Query-by-Example (QBE)

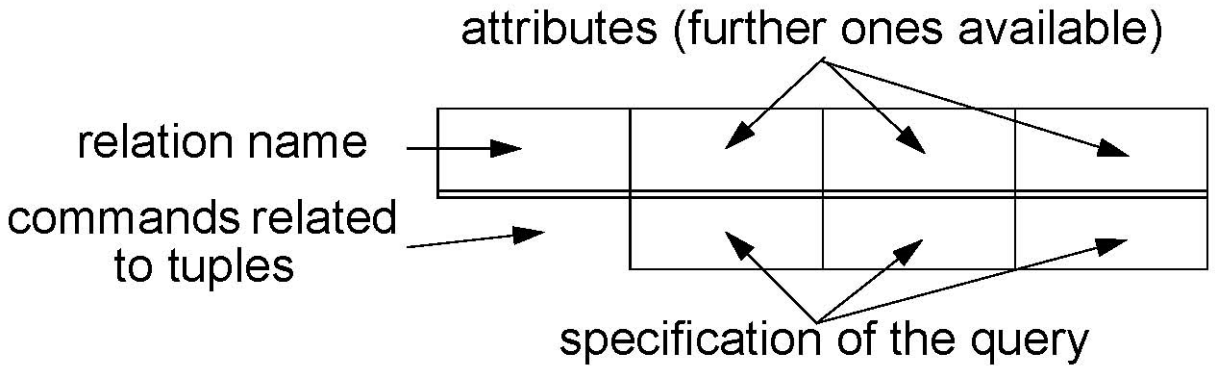
Features of Query-by-Example

- Acronym: QBE
- QBE
  - was developed at the beginning of the seventies by IBM
  - is a *graphical* query language where queries *look* like tables
  - uses *skeleton tables* for the specification of a query
  - has a two-dimensional syntax
  - queries are expressed “by example”
  - generalizes the examples in order to compute answers to queries
  - pursues a declarative approach
  - is based on the domain relational calculus: variables are bound to attribute domains (domain variables)
- Database schema used in examples

  customer(cname, caddr, account)
  order(cname, product, amount)
  vendor(vname, vaddr, product, price)

Hypothetical Onscreen Dialog (I)

- Query 1: Find the names and addresses of customers with a negative balance.
  - Request a skeleton table



  - Insert the name of the requested table, followed by the QBE command "P." ("P." stands for "print", "display"; corresponds to a projection in the RA)

customer P.			

Hypothetical Onscreen Dialog (II)

- Attribute names are inserted by the system

customer	cname	caddr	account

- Specify the query by making entries in the table cells

customer	cname	caddr	account
	P.	P.	< 0

- Table is filled with values by the system

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

Queries on a Single Table (I)

- Language elements
  - Commands: e.g. P. (print), I. (insert), D. (delete), U. (update)
  - Example elements, domain variables: _X, _Meyer
  - Constants: Smith, 123
  - Boolean (e.g., and, or, not), arithmetic (e.g., +, –, *, /), and relational operators (e.g., =, <, >=, <>)
- Query 2: List all vendor names.

vendor	vname	vaddr	product	price
	P.			

- QBE (unlike SQL) performs duplicate elimination automatically
- Use the ALL. command to suppress duplicate elimination

vendor	vname	vaddr	product	price
	P.ALL.			

Queries on a Single Table (II)

- Query 3: Display the entire table *order*.

order	cname	product	amount
	P.	P.	P.

Alternatively, one can use the following shorthand notation:

order	cname	product	amount
P.			

- Query 4: Find the names and addresses of the vendors who deliver milk.

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

The semantics (i.e., meaning) of the QBE query corresponds in the domain relational calculus to the expression

$\{[x, y] \mid \exists z\ ([x, y, \text{“milk”}, z] \in \text{vendor})\}$

Queries on a Single Table (III)

- Free domain variables that 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	

- Query 5: Find the names and addresses of the vendors who deliver milk or flour.

