Query-by-Example (QBE)

Chapter 6

Example is the school of mankind,
and they will learn at no other.
-- Edmund Burke (1729-1797)

QBE: Intro

- A "GUI" for expressing queries.
  - Based on the Domain Relational Calculus (DRC)
  - Actually invented before GUIs.
  - Very convenient for simple queries.
  - Awkward for complex queries.
- QBE an IBM trademark.
  - But has influenced many projects
  - Especially PC Databases: Paradox, Access, etc.

`Example Tables' in QBE

- Users specify a query by filling in *example tables*, or *skeletons*; we will use these skeletons in our examples.

Reserves	sid	bid	day

Boats	bid	bname	color

Sailors	sid	sname	rating	age

Basics

- To print names and ages of all sailors:

Sailors	sid	sname	rating	age
		P._N		P._A

- Print all fields for sailors with *rating* > 8, in ascending order by (*rating, age*):

Sailors	sid	sname	rating	age
P.			AO(1). >8	AO(2).

- QBE puts unique new variables in blank columns. Above query in DRC (no ordering):

$$\left\{\langle I, N, T, A\rangle \mid \langle I, N, T, A\rangle \in \ Sailors \ \wedge \ T > 8\right\}$$

And/Or Queries

Note: MiniQBE uses a slightly different syntax!

- Names of sailors younger than 30 *or* older than 20:

Sailors	sid	sname	rating	age
		P.		< 30
		P.		> 20

- Names of sailors younger than 30 *and* older than 20:

Sailors	sid	sname	rating	age
	_Id	P.		< 30
	_Id	P.		> 20

- Names of sailors younger than 30 *and rating* > 4:

Sailors	sid	sname	rating	age
	_Id	P.	> 4	< 30

Duplicates

- *Single row with* P: Duplicates not eliminated by default; can force elimination by using UNQ.

Sailors	sid	sname	rating	age
UNQ.		P.		< 30

- *Multiple rows with* P: Duplicates eliminated by default! Can avoid elimination by using ALL.

Sailors	sid	sname	rating	age
ALL.	_Id	P.		< 30
	_Id	P.		> 20

Join Queries

- Names of sailors who've reserved a boat for 8/24/96 and are older than 25 (note that dates and strings with blanks/special chars are quoted):

Sailors	sid	sname	rating	age
	_Id	P._S		> 25

Reserves	sid	bid	day
	_Id		'8/24/96'

Note: MiniQBE uses double quotes

- Joins accomplished by repeating variables.

Join Queries (Contd.)

- Colors of boats reserved by sailors who've reserved a boat for 8/24/96 and are older than 25 :

Sailors	sid	sname	rating	age
	_Id	_S		> 25

Reserves	sid	bid	day
	_Id	_B	'8/24/96'

Boats	bid	bname	color
	_B	'Interlake'	P.

Join Queries (Contd.)

- Names and ages of sailors who've reserved some boat that is also reserved by the sailor with *sid* = 22:

Sailors	sid	sname	rating	age
	_Id	P.		P.

Reserves	sid	bid	day
	22	_B	
	_Id	_B	

Unnamed Columns

MiniQBE allows P. in multiple tables

- Useful if we want to print the result of an expression, or print fields from 2 or more relations.
  - QBE allows P. to appear in at most one table!

Sailors	sid	sname	rating	age		
	_Id	P.	_R	_A	P._D	P.(_R/_A)

Reserves	sid	bid	day
	_Id		_D

"Negative Tables"

- Can place a negation marker in the relation column:

Sailors	sid	sname	rating	age
	_Id	P._S		

Reserves	sid	bid	day
¬	_Id	_B	

- Variables appearing in a negated table must also appear in a positive table!

Note: MiniQBE uses NOT or ~.

Aggregates

- QBE supports AVG, COUNT, MIN, MAX, SUM
  - None of these eliminate duplicates, except COUNT
  - Also have AVG.UNQ. etc. to force duplicate elimination

Sailors	sid	sname	rating	age	
	_Id	G.	G.P.AO	_A	P.AVG._A

- The columns with G. are the *group-by* fields; all tuples in a group have the same values in these fields.
  - The (optional) use of .AO orders the answers.
  - Every column with P. must include G. or an aggregate operator.

