Databases and the Relational Data Model

Gilles Falquet

Centre universitaire d'informatique

UniGE

Content

- Databases and Database management systems
- The Relational Data Model
- Querying Relational Databases

查询关系数据库

Database

- A database is a collection of related data
  - known facts that can be recorded and that have implicit meaning.

- A database has the following implicit properties
  - represents some aspect of the real world (universe of discourse)
  - is a logically coherent collection of data with some inherent meaning (not random)
  - designed, built, and populated with data for a specific purpose
    - an intended group of users
    - some preconceived applications for these users

Database management system

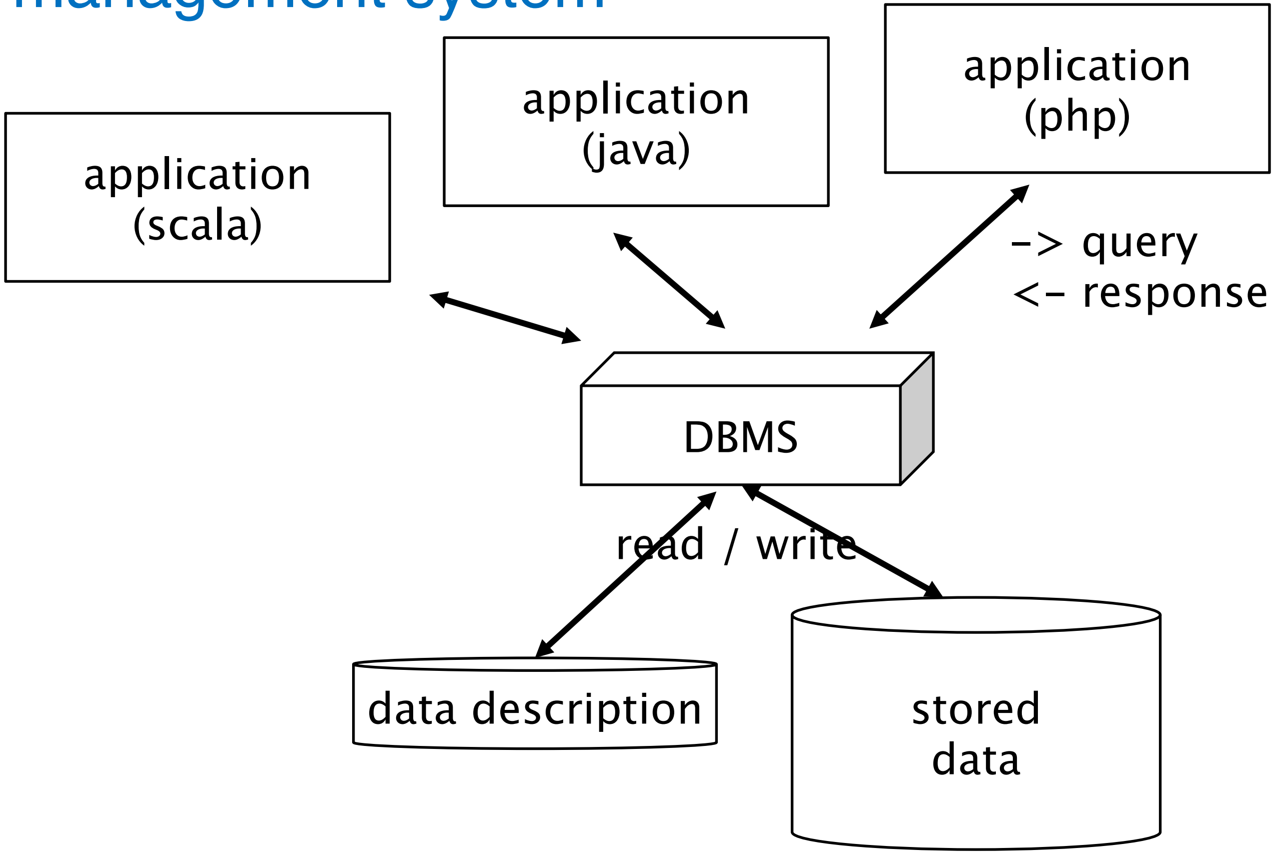
a piece of software that

- manages the physical storage of persistent date (on hard disks, solid state disks, RAM memory, etc.)
- executes application or user requests to
  - select and retrieve data
  - update data
- satisfies data management requirements

