INFORMATION MODELING

- A **symbolic model** is a subjective formal representation of ideas and knowledge about some aspects of the real world (*domain of discourse*), designed to serve an explicit purpose.

- A **data model** is a set of abstraction mechanisms to describe abstract knowledge

What is the problem?

  - Conceptual data model: to analyse a problem, given user requirements
    - Operational databases:
      - E.g., E-R or Entity-Relationship, **ODM or Object Data Model**
    - Data warehouses:
      - **Dimensional Fact Model**

How to solve it?

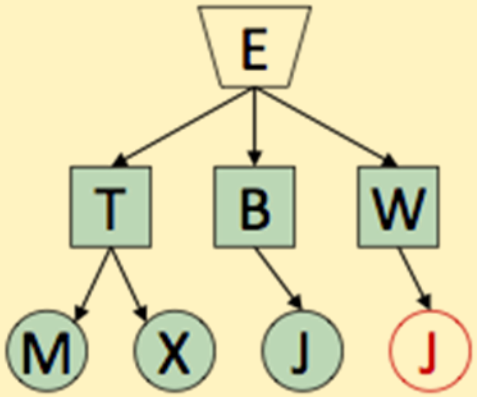
  - Logical model: to design a solution independently of actual DBMS
    - E.g., Relational Data Model ⇦ **Today: RECALLS ON RELATIONAL MODEL**

How to implement a solution?

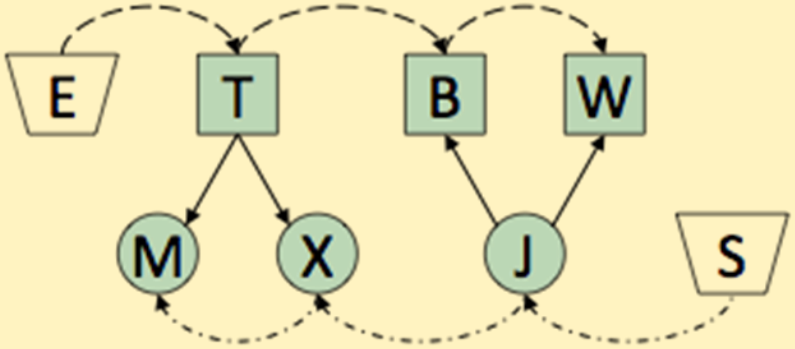
  - Physical model: to realize a project on a specific DBMS

DATABASE LOGICAL MODELS

Hierarchical (tree)



Network (graph)



Relational (table)

E

B	...
T	...
W	...

S

J	...
M	...
X	...

R

M	T
X	T
J	B
J	W

The relational data model is **based on typed finite sets of ...**

Simplicity is Beautiful

THE RELATIONAL DATA MODEL

Definition.

- integers, floats, booleans, and strings are **primitive** types;
- if **T1**, …, **Tn** are primitive types, and **A1**, …, **An** are distinct **attribute names**, then **(A1: T1, …, An: Tn)** is a **tuple type** of degree n. The **attributes order** is **unimportant**;
- A tuple **(A1 := V1, …, An := Vn)** of type T = (A1: T1, …, An: Tn) is a **set of pairs (Ai, Vi)** with Vi of primitive type Ti.

(Age :int, Name :string) (Age := 23, Name := 'mary')

= { (Age, 23), (Name, 'mary') }

= { (Name, 'mary'), (Age, 23) } =

(Name :string, Age :int) (Name := 'mary' , Age := 23)

Definition.

- integers, floats, booleans, and strings are **primitive** types;
- if **T1**, …, **Tn** are primitive types, and **A1**, …, **An** are distinct **attribute names**, then **(A1: T1, …, An: Tn)** is a **tuple type** of degree n. The **attributes order** is **unimportant**;
- A tuple **(A1 := V1, …, An := Vn)** of type T = (A1: T1, …, An: Tn) is a **set of pairs (Ai, Vi)** with Vi of primitive type Ti.

When two tuple types are equal?

Two **tuple types are equal** iff they have the same degree, same attributes and same type of the attributes with the same name.

When two tuples are equal?

Two **tuples** of the **same type** are **equal** iff they have the same set of pairs (Attribute, Value)

THE RELATIONAL DATA MODEL

Definition. A relational database is described by a set of **relation schemes** R:{T} defined as follows:

- **T = (A1: T1, …, An: Tn)** is a **tuple type;**
- **{T}** is called a **relation type**;
- a **relation scheme** R:{T} is a variable R with a relation type T;

  For brevity, instead of R:{T} we will write R(T).

