

- forall quantification
 - Syntax: $\forall t \in R(P(t))$
 - semantics: for all tuples t in relation R , $P(t)$ has to be fulfilled
- example query: Determine all students who have at least attended one lecture held by a professor with name Curie.

$$\{s \mid s \in \text{students} \\ \wedge \exists a \in \text{attends}(s.\text{reg-id} = a.\text{reg-id} \\ \wedge \exists l \in \text{lectures}(a.\text{id} = l.\text{id} \\ \wedge \exists p \in \text{professors}(p.\text{pers-id} = l.\text{held_by} \wedge l.\text{name} = \text{„Curie“}))))\}$$

- example query: Find the registration ids of those students who attended *all* lectures with four credits.

$$\{[s.\text{reg-id}] \mid s \in \text{students} \wedge \forall l \in \text{lectures}(l.\text{credits} = 4 \Rightarrow \\ \exists a \in \text{attends}(a.\text{id} = l.\text{id} \wedge a.\text{reg-id} = s.\text{reg-id}))\}$$

Formal definition of the tuple relational calculus

- an expression of the tuple calculus has the form $\{t \mid P(t)\}$
 - result specification left from the “|” sign
 - formula (predicate) $P(t)$ with free tuple variable t
 - formula P can contain further tuple variables which must not be free

- a formula is constructed from **atoms** where an atom has the following form:
 - $s \in R$ s tuple variable, R relation name
 - $s.A \theta t.B$ s and t tuple variables, A and B attribute names, θ comparison operators, $\theta \in \{=, \neq, <, \leq, >, \geq\}$, matching domains for $s.A$ and $t.B$
 - $s.A \theta c$ meaning of s and A like above, c is constant with $c \in \text{dom}(s.A)$
- structure of formulae (bottom-up approach)
 - All atoms are formulae.
 - If P is a formula, then so are $\neg P$ and (P) .
 - If P and Q are formulae, then so are $P \wedge Q$, $P \vee Q$ and $P \Rightarrow Q$.
 - If $P(t)$ is a formula containing a free variable t , and R is a relation, then $\exists t \in R(P(t))$ and $\forall t \in R(P(t))$ are also formulae.
- problem: expressions of the tuple relational calculus can specify infinite results
 - examples: $\{p \mid \neg(p \in \text{professors})\}$
 - solution: restriction to safe queries: An expression of the tuple relational calculus is **safe**, if the result of the expression is a subset of the **domain** of the formula.
 - domain of a tuple relational formula: all values that appear in the formula as constants, and all values (i.e. attribute values in tuples) of relations whose names are mentioned in the formula.

3.6 The Domain Relational Calculus

- ❑ non-procedural, declarative query language

Queries in the domain relational calculus

- ❑ variables are bound to domains, i.e., to value sets of attributes
- ❑ form: $\{[v_1, \dots, v_n] \mid P(v_1, \dots, v_n)\}$
- ❑ the v_i ($1 \leq i \leq n$) are **domain variables** representing an attribute value, P is a predicate (a formula) containing the free variables v_1, \dots, v_n .
- ❑ single variables are not bound to a relation, but a list of domain variables is bound to a relation
- ❑ a formula is composed of **atoms**, an atom has the following form:
 - $[v_1, \dots, v_n] \in R$ R n -ary relation, assignment of the n domain variables v_1, \dots, v_n to the attributes of R according to the order of the attributes in the schema
 - $x \theta y$ x and y are domain variables, $\theta \in \{=, \neq, <, \leq, >, \geq\}$, θ must be applicable to the domain
 - $x \theta c$ x domain variable, c is constant with $c \in \text{dom}(x)$, θ comparison operator, must be applicable to the domain

