- RDB language
  - SQL
    - English Structured style
  - Relational algebra
    - operators
  - Relational Calculus

Chapter 6 Relational calculus

related to text book chapter7 (version 7)

related to text book chapter 8 (version 8)

Contents

- Introduction
- Tuple Calculus
- Examples
- Calculus VS Algebra
- Domain Calculus
- Other Calculus Languages

Introduction

- Based on a branch of mathematical logical called the predicate calculus.
- ALPHA, QUEL language

Introduction-cont.

- Range variable

  a fundamental feature of the calculus

  e.g. Range of sx is S

- Tuple calculus and Domain calculus

Tuple Calculus

- Syntax

<Range var definition>::= RANGEVAR <range var name>
RANGES OVER <relational exp commlist>

<range attr reference>::= <range var name>.<attr reference>
[AS <atrr name>]

<boolean exp>::= … all the usual possibilities, together with:
| <quantified boolean exp>

<quantified boolean exp>::= EXISTS <range var name>
(<boolean exp>)| FORALL <range var name>(boolean exp>)

<relational operation>::= <proto tuple>[WHERE <boolean exp>]

<proto tuple> ::= <tuple exp>

Tuple Calculus – cont.

- Range Variables

  Not variables in the usual programming language sense, they are variables in the sense of logical.

- e.g.

  RANGEVAR sx RANGES OVER S;

  RANGEVAR sy RANGES OVER S;

  RANGEVAR spx RANGES OVER SP;

  RANGEVAR spy RANGES OVER SP;

  RANGEVAR px RANGES OVER P;

Tuple Calculus – cont.

RANGEVAR su RANGES OVER

(sx WHERE sx.city = ‘London’),

(sx WHERE EXISTS

spx (spx.s# = sx.s# AND

spx.p# = p#(‘p1’)));

Tuple Calculus – cont.

- Quantifiers

  EXISTS v (P)

  FORALL v (P)

- Definition

  EXISTS v (P (v))

  false OR P(t1) OR … OR P(tm)

  FORALL v (P (v))

  true AND P(t1) AND … AND P(tm)

FORALL v (p) = NOT EXISTS v (NOT p)

Tuple Calculus – cont.

- e. g.

r(a, b, c)={ (1, 2, 3) (1, 2, 4) (1, 3, 4)}

EXISTS V (V.c>1) : *true*

EXISTS V (V.b>3) : *false*

EXISTS V (V.a>1 OR V.c=4) : *true*

FORALL V (V.a>1) : *false*

FORALL V (V.b>1) : *true*

FORALL V (V.a=1 AND V.c>2) : *true*

Examples

- Find supplier numbers and status for suppliers in Paris with status >20

(sx.s#, sx.status)
WHERE sx.city = ‘paris’ AND
sx.status>20

Examples-cont.

- Find all pairs of supplier numbers such that the two suppliers are colocated.

(sx.s# AS SA, sy.s# AS SB)

WHERE sx.city = sy.city

AND sx.s#<sy.s#

SX

S

S#	Sname	Status	City
S1	Smith	10	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Result

SA	SB
S1	S4
S2	S3

SY

S

S#	Sname	Status	City
S1	Smith	10	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Examples-cont.

- Find full supplier information for suppliers who supply part p2.

sx WHERE

EXISTS spx (spx.s#=sx.s# AND

spx.p#=p#('p2'))

S

sx

S#	Sname	Status	City
S1	Smith	10	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

spx

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Result

S#	Sname	Status	City
S1	Smith	10	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London

Examples-cont.