DBMS Requirements

- persistent storage of data
- provide a description of the stored data (schema)
- data access and update functions
- content-based data retrieval (selection criteria)
- multiple simultaneous access (concurrency)
- maintain data integrity
- reliability (crash recovery mechanisms)
- prevent unauthorized access to confidential data

Database management system



Can't we Use a File System?

- persistent storage of data OK
- provide a description of the stored data (schema) ✗
- data access and update functions ~ the fs doesn't know the internal structure of a file
- content-based data retrieval (selection criteria) ✗
- multiple simultaneous access (concurrency) ✗ (not within one file)
- maintain data integrity ✗(anything can be written in a file)
- reliability (crash recovery mechanisms) OK
- prevent unauthorized access to confidential data OK

Data Description and Files

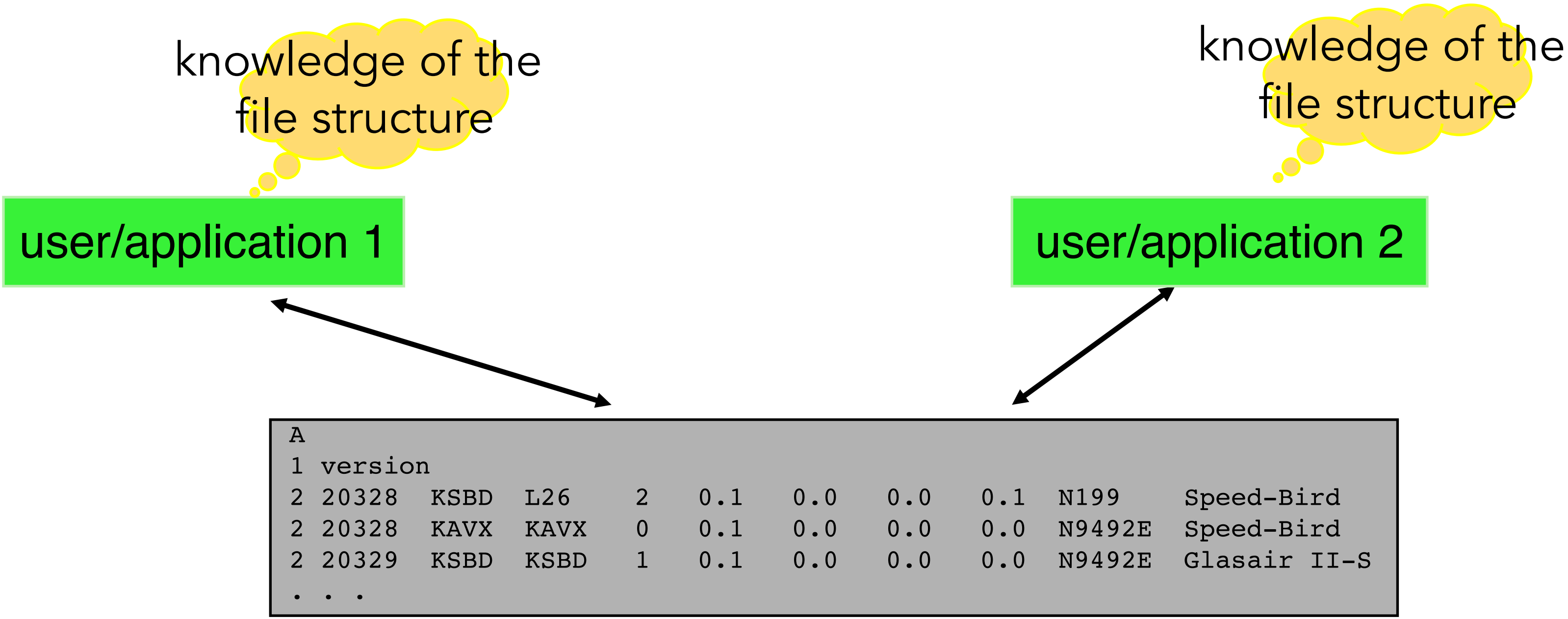
What is the meaning of these symbols?

```
A
4 SoCal:hangarWall // polygon # 5
28.7   9.8  -43.7
28.7   0  -43.7
28.7   0    4
28.7   9.8    4 4
SoCal:hangarIn // polygon # 6
-28.6   9.8  -43.7
```

```
A
1 version
2 20328  KSBD  L26    2   0.1   0.0   0.0   0.1  N199    Speed-Bird
2 20328  KAVX  KAVX   0   0.1   0.0   0.0   0.0  N9492E  Speed-Bird
2 20328  KSBD  KSBD   0   0.1   0.0   0.0   0.0  N9492E  Glasair II-S
2 20329  KSBD  KSBD   1   0.1   0.0   0.0   0.0  N9492E  Glasair II-S
. . .
```

