Relational Algebra

ISYS2120 Data and Information Management

Prof Alan Fekete

University of Sydney

Acknowledge: slides from Uwe Roehm and Alan Fekete, and from the materials associated with reference books (c) McGraw-Hill, Pearson

Agenda

- *Motivation (not examinable)*
- Mathematical definition of relations
- The operations
- Some rules
- Conversion skills

Motivation

Not examinable

- Relational database consists of tables, and a query result is a table
- It is desirable to describe how to perform computation which combines tables to produce a new one
- This is used as internal language in DBMS as it aims to compute efficiently
  - Rules allow platform to consider several ways to compute same result, and choose one which is most efficient
  - It is also visible to DBA etc, to know what the system is doing

Database Querying with SQL

"List studID of students who are enrolled in the unit named ***Database Systems I***"

Enter SQL, PL/SQL and SQL*Plus statements.

```
SELECT studID
  FROM Enrolled E, UnitOfStudy U
 WHERE E.uosCode = U.uosCode
     AND U.uosName= 'Database Systems I'
```

Execute | Load Script | Save Script | Cancel

STUDID
305422153
305678453
307088592
316424328
309187546
309145324

6 rows selected.

Many Ways to Write this Query…

"List studID of students who are enrolled in ***Database Systems I***"

```
SELECT studID
  FROM Enrolled E, UnitOfStudy U
 WHERE E.uosCode = U.uosCode
   AND uosName = 'Database Systems I'
```

```
SELECT sid AS studID
  FROM (SELECT u.uosCode AS u1,  e.uosCode AS u2,
               e.studId  AS sid, u.uosName AS title
          FROM Enrolled e, UnitOfStudy u) SubQ
 WHERE u1=u2 AND title = 'Database Systems I'
```

```
SELECT R.studID
  FROM (SELECT studID, uosName
          FROM Enrolled NATURAL JOIN UnitOfStudy)R
 WHERE R.uosName= 'Database Systems I'
```

```
SELECT studID
  FROM Enrolled
 WHERE uosCode IN (SELECT uosCode
                     FROM UnitOfStudy
                    WHERE uosName = 'Database Systems I')
```

How do we know what a DBMS is doing?

- All the SQL queries on the previous slide are *equivalent*
  - On the same data set, they produce the same result
  - Even when run on different database systems by various vendors, their results will be the same
- Why?
- Why do we know that this is the case?
- How can database system vendors even guarantee this?
- And why are some not equivalent?
  - Wrong:

