

# Query languages 2 (NDBI006) part 0

(revision of basic DB notions)

**J. Pokorný**  
**MFF UK**

lecturing  $\neq$  talking !!

# *Some rules for presentation*

*Credit:* based on a preparation of a paper for the course colloquium.

*Slide presentation:* in PowerPoint and presented on the basis of materials given by the teacher.

*Exam:* examples in written form (1,5 hours)

*Style:* Introduction - title, link to the source article, what will it be?

*Proofs:* only for important theorems

*Relationship to lectures:* do not repeat things from lectures

*Do not use:*

- fonts smaller than 18 p.
- "wild" templates - just a simple structure + color
- a lot of text on one slide

*Do not change slides very quickly:* rather explain in detail what it is about

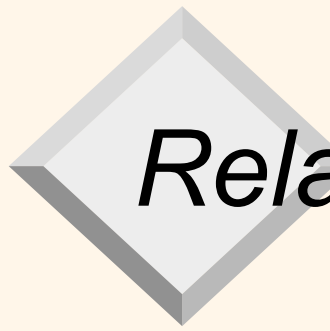
*Conclusion:* summarize the talk

Then: send your presentation to [pokorny@ksi.mff.cuni.cz](mailto:pokorny@ksi.mff.cuni.cz)



# *Basic notions*

- Relational data model (RDM)
- Relational algebra (RA)
- Domain Relational Calculus (DRC)



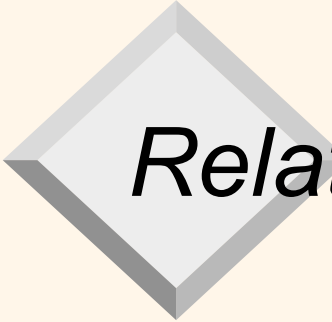
# *Relational data model*

Shows	C_name	F_name	Date
	Flora	Top gun	3.2.2018
	Flora	Black Panther	5.2.2018
	Atlas	Yellowstone	12.2.2018
	Atlas	Top gun	15.2.2018
	Atlas	Black Panther	20.2.2018

Relational schemas: Shows(C\_name, F\_name, Date)

Cinema(C\_name, Address, Head\_of\_c)

Movie(F\_name, Actor, Director)



# *Relational algebra – query language*

Assumptions: DB schema  $\mathbf{R}$ ;  $R(A), S(B) \in \mathbf{R}$

- **projection** of  $R$  on the set of attributes  $C$ , where  $C \subseteq A$

Notation:  $R[C]$

- **selection** of  $R$  by a selection condition  $\varphi$

Notation:  $R(\varphi)$

- **join** of  $R$  and  $S$

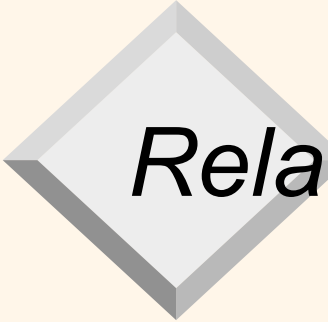
Notation:  $R * S$

Ex.:  $(Shows(Cinema\_n = Atlas)[F\_name, Time] * Movie)[Actor]$

Other: **union**  $\cup$ , **intersection**  $\cap$ , **difference**  $-$ ,

**Cartesian product**  $\times$

What is enough:  $\times, \cup, -, \text{projection, selection}$ . Other operations are derivable from the basic ones.



# *Relational algebra – query language*

Other (derived) operations:

- join of relations (natural,  $\theta$ -join, semijoin)
- division of relations
- composition of relations

! outer join is out of basic RMD

it requires empty values

Remark: Properties of relational operations allow to do algebraic query optimization.



# *DRC* - domain relational calculus

DRC is a subset of 1st predicate order calculus

- terms: variables, constants
- predicate symbols:  $\mathbf{R}$ , comparisons ( $=, \neq, <, >, \geq, \leq$ )
- logical connectives ( $\neg, \wedge, \vee, \Rightarrow$ )
- quantifiers ( $\exists, \forall$ )

Other notions: free and bound variables

TRUE-assignment of free variables, interpretation of predicate symbols, evaluation of formulas

Query in DRC is an expression  $\{x_1, \dots, x_k \mid A(x_1, \dots, x_k)\}$



## *DRC - domain relational calculus*

*Ex.:*

$\{x,y \mid \text{Cinema}(x, \text{'Národní třída'}, y)\}$

$\{\text{actor}, \text{dir} \mid \text{Movie}(\text{'Top gun'}, \text{actor}, \text{dir})\}$

$\{\text{actor} \mid \exists \text{dir} \text{Movie}(\text{'Top gun'}, \text{actor}, \text{dir})\}$

syntactical simplification:

– introducing attribute names

– removing unnecessary  $\exists$

$\{a, \text{fn} \mid \exists c (\text{Shows}(\text{Cinema\_n}:c, \text{F\_name}: \text{fn})$   
 $\quad \wedge \text{Cinema}(\text{Cinema\_n}:c, \text{Address}:a))\}$