- structure of formulae (bottom-up approach)
 - All atoms are formulae.
 - If P is a formula, then so are $\neg P$ and (P) .
 - If P and Q are formulae, then so are $P \wedge Q$, $P \vee Q$ and $P \Rightarrow Q$.
 - If $P(v)$ is a formula containing a free variable v , then $\exists v(P(v))$ and $\forall v(P(v))$ are also formulae.
- shorter notation: e.g. $\exists v_1, v_2, v_3(P(v_1, v_2, v_3))$ instead of $\exists v_1(\exists v_2(\exists v_3(P(v_1, v_2, v_3))))$
- query example: Determine the registration ids and names of students who have passed at least one exam from professor Curie.
 - $\{[m, n] \mid \exists s([m, n, s] \in \text{students} \wedge \exists v, p, g([m, v, p, g] \in \text{tests} \wedge \exists a, r, b([p, a, r, b] \in \text{professors} \wedge a = \text{“Curie”})))\}$
 - Join conditions are implicitly specified by using the same domain variables. In the example the variable m is used to perform the join between *students* and *test*. Variable m represents in both tuples $[m, n, s] \in \text{students}$ and $[m, v, p, g] \in \text{tests}$ the same registration id.
- considerations for safe expressions of the domain relational calculus similar to those for the tuple relational calculus
- relational algebra, tuple relational calculus, and domain relational calculus, the latter two restricted to safe expressions, have the same expressive power.

3.7 Some Extensions of the Relational Algebra

- ❑ problem: their definition cannot be based on the relational model

Generalized Projection

- ❑ arithmetic functions may be used in the projection list
- ❑ general form: $\pi_{F_1, F_2, \dots, F_n}(E)$, E relational algebra expression, F_1, \dots, F_n arithmetic expressions with constants and attributes from the schema of E
- ❑ example: Which credit line do the customers still have?

$\pi_{\text{name, credit-line-debit}}(\text{customer})$

customer		
name	credit-line	debit
Benson	6000	700
Smith	2000	400
Miller	1500	1500
Jones	2000	1750

$\pi_{\text{name, credit-line-debit}}(\text{customer})$	
name	credit-line – debit
Benson	5300
Smith	1600
Miller	0
Jones	250

Aggregate functions

- ❑ **Aggregate functions** are functions that take a set of values and that yield a single value in a result tuple.
- ❑ examples: **sum**, **avg**, **count**, **min**, **max** for sum, average, number, minimum, maximum of a *multiset* (*bag*) of values (not contained in the relational model)
- ❑ multiset = set with frequency of occurrence
- ❑ elimination of duplicates with the aid of the keyword **distinct**
- ❑ example: relation employees(name, department, salary)
 - What is the whole amount of the employees' salaries?
sum_{salary}(employees)
 - What is the average salary of employees?
avg_{salary}(employees)
 - How many employees are occupied?
count(employees)
 - What is the highest salary of an employee?
max_{salary}(employees)

- In how many departments are the employees occupied?

count-distinct_{department}(employees)

Grouping

- ❑ First the tuples of a relation are **grouped** (collected) according to equal values of some selected attributes of the relation. Afterwards an aggregate function is applied to each group.

- ❑ example: relation employees(name, department, salary)

- What is the number of employees per department?

department **count**(employees)

- What is the average salary of the employees in each department?

department **avg**_{salary}(employees)

3.8 Extension of the relational algebra on the basis of multi-relations

Problems of the relational algebra and the relational calculus

- ❑ Since the relational algebra is a universal algebra, some important query types are not supported by the model.
- ❑ sorting of data (especially as the result of a query) desirable
- ❑ storing of duplicates, which e.g. have been computed by the projection operation, is frequently desired
- ❑ insufficient functionality of the relational algebra: aggregation (e.g., sum, average, maximum) of the data of a relation desirable but not possible by the model
- ❑ In SQL, which is the standard database query language for relational systems, these design requirements have been taken into account.