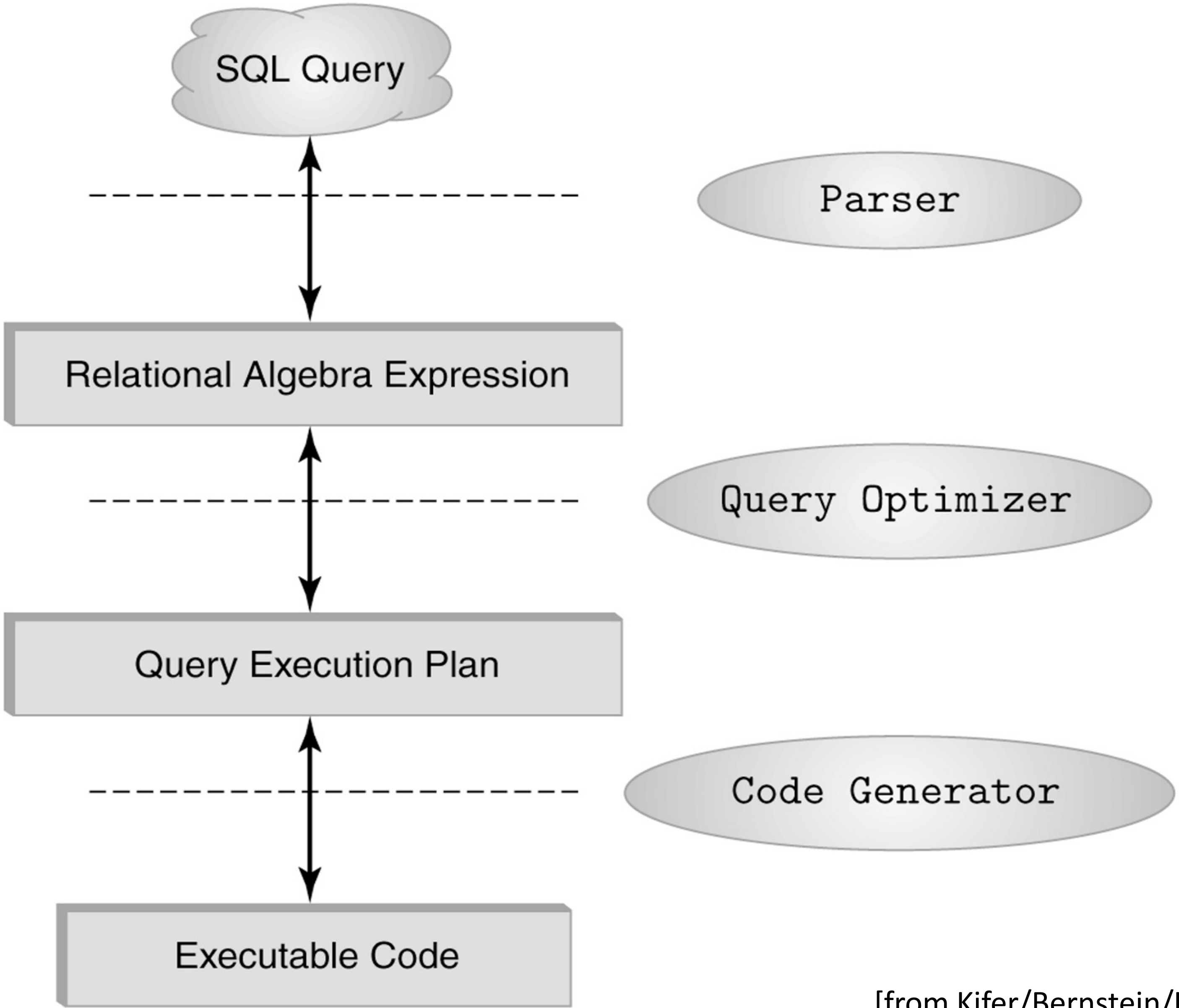
```
SELECT studID
  FROM Enrolled E, UnitOfStudy U
 WHERE uosName = 'Database Systems I'
```

The Big Idea

- Users request information from a database using a **query language**
- A query that extracts information can be seen as calculating a relation from combining one or more relations in the current state of the database
  - SQL expresses a query declaratively (“what to return” not “how to calculate”)
- **Relational algebra** (RA) can be used to express how to do a calculation of that kind (give a relation, based on existing relations), step-by-step
  - Each operator takes one or more relation instances, and produces a relation instance *(the operation part of the relational data model)*
- Why is it important?
  - Helps us understanding the precise meaning of declarative SQL queries.
  - Intermediate language used within DBMS (cf. chapter on db tuning)

The Role of Relational Algebra in a DBMS

Not examinable



[from Kifer/Bernstein/Lewis, Figure 5.1]

Agenda

- Motivation (not examinable)
- *Mathematical definition of relations*
- The operations
- Some rules
- Conversion skills

The relational approach to data

- **Informal Definition:** A ***relation*** is a named, two-dimensional table of data
  - Table consists of rows (record) and columns (attribute or field)

Attributes (also: columns, fields)

eid	fname	lname	gender	address
1234	Peter	Pan	M	Neverland
5658	Dan	Murphy	M	Alexandria
8877	Sarah	Sander	F	Glebe

Tuples (rows, records)

Maths Definition of a Relation

- **A Relation is a mathematical concept from discrete maths, based on the ideas of sets.**
  - ***Relation R***
    Given sets $D_1, D_2, \ldots, D_n$, a relation R is a subset of $D_1 \times D_2 \times \ldots \times D_n$
    Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where $a_i \in D_i$
- Example:

If

studentid = {12345678, 23456789, 345354345, 44455666, etc}
name = {Jones, Smith, Kerry, Lindsay, etc}
date = {1985-11-09, 1984-07-15, 1984-12-01, 1986-01-01, etc}

then

R = { (12345678, Jones, 1984-07-15),
(345354345, Lindsay, 1986-01-01),
(44455666, Kerry, 1985-11-09),
(23456789, Kerry, 1994-07-15) }

is a relation over *studentid* x *name* x *date*

Theory vs. Technology

- A relational DBMS supports data in a form close to, but not exactly, matching the mathematical relation
  - RDBMS allows null values for unknown information
  - RDBMS insists that datatype for any column is from a limited variety, all simple
  - RDBMS considers the rows as being arranged in an order
  - RDBMS allows duplicate rows

What is an Algebra?

