WEEK 6

Query-by-Example (QBE)

QBE: Intro

- A "GUI" for expressing queries.
  - Based on the 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.

Some Features of QBE

- 2-D relation language.
- Display results.
- Fill up templates of relations for query formulation.
- Example of a possible answer is entered by the user.

`Example Tables' in QBE

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

SP	S#	P#	Qty

Parts	P#	Pname	color

Suppliers	S#	sname	City	status

4 types of areas

Table-name area	Column-name area
Row-operation area	Data entry area

Data Manipulation in QBE

- Retrieval
- Modification

- Simple retrieval.
  Prefix P. to indicate attribute to be retrieved.

P	P#	Color	Pname
		≠ Green	P._C

Query By Example

Ex: Find the P# that is Red.

(A variable that appears only once can be omitted)

P	P#	Color	Pname
	P.	Red	

Basics

- To print names and status of all suppliers:

Suppliers	S#	sname	City	status
		P._N		P._S

- Print all fields for suppliers with *status* > 10, in ascending order by (*status*, *city*):

Suppliers	S#	sname	city	status
P.			AO(2).	AO(1). > 10

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

$\{ \langle I,N,C,S \rangle \mid \langle I,N,C,S \rangle \in \text{Suppliers} \wedge S > 10 \}$

Query By Example

Ex: Give all the colors of the part 'Bolt'

P	P#	Pname	Color
		Bolt	P._L

Ex: Give all the available colors

			Color
			P._L

Data Manipulation in QBE

- Qualified (simple condition)
  - OR (conditions on different rows)
  - AND (condition on same row)

on the same row

>,≤ ,≥ ,= ,≠

And/Or Queries

- Names of suppliers with status < 30 *or* status > 20:

Supplier	S#	sname	city	status
		P.		< 30
		P.		> 20

- P# which do not come in green *or* red.

		Color	P#
		≠ Green	P._A
		≠ Red	P._A

And/Or Queries

- Names of suppliers with status < 30 in London:

Supplier	S#	sname	city	status
		P.	London	< 30

- Names of suppliers with status < 30 *and* status > 20:

Supplier	S#	sname	city	status
	_Id	P.		< 30
	_Id	P.		> 20

Duplicates

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

Supplier	S#	sname	city	status
UNQ		P.		< 30

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

Supplier	S#	sname	city	status
ALL	_Id	P.		< 30
	_Id	P.		> 20

Join Queries

- Names of suppliers who supply parts in quantity of 100 and have status higher than 25

Supplier	S#	sname	city	status
	_Id	P._N		> 25

SP	S#	P#	Qty
	_Id		100

- Joins accomplished by repeating variables.

Join Queries (Contd.)

- Colors of part "bolt" which is supplied by suppliers who supply them in quantity of 100 and have status more than 25 :

Suppliers	S#	sname	city	status
	_Id			> 25

SP	S#	P#	Qty
	_Id	_P	100

Parts	P#	Pname	color
	_P	'bolt'	P.

Join Queries (Contd.)

- Names and status of suppliers who supply some part that is also supplied by the supplier with *S#* = 'S3':

Supplier	S#	sname	city	status
	_Id	P.		P.

SP	S#	P#	Qty
	'S3'	_P	
	_Id	_P	

Use of 'ALL'

Ex: Give Sname who supply all parts.

S	S#	Sname
	_Id	P._N

SP	S#	P#
	_Id	ALL._P

Ex: Give P# which come in ALL colors

P	P#	Color
	P._P	ALL._L

Unnamed Columns

- 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!

Suppliers	S#	sname	City	status		
	_Id	P.		_S	P._B	P.(_S/10)

SP	S#	P#	Qty
	_Id		_B

"Negative Tables"

- Can place a negation marker in the relation column (kind of NOT EXISTS):

Suppliers	S#	sname	City	Status
	_Id	P._N		

Suppliers	S#	P#	Qty
¬	_Id		

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

"Negative Tables"

Ex: Give a supplier not located in London.

S	S#	city
¬	_Id	London
	P._Id	

"Negative Tables"

Ex: Get J# for projects supplied entirely by S1.

SPJ	S#	P#	J#
	S1		P._J
¬	≠S1		_J

"Negative Tables"

Ex: Give P# so that no order involving P# has quantity equal to or less than 50.