  In the following we will use as schema notation R(A1, …, An) when the type of the attributes is not important.

Student(Age :int, Name :string) **Student(Name :string, Age :int)**

When two relation types are equal?

Two **relation types are equal** iff they have the same **tuple types**.

Definition. Consider a **relation scheme** R:{T}:

- A **schema instance** (or **relation**) is a finite set of tuples with type T.

Student(Age :int, Name :string)

{(Age := 23, Name := 'mary'),
(Age := 24, Name := 'john')}

{(Name := 'john, Age := 24),
(Age := 23, Name := 'mary')}

Important properties of relations: finite sets of tuples without duplicates,
The **order** of a relation elements and of the attributes **is irrelevant**.

When two relations are equal?

Two **relations** of the **same type** are **equal** iff have the equal tuples.

**In some books on relational algebra,
and in relational DBMS (SQL)
the order of the attributes is IMPORTANT**

Two relations R and S have the same type if
- they have the same set of attributes,
- the order of the attributes is the same for both relations.

RELATION SCHEMA INSTANCE

Relational schema: R(A :int, B :string)

An instance of R is a finite set of tuples with type **(A :int, B :string)**

R = { (A := 10, B := "Mario"),
(A := 18, B := "Giovanna"),
(A := 77, B := "Luca"),
(A := 18, B := "Francesco"),
(A := 14, B := "Carla") }

R

A	B
10	Mario
18	Giovanna
77	Luca
18	Francesco
14	Carla

KEYS AND RELATIONSHIPS

Key: A key for a relation is a minimal subset of attributes whose values identify a tuple;

Primary key: one of the possible keys, usually the shortest.

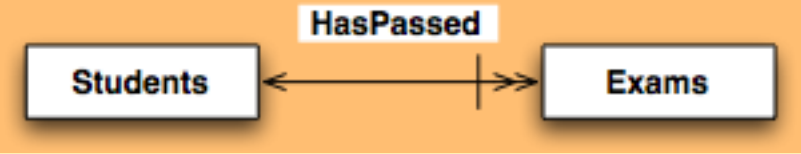
Which are the keys?

Students(Name: string, StudCode: string, City: string, BirthYear:int)

Exams(Subject: string, Candidate: string, Date: string, Grade: int)

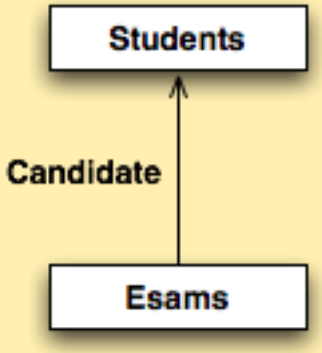
Relationships between tuples are represented with values of a set of attributes (**foreign key**) which take as values those of the primary key of another relation.

EXAMPLE



Schema:

Students(Name: string, StudCode: string, City: string, BirthYear:int)
Exams(Subject: string, Candidate*: string, Date: string, Grade: int)



Students

Name	StudCode	City	BirthYear
Isaia	171523	PI	1982
Rossi	167459	LU	1980
Bianchi	179856	LI	1981
Bonini	175649	PI	1982

Exams

Subject	Candidate	Grade	Date
BD	171523	20	12/01/05
ALG	167459	30	15/09/05
MP	171523	30	25/10/05
IS	167459	20	10/10/05

EXAMPLE ALGEBRA: INTEGER ARITHMETIC

DOMAIN: integers

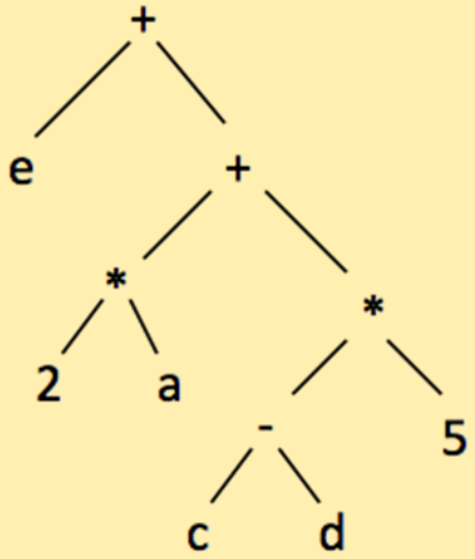
OPERATORS: -, + , *, ...

EXPRESSIONS: e + ((2*a) + ((c + (-d)) * 5))

LAWS: Commutative, Associative, Distributive

- a * b = b * a
- a * (b * c) = (a * b) * c
- a * (b + c) = a * b + a * c

Tree form



RELATIONAL ALGEBRA

Relational algebra is a set of **operators on relations whose results are themselves relations**. An **expression** is called a **query**.

