Chapter 5: Other Relational Languages

- Query-by-Example (QBE)
- Quel
- Datalog

Query-by-Example (QBE)

- Basic Structure
- Queries on One Relation
- Queries on Several Relations
- The Condition Box
- The Result Relation
- Ordering the Display of Tuples
- Aggregate Operations
- Modification of the Database

QBE — Basic Structure

- A graphical query language which is based (roughly) on the domain relational calculus
- Two dimensional syntax – system creates templates of relations that are requested by users
- Queries are expressed “by example”

Skeleton Tables

branch	*branch-name*	*branch-city*	*assets*

customer	*customer-name*	*customer-street*	*customer-city*

loan	*branch-name*	*loan-number*	*amount*

Skeleton Tables (Cont.)

borrower	*customer-name*	*loan-number*

account	*branch-name*	*account-number*	*balance*

depositor	*customer-name*	*account-number*

Queries on One Relation

- Find all loan numbers at the Perryridge branch.

loan	*branch-name*	*loan-number*	*amount*
	Perryridge	P._x	

  - _x is a variable (optional)
  - P. means print (display)
  - duplicates are removed

loan	*branch-name*	*loan-number*	*amount*
	Perryridge	P.ALL.	

  - duplicates are not removed

Queries on One Relation (Cont.)

- Display full details of all loans
  - Method 1:

loan	*branch-name*	*loan-number*	*amount*
	P._x	P._y	P._z

  - Method 2: shorthand notation

loan	*branch-name*	*loan-number*	*amount*
P.			

- Find the loan number of all loans with a loan amount of more than $700.

loan	*branch-name*	*loan-number*	*amount*
		P.	>700

Queries on One Relation (Cont.)

- Find the loan numbers of all loans made jointly to Smith and Jones.

borrower	*customer-name*	*loan-number*
	"Smith"	P._x
	"Jones"	_x

- Find the loan numbers of all loans made to Smith, Jones or both.

borrower	*customer-name*	*loan-number*
	"Smith"	P._x
	"Jones"	P._y

Queries on Several Relations

- Find the names of all customers who have a loan from the Perryridge branch.

loan	*branch-name*	*loan-number*	*amount*
	Perryridge	$\_x$	

borrower	*customer-name*	*loan-number*
	P.$\_y$	$\_x$

Queries on Several Relations (Cont.)

- Find the names of all customers who have both an account and a loan at the bank.

depositor	*customer-name*	*account-number*
	P._*x*	

borrower	*customer-name*	*loan-number*
	_*x*	

Queries on Several Relations (Cont.)

- Find the names of all customers who have an account at the bank, but do not have a loan from the bank.

depositor	*customer-name*	*account-number*
	P._x	

borrower	*customer-name*	*loan-number*
¬	_x	

¬ means “there does not exist”

Queries on Several Relations

- Find all customers who have at least two accounts.

depositor	*customer-name*	*account-number*
	P.$\_x$	$\_y$
	$\_x$	$\neg$ $\_y$

$\neg$ means “not equal to”

The Condition Box

- Allows the expression of constraints on domain variables that are either inconvenient or impossible to express within the skeleton tables.
- Find all account numbers with a balance between \$1,300 and \$2,000 but not exactly \$1,500.

account	*branch-name*	*account-number*	*balance*
		P.	$\_x$

conditions
$\_x = (\geq 1300$ **and** ≤ 2000 **and** $\neg$ $1500)$

The Result Relation

- Find the *customer-name*, *account-number*, and *balance* for all customers who have an account at the Perryridge branch.
  - We need to:
    - Join *depositor* and *account*.
    - Project *customer-name*, *account-number*, and *balance*.
  - To accomplish this we:
    - Create a skeleton table, called *result*, with attributes *customer-name*, *account-number*, and *balance*.
    - Write the query.

The Result Relation (Cont.)

- The resulting query is:

branch-name	*account-number*	*balance*
Perryridge	$\_y$	$\_z$

depositor	*customer-name*	*account-number*
	$\_x$	$\_y$

result	*customer-name*	*account-number*	*balance*
P.	$\_x$	$\_y$	$\_z$

Ordering the Display of Tuples

- AO = ascending order; DO = descending order.
  When sorting on multiple attributes, the sorting order is specified by including with each sort operator (AO or DO) an integer surrounded by parentheses.