- Find supplier names for suppliers who supply at least one red part.

```
sx.sname   WHERE
           EXISTS spx (sx.s#=spx.s# AND
                 EXISTS px (px.p#=spx.p#
                        AND
                     px.color = color(‘red’)))
```

More Examples

- Find supplier names for suppliers who supply at least one part supplied by supplier s2.

sx.sname WHERE

EXISTS spx (EXISTS spy

(sx.s#=spx.s# AND

spx.p#=spy.p# AND

spy.s#=s#('s2')))

More Examples

- Find supplier names for suppliers who supply all parts.

```
sx.sname WHERE
    FORALL px ( EXISTS spx
                (spx.s# = sx.s# AND
                 spx.p# = px.p# ))
```

More Examples

- Find supplier names for suppliers who do not supply part p2.

```
sx.s#
 WHERE NOT EXISTS  spx
          (spx.s# =sx.s# AND
           spx.p# = p#(‘p2’))
```

More Examples

- Find supplier numbers for suppliers who supply at least all those parts supplied by supplier s2.

sx.s#   WHERE FORALL spx

(spx.s# ≠ s#(‘s2’)

OR EXISTS spy

(spy.s# = sx.s# AND

spy.p# = spx.p#))

$(\forall_{spx})$If spx.s#=‘s2’ then $(\exists_{spy})$ spy.s#=sx.s# and spy.p#=spx.p#

S

sx

S#	Sname	Status	City
S1	Smith	10	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

spy

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Result

S#
S1

More Examples

- Find part numbers for parts that either weight more than 16 pounds or are supplied by supplier s2, or both.

```
RANGEVAR pu RANGE OVER
      (px.p# WHERE
      px.weight>weight(16.0)),
   (spx.p# WHERE spx.s# = s#('s2'));
pu.p#
```

More Examples

OR equal to

px.p#

WHERE px.weight>weight(16.0)

OR EXISTS spx

(spx.p# = px.p# AND

spx.s# = s#('s2'))

Calculus VS Algebra

- Find supplier names for suppliers who supply at least one red part.

```
sx.sname   WHERE
    EXISTS spx (sx.s#=spx.s# AND
            EXISTS px (px.p#=spx.p# AND
                    px.color = color('red')))
```

$$\Pi_{sname}\ ((\sigma_{color=\text{"red"}}(P) \bowtie SP) \bowtie S))$$

Calculus VS Algebra – cont.

- Find supplier names for suppliers who supply all parts.

sx.sname WHERE
FORALL px (EXISTS spx
(spx.s# = sx.s# AND
spx.p# = px.p#))

$$\Pi_{sname}((\Pi_{s\#,p\#}(SP) \div \Pi_{p\#}(P)) \bowtie S)$$

- Find names and cities for suppliers who supply at least one Athenis project with at least 50 of every part.

(sx.sname,sx.city) WHERE
EXISTS Jx FORALL Px
EXISTS Spjx
(Jx.city = 'Athenis' AND
Jx.J# = Spjx.J# AND
Px.P# = Spjx.P# AND
Sx.S# = Spjx.S# AND
Spjx.QTY >= QTY(50))

The expression can be evaluated as follows:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Rome
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

J

J#	JNAME	CITY
J1	Sorter	Paris
J2	Display	Rome
J3	OCR	Athens
J4	Console	Athens
J5	RAID	London
J6	EDS	Oslo
J7	Tape	London

SPJ

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

- Step1: For each range variable, retrieve the range, restricted if possible. In the example, the set of tuples retrieved are as follows:

Sx:	all tuples of S	5 tuples
Px:	all tuples of P	6 tuples
Jx:	tuples of where city=‘Athens’	2 tuples
Spjx:	tuples of SPJ where QTY >= QTY(50)	24 tuples

- Step2: Construct the cartesian product of the range retrieved in Step 1, so yield:

S#	SN	STATUS	CITY	P#	PN	COLOR	WEIGHT	CITY	J#	JN	CITY	S#	P#	J#	QTY
S1	Sm	20	Lon	P1	Nt	Red	12.0	Lon	J3	OR	Ath	S1	P1	J1	200
S1	Sm	20	Lon	P1	Nt	Red	12.0	Lon	J3	OR	Ath	S1	P1	J4	700
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..