Relational Algebra describes **HOW** to get results
with "logical query plan" of relational operators.

A naive way to evaluate an expression would be to compute the results of the relational operators directly as specified... but there are equivalence rules for **query rewriting**.

The language is not used by DBMS to query but to represent queries internally.

The **standard language** used by DBMS to query a relational database is **SQL**.

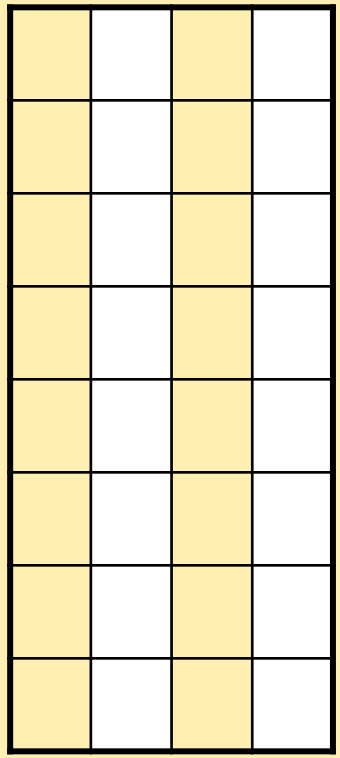
SQL is a declarative rather than a procedural language.

It describes **WHAT** we are looking for, but not how to get it.

RELATIONAL ALGEBRA: UNARY OPERATORS

Note: R through W stand for any expressions, not necessarily for stored relations.

Semantics



Project (π): $\pi_{A_1,A_2,\ldots,A_n}(R)$

Which is the result type? $\{(A_1:T_1, A_2:T_2, \ldots, A_n:T_n)\}$

If R has M elements, how many elements has the result?

Rename (ρ): ρ $\rho_{A \to B}(R)$

Generalized Project

$$\pi_{Exp_1\ \mathbf{AS}\ A_1, Exp_2\ \mathbf{AS}\ A_2, \ldots, Exp_n\ \mathbf{AS}\ A_n}(R)$$

Which is the result type?

EXAMPLE

Students

Name	StudCode	City	BirthYear
Isaia	171523	PI	1982
Rossi	167459	LU	1980
Bianchi	179856	LI	1981
Bonini	175649	PI	1982

Find name, student code and city of students

$\pi_{\text{Name, StudCode, City}}(\text{Students})$

Name	StudCode	City
Isaia	171523	PI
Rossi	167459	LU
Bianchi	179856	LI
Bonini	175649	PI

Find city of students

$\pi_{\text{City}}(\text{Students})$?

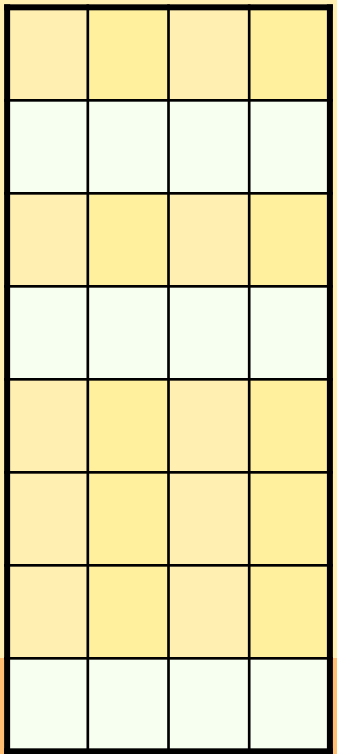
Which are the query result types?

Selection (σ): selects the tuples from a relation that satisfy a condition.

$$\sigma_{Condition}(R)$$

Condition is a propositional logic expression ($\wedge \vee \neg$) over comparison predicates (<=, <, =, >, >=, $\neq$) and arithmetic expressions (+,-,*,/)

Semantics



Which is the result type?

If R has M elements, how many elements has the result?

EXAMPLE

Students

Name	StudCode	City	BirthYear
Isaia	171523	PI	1982
Rossi	167459	LU	1980
Bianchi	179856	LI	1981
Bonini	175649	PI	1982

Find all data of Pisa students:

$\sigma_{City = 'PI'}$ (Students)

Name	StudCode	City	BirthYear
Isaia	171523	PI	1980
Bonini	175649	PI	1980

Find name, student code and birth year of Pisa students:

$\pi_{Name,StudCode,BirthYear}$ ($\sigma_{city = 'PI'}$ (Students))

Name	StudCode	BirthYear
Isaia	171523	1980
Bonini	175649	1980

Which are the query result types?