Not examinable

- A language based on operators and a domain of values:
  - basic operators
  - atomic operands (constants and variables)
  - rules to form complex *expressions* from operators and operands, typically using nesting with parentheses [and precedence rules, to leave out parenthesis sometimes]
  - Equations that say when two expressions are equal to one another
- Example: Algebra of Arithmetic (high-school maths)
  - Operators for usual arithmetic operations: + - * /
  - Operands: variables (*x, y, z, …)* and numerical constants
  - Example arithmetic expression: $100 - ((x + y) / 2)$
  - Example equations: $x*(y+z) = x*y + x*z$, $x+0 = x$
- In databases, operators and operands are relations, which leads to the **Relational Algebra**
  - We refer to expression as a *query* and the value produced as *query result*

Relational Algebra as a model

- In a SQL DBMS, the data are in tables
- Mathematical relation is an idealised mathematical concept that is very similar to a database table
  - However, in a relation, NULL is not allowed for values, duplicate rows are not allowed, rows are not ordered
- So the calculation of a relational algebra expression, is an *idealised* way to express a calculation done in a SQL DBMS
- A real system has some extra complexity in how it deals with duplicates and order, NULL etc

Agenda

- Motivation (not examinable)
- Mathematical definition of relations
- *The operations*
- Some rules
- Conversion skills

Relational Algebra (RA)

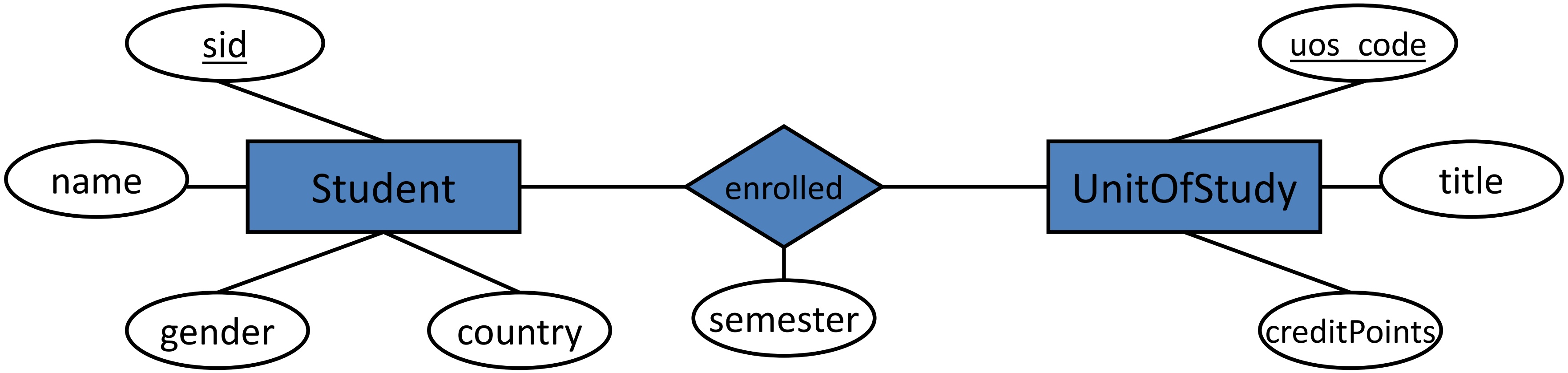
- 1. Set Operations [these operate on two relations]
  - **Union** ($\cup$) tuples in relation 1 or in relation 2.
  - **Intersection** ($\cap$) tuples in relation 1, as well as in relation 2.
  - **Difference** (**-**) tuples in relation 1, but not in relation 2.
- 2. Operations that produce a relation with only some parts of another relation
  - **Selection** (σ) picks a subset of rows from relation.
  - **Projection** (π) picks a subset of columns from relation.
- 3. Operations that combine tuples from two relations
  - **Cross-product** (**X**) allows us to fully combine two relations
  - **Join** ($\bowtie$) to combine *matching* tuples from two relations.
- 4. A schema-level 'rename' operation
  - **Rename** (ρ) allows us to rename one field or even whole relation