- Step 3: Restrict the Cartesian product constructed in step 2 in accordance with the “Join portion” of the WHERE clause. In the example, that portion is

  Jx.J#=Spjx.J# AND Px.P#=Spjx.P# AND Sx.S#=Spjx.S#

  the subset of the Cartesian product consisting of just 10 tuples:

S#	SN	STATUS	CITY	P#	PN	COLOR	WEIGHT	CITY	J#	JN	CITY	S#	P#	J#	QTY
S1	Sm	20	Lon	P1	Nt	Red	12.0	Lon	J4	Cn	Ath	S1	P1	J4	700
S2	Jo	10	Par	P3	Sc	Blue	17.0	Rom	J3	OR	Ath	S2	P3	J3	200
S2	Jo	10	Par	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath	S2	P3	J4	200
S4	Cl	20	Lon	P6	Cg	Red	19.0	Lon	J3	OR	Ath	S4	P6	J3	300
S5	Ad	30	Ath	P2	Bt	Green	17.0	Par	J4	Cn	Ath	S5	P2	J4	100
S5	Ad	30	Ath	P1	Nt	Red	12.0	Lon	J4	Cn	Ath	S5	P1	J4	100
S5	Ad	30	Ath	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath	S5	P3	J4	200
S5	Ad	30	Ath	P4	Sc	Red	14.0	Lon	J4	Cn	Ath	S5	P4	J4	800
S5	Ad	30	Ath	P5	Cm	Blue	12.0	Par	J4	Cn	Ath	S5	P5	J4	400
S5	Ad	30	Ath	P6	Cg	Red	19.0	Lon	J4	Cn	Ath	S5	P6	J4	500

- Step 4: Apply the quantifiers from right to left.
  - EXISTS Spjx   Project away the attributes of SPJ. Result:

S#	SN	STATUS	CITY	P#	PN	COLOR	WEIGHT	CITY	J#	JN	CITY
S1	Sm	20	Lon	P1	Nt	Red	12.0	Lon	J4	Cn	Ath
S2	Jo	10	Par	P3	Sc	Blue	17.0	Rom	J3	OR	Ath
S2	Jo	10	Par	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath
S4	Cl	20	Lon	P6	Cg	Red	19.0	Lon	J3	OR	Ath
S5	Ad	30	Ath	P2	Bt	Green	17.0	Par	J4	Cn	Ath
S5	Ad	30	Ath	P1	Nt	Red	12.0	Lon	J4	Cn	Ath
S5	Ad	30	Ath	P3	Sc	Blue	17.0	Rom	J4	Cn	Ath
S5	Ad	30	Ath	P4	Sc	Red	14.0	Lon	J4	Cn	Ath
S5	Ad	30	Ath	P5	Cm	Blue	12.0	Par	J4	Cn	Ath
S5	Ad	30	Ath	P6	Cg	Red	19.0	Lon	J4	Cn	Ath

– FORALL Px   Divide by P. Result:

S#	SNAME	STATUS	CITY	J#	JNAME	CITY
S5	Adams	30	Athens	J4	Console	Athens

– EXISTS Jx   Project away the attributes of J. Result:

S#	SNAME	STATUS	CITY
S5	Adams	30	Athens

- Step 5: Project the result of step 4 in accordance with the specifications in the proto tuple. In tne example, the proto tuple is (sx.sname, sx.city) Hence the final result:

SNAME	CITY
Adams	Athens

Computational Capability

- Find the part number and the weight in grams for each part with weight > 10000 grams.

(px.P#, px.weight*454 as GMWT)
WHERE
px.weight*454>weight(10000.0)

Computational Capability-cont.

- Find all suppliers and tag each one with the literal value “Supplier”

  (sx, ‘supplier’ AS Tag)

Computational Capability-cont.

- For each shipment, get full shipment details, including total shipment weight.

(spx, px.weight*spx.weight AS shipwt)
WHERE px.p# = spx.p#

Computational Capability-cont.

- For each part, get the number and total shipment quantity.

```
(px.p#, SUM(spx WHERE
                    spx.p#=px.p#, QTY)
                         AS totqty)
```

Computational Capability-cont.