Problem

- Each application/user must know the meaning and organization of data in the files
- Nothing guarantees that everyone has the same interpretation



knowledge of the file structure

user/application 1

knowledge of the file structure

user/application 2

```
A
1 version
2 20328  KSBD  L26   2   0.1   0.0   0.0   0.1  N199    Speed-Bird
2 20328  KAVX  KAVX  0   0.1   0.0   0.0   0.0  N9492E  Speed-Bird
2 20329  KSBD  KSBD  1   0.1   0.0   0.0   0.0  N9492E  Glasair II-S
. . .
```

Principle: Define a Common Schema

- A schema describes the data structure

$$\text{Database} = \text{Schema} + \text{Data}$$

- The schema is an integral part of the database
- Data may not exist without a schema

Data

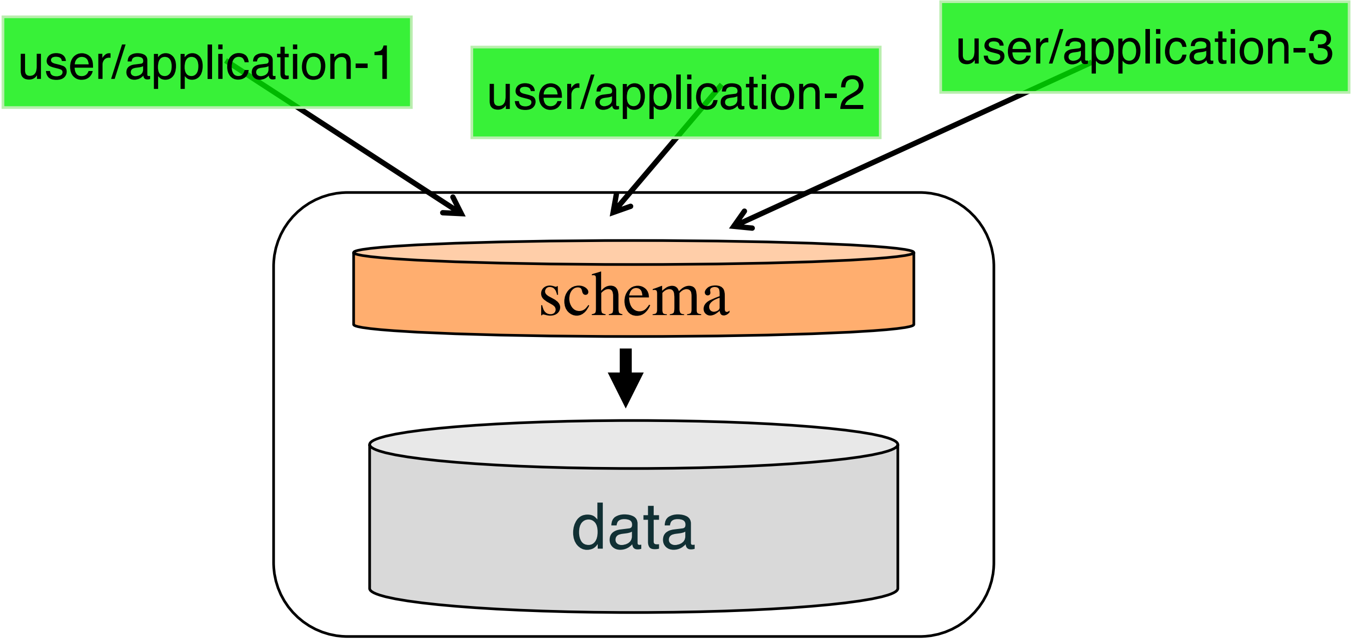
ZK567	GVA	ZRH	43	106
KL1122	AMS	CDG	50	77
KL232	AMS	PPP	560	230
LX441	GVA	NCE	35	101