SP	S#	P#	Qty
¬		_P	<=50
		P._P	

Join and "Negative Tables"

Ex: Find S.name who do not supply red parts.

S	S#	Sname
	_Id	P._N

P	P#	Color
¬	_P	red

Red part does not exists

SP	S#	P#
¬	_Id	_P

Does not exist and matches, so print that S.sname.

≡ WHERE in SQL using join.

Find suppliers who supply all parts

- We want to find suppliers _Id such that there is no part _P that is not supplied by _Id:

Suppliers	S#	sname	City	status
	_Id	P._N		

Parts	P#	pname	color
¬	_P		

SP	S#	P#	Qty
¬	_Id	_P	

- Illegal query! Variable _P does not appear in a positive row. In what order should the two negative rows be considered? (Meaning changes!) (no nesting)

Aggregates

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

Suppliers	S#	sname	City	status	
			G.P.AO	_S	P.AVG._S

- 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 (SQL has a similar restriction).

QBE

SUM.ALL
COUNT.ALL (CNT.ALL)
AVG.ALL
DISTINCT.ALL
MIN.ALL
MAX.ALL

CNT.UNQ.ALL
SUM.UNQ.ALL
AVG.UNQ.ALL

Ex: Get the total number of suppliers

S	S#	Sname	Status	City
	P.CNT.ALL._Id			

Conditions Box

- Used to express conditions involving 2 or more columns, e.g., _S / 10 > 0.2.
- Can express a condition that involves a group, similar to the HAVING clause in SQL:

Suppliers	S#	sname	City	status
			G.P.	_S

CONDITIONS
AVG._S > 30

- Express conditions involving AND and OR:

Suppliers	S#	sname	city	status
		P.		_S

CONDITIONS
20 < _S AND _S < 30

Find suppliers who supply all parts

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

Suppliers	S#	sname	City	status
	P.G._Id			

SP	S#	P#	day
	_Id	_B1	

CONDITIONS
COUNT._B1=COUNT._B2

Parts	P#	pname	color
	_B2		

- How can we modify this query to print the names of suppliers who supply all parts?

Conditions Box

Ex: Find part names which come in three different colors

P	Pname	Color
	P._Id	_C1
	_Id	_C2
	_Id	_C3

Condition box: $_C1 \neq _C2 \neq _C3$

Conditions Box

Ex: P# which comes in ALL available colors

Ans 1:

P#	Color
P._P	ALL._L1

Ans 2:

P#	Color
	_L1
P._P	_L2

ALL._L2 = ALL._L1

all colors of _P = universal set

Conditions Box

Ex: Get all the J# which use those parts available from S1

No binding of J# with S1

S#	P#	J#
	_P1	P._J
'S1'	_P2	

ALL._P2 ≥ ALL._P1

Set of parts for S1

Inserting Tuples

- Single-tuple insertion:

Suppliers	S#	sname	City	status
I.	'S2'	Smith	'London'	10

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

Suppliers	S#	sname	City	status
I.	_Id	_N		_A

CONDITIONS
_A > 18 OR
_N LIKE 'C%'

Students	sid	name	login	age
	_Id	_N		_A

Delete and Update

- Delete all SP tuples for suppliers with city='London'

Suppliers	S#	sname	City	status
	_Id		='London'	

SP	S#	P#	Qty
D.	_Id		

- Increment the status of the supplier with *S#* = 'S4'

Suppliers	S#	sname	City	status
	'S4'			U._S+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:

Suppliers S#	sname	City	status
	Jones		_S
	Adams		U._S+1

Should we update *every* Adams' status?
Which Jones status should we use?

Find suppliers who supply all parts (Again!)

- We want to find suppliers _Id such that there is no part _P that is not supplied by _Id:

Suppliers	S#	sname	City	status
	_Id	P._N		

Parts	P#	pname	color
¬	_P		

SP	S#	P#	Qty
¬	_Id	_P	

- Illegal query

A Solution Using Views

- Find suppliers who do not supply some part _P:

Suppliers	S#	sname	City	status
	_Id	P._S		

BadS#s	S#
I.	_Id

Parts	P#	Pname	color
	_P		

SP	S#	P#	Qty
¬	_Id	_P	

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

Suppliers	S#	sname	City	status
	_Id	P._S		

BadS#s	S#
¬	_Id

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.