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

In the domain relational calculus this QBE query corresponds to the expression

$\{[x, y] \mid \exists z\, ([x, y, \text{“milk”}, z] \in \text{vendor} \vee [x, y, \text{“flour”}, z] \in \text{vendor})\}$

Queries on a Single Table (IV)

- Query 6: Determine the vendor names and product names of those products that cost less than or equal to $5.

vendor	vname	vaddr	product	price
	P.		P.	<=5

In the domain relational calculus the QBE query corresponds to:

$\{[n, p] \mid \exists a \exists b ([n, a, p, b] \in \text{vendor} \wedge b <= 5)\}$

- Comparisons correspond to selections in the RA and can be used in table cells
  - They can only involve one arithmetic expression on the right-hand side of the comparison operator (e.g., >, <=, <>)
  - The expression must include constants but not tuple variables
  - The space on the left-hand side must be blank since the left-hand side corresponds to the attribute name ("*price* <= 5" above)

Queries on a Single Table (V)

- Query 7: Find the products that are delivered by both Smith and Jones.

vendor	vname	vaddr	product	price
	Smith		P._x	
	Jones		_x	

In the domain relational calculus the QBE query corresponds to:

$\{[x] \mid \exists a \exists b$ (["Smith", a, x, b] $\in$ vendor) $\wedge \exists a \exists b$ (["Jones", a, x, b] $\in$ vendor)$\}$

Note: This expression is equivalent to

$\{[x] \mid \exists a \exists b$ (["Smith", a, x, b] $\in$ vendor) $\wedge \exists c \exists d$ (["Jones", c, x, d] $\in$ vendor)$\}$

since bound tuple variables are *locally* bound only

- The primary purpose of tuple variables in QBE is to force values of different tuples to have the same value on certain attributes
- 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".

Queries on a Single Table (VI)

- Query 8: Find the names of all vendors who deliver the same products as Jones.

vendor	vname	vaddr	product	price
	P._x		_y	
	Jones		_y	

In the domain relational calculus the QBE query corresponds to:

$\{[x] \mid \exists y\, (\exists a\, \exists b\, ([x, a, y, b] \in \text{vendor}) \wedge \exists c\, \exists d\, ([\text{“Jones”}, c, y, d] \in \text{vendor}))\}$

- Query 9: Find the names of all vendors who deliver milk for a price between $1 and $1.20.

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

In the domain relational calculus the QBE query corresponds to:

$\{[x] \mid \exists a\, \exists p\, ([x, a, \text{“milk”}, p] \in \text{vendor}) \wedge p >= 1.00 \wedge p <= 1.20)\}$

Negation and Inequality (I)

- Query 10: Find for each product the name(s) of the cheapest vendor.

vendor	vname	vaddr	product	price
	P._n		_w	_p
¬	_m		_w	< _p

In the domain relational calculus the QBE query corresponds to:

$$\{[n] \mid \exists w\, (\exists a\, \exists p\, ([n, a, w, p] \in \text{vendor}) \wedge \neg\exists m\, \neg\exists b\, \neg\exists q\, ([m, b, w, q] \in \text{vendor} \wedge q < p))\}$$

- Negation ¬ (or: **not**) means that no such tuple exists (such that …)

- If ¬ is used in a table cell in front of a tuple variable or a constant, this is a shortcut for <> (≠).

Negation and Inequality (II)

- Query 11: Find all vendors who can deliver at least two products.

vendor	vname	vaddr	product	price
	P._x		_w	
	_x		¬_w	

In the domain relational calculus the QBE query corresponds to:

$$\{[x] \mid \exists w\, (\exists a\, \exists p\, ([x, a, w, p] \in \text{vendor}) \wedge \exists b\, \exists v\, \exists q\, ([x, b, v, q] \in \text{vendor} \wedge w <> v))\}$$

Queries on Several Tables (I)

- QBE allows queries that span several different tables (analogous to the Cartesian product or join in the Relational Algebra)
- Connections among the various tables are achieved through the use of the same domain variable in different table cells to force certain tuples to have the same value on certain attributes ("the same domain variable denotes the same value")
- Query 12: Determine all vendors that deliver to Meyer.

vendor	vname	vaddr	product	price
	P.		_p	

order	cname	product	amount
	Meyer	_p	

In the domain relational calculus the QBE query corresponds to:

$\{[n] \mid \exists p\, (\exists b\, \exists d\, ([n, b, p, d] \in \text{vendor}) \wedge \exists a\, ([\text{“Meyer”}, p, a] \in \text{order}))\}$

Queries on Several Tables (II)

- Query 13: Find the names and addresses of all customers who have a balance larger than $800 and have made an order for a product delivered by a vendor. Output the vendor's name and the product delivered too.

customer	cname	caddr	account
	P._n	P._a	> 800

order	cname	product	amount
	_n	P._p	

vendor	vname	vaddr	product	price
	P._v		_p	

In the domain relational calculus the QBE query corresponds to:

$$\{[n, a, p, v] \mid \exists b\,([n, a, b] \in \text{customer} \wedge b > 800) \wedge \exists c\,([n, p, c] \in \text{order}) \wedge \exists d\,\exists e\,([v, d, p, e] \in \text{vendor})\}$$

Queries on Several Tables (III)