# *DRC - domain relational calculus*

A more complex query:

Q: Find films, they give in all cinemas, where they give something.

$$\{f \mid \forall c(\text{Shows}(\text{Cinema\_n}:c) \Rightarrow \text{Shows}(\text{Cinema\_n}:c, \text{F\_name}:f))\}$$

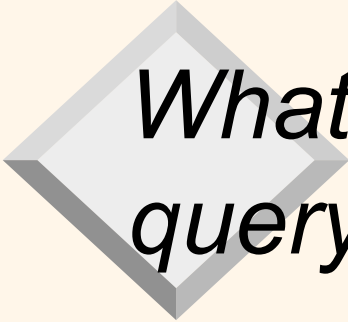
Problems:

- how to quantify, when the domain is infinite
- how to solve some queries with negation and disjunction

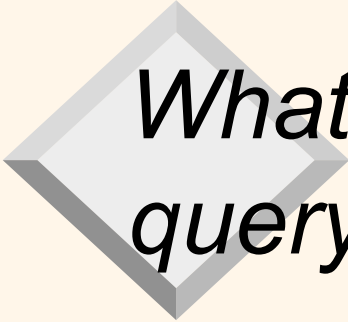
Ex.:  $\{x \mid \neg R(x)\}$

$$\{x,y \mid R('a',x) \vee S('b',y)\}$$

Solution: limited interpretation, save expressions



*What is a database query, what is a query language?*

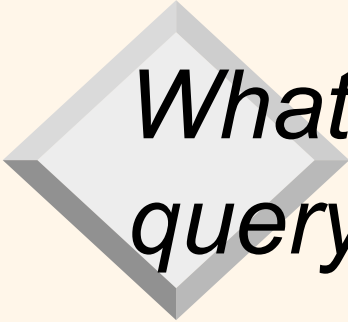


# *What is a database query, what is a query language?*

Q: Find films, they give in all cinemas, where they give something.

$$\{f \mid \forall c(\text{Shows}(\text{Cinema}_n:c) \Rightarrow \text{Shows}(\text{Cinema}_n:c, \text{F\_name}:f))\}$$
$$\{f \mid \neg \exists c(\text{Shows}(\text{Cinema}_n:c) \wedge \neg \text{Shows}(\text{Cinema}_n:c, \text{F\_name}:f))\}$$

The expressions denote the same query.

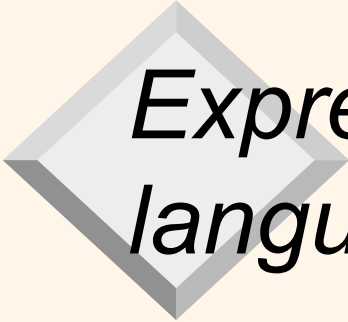


# *What is a database query, what is a query language?*

- (Database) query of type  $(S \rightarrow T)$  is a partial recursive function  $q$ , which for each database  $S^*$  provides an answer  $q(S^*)$  of type  $T$ , or it is not defined on  $S^*$ .

## Restrictions:

- values in  $q(S^*)$  are from  $S^*$ ,
- the answer to a query does not depend on representation of data in DB,
- elements of DB are conceived as non-interpreted objects.
- A query language over  $S$  is a set of expressions over a finite alphabet + meaning function assigning to each expression a query.



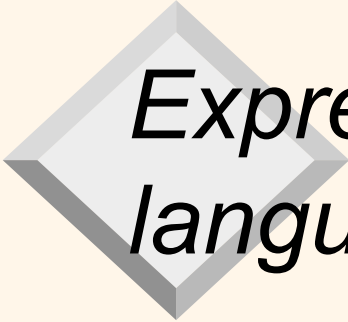
# *Expressive power of relational languages*

- **Expressive power of a query language L over S** is a set of all queries  $M(L)$ , which are expressible by L.

$L_1 < L_2$  if and only if  $M(L_1) \subset M(L_2)$

$J_1 \cong J_2$  if and only if  $M(L_1) = M(L_2)$

- Query language is called **complete**, if it can express all database queries.



# *Expressive power of relational languages*

- Programming vs. relational algebra
  - relational algebra is a high-level language
- A query language is called **relationally complete**, if it is (at least) as expressive as the relational algebra.
- Commercial world:
  - SQL,
  - languages forms,
  - picture languages



# *Extension of relational languages*

Problems with queries:

- Query on the number of something (COUNT), or AVARAGE, or calculating the value in a n-tuple,
- Find all subordinates of John (in all levels)  
(transitive closure of a relation).

Question: is it possible to propose a non-procedural computationally complete language?

Partial solutions:

- introducing aggregation functions

{c, number | number=COUNT(f | Shows(Cinema\_n:c, F\_name:f))}

- introducing a least fixpoint
- procedural constructs: while, repeat, ...

Compromise in practice: SQL + stored procedures