EXERCISE

Students

Name	StudCode	City	BirthYear
Isaia	171523	PI	1982
Rossi	167459	LU	1980
Bianchi	179856	LI	1981
Bonini	175649	PI	1982

Find the name of Pisa students:

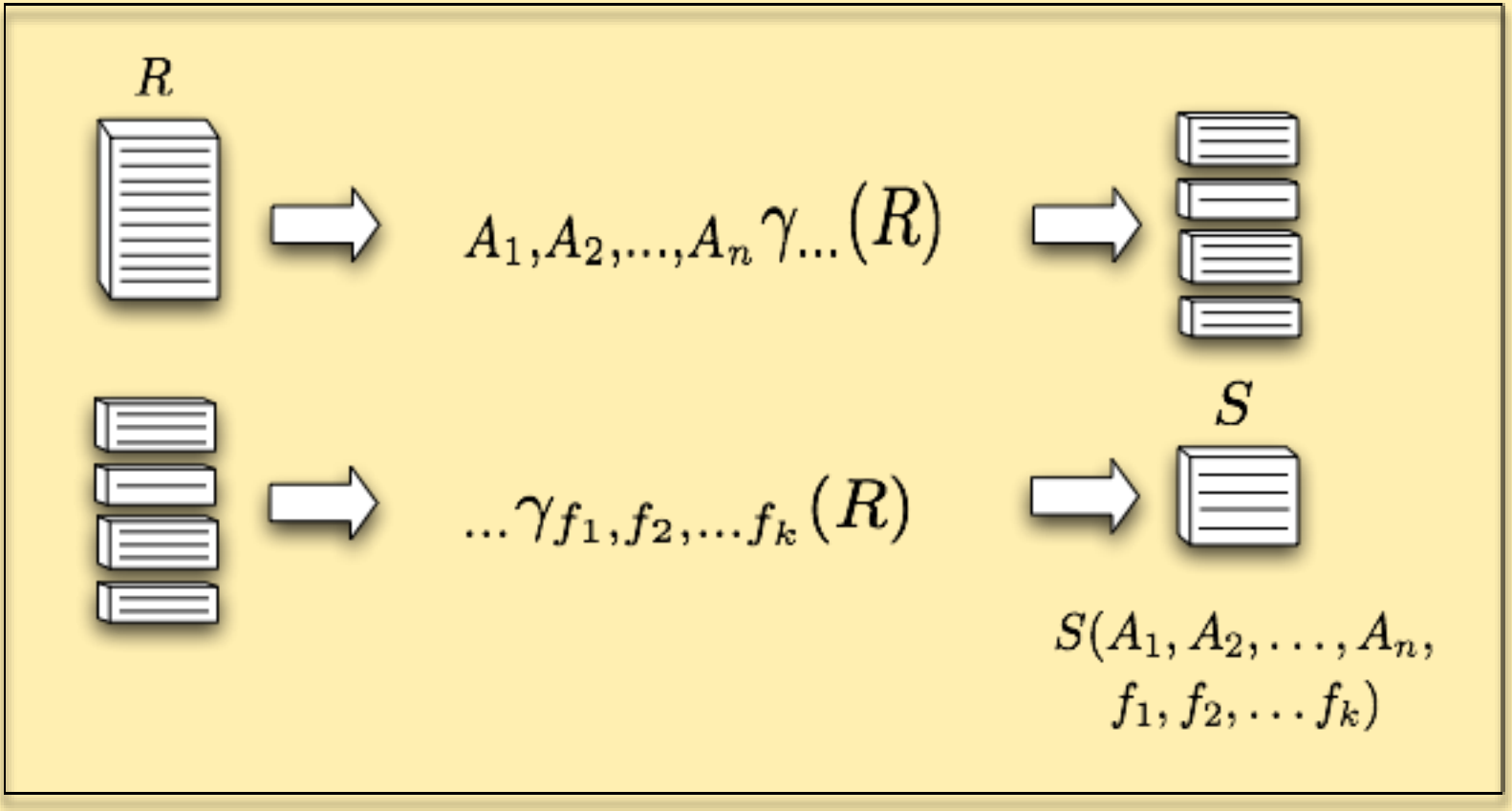
$\pi_{Name}(\sigma_{City = 'PI'}(Students))$

is equivalent to: $\sigma_{City = 'PI'}(\pi_{Name}(Students))$?