- Query 14: Find the names of all customers who have not made an order.

customer	cname	caddr	account
	P._n		

order	cname	product	amount
¬	_n		

In the domain relational calculus the QBE query corresponds to:

$\{[n] \mid \exists b \exists d ([n, b, d] \in \text{customer}) \wedge \neg\exists p \neg\exists a ([n, p, a] \in \text{order})\}$

Queries on Several Tables (IV)

- Query 15: Determine the names and addresses of all customers who have ordered all products that customer Benson has ordered.

customer	cname	caddr	account
	P._n	P._a	

order	cname	product	amount
	_n	_p	
	Benson	_p	

In the domain relational calculus the QBE query corresponds to:

$$\{[n, a] \mid \exists b\,([n, a, b] \in \text{customer}) \wedge \exists p\,(\exists c\,([n, p, c] \in \text{order}) \wedge \exists d\,([\text{“Benson”}, p, d] \in \text{order}))\}$$

Condition Box (I)

- Sometimes it is either inconvenient or impossible to express all constraints on the domain variables within the skeleton tables
- In particular, a constraint in a table cell is restricted to the contents of the respective column
- QBE provides a condition box feature that allows the expression of general constraints over any of the domain variables
- QBE allows logical expressions in a condition box that may make use of the logical operators **and** and **or**
- Query 1: Find the names of all vendors who deliver milk or flour.

vendor	vname	vaddr	product	price
	P.		_p	

Conditions
_p = milk **or** _p = flour

Condition Box (II)

Alternatively, the logical operator **or** can be used in an unconventional way to allow comparisons with a set of constant values

vendor	vname	vaddr	product	price
	P.		_p	

Conditions
_p = (milk **or** flour)

- Query 2: Find the names of all vendors who deliver milk for a price between $1 and $1.20.

vendor	vname	vaddr	product	price
	P.		milk	_USD

Conditions
_USD >= 1.00 **and** _USD <= 1.20

Alternatively, we can write:

vendor	vname	vaddr	product	price
	P.		milk	_USD

Conditions
_USD = (>= 1.00 **and** <= 1.20)

Condition Box (III)

- Query 3: Give out all products that are also available for a cheaper price.

vendor	vname	vaddr	product	price
			P._x	_p
			_x	_q

Conditions
_p > _q

- Query 4: Find the names of vendors that deliver Brie and Perrier for a total price not more than $7.

vendor	vname	vaddr	product	price
	P._n		Brie	_p
	_n		Perrier	_q

Conditions
_p + _q <= 7.00

- Query 5: Find the names of customers that have ordered the same product more than five times as another customer.

order	cname	product	amount
	P.	_p	_a
		_p	_b

Conditions
_a > 5 * _b

Condition Box (IV)

- Query 6: Output the products and the names of customers who can pay the corresponding orders individually (not in total).

customer	cname	caddr	account
	P._n		_a

order	cname	product	amount
	_n	P._p	_b

Conditions
_a >= _b * _c

vendor	vname	vaddr	product	price
			_p	_c

- To also obtain the vendor who can deliver the product, we put the command "P." under the attribute "vname"

Aggregation (I)

- All aggregate functions require that duplicates are initially not removed (realized by “ALL.”).
- The keyword “UN.” (for “unique”) makes it possible to remove duplicates and apply the aggregate functions CNT., SUM., or AVG. afterwards.
- Aggregate functions are: SUM., AVG., MAX., MIN., CNT.
- Query 7: Calculate the number of customers.

customer	cname	caddr	account
	P.CNT.ALL._x		

- No duplicate customer names exist since “cname” is primary key
- The domain variable can also be omitted
- Query 8: Calculate the number of different products offered by vendors.

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

Aggregation (II)

- In order to maintain (eliminate) duplicates in the context of aggregate functions, the clause ALL. (UN.ALL.) has to be used
- Query 9: Compute how many liters of milk have been ordered.

order	cname	product	amount
		milk	P.SUM.ALL.

- Query 10: Compute the average price of all products.

vendor	vname	vaddr	product	price
				P.AVG.ALL.

- Speculation about the use of the '.' notation
  - Terms in front of the '.' notations are functions; functions can be composed; hence, the '.' can be interpreted as the *function composition operator* '∘'
  - For example, the expression P.CNT.UN.ALL._x corresponds to
    P ∘ CNT ∘ UN ∘ ALL(x) = P(CNT(UN(ALL(x)))) [$f \circ g(x) = f(g(x))$]

Grouping (I)

- Grouping of tuples is possible with the “G.” (group) operator
- Query 11: Provide the average price structure of the products sold by each vendor.

