

- ❑ example: Which lectures are required for the immediate predecessors?

**select** predecessor

**from** is\_precondition\_of

**where** successor **in** ( **select** predecessor

**from** is\_precondition\_of, lectures

**where** successor = id **and** title = "The Vienna Circle")

- ❑ SQL, relational algebra and relational calculus do not offer possibilities for an efficient computation of recursive queries.

## Insertion of tuples

- ❑ immediate input of constant values to fill relations with data
  - **insert into** <relation name>[(<attribute name> [, <attribute name>]\*)]
  - **values** (<constant> [, <constant>]\*)
  - **insert into** professors **values**(2136, "Curie", 536, "C4")
  - input of attribute values according to the order in the schema definition
  - It is possible to insert only a part of the attribute values of a tuple, if, e.g., some values are unknown. The undefined fields are automatically filled by the system with null values.
  - **insert into** students (reg-id, name) **values**(28121, "Archimedes")

- ❑ Generation of tuples by means of a query
  - **insert into** <relation name>[(<attribute name> [, <attribute name>]\*)]  
**select ... from ... where ...**
  - **insert into** attends **select** reg-id, id  
**from** students, lectures  
**where** title = “logic”

## Deletion of tuples

- ❑ with a given condition those tuples are selected that are to be deleted
- ❑ **delete from** <relation name> [**where** <condition>]
- ❑ example: students who study longer than 8 semester are to be deleted from the relation  
**delete from** students **where** sem > 8

## Change of tuples

- ❑ with a given condition those tuples are selected that are to be changed
- ❑ **update** <relation name>  
**set** <attribute name> = <expression> [, <attribute name> = <expression>]\*  
[**where** <condition>]
- ❑ increase of the semester number of each student at the beginning of a new semester  
**update** students  
**set** sem = sem + 1

## 4.4 Views in SQL

### Creation of a view

- ❑ The purpose of views is to adapt a DBS to the requirements and access rights of different user groups. They correspond to the external DB schemas.
- ❑ A **view** is a **virtual relation** (virtual table) that is derived from other relations (tables). “Virtual” means that no new tables are created. They are recalculated for each new application.
- ❑ A view determines which data a user may access and which data a user must not access.
- ❑ **create view** <view name> [(<attribute name> [, <attribute name>]\*)] **as** <subquery> [**with check option**]
- ❑ example: A view on *tests* shall express the restriction that not each user is allowed to see the results of an exam.

```
create view tests_view as  
select reg-id, id, pers-id  
from tests
```

## Change of a view

- ❑ Views have the inherent problem that they frequently cannot be updated.
- ❑ example:
  - **create view** grading(pers-id, avg-grade) **as**  
**select** pers-id, **avg**(grade)  
**from** tests  
**group by** pers-id
  - This view is not changeable since it contains the computed attribute *avg-grade*. An update operation cannot be transferred to the original base relation *tests*. The following operation is rejected by the DBMS:  
**update** grading  
**set** avg-grade = 3.0  
**where** pers-id = (**select** pers-id **from** professors **where** name = "Sokrates")
- ❑ example:
  - **create view** lecture-view **as**  
**select** title, credits, name **from** lectures, professors  
**where** held\_by = pers-id

- insertion of a new lecture is impossible

**insert into** lecture-view  
**values** (“nihilism”, 2, “Nobody”)

In order to insert tuples, the DBMS would have to be able to assign the entered values to the original relations. This is not always possible. Here the view removes the keys of the original relations.

- ❑ In general views can be updated if
  - they contain neither aggregate functions nor constructs like **distinct**, **group by**, **having**, **union** and **minus**,
  - only unique column names are in the **select** list and a key of the base relation is contained and
  - they use exactly one base relation or *changeable* view in the **from** clause
- ❑ There are also views which can be updated although they do not fulfil the three aforementioned conditions.