M-relations

- ❑ A **multi-relation (M-relation)** is based on the concept of a **multiset**, i.e., equal elements (tuple) may occur in a relation multiple times. The M-relation schema is equal to the respective relation schema (without duplicates in the relations).
- ❑ notation: special brackets “⟨” and “⟩”, e.g. ⟨6, 7, 6⟩
- ❑ Let M be a multiset. Let $F(M, x)$ denote the frequency of occurrence of an element x in M .
- ❑ Let M and N be multisets. Then gilt $M = N$ if and only if

$$\forall x : F(M, x) = F(N, x)$$
- ❑ Let $MRel$ be the type for M-relations and let Rel be the type for relations. We additionally introduce the operation $\delta : MRel \rightarrow Rel$ which removes the duplicates from an M-relation.
- ❑ example: Let $R = \langle (3), (4), (1), (6), (4), (1), (7) \rangle$ be an M-relation. Then $\delta(R) = \{(3), (4), (1), (6), (7)\}$ is a relation.
- ❑ relational operators γ have the form $\gamma : MRel \rightarrow MRel$ oder $\gamma : MRel \times MRel \rightarrow MRel$

Extension of the relational operators

- Let R and S be two M-relations, and let T be the M-relation of the output.

- Union
 - R and S be schema compliant, $R \cup S$ has the same schema as R resp. S
 - $\forall x: F(R \cup S, x) = F(R, x) + F(S, x)$
 - no elimination of duplicates
 - example: Let $R = \langle(1), (2), (1)\rangle$ and $S = \langle(2), (3)\rangle$. Then $R \cup S = \langle(1), (2), (1), (2), (3)\rangle$.

- Difference
 - R and S be schema compliant, $R - S$ has the same schema as R resp. S
 - $\forall x: (F(S, x) = 0 \wedge F(R - S, x) = F(R, x)) \vee (F(S, x) > 0 \wedge F(R - S, x) = 0)$
 - The difference removes *all* elements from the first multiset that appear in the second multiset.
 - example: Let $R = \langle(1), (2), (1), (2)\rangle$ and $S = \langle(2), (3)\rangle$. Then $R - S = \langle(1), (1)\rangle$.
 - alternative definition conceivable

□ Cartesian product

- result schema of $R \times S$ is the concatenation of the schemas of R and S resp. the union of the attributes of R and S
- $\forall x \forall y: F(R \times S, xy) = F(R, x) \cdot F(S, y)$
- example: Let $R = \langle(1), (2), (1)\rangle$ and $S = \langle(2), (3)\rangle$. Then $R \times S = \langle(1, 2), (1, 3), (2, 2), (2, 3), (1, 2), (1, 3)\rangle$.

□ Projection

- relation schema of the projection is implicitly defined as subset of the argument schema
- $\forall x: F(\pi(R), x) = F(R, x)$
- example: for $R = \langle(1, 2), (1, 3)\rangle$ holds: $\pi_1(R) = \langle(1), (1)\rangle$.

□ Selection

- no change of the relation schema
- $\forall x: (P(x) \wedge F(\sigma_P(R), x) = F(R, x)) \vee (\neg P(x) \wedge F(\sigma_P(R), x) = 0)$
- The selection on M-relations hence corresponds to the selection on relations.
- example: Let $\{A, B\}$ be the schema of R , and let $R = \langle(1, 2), (1, 3)\rangle$. Then $\sigma_{B=2}(R) = \langle(1, 2)\rangle$ holds.

❑ Renaming

- corresponds to the operation on relations

❑ R relation $\Rightarrow R$ M-relation

- hence relations are permitted as operands
- the δ -operator has to be used to obtain a relation as a result

❑ example:

- M-relation schemas:
 - + cities(cname, cpop, sname)
 - + states(sname, spop, party)
- Find the names of cities in all states.

$$\delta(\pi_{\text{sname}}(\sigma_{\text{cpop} \geq 8000}(\text{cities})))$$

❑ Aggregation

- only numerical aggregation operations considered
- **sum, avg, count, min, max** : MRel \rightarrow *numeric* with *numeric* \in {integer, real, ...}
- For the application of aggregation functions an additional attribute has to be specified which is used for forming the aggregation.
- The operation **count** yields the number of tuples in a relation.