- RA is the algebra, with expressions built from relations using these operations
  - operators take one/more relations as inputs and gives a new relation as result

  => **RA queries by nesting of multiple operators**

Running Example

Conceptual data model of domain



Relational schema and example instance

Student			
sid	**name**	**gender**	**country**
1001	Ian	M	AUS
1002	Ha Tschi	F	ROK
1003	Grant	M	AUS
1004	Simon	M	GBR
1005	Jesse	F	CHN
1006	Franzisca	F	GER

Enrolled		
sid	**uos_code**	**semester**
1001	COMP5138	2005-S2
1002	COMP5702	2005-S2
1003	COMP5138	2005-S2
1006	COMP5318	2005-S2
1001	INFS6014	2004-S1
1003	ISYS3207	2005-S2

UnitOfStudy		
uos_code	**title**	**credit Points**
COMP5138	Relational DBMS	6
COMP5318	Data Mining	6
INFO6007	IT Project Management	6
SOFT1002	Algorithms	12
ISYS3207	IS Project	4
COMP5702	MIT Research Project	18

Set Operations

- These operations take two input relations *R* and *S*
  - Set Union $R \cup S$
    - Has a row which is in R or in S (or both)
    - Maths definition: $R \cup S = \{ t \mid t \in R \lor t \in S \}$
  - Set Intersection $R \cap S$
    - Has a row which is both in R and in S
    - Maths definition: $R \cap S = \{ t \mid t \in R \land t \in S \}$
  - Set Difference $R - S$
    - Has a row which is in R but not in S
    - Maths definition: $R - S = \{ t \mid t \in R \land t \notin S \}$
- Important restriction: the operation is only defined when *R* and *S* have the same structure
  - *R*, *S* have the *same arity* (same number of fields)
  - `Corresponding' fields must have the same names and domains

Projection

- keeps only attributes that are in *projection* list
  - Structure of result contains exactly the fields in the projection list, with the same names that they had in the input relation
- Duplicates are not in the result (so it is a relation)
- Examples:

$\Pi_{name,\ country}\ (Student)$

$\Pi_{title}\ (UnitOfStudy)$

Student	
name	**country**
Ian	AUS
Ha Tschi	ROK
Grant	AUS
Simon	GBR
Jesse	CHN
Franzisca	GER

UnitOfStudy
title
Relational DBMS
Data Mining
IT Project Management
Algorithms
IS Project
MIT Research Project

Selection

- Keeps those rows that satisfy a (Boolean) ***selection condition*** based on the attributes
  - Example:

$$\sigma_{country='AUS'}(Student)$$

Student			
<u>sid</u>	**name**	**gender**	**country**
1001	Ian	M	AUS
1003	Grant	M	AUS

Cross-Product

- Maths definition $R \times S = \{t\, s \mid t \in R \wedge s \in S\}$
  - each tuple of *R* is paired with each tuple of *S*.
  - Resulting schema has one field per field of *R* and *S*, with field names `inherited' if possible.
    - It might end in a conflict with two fields of the same name -> rename needed
- Sometimes also called *Cartesian product*
- Example:

R

A	*B*
α	1
β	2

x

S

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

=

result

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	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

Composing operations

- Result relation from an operation can be the input for another relational algebra operation
  - Indicate order of operations with parentheses if needed
  - Example:

$$\Pi_{name}(\,\sigma_{country='AUS'}(Student)\,)$$

Student
name
Ian
Grant

Joins

- **Conditional Join (also called theta-join)**:
  - Consider all pairs of tuples from the relations, but only keep those that satisfy a (Boolean) condition based on the attributes

$$R \bowtie_c S = \sigma_c (R \times S)$$

- Result schema same as the cross-product's result schema.
- ***Equi-Join***: Special case where the condition *c* contains only equalities.