# 5. Other Relational Database Query Languages

## 5.1 Query-by-Example (QBE)

### Features of QBE

- ❑ developed at the beginning of the seventies by IBM, later part of DB2
- ❑ uses a skeleton table for the specification of a query, i.e., QBE has a two-dimensional syntax
- ❑ QBE queries are expressed by means of examples. The system generalizes the examples in order to compute answers for queries.
- ❑ declarative approach
- ❑ QBE is based on the domain relational calculus: variables are bound to attribute domains
- ❑ example database schema

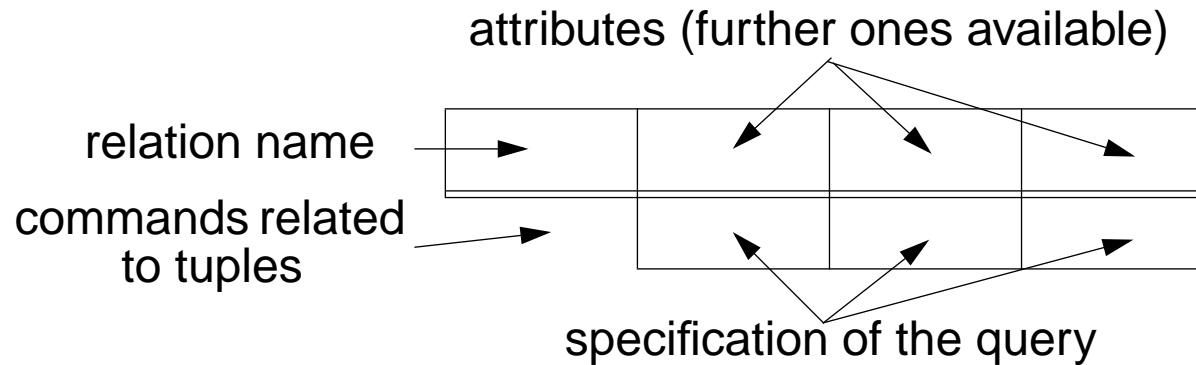
customer(cname, caddr, account)

order(cname, product, amount)

vendor(vname, vaddr, product, price)

## Onscreen dialog

- query: Find names and addresses of customers with a negative balance.
  - request for a skeleton table



- the name of the requested relation is inserted, followed by “P.” (= print)

customer P.			

- attribute names are inserted by the system

customer P.	cname	caddr	account

- query is specified by an entry in the table

customer	cname	caddr	account
	P.	P.	< 0

- table is filled with values

customer	cname	caddr
	Smith	Gainesville, FL 32611
	Jones	Ocala, FL 35768
	Meyer	Orlando, FL 40567

- language elements
  - commands: e.g. P., I., D.
  - domain variables: `_X`, `_Meyer`
  - constants: Smith, 123
  - boolean, arithmetic and relational operators

- example: Which vendors deliver milk?

vendor	vname	vaddr	product	price
	P._x	P._y	milk	

in the domain relational calculus this corresponds to:

$$\{[x_1, x_2] \mid \exists x_4([x_1, x_2, \text{milk}, x_4] \in \text{vendor})\}$$

- Free domain variables, which are used only once, need not be mentioned. This leads to a simpler notation of the query:

vendor	vname	vaddr	product	price
	P.	P.	milk	

- example: Which vendors deliver milk *or* flour?

vendor	vname	vaddr	product	price
	P.	P.	milk	
	P.	P.	flour	

in the domain relational calculus this corresponds to:

$$\{[x_1, x_2] \mid \exists x_4([x_1, x_2, \text{milk}, x_4] \in \text{vendor} \vee [x_1, x_2, \text{flour}, x_4] \in \text{vendor})\}$$

If several pattern rows are inserted, the use of domain variables decides whether the rows are connected by a logical “or” or a logical “and”.