Data + Schema

Flight	From	To	Duration	Passagers
ZK567	GVA	ZRH	43	106
KL1122	AMS	CDG	50	77
KL232	AMS	PPP	560	230
LX441	GVA	NCE	35	101

Single schema principle

- Everyone sees the data through the schema



Data Model

- A conceptual tool to structure data
- A database schema is expressed with a data model
- A data model is based on a set of concepts
  - hierarchical models (nodes, descendant links, ...)
  - tabular models (tables, rows, columns, ...)
  - graph models (nodes, links, ...)
  - key-value (keys, value sets, ...)

Relational Model of Data

E. F. Codd (1970)

- Based on simple mathematical constructs: sets, n-ary relations
- Has a simple intuitive interpretation: tables, rows, columns
- Well defined operations with a semantics based on relational algebra or relational calculus
- Currently the most frequently used model

Relational Database = Set of Tables

- table = set of rows and columns
- each column has a name and a type (character string, number, date, ...)

Wine

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
ais	2013	Average
rdeaux	2011	Good

Order

Order#	Quantity	Region	Year
1022	455	Bordeaux	2010
1322	112	Bordeaux	2010
0998	14	Valais	2014

Formal Model

Domain (or Type): a set of values considered as atomic

Relation schema: $R(A_1, A_2, ..., A_n)$

- a relation name R
- list of attributes, $(A_1, A_2, ..., A_n)$.

Attribute: role played by some domain D in the relation schema R .

- each A_k has a domain dom(A_k)

Relation (relation state)

for a relation schema $R(A_1{:}T_1, A_2{:}T_2, \ldots, A_n{:}T_n)$

a relation r on R is a set of n-tuples (the rows)

$$r = \{(t_1 = (a_1^1, \ldots, a_n^1), t_2 = (a_1^2, \ldots, a_n^2), \ldots, t_k = (a_1^k, \ldots, a_n^k))\}$$

where

$$a_1^i \in T_1,\ a_2^i \in T_2,\ \ldots,\ a_n^i \in T_n \text{ for } i = 1, \ldots, k.$$

the j th value a_j^i in tuple $t_i = (a_1^i, \ldots, a_n^i)$, corresponds to the attribute A_j ,

notation: $\boldsymbol{t_i.A_j}$

In SQL

- There is a set of standard types for the domains

- Definition of a relation schema (and an empty relation)

```
create table Wine(
   Region varchar,      -- character string
   Year integer,
   Quality varchar,
   AveragePrice real)
```

Interpretation of a Relation

- Each row represents a fact about an entity or a relationship between entities

- Exemple

a row

r	y	q

in the table Wine**(Region, Year, Quality)** represents the fact

"Wines in region r have quality q for year y"

III. Querying a Relational Database

- Extract desired information from a database

- Basic operations
  - selection, projection, join

- Standardized query language: SQL(**S**tructured **Q**uery **L**anguage)

- A **declarative** (non-procedural language)
  - specify the result you want to obtain (what, no how)

Selection

r a relation on the schema $(A_1, A_2, \ldots, A_n)$

$$\sigma_{Condition}\, r = \{ t \in r \mid Condition(\mathrm{t}) = \mathrm{true} \}$$