$$Student \bowtie_{family\_name = last\_name} Lecturer$$

sid	given	family_name	gender	country	empid	first_name	last_name	room
1001	Cho	Chung	M	AUS	47112344	Vera	Chung	321
1004	Ciao	Poon	M	CHN	12345678	Simon	Poon	431
1004	Ciao	Poon	M	CHN	99004400	Josiah	Poon	482
1111	Alice	Poon	F	AUS	12345678	Simon	Poon	431
1111	Alice	Poon	F	AUS	99004400	Josiah	Poon	482
…		…	…	…	…		…	…

Natural Join

- **Natural Join** $R \bowtie S$
- Like Equijoin on all same-named fields, except that result structure has only one copy of each of those fields

Enrolled	
sid	uos_code
1001	COMP5138
1002	COMP5702
1003	COMP5138
1006	COMP5318
1001	INFO6007
1003	ISYS3207

$\bowtie$

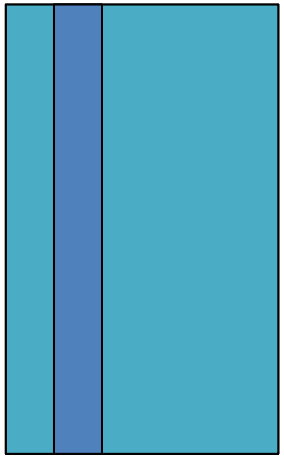
UnitOfStudy		
uos_code	title	points
COMP5138	Relational DBMS	6
COMP5318	Data Mining	6
INFO6007	IT Project Mgmt.	6
SOFT1002	Algorithms	12
ISYS3207	IS Project	4
COMP5702	MIT Research Project	18

=

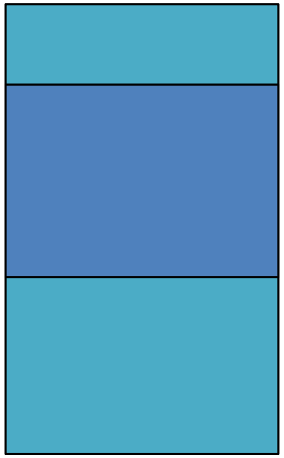
result			
sid	uos_code	title	points
1001	COMP5138	Relational DBMS	6
1002	COMP5702	MIT Research Project	18
1003	COMP5138	Relational DBMS	6
1006	COMP5318	Data Mining	6
1001	INFO6007	IT Project Mgmt.	6
1003	ISYS3207	IS Project	4

Visualisation of Relational Algebra

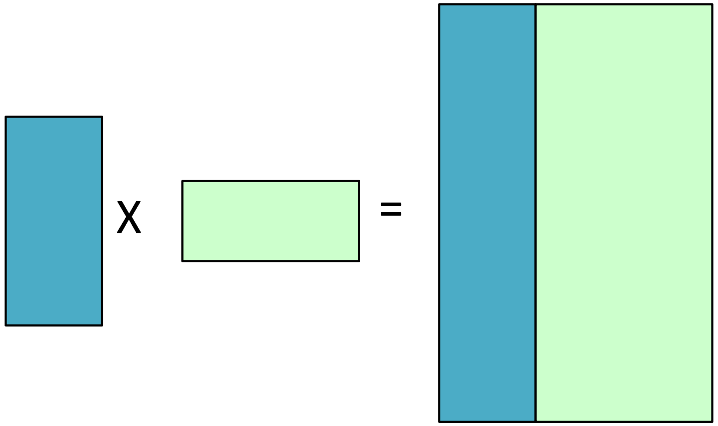
Projection (π)



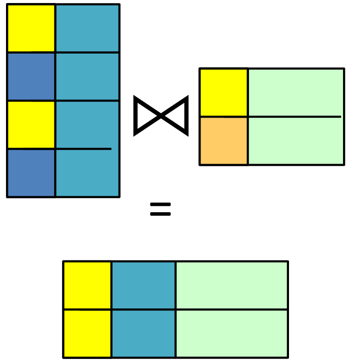
Selection (σ)