is equivalent to: $\sigma_{City = 'PI'}(\pi_{Name, City}(Students))$?

is equivalent to: $\pi_{Name}(\sigma_{City = 'PI'}(\pi_{Name, City}(Students)))$?

$$S = {}_{A_1,A_2,\ldots,A_n}\gamma_{f_1,f_2,\ldots f_k}(R)$$



EXAMPLE

Find for each student, the number of exams, min, max and avg grade.

$$\text{Candidate}\,\gamma\,\text{COUNT(*), MIN(Grade), MAX(Grade), AVG(Grade)}\,(Exams)$$

Subject	Candidate	Grade	Date
BD	171523	20	12/01/18
ALG	167459	30	15/09/18
MP	171523	30	25/10/18
IS	167459	28	10/10/18

Subject	Candidate	Grade	Date
ALG	167459	30	15/09/18
IS	167459	28	10/10/18
BD	171523	20	12/01/18
MP	171523	30	25/10/18

Candidate	COUNT(*)	MIN(Grade)	MAX(Grade)	AVG(Grade)
167459	2	28	30	29.0
171523	2	20	30	25.0

Type of result is:

{(Candidate:int, COUNT(*):int, MIN(Grade):int, MAX(Grade):int, AVG(Grade):float)}

Grouping (γ):

$$A_1, A_2, \ldots, A_n \gamma f_1, f_2, \ldots f_k (R)$$

where A_i are attributes of R and the f_i are aggregation function (**min**, **max**, **count**, **sum**, **avg**, ...)

Which is the result type? $\{(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n, \; f_1 : T_{f1}, f_2 : T_{f2}, \ldots, f_k : T_{fk})\}$

If R has **M** elements, how many elements has the result?

Generalized Grouping

$$A_1, A_2, \ldots, A_n \gamma f_1 \; \mathbf{AS} \; F_1, f_2 \; \mathbf{AS} \; F_2, \ldots, f_k \; \mathbf{AS} \; F_k (R)$$

Which is the result type?

Union ($\cup$): the arguments R and S **must have the same type** and the result is the union of the two set of tuples.

$$R \cup S$$

Set-difference (-): the arguments R and S **must have the same type** and the result is the tuples in R but not in S .

$$R - S$$

Which is the query result type?

If R and S have **Mr** and **Ms** elements, how many elements has the result?

RELATIONAL ALGEBRA: BINARY OPERATORS (cont.)

Product (×): of two relations R ($A_1 : T_1$, ..., $A_n : T_n$) and S($B_1 : T_1$, ..., $B_m : T_m$) with different attributes

$$R \times S$$

{($A : T_1$, $B : T_2$)}

A	B
a1	b1
a2	b2

×

{($C : T_3$, $D : T_4$)}

C	D
c1	d1
c2	d2
c3	d3

=

{($A : T_1$, $B : T_2$, $C : T_3$, $D : T_4$)}

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a1	b1	c3	d3
a2	b2	c1	d1
a2	b2	c2	d2
a2	b2	c3	d3

Which is the result type?

If R and S have **Mr** and **Ms** elements, how many elements has the result?

Students(Name, StudCode, City, BirthYear)
Exams(Subject, Candidate*, Date, Grade)

Find the name of students which have passed BD with 30

π Name (σ Subject = 'BD' σ Grade = 30 (σ StudCode = Candidate (Students × Exams)))

Is it meaningful?

Simpler with **join**! $\sigma_{R.A = S.D}(R \times S) = R \underset{R.A = S.D}{\bowtie} S$

$\pi_{Name}(\sigma_{Subject = 'BD' \wedge Grade = 30}(Students \overset{\bowtie}{_{StudCode=Candidate}} Exams))$

Intersect

$$R \cap S$$

Natural Join

$$R \bowtie S$$

EQUIVALENCE RULES FOR SIMPLIFYING EXPRESSIONS

Examples for the relations R(A, B, C, D), S(E, F), and T(G, H) :

$$\pi_A(\pi_{A,B}(R)) \equiv \pi_A(R)$$
$$\sigma_{C_1}(\sigma_{C_2}(R)) \equiv \sigma_{C_1 \wedge C_2}(R)$$
$$\sigma_{C_R \wedge C_S}(R \bowtie S) \equiv \sigma_{C_R}(R) \bowtie \sigma_{C_S}(S)$$
$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$
$$(R \bowtie S) \equiv (S \bowtie R)$$
$$\sigma_{C_X}({}_X\gamma_F(R)) \equiv {}_X\gamma_F(\sigma_{C_X}(R))$$