Retain the tuples of r that satisfy *Condition*

- the condition is a logical expression with the attribute names $A_1, A_2, \ldots, A_n$ as variables

Exemple

Wine

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
Valais	2013	Average
Bordeaux	2011	Good

$$\sigma_{\mathit{Quality} = \text{'Excellent'}} \text{Wine}$$

Region	Year	Quality
Bordeaux	2010	Excellent
Valais	2014	Excellent

SQL:

```
select *
from Wine
where Quality = 'Excellent'
```

Exemple

Wine

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
Valais	2013	Average
Bordeaux	2011	Good

$$\sigma_{\mathit{Quality} = \text{‘Excellent’ and Year} > 2010}\,\text{Wine}$$

Région	Year	Quality
Valais	2014	Excellent

```
select *
from Wine
where Quality = 'Excellent'
    and Year > 2010
```

Projection

r a relation on the schema $(A_1,A_2, \ldots, A_n)$

$B = \{A_{i1}, A_{i2}, \ldots, A_{ik}\} \subseteq \{A_1,A_2, \ldots, A_n\}$

$$\pi_B\, r = \{(t.A_{i1}, t.A_{i2}, \ldots, t.A_{ik} \mid t \in r \}$$

- Retain only the columns $A_{i1}, A_{i2}, \ldots, A_{ik}$ of r.

Exemple

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
Valais	2013	Average
Bordeaux	2011	Good

$\pi_{Region} Wine$

Region
Bordeaux
Valais
Bourgogne

In SQL

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
Valais	2013	Average
Bordeaux	2011	Good

```
select Region
from Wine
```

Region
Bordeaux
Valais
Valais
Bordeaux
Bourgogne

Duplicates are not removed !
⇒ a multiset (bag), not a set

Reason: performance

In SQL (2)

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	Average
Valais	2014	Excellent
Valais	2013	Average
Bordeaux	2011	Good

```
select distinct Region
from Wine
```

Region
Bordeaux
Valais
Bourgogne

Duplicates are removed

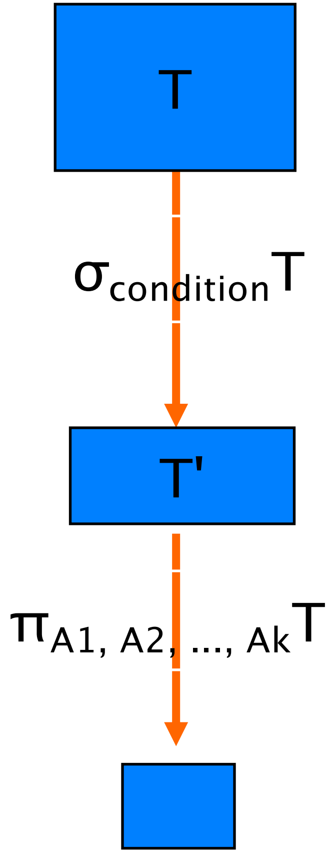
(less efficient)