□ Generalization of Projection

- For M-relations the projection operation is an operator which for each tuple of the input creates a tuple of the output. Such mappings are also called *maps*.
- Let R and S be M-relations with the schemas R resp. S , and let $f: (R \rightarrow \text{dom}(R)) \rightarrow (S \rightarrow \text{dom}(S))$ be a mapping. Then mapping μ_f defines the **relational map**, where
 - + $\mu_f(R) = S$
 - + $\forall x: F(\mu_f(R), f(x)) = F(R, x)$
- f is a function which maps a tuple of relation R to a tuple of relation S . We here assume a definition of a tuple as mapping.
- example: Let $R = \{(1, 2), (3, 4), (4, 6)\}$ be a relation with the schema $R = \{A, B\}$. Let furthermore S be the result relation with the schema $S = \{C, D\}$. Then the function $f(t) = (C \rightarrow t[A] * t[B], D \rightarrow t[A] + t[B])$ defines an example map with schema S . We obtain the relation $S = \{(2, 3), (12, 7), (24, 10)\}$.

□ Grouping

- **Grouping** is used to subdivide a relation into partitions according to a certain criterion (predicate) and to aggregate over each partition in a suitable way.
- A partition can conceptually be also regarded as a (temporary) relation which inherits its schema from the original relation.
- A partition must not be empty. All partitions are disjoint.
- For a relation R with schema R a partitioning is defined by a set $\{A_1, \dots, A_n\} \subseteq R$. A specific partition contains all tuples from R which have the same value with respect to the attributes A_1, \dots, A_n .
- For each partition a value is computed with the aid of an aggregation function and with respect to a specified attribute. This value together with the values of the partition attributes is inserted into the result relation.
- formal: $\gamma_{X, A, agg, B} : MRel \rightarrow Rel$
 X set of partition attributes, A aggregation attribute (partially optional), agg aggregation function, B further (in general numerical) attribute
- generalization: calculation of several aggregates for each partition

- examples
 - + How many C3 professors are there?
 $\mathbf{count}(\sigma_{\text{rank} = \text{"C3"}}(\text{professors}))$
 - + How many C4 and how many C3 professors are there?
 $\gamma_{\{\text{rank}\}, \mathbf{count}, \text{total}}(\text{professors})$
 - + What is the average semester number of the students?
 $\gamma_{\emptyset, \text{sem}, \mathbf{avg}, \text{average}}(\text{students})$
 - + How many hours per week in a semester do the professors give lectures (relation *lectures* here extended by an attribute *duration*)?
 $\gamma_{\{\text{held_by}\}, \text{duration}, \mathbf{sum}, \text{number}}(\text{lectures})$

□ S-relation

- An **S-relation (sorted relation)** is a triple $(R, <_R, \varphi)$ where
 - + R is an M-relation of arity r ,
 - + $<_R$ is an order relation defined over $D_1 \times \dots \times D_r$
 - + $\varphi : \{1, \dots, n\} \rightarrow R$ is a bijective mapping with $n = |R|$ so that $\varphi(i) <_R \varphi(j)$ for $i < j$
- $\omega_{<} : \text{MRel} \rightarrow \text{SRel}$ maps an M-relation R to an S-relation $S = (R, <_R, \varphi)$ where $<_R$ defines an order relation on R and φ is a mapping sorting the relation (underlying model used here: relation as a sequence of tuples, like in a table).

– Beispiel:

R		
A	B	C
4	1	3
9	2	3
7	8	4

Let $<_1 = \{((a, b, c), (d, e, f)) \mid a < d\}$ and

$<_2 = \{((a, b, c), (d, e, f)) \mid (c < f) \vee (c = f \wedge b > e)\}$ be order relations. Then:

$\omega_{<_1}(R)$		
A	B	C
4	1	3
7	8	4
9	2	3

$\omega_{<_2}(R)$		
A	B	C
9	2	3
4	1	3
7	8	4

- Mapping ω creates an S-relation. Since the operations of the relational algebra are not defined for S-relations, it is only possible in an algebra expression to apply ω as the last operation.
- example: Compute the list of names of all cities whose population numbers exceed the population number of a state. Sort the list concerning *cpop*.

$\omega_{<}(\pi_{cname, cpop}(\sigma_{cpop > spop}(\text{cities} \times \text{states})))$ [assumed: “<” defined on *cpop*]

- expression computes too much information, since in addition to the name of the city also the city population is stored.