Relational Model

Chapter -3

Chapter 3: Relational Model

- Structure of Relational Databases
- Relational Algebra
- Modification of the Database
- Views

Example of a account Relation

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

The *branch* Relation

branch-name	*branch-city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

The *customer* Relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

The *depositor* Relation

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

The *loan* Relation

loan-number	*branch-name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

The *borrower* Relation

customer-name	*loan-number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Basic Structure

- Database consists of tables.
- Each table is contain a attribute.
- And each attribute has a set of permitted values called the domain of attribute.
- D1,D2.. D3 is attribute domain of table .
- D1 → set of acc_no , D2 → branch_name , D3→ balance
- Then any row of account table must consist of value(v1,v2,v3) where v1 $\in$ D1, v2 $\in$D2 ,v3 $\in$ D3

- Formally, given sets D_1, D_2, …. D_n a **relation** r is a subset of
  $D_1 \times D_2 \times \ldots \times D_n$
  Thus a relation is a set of n-tuples (a_1, a_2, …, a_n) where
  each $a_i \in D_i$
- Example: if

  customer-name = {Jones, Smith, Curry, Lindsay}
  customer-street = {Main, North, Park}
  customer-city = {Harrison, Rye, Pittsfield}

Then *r* = { (Jones, Main, Harrison),
(Smith, North, Rye),
(Lindsay, Park, Pittsfield)}

is a relation over *customer-name x customer-street x customer-city*

- If tuple variable t is refer to the first tuple of relation the notation t[acc_no] represent A101. same
- t[1] represent value of first attribute in the first tuple.

Attribute Types

- Attribute values are (normally) required to be **atomic**, that is, indivisible
  - (each attribute contain a one value for one tuple)
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations

Relation Schema

– $A_1, A_2, \ldots, A_n$ are *attributes*

– $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

E.g. *Customer-schema* =
(*customer-name, customer-street, customer-city*)

– $r(R)$ is a *relation* on the *relation schema* R

E.g. *customer (Customer-schema)*

Relation Instance

– The current values (*relation instance*) of a relation are specified by a table

– An element *t* of *r* is a *tuple*, represented by a *row* in a table

attributes (or columns)

customer-name	*customer-street*	customer-city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

tuples (or rows)

customer

Order of tuples is irrelevant (tuples may be stored in an arbitrary order)

Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  E.g.: *account* : stores information about accounts
- *depositor* : stores information about which customer owns which account
  customer : stores information about customers
- Storing all information as a single relation such as
  bank(*account-number, balance, customer-name*, ..)

- results in
  - repetition of information (e.g. two customers own an account)
- the need for null values (e.g. represent a customer without an account)

Relational Algebra

– Procedural language

– Six basic operators

- select
- project

Unary operation

- union
- set difference
- Cartesian product

binary operation

- Rename unary operation

– The operators take one or more relations as inputs and give a new relation as a result.

Select Operation

- It select tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- *p* is called the selection predicate
- Defined as:

  where *p* is a formula in propositional calculus consisting of terms connected by $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
  Each term is one of

  <attribute>*op* <attribute> or <constant>

  where *op* is one of: $=, \neq, >, \geq. <. \leq$
- Example of selection:
- Select the tuples from account which are from branch perryidge.

  $\sigma_{\textit{branch-name}=\text{“Perryridge”}}(\textit{account})$

- $\sigma_{\text{balance>500}}(\textit{account})$

Select Operation – Example

- Relation *r*

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Loan information having in branch perryridge account >1200

Result of $\sigma_{\textit{branch-name} = \text{"Perryridge"} \wedge \text{amount} > 1200}$ (*loan*)

loan-number	*branch-name*	*amount*
L-15	Perryridge	1500
L-16	Perryridge	1300

Project Operation

– Notation:

$$\Pi_{A1,\ A2,\ \ldots,\ Ak}\ (r)$$

where A_1, A_2 are attribute names and r is a relation name.

– The result is defined as the relation of k columns obtained by erasing the columns that are not listed

– Duplicate rows removed from result, since relations are sets

– E.g. To eliminate the *branch-name* attribute of *account*

$$\Pi_{\text{account-number, balance}}\ (\textit{account})$$

select account-number,balance from account;