Selection + Projection in SQL

select A1, A2, ..., Ak

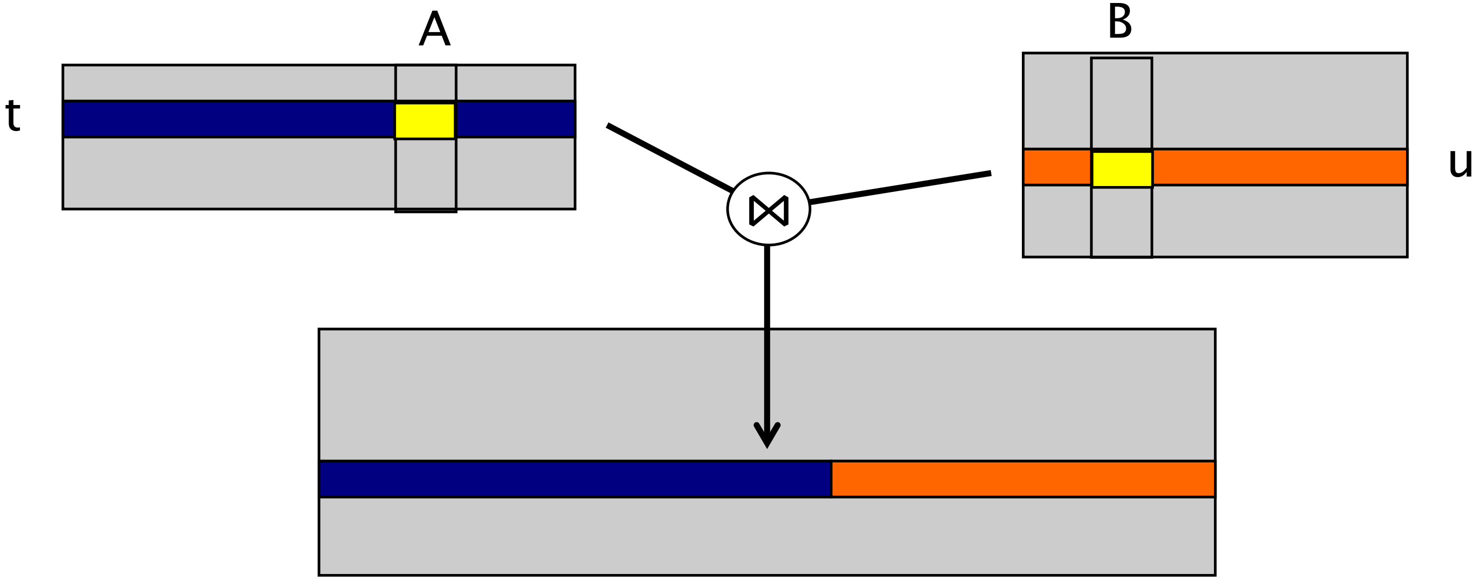
from T

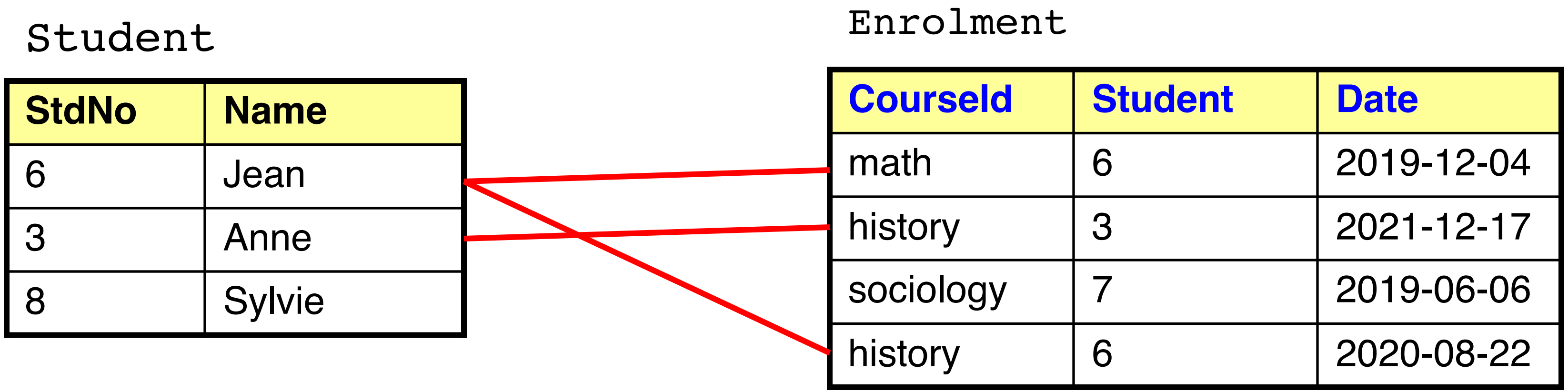
where Condition



Equi Join ⋈

$$r \bowtie_{A=B} s = \{t \text{ concatenated with } u \mid t \in r \text{ and } u \in s \text{ and } t.A = s.B\}$$