- Find the total shipment quantity.

SUM(spx, QTY) AS Grandtotal

- For each supplier, get the supplier number and the total number of parts supplied.

(sx.s#, COUNT(spx WHERE spx.s#=sx.s#) AS #_of_parts)

Computational Capability-cont.

- Find part cities that store more than five red parts.

```
RangeVar   py  Over  P
px.city
   WHERE COUNT(py WHERE py.city=px.city
                AND  py.color = color('red') ) > 5
```

Domain calculus

- Range variables range over domain

RANGEVAR sx, sy RANGES OVER S#;

RANGEVAR px, py RANGES OVER P#;

RANGEVAR namex,namey RANGES OVER name;

RANGEVAR qtyx,qtyy RANGES OVER qty;

RANGEVAR cityx,cityy RANGES OVER char;

Domain calculus-cont.

- Membership condition

  R (pair, pair, …)

  – R is a relvar
  – each pair is of the form A:v
  – e.g. *SP* (s#:s#(‘s1’), p#:p#(‘p1’))

Domain calculus – cont.

- Examples
  - Find supplier numbers for suppliers in Paris with status >20

sx WHERE EXISTS statusx

(statusx > 20 AND

S (s# : sx, status : statusx, city :

‘Paris’))

Domain calculus – cont.

– Find all pairs of supplier numbers such that the two suppliers are colocated.

```
( sx AS sa, sy AS sb )
    WHERE  EXISTS cityz
        ( S (s# : sx, city : cityz ) AND
          S (s# : sy, city : cityz ) AND
              sx < sy )
```

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{t \mid \neg\, t \in r\}$ results in an infinite relation if the domain of any attribute of relation *r* is infinite

Safety of Expressions – cont.

- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression {*t* | *P*(*t*)} in the tuple relational calculus is *safe* if every component of *t* appears in one of the relations, tuples, or constants that appear in *P*

Other Relational Languages

- Query-by-Example (QBE)
- Datalog

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”

QBE Skeleton Tables for the Bank Example

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

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

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

Queries on One Relation

- Find all loan numbers at the Perryridge branch.

loan	*loan-number*	*branch-name*	*amount*
	P._*x*	Perryridge	

- _x is a variable (optional; can be omitted in above query)
- P. means print (display)
- duplicates are removed by default
- To retain duplicates use P.ALL

Queries on Several Relations

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

loan	*loan-number*	*branch-name*	*amount*
	_*x*	Perryridge	

borrower	*customer-name*	*loan-number*
	P._*y*	_*x*

Aggregate Operations

- The aggregate operators are AVG, MAX, MIN, SUM, and CNT
- The above operators must be postfixed with “ALL” (e.g., SUM.ALL.or AVG.ALL._x) to ensure that duplicates are not eliminated.

Aggregate Operations - Cont.

- E.g. Find the total balance of all the accounts maintained at the Perryridge branch.

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

Microsoft Access QBE

- Microsoft Access supports a variant of QBE called Graphical Query By Example (GQBE)
- GQBE differs from QBE in the following ways
  - Attributes of relations are listed vertically, one below the other, instead of horizontally

Microsoft Access QBE – cont.