- List all account numbers at the Perryridge branch in ascending alphabetic order with their respective account balances in descending order.

account	*branch-name*	*account-number*	*balance*
	Perryridge	P.AO(1).	P.DO(2).

Aggregate Operations

- The aggregate operators are AVG, MAX, MIN, SUM,and CNT
- The above operators must always be postfixed with “ALL.” (e.g., SUM.ALL.or AVG.ALL._x).
- Find the total balance of all the accounts maintained at the Perryridge branch.

account	*branch-name*	*account-number*	*balance*
	Perryridge		P.SUM.ALL.

Aggregate Operations (Cont.)

- Find the total number of customers having an account at the bank.

depositor	*customer-name*	*account-number*
	P.CNT.UNQ.ALL.	

Note: UNQ is used to specify that we want to eliminate duplicates.

Query Examples

- Find the average balance at each branch.

account	*branch-name*	*account-number*	*balance*
	P.G.		P.AVG.ALL._x

  Note:

  – The “G” in “P.G” is analogous to SQL’s **group by** construct

  – The “ALL” in the “P.AVG.ALL” entry in the *balance* column ensures that all balances are considered

- Find the average account balance at only those branches where the average account balance is more than \$1,200. Add the condition box:

conditions
AVG.ALL._x > 1200

Query Example

- Find all customers who have an account at all branches located in Brooklyn:

depositor	*customer-name*	*account-number*
	P.G._x	_y

account	*branch-name*	*account-number*	*balance*
	CNT.UNQ.ALL._z	_y	

branch	*branch-name*	*branch-city*	*assets*
	_z	Brooklyn	
	_w	Brooklyn	

Query Example (Cont.)

conditions
CNT.UNQ.ALL._z = CNT.UNQ.ALL._w

- CNT.UNQ.ALL._w specifies the number of distinct branches in Brooklyn.
- CNT.UNQ.ALL._z specifies the number of distinct branches in Brooklyn at which customer x has an account.

Modification of the Database – Deletion

- Deletion of tuples from a relation is expressed by use of a D. command. In the case where we delete information in only some of the columns, null values, specified by –, are inserted.
- Delete customer Smith

customer	*customer-name*	*customer-street*	*customer-city*
D.	Smith		

- Delete the *branch-city* value of the branch whose name is "Perryridge".

branch	*branch-name*	*branch-city*	*assets*
	Perryridge	D.	

Deletion Query Examples

- Delete all loans with a loan amount between \$1300 and \$1500.

loan	*branch-name*	*loan-number*	*amount*
D.		$\_y$	$\_x$

borrower	*customer-name*	*loan-number*
D.		$\_y$

conditions
$\_x = (\ \geq 1300$ **and** $\leq 1500)$

Deletion Query Examples (Cont.)

- Delete all accounts at branches located in Brooklyn.

account	*branch-name*	*account-number*	*balance*
D.	$\_x$	$\_y$	

depositor	*customer-name*	*account-number*
D.		$\_y$

branch	*branch-name*	*branch-city*	*assets*
	$\_x$	Brooklyn	

Modification of the Database – Insertion

- Insertion is done by placing the I. operator in the query expression.
- Insert the fact that account A-9732 at the Perryridge branch has a balance of $700.

account	*branch-name*	*account-number*	*balance*
I.	Perryridge	A-9732	700

- Provide as a gift for all loan customers of the Perryridge branch, a new $200 savings account for every loan account they have, with the loan number serving as the account number for the new savings account.

(next slide)

Modification of the Database – Insertion (Cont.)

account	*branch-name*	*account-number*	*balance*
I.	Perryridge	_x	200

depositor	*customer-name*	*account-number*
I.	_y	_x

loan	*branch-name*	*loan-number*	*amount*
	Perryridge	_x	

borrower	*customer-name*	*account-number*
	_y	_x

Modification of the Database – Updates

- Use the U. operator to change a value in a tuple without changing *all* values in the tuple. QBE does not allow users to update the primary key fields.
- Update the asset value of the of the Perryridge branch to $10,000,000.

branch	*branch-name*	*branch-city*	*assets*
	Perryridge		U.10000000

- Increase all balances by 5 percent.

account	*branch-name*	*account-number*	*balance*
U.			_x * 1.05
			_x