- ❑ example: Which vendors deliver milk for a prize between \$1 and \$1.20?

vendor	vname	vaddr	product	price
	P._X	P.	milk	>= 1.00
	_X	P.	milk	<= 1.20

in the domain relational calculus this corresponds to:

$$\{[x_1, x_2] \mid \exists x_4([x_1, x_2, \text{milk}, x_4] \in \text{vendor} \wedge x_4 \geq 1.00 \wedge x_4 \leq 1.20)\}$$

### Condition Box

- ❑ formulation of conditions, that do not fit into the tables, in special **condition boxes**
- ❑ comparison of values of two different table columns possible
- ❑ conditions inserted into a column can only refer to the contents of the column

- example: Which vendors deliver milk or flour?

vendor	vname	vaddr	product	price
	P.	P.	_M	

Conditions
_M = (milk or flour)

- example: Which vendors deliver Brie and Perrier for a total price not more than \$7?

vendor	vname	vaddr	product	price
	P._N		Brie	_X
	_N		Perrier	_Y

Conditions
_X + _Y <= 7.00

- example: Which vendors deliver milk for a prize between \$1 and \$1.20?

vendor	vname	vaddr	product	price
	P._X		milk	_USD

Conditions
_USD >= 1.00 and _USD <= 1.20

## Join queries

- examples: Which vendors deliver to Meyer?

vendor	vname	vaddr	product	price
	P.		_W	

order	cname	product	amount
	Meyer	_W	

in the domain relational calculus this corresponds to:

$$\{[x_1] \mid \exists x_2, x_3, x_4([x_1, x_2, x_3, x_4] \in \text{vendor} \wedge \exists y_3([“Meyer”, x_3, y_3] \in \text{order}))\}$$

- QBE automatically removes duplicates. To avoid this, the ALL. operator is applied.

## Result relation

- ❑ example: Output all customers, their accounts, what they order, and how much they order.

customer	cname	caddr	account
	_N		_K

order	cname	product	amount
	_N	_W	_M

subscription	name	commodity	amount	balance
P.	_N	_W	_M	_K

- ❑ New skeleton table must be created before the query.
- ❑ “P.” under the relation name means that all attributes of the relation belong to the output.

## Negation

- example: Find for each product the cheapest vendor.

vendor	vname	vaddr	product	price
P.			_W	_P
$\neg$			_W	< _P

- Negation  $\neg$  means that no such tuple exists.
- If  $\neg$  is used in front of an attribute, this is a shortcut for  $\neq$ .
- example: Find all vendors which can deliver at least two articles.

vendor	vname	vaddr	product	price
	P._x		_W	
	_x		$\neg$ _W	

## Aggregate functions

- ❑ All aggregate functions require that duplicates are initially not removed (realized by “ALL.”).
- ❑ The keyword “UN.” makes it possible to remove duplicates and to apply the aggregate functions nevertheless.
- ❑ aggregate functions: SUM., AVG., MAX., MIN., CNT.
- ❑ example: How many vendors are there?

vendor	vname	vaddr	product	price
	P.CNT.UN.ALL._X			

- ❑ example: How many liters of milk have been ordered?

order	cname	product	amount
		milk	P.SUM.ALL._X

## Grouping

- ❑ grouping of tuples possible with the “G.” operator
- ❑ example: average price structure of all vendors

vendor	vname	vaddr	product	price
	P.G			P.AVG.ALL._X

- ❑ example: average price structure of all vendors whose average price is more than \$1000

additionally the following condition box:

Conditions
AVG.ALL._X > 1000

corresponds to the **having** clause in SQL

- ❑ vendor names in ascending order: P.AO.
- ❑ vendor names in descending order: P.DO.

## Insertion, change and deletion

- ❑ commands “I.” und “D.” are used for the insertion and deletion of tuples.
- ❑ example: Smith orders 20 pieces of an article which has been ordered by Jones more than hundred times.

order	cname	product	amount
I.	Smith	_W	20
	Jones	_W	> 100

- ❑ Command “U.” is used for updating the values in non-key attributes.
- ❑ Empty fields are not changed.
- ❑ Values of the key attributes must be specified.
- ❑ example: Jones prices milk to \$1.30.

vendor	vname	vaddr	product	price
U.	Jones		milk	1.30