Conditions Box

- Used to express conditions involving 2 or more columns, e.g., _R/_A > 0.2.
- Can express a condition that involves a group, similar to the HAVING clause in SQL:

Sailors	sid	sname	rating	age
			G.P.	_A

CONDITIONS
AVG._A > 30

- Express conditions involving AND and OR:

Sailors	sid	sname	rating	age
		P.		_A

CONDITIONS
20 < _A AND _A < 30

Find sailors who've reserved all boats

- A division query; need aggregates (or update operations, as we will see later) to do this in QBE.

Sailors	sid	sname	rating	age
	P.G._Id			

Reserves	sid	bid	day
	_Id	_B1	

CONDITIONS
COUNT._B1= COUNT._B2

Boats	bid	bname	color
	_B2		

- How can we modify this query to print the names of sailors who've reserved all boats?

Inserting Tuples

- Single-tuple insertion:

Sailors	sid	sname	rating	age
I.	74	Janice	7	14

- Inserting multiple tuples (*rating* is *null* in tuples inserted below):

Sailors	sid	sname	rating	age
I.	_Id	_N		_A

Students	sid	name	login	age
	_Id	_N		_A

CONDITIONS
_A > 18 OR
_N LIKE 'C%'

Delete and Update

- Delete all reservations for sailors with *rating* < 4

Sailors	sid	sname	rating	age
	_Id		< 4	

Reserves	sid	bid	day
D.	_Id		

- Increment the age of the sailor with *sid* = 74

Sailors	sid	sname	rating	age
	74			U._A+1

Restrictions on Update Commands

- Cannot mix I., D. and U. in a single example table, or combine them with P. or G.
- Cannot insert, update or modify tuples using values from fields of other tuples in the same table. Example of an update that violates this rule:

Sailors	sid	sname	rating	age
		john		_A
		joe		U._A+1

Should we update *every* Joe's age?
Which John's age should we use?

Find sailors who've reserved all boats (Again!)

- We want to find sailors _Id such that there is no boat _B that is not reserved by _Id:

Sailors	sid	sname	rating	age
	_Id	P._S		

Boats	bid	bname	color
¬	_B		

Reserves	sid	bid	day
¬	_Id	_B	

- Illegal query! Variable _B does not appear in a positive row. In what order should the two negative rows be considered? (Meaning changes!)

A Solution Using Views

- Find sailors who've not reserved some boat _B:

Sailors	sid	sname	rating	age
	_Id	P._S		

BadSids	sid
I.	_Id

Boats	bid	bname	color
	_B		

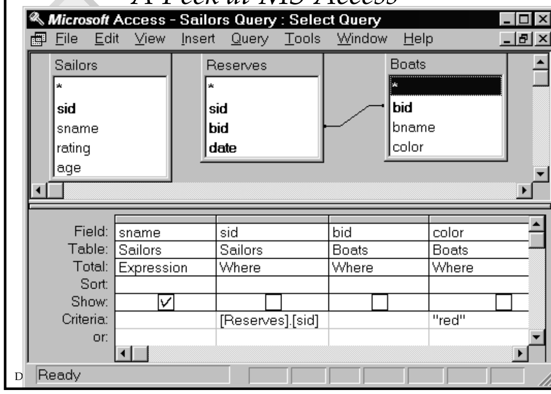
Reserves	sid	bid	day
¬	_Id	_B	

- Next, find sailors not in this `bad' set:

Sailors	sid	sname	rating	age
	_Id	P._S		

BadSids	sid
¬	_Id

A Peek at MS Access



Summary

- QBE is an elegant, user-friendly query language based on DRC.
- It is quite expressive (relationally complete, if the update features are taken into account).
- Simple queries are especially easy to write in QBE, and there is a minimum of syntax to learn.
- Has influenced the graphical query facilities offered in many products, including Borland's Paradox and Microsoft's Access.