Student

StdNo	Name
6	Jean
3	Anne
8	Sylvie

Enrolment

CourseId	Student	Date
math	6	2019-12-04
history	3	2021-12-17
sociology	7	2019-06-06
history	6	2020-08-22

```
select * from Student, Enrolment
where Student.StdNo = Enrolment.Student
```

StdNo	Name	CourseId	Student	Date
6	Jean	math	6	2019-12-04
6	Jean	history	6	2020-08-22
3	Anne	history	3	2021-12-17

Self Join

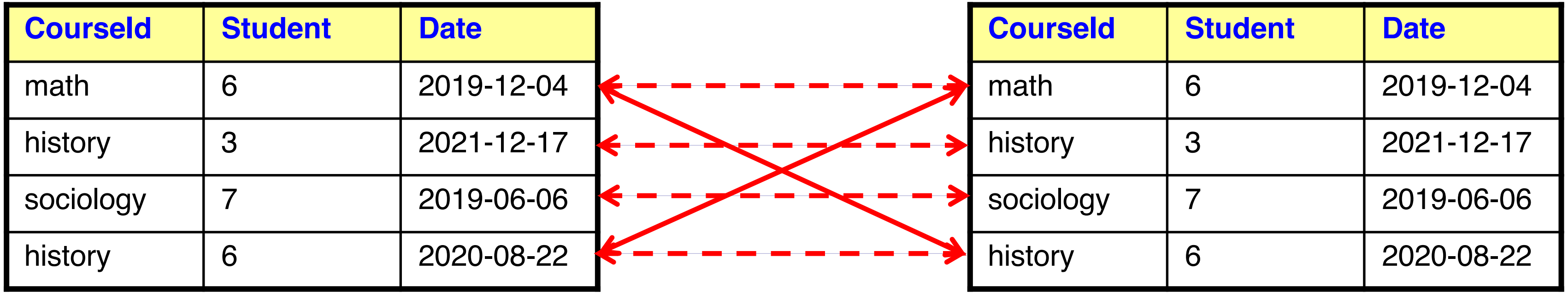
- A relation can be joined with itself
- Like joining two copies of the same relation

"Find the courses that share (at least) one student"

```
select E1.CourseId, E2.CourseId
from Enrolment E1, Enrolment E2
where E1.Student = E2.Student
    and E1.CourseId <> E2.CourseId
```