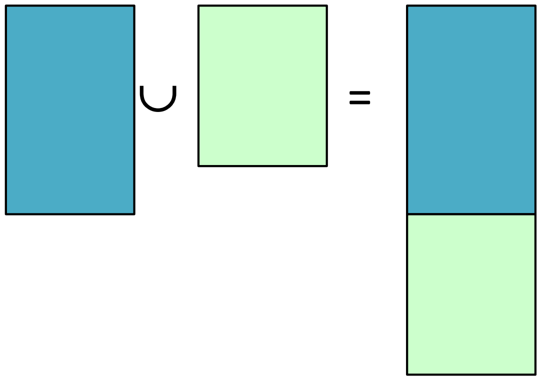
Cross-product (X)



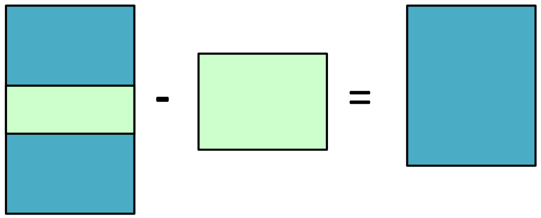
Join ($\bowtie$)



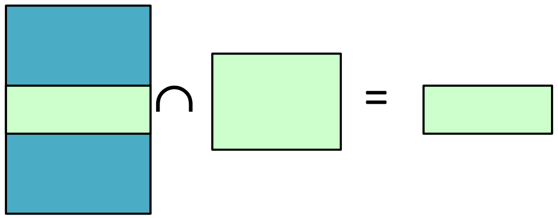
Set Union ($\cup$)



Set Difference (-)



Set Intersection ($\cap$)



Rename Operation

- Allows to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows to refer to a relation by more than one name.
- Notation 1: $\rho_X(E)$
  - returns the expression *E* under the name *X* (rename whole relation)
- Notation 1: $\rho_{X(a1 \rightarrow b1)}(E)$
  - returns the result of expression *E* under the name *X*, and with the attributes a_1…. renamed to b_1 … (rename individual attributes)
  - (assumes that the relational-algebra expression *E* has arity *n*)

- Example:

$$\rho_{\text{Classlist}(\,sid \rightarrow student\,)}\left(\sigma_{uos\_code='ISYS2120'}(Enrolled)\right)$$

Working without symbols

- The RA notation uses many special symbols, subscripts etc
- (i) $\rho_{\text{Classlist}(\textit{sid} \rightarrow \textit{student})} (\sigma_{\textit{uos\_code='ISYS2120'}} (\textit{Enrolled}))$
- (ii) $Student \bowtie_{family\_name=last\_name} Lecturer$
- If you are working just with a usual keyboard, you can instead do it with text as
  (i) RENAME_{Classlist(sid -> Student)} (SELECT_{uos_code='ISYS2120'}(Enrolled))
  (ii) Student JOIN_{family_name = last_name} Lecturer

Basic versus Derived Operators

Not examinable

- We can distinguish between basic and derived RA operators
- Only 6 basic operators are required to express everything else:
  - **Union** ($\cup$) tuples in relation 1 or in relation 2.
  - **Set Difference** (-) tuples in relation 1, but not in relation 2.
  - **Selection** (σ) selects a subset of rows from relation.
  - **Projection** (π) deletes unwanted columns from relation.
  - **Cross-product** (X) allows us to fully combine two relations.
  - **Rename** (ρ) allows us to rename one field to another name.
- Additional (derived) operations:
  - intersection, join, division:
    - Not essential, but (very!) useful.
  - Eg Join: $R \bowtie_c S = \sigma_c(R \times S)$