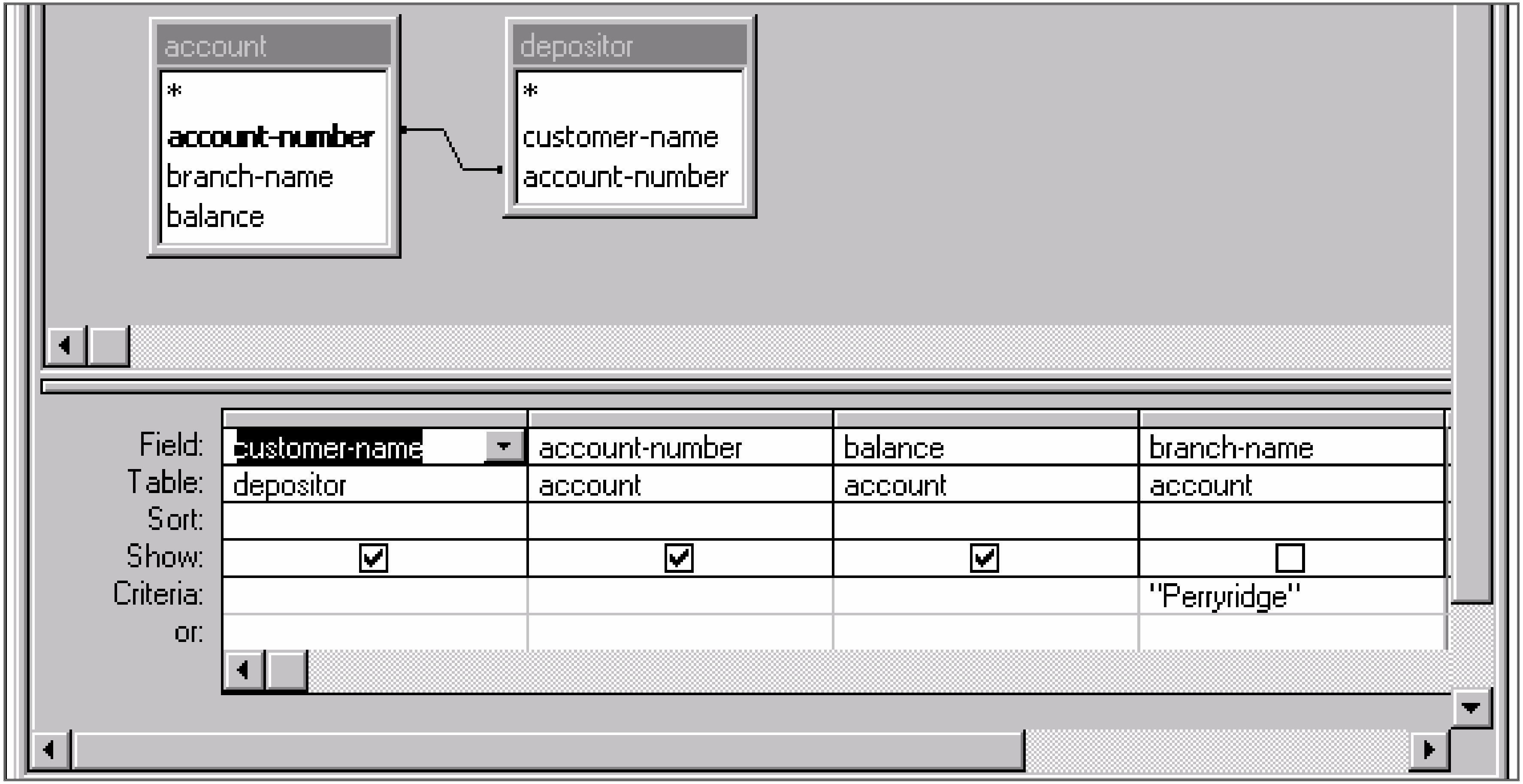
– Instead of using variables, lines (links) between attributes are used to specify that their values should be the same.
  - Links are added automatically on the basis of attribute name, and the user can then add or delete links
  - By default, a link specifies an inner join, but can be modified to specify outer joins.

Microsoft Access QBE – cont.