Enrolment E1

Enrolment E2

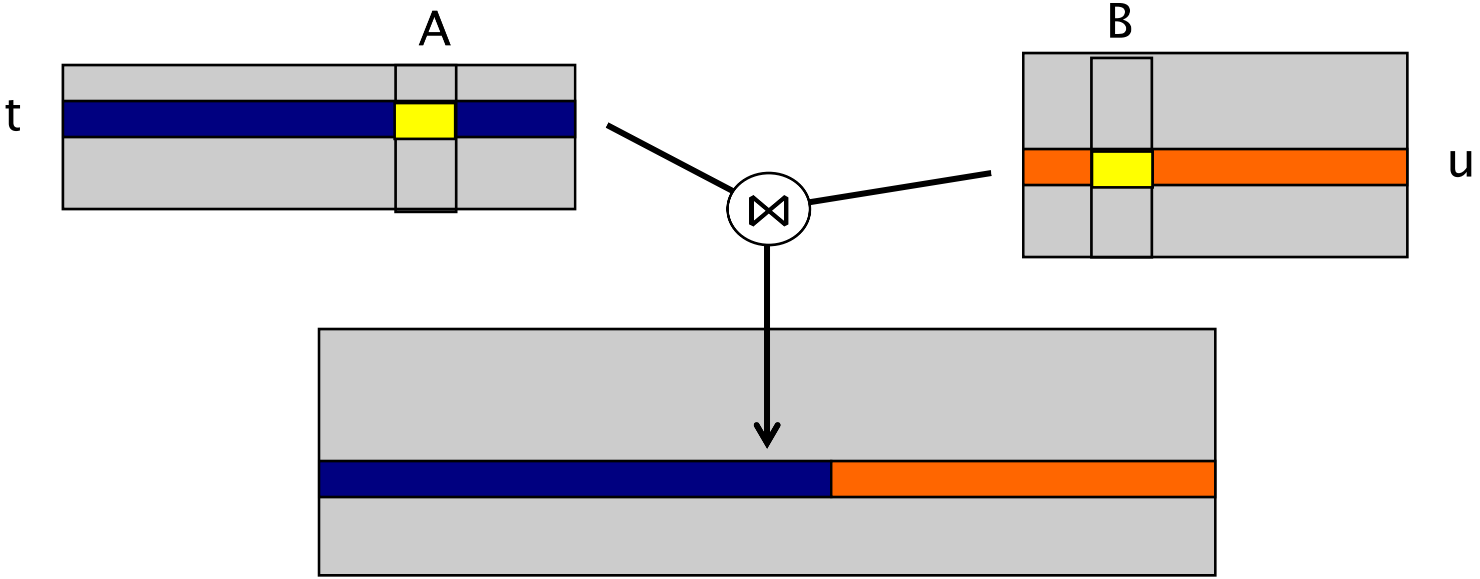


CourseId	Student	Date
math	6	2019-12-04
history	3	2021-12-17
sociology	7	2019-06-06
history	6	2020-08-22

CourseId	Student	Date
math	6	2019-12-04
history	3	2021-12-17
sociology	7	2019-06-06
history	6	2020-08-22

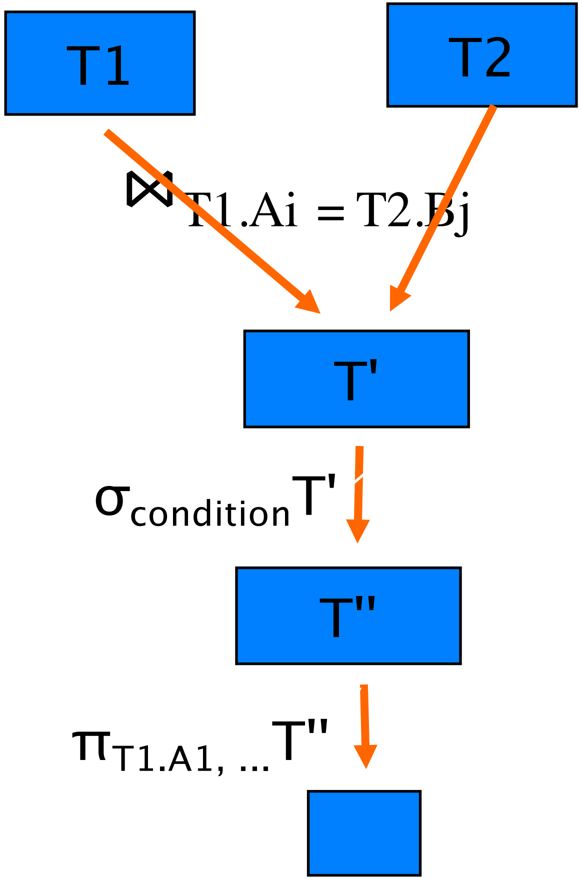
Theta Join ⋈

$r \bowtie_{A\,\theta\,B} s = \{t \text{ concatenated with } u \mid t \in r \text{ and } u \in s \text{ and } t.A\ \theta\ s.B\ \}$



Join +Selection + Projection in SQL

```
select T1.A1,   ...
from T1, T2, ...
where Condition and T1.Ai = T2.Bj
```



Exercises

Schema

- city(Name, Country, Latitude, Longitude, Population, Province)
- country(Code, Area, Capital, Name, Population, Province)
- located(City, Country, Lake, Province, River, Sea)

Write SQL queries that retrieve the following information

- cities with more than 10 000 000 inhabitants
- countries with a city on the Indian Ocean
- cities with a population > 30% of the country population
- name of the countries that have a city located on the Baltic Sea
- cities that are more populated than the capital of their country (*)