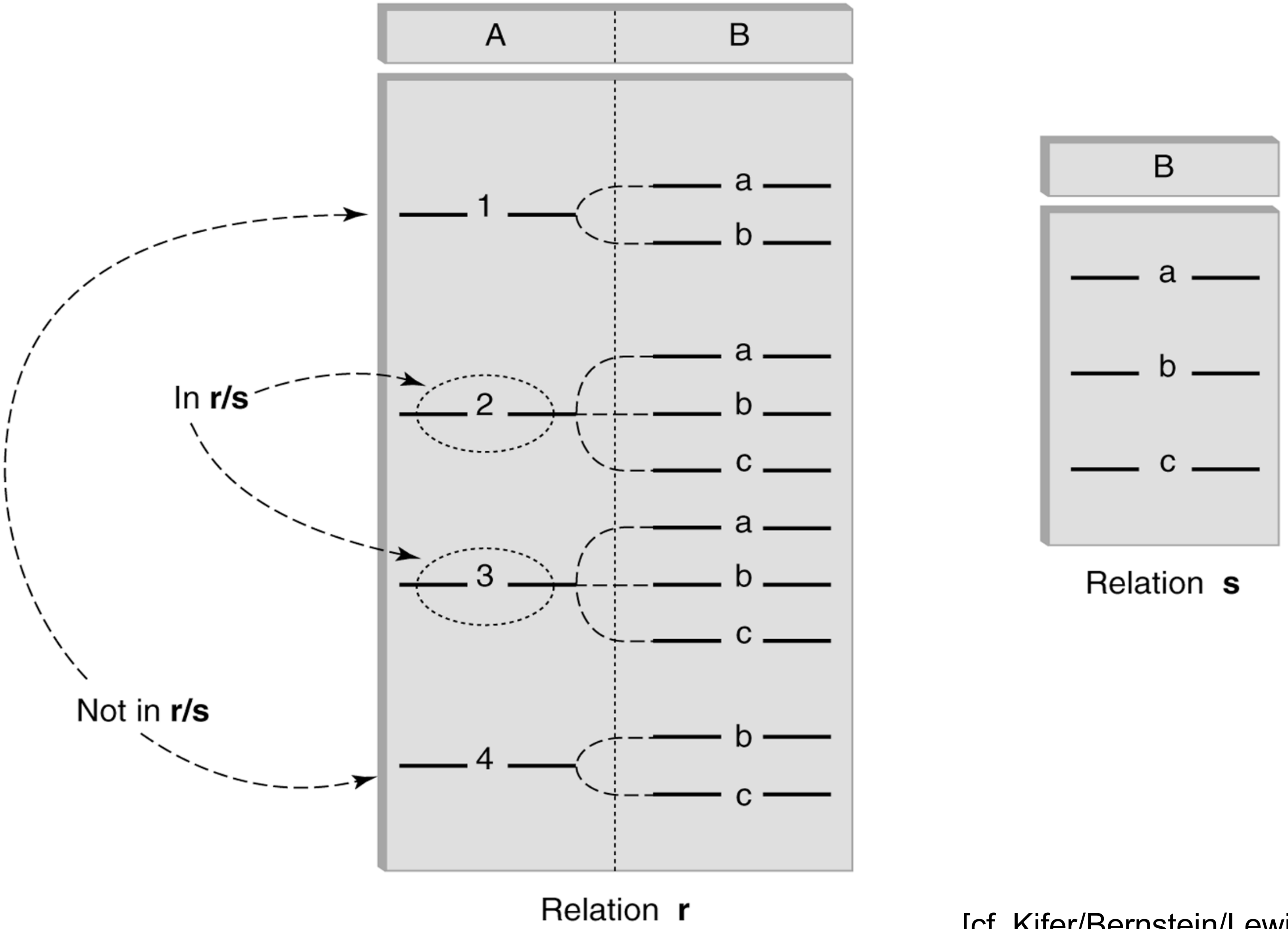
Relational Division

- *Purpose*: Find the items in a set that are related to *all* tuples in another set
- Relational Division operator (R / S)
  - We call the base set (S) the *divisor (*or *denominator)*
  - and the candidate set (R) the *dividend (*or *numerator)*
  - **Note:** This can be seen as the inverse of the cross product (x)

Relational Division

- $R\ (a_1, \ldots a_n, b_1, \ldots b_m)$
- $S\ (b_1 \ldots b_m)$
- *R/S*, with attributes $a_1, \ldots a_n$, is the set of all tuples *<a>* such that for <u>every</u> tuple *<b>* in *S,* there is an *<a,b>* tuple in *R*
  - Maths definition R/S := $\left\{ \langle a \rangle \mid \forall \langle b \rangle \in S : \exists \langle a,b \rangle \in R \right\}$

Visualisation of Division



[cf. Kifer/Bernstein/Lewis, Figure 5.6]