– Conditions, values to be printed, as well as group by attributes are all specified together in a box called the **design grid**

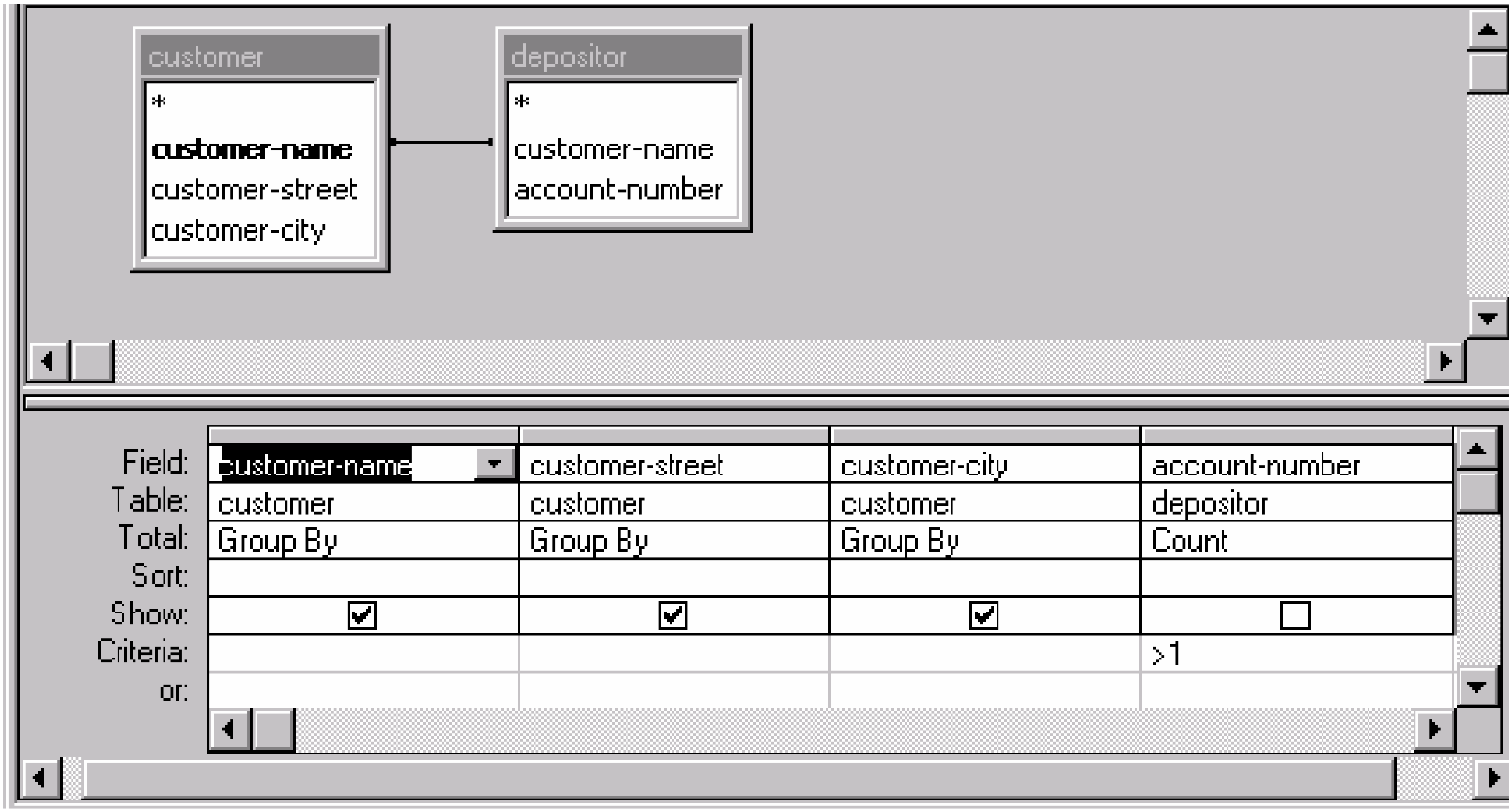
An Example Query in Microsoft Access QBE

- Example query: Find the *customer-name*, *account-number* and *balance* for all accounts at the Perryridge branch



An Aggregation Query in Access QBE

- Find the *name, street* and *city* of all customers who have more than one account at the bank



- exercises

  3,

  13 choose some exercises as chapter 5 asked.

- Next class

Integrity

text book chapter8 (version 7)

chapter 9 (version 8)