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

- Query 12: Provide the average price structure of the products sold by each vendor where the average price is more than $1000.

  Add the following condition box to the query above:

Conditions
AVG.ALL._x > 1000

  It corresponds to the **having** clause in SQL

- If “G.” appears in more than one column, these columns together form the grouping attributes, and the groups are formed with respect to equal values in the grouping attributes

Grouping (II)

- Query 13: Determine the names of the customers who have ordered all products for which there is some order.

customer	cname	caddr	account
	P.G._n		

order	cname	product	amount
	_n	_p1	
		_p2	

Conditions
CNT.UN.ALL._p1 = CNT.UN.ALL._p2

For each "_n" value (notice the "G." operator), we count all "_p1" values (after duplicate elimination) to get the number of distinct products ordered by customer "_n". We compare this count against the count of all "_p2" values (after duplicate elimination), which represents the total number of distinct products in the table *order* (i.e., the number of products found in orders). If both counts are equal for a customer, this customer has ordered all products for which there is some order.

Ordering

- Sorted output
  - in ascending order: P.AO.
  - in descending order: P.DO.
- Query 14: Provide an ordered output of all customer names.

customer	cname	caddr	account
	P.AO.		

- An optional integer argument allows one to sort on more than one attribute
- Query 15: Provide an output of all products and their vendors, ordered first by product and second by vendor name.

vendor	vname	vaddr	product	price
	P.AO(2).		P.AO(1).	

Specification of Result Tables

- Users can specify the desired schema of a query result before posing the query
- Query 16: Output all customer names, their accounts, the products they order, the amount they order, and the total cost of each order into a table *subscription* with the schema (name, commodity, amount, balance, totalCost).

customer	cname	caddr	account
	_n		_a

order	cname	product	amount
	_n	_p	_b

vendor	vname	vaddr	product	price
			_p	_c

subscription	name	commodity	amount	balance	totalCost
P.	_n	_p	_b	_a	_b * _c

Insertion, Update, and Deletion (I)

- Commands “I.”, “U.”, and “D.” are used for the insertion, update, and deletion of tuples
- Query 17: Insert the tuple (“Jones”, “Highway 15, New York”, 3000) into the table *customer*.

customer	cname	caddr	account
I.	Jones	Highway 15, New York	3000

- Query 18: Insert tuples into the table *order* for the situation where Smith orders 20 pieces of each product that has been ordered by Jones more than 100 times.

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

Insertion, Update, and Deletion (II)

- Query 19: Insert tuples into a new table *customerB* that stores the complete information of those customers who have an account larger than $10000 and live in Boston.

customer	cname	caddr	account
	_n	_a	_b

customerB	cnameB	caddrB	accountB
I.	_n	_a	_b

Conditions
_b > 10000 **and** _a LIKE "%Boston%"

- QBE's LIKE operator is similar to the SQL counterpart

Insertion, Update, and Deletion (III)

- Restrictions for updates
  - Command “U.” can only be used to update values in non-key attributes but not key attributes
  - Empty fields are not changed
  - Values of the key attributes must be specified
- Query 20: Jones prices milk to $1.30.

vendor	vname	vaddr	product	price
U.	Jones		milk	1.30

We can also write:

vendor	vname	vaddr	product	price
	Jones		milk	U.1.30

- Query 21: Increment the current milk price from Jones by $0.20.

vendor	vname	vaddr	product	price
	Jones		milk	U._p+0.20

Insertion, Update, and Deletion (IV)

- Query 22: Benson increases all his prices by 10%.

vendor	vname	vaddr	product	price
U.	Benson		_c	1.1 * _p
	Benson		_c	_p

- Query 23: Delete all orders of customers with a negative balance

customer	cname	caddr	account
	_n		< 0

order	cname	product	amount
D.	_n		

Creation of Table Schemas

- Example: Creation of the table schema *customer*

I. customer I.	cname	caddr	account
TYPE I.	CHAR	CHAR	FLOAT
LENGTH I.	40	50	8
KEY I.	Y	N	N
SYS NULL I.		–	–
INVERSION I.	Y	N	N

- Specifications
  - TYPE: Data type of an attribute (e.g., CHAR, FLOAT, FIXED)
  - LENGTH: Maximum representation length of an attribute in bytes
  - KEY: Definition of the primary key attributes
  - SYS NULL: Optional symbol to be used as the null value, here: –
  - INVERSION: Creation of an index structure on selected attributes

Creation of Views

- Example: Creation of a view *goodCustomer* for all customers with a positive balance

customer	cname	caddr	account
	_n	_a	_k

I.VIEW goodCustomer I.	cname	caddr	balance
	_n	_a	_k > 0