those tuples t in

  $\Pi_{R-S}(r)$ such that for some tuple $u \in s$, $tu \notin r$.

Example Queries

- Find the students who learned course “Database system” and “Advanced programing”
  - Query 1

$$\Pi_{ID}\,(\sigma_{title\,=\,\text{“Database system”}}\,(takes \bowtie course\,))\ \cap$$

$$\Pi_{ID}\,(\sigma_{title\,=\,\text{“Advanced Programing”}}\,(takes \bowtie course\,))$$

  - Query 2

$$\Pi_{ID,\ title}(takes \bowtie course)$$

$$\div\ \rho_{temp(title)}\,(\{(\text{“Database System”}),\ (\text{“Advanced Programming”})\})$$

Example Queries

- Find the students who learned all the courses in “Comp.Sci” department.

$$\Pi_{ID,\ course\_id}(takes) \div \Pi_{course\_id}(\sigma_{tdept\_name = \text{“Comp.Sci”}}(\text{course}))$$

Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department
  - Query 1

$$\sigma_{dept\_name=\text{“Physics”}}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$$

  - Query 2

$$(\sigma_{dept\_name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$$

- The two queries are not identical; they are, however, equivalent
  - They give the same result on any database.
  - Which one is better?

Example Queries

Find the instructor who has the highest salary

- *Self join: Rename instructor relation as d*
- The query is:

$$\Pi_{id,\ name}(instructor) - \Pi_{instructor.id,\ instructor.name}(\sigma_{instructor.salary\ <\ d.salary}(instructor \times \rho_d(instructor)))$$

Extended Relational-Algebra Operations*

- Some of the data query needs could not be expressed using the basic relational algebra. Therefore, some additional operations been defined.

- Generalized projection, allowing calculations

  $\Pi_{\textit{ID, name, dept\_name,salary/12}}$ (*instructor*)

- Aggregations
  - Aggregation functions take a collection of values and return a single value as a result.

Aggregation Operations*

- Notation

$$_{G1,G2,\ldots,Gn}\ \mathcal{G}\ _{F1(A1),F2(A2),\ldots,Fm(Am)}\ (E)$$

  - All tuples in a group has the same value for G1,G2,…, Gn
  - Tuples in different groups has different values for G1,G2,…, Gn
  - For each group, has one tuple in the result. Fi are aggregation functions.
  - Aggregation functions including SUM/MAX/MIN/AVG/COUNT
- Examples

$$\mathcal{G}_{sum(salary)}\ (\text{instructor})$$

$$_{dept\_name}\mathcal{G}_{sum(salary)}\ (\text{instructor})$$

$$\mathcal{G}_{count\text{-}distinct(ID)}\ (\sigma_{semester=\text{“Spring”}\ \wedge\ year=2010}\ (\text{teaches}))$$