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

R

Example 1

pno
p2

$S1$

sno
s1
s2
s3
s4

$R/S1$

Example 2

pno
p2
p4

$S2$

sno
s1
s4

$R/S2$

Example 3

pno
p1
p2
p4

$S3$

sno
s1

$R/S3$

[cf. Ramakrishnan/Gehrke]

Relational Expressions

- A basic expression in the relational algebra consists of either one of the following:
  - Variable that refers to a relation of the same name in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; then the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $E_1 \bowtie E_2$
  - $\sigma_P (E_1)$, P is a Boolean predicate on attributes in E_1
  - $\pi_S (E_1)$, 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

Agenda

- Motivation (not examinable)
- Mathematical definition of relations
- The operations
- *Some rules*
- Conversion skills

Equivalence Rules

The following equivalence rules hold:

- Commutation rules

  1. $\pi_A(\sigma_p(R)) = \sigma_p(\pi_A(R))$ when p only mentions attributes in A

- Association rule

  1. $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$

- Idempotence rules

  1. $\pi_A(\pi_B(R)) = \pi_A(R)$ if $A \subseteq B$
  2. $\sigma_{p1}(\sigma_{p2}(R)) = \sigma_{p1 \wedge p2}(R)$

- Distribution rules

  1. $\pi_A(R \cup S) = \pi_A(R) \cup \pi_A(S)$
  2. $\sigma_P(R \cup S) = \sigma_P(R) \cup \sigma_P(S)$
  3. $\sigma_P(R \bowtie S) = \sigma_P(R) \bowtie S$ when *P* only mentions attributes of *R*
  4. $\pi_{A,B}(R \bowtie S) = \pi_A(R) \bowtie \pi_B(S)$ when *join-attr.* in $(A \cap B)$
  5. $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

- These are the basis for the automatic optimisation of relational queries

Agenda

- Motivation (not examinable)
- Mathematical definition of relations
- The operations
- Some rules
- *Conversion skills*

Give RA expression to extract information

- Eg Give the names of all Australian students

  $\pi_{\,name}\,(\,\sigma_{country='AUS'}\,(\text{Student})\,)$

Give purpose of RA expression

- Eg

$\pi_{name} (\sigma_{country='AUS'} (Student))$

This will give the names of all Australian students

Note: in exams etc, do not simply state the operations in order

[eg "select Australian students and then project only the names" will not get full marks; it is not a natural way a stakeholder would ask for the information!]

SQL -> Relational Algebra

- Given a Select-From-Where (SFW) query

SELECT A1, A2, …, An

FROM R1, R2, …, Rm

WHERE condition

- This can be calculated by the relational algebra expression:

$$\Pi_{A1,\ A2,\ \ldots,\ An} \left(\sigma_{condition} \left(R_1 \ \mathrm{x}\ R_2 \ \ \mathrm{x}\ \ldots\ \mathrm{x}\ \ R_m \right) \right)$$

Warning: The SELECT clause in SQL corresponds to a PROJECT operation in RA;
The SELECT operation in RA corresponds to the WHERE clause in SQL

You should now be able …

- to understand the **Foundations of SQL**
  - basic operations of Relational Algebra:
    Projection, Selection, Rename, Set Operations, Cross Product
  - the meaning of a relational join
  - differences between an equi-join, a theta-join and a natural join

- to give a relational algebra expression that will calculate the result for a (simple) English data-request on a given schema
- to give an English expression of the purpose of a given calculation in relational algebra

- to translate a relational algebra expression into a SQL query
  - and vice versa…

References

- Silberschatz/Korth/Sudarshan(7ed)
  – Chapter 2.6

Also

- Kifer/Bernstein/Lewis(complete version, 2ed)
  – Chapter 5.1
- Ramakrishnam/Gehrke(3ed)
  – Chapter 4.1, 4.2
- Garcia-Molina/Ullman/Widom(complete book, 2ed)
  – Chapter 2.4, 5.1, 5.2
  – *More extensive coverage than what we do in ISYS2120*

Summary

- Mathematical model of relations
- Relational algebra operations
- Equivalence rules in RA
- Conversion skills between English purpose, SQL queries, and RA expressions