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

find the branch name who have branch in city brooklyn

$$\Pi_{branch\text{-}name}(\sigma_{branch\text{-}city\,=\,\text{“Brooklyn”}}(branch))$$

$$\sigma_{branch\text{-}city\,=\,\text{“Brooklyn”}}(branch)$$

$$\Pi_{branch\text{-}name}(\sigma_{branch\text{-}city\,=\,\text{“Brooklyn”}}(branch)$$

branch-name
Brighton
Downtown

branch-name	*branch-city*	*asse*
Brighton	Brooklyn	7100
Downtown	Brooklyn	9000
~~Mianus~~	~~Horseneck~~	~~400~~
~~North Town~~	~~Rye~~	~~3700~~
~~Perryridge~~	~~Horseneck~~	~~1700~~
~~Pownal~~	~~Bennington~~	~~300~~
~~Redwood~~	~~Palo Alto~~	~~2100~~
[illegible]	[illegible]	[illegible]

- Display customername who lives in stampford

- Sigma $_{\text{customer-city=“stampford”(customer)}}$
- Pi $_{\text{customer-name}}$ (Sigma $_{\text{customer-city=“stampford”}}$(customer)

Union Operation

– Notation: $r \cup s$

– Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

– For $r \cup s$ to be valid.

1. *r, s* must have the *same arity* (same number of attributes)

2. The attribute domains must be *compatible* (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

Union Operation – Example

– Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

- Duplication is avoided(duplicate row are eliminate)
- E.g. to find all customers name with either an account or a loan

$$\Pi_{customer\text{-}name}\ (depositor) \ \cup \ \Pi_{customer\text{-}name}\ (borrower)$$

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

customer-name	*loan-number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Set Difference Operation

– It allow to find tuples that are in one relation but not in another.

– Notation $r – s$

– Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

that is tuples in r but not in s.

– Set differences must be taken between *compatible* relations.

- *r* and *s* must have the *same arity*
- attribute domains of *r* and *s* must be compatible

Set Difference Operation – Example

A	B
α	1
α	2
β	1

r

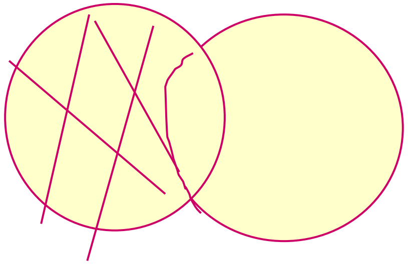
A	B
α	2
β	3

s

- Relations *r, s:*

r – s:

A	B
α	1
β	1



- Find the customer name who have account in bank but not taken any loan.

$$\Pi_{\text{customer-name}}(\text{depositor}) - \Pi_{\text{customer-name}}(\text{borrower})$$

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

customer-name	*loan-number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Customer-name
Johnson
Lindsay
Turner

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\text{-}name}\,(\sigma_{borrower.loan\text{-}number\,=\,loan.loan\text{-}number}(\sigma_{branch\text{-}name\,=\,\text{“Perryridge”}}(borrower \times loan)))\ -\ \Pi_{customer\text{-}name}(depositor)$$

Cartesian-Product Operation

- It allows to combine information from any two relation.
- Notation *r* x *s*
- Defined as:

$$r \times s = \{t\ q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of *r(R)* and *s(S*) are not disjoint, then renaming must be used.

Cartesian-Product Operation-Example

Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Composition of Operations

– Can build expressions using multiple operations

– Example:

– $\sigma_{A=C}(r \ x \ s)$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

r xs

– $\sigma_{A=C}(r \ x \ s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

– If the relation schema for r = borrower X loan then

– It conatinas all attributes of borrower and loan

– [borrwer.customer_name , borrower.loan_no, loan.loan_no, loan.branch_name, loan.amount]

E.G Find the name of all customer who have a loan at the perryridge branch

σ

custome
Ada
Curr
Hay
Jack
Jone
Smit
Smit
Will

loan-number	*branch-name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

- Assume that we have n1 tuples in borrower and n2 tuples in loan. Then, there are
- n1 $\ast$ n2 ways of choosing a pair of tuples—one tuple from each relation; so there are n1 $\ast$ n2 tuples in r
- In general, if we have relations r1(R1) and r2(R2), then r1 $\times$ r2 is a relation whose
- schema is the concatenation of R1 and R2.

- relation schema for r = borrower × loan, we construct a tuple of r out of each possible pair of tuples: one from the borrower relation and one from the loan relation

customer-name	borrower. loan-number	loan. loan-number	branch-name	amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
Hayes	L-15	L-11		900
Hayes	L-15	L-14		1500
Hayes	L-15	L-15		1500
Hayes	L-15	L-16		1300
Hayes	L-15	L-17		1000
Hayes	L-15	L-23		2000
Hayes	L-15	L-93		500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Smith	L-23	L-11	Round Hill	900
Smith	L-23	L-14	Downtown	1500
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Smith	L-23	L-17	Downtown	1000
Smith	L-23	L-23	Redwood	2000
Smith	L-23	L-93	Mianus	500
Williams	L-17	L-11	Round Hill	900
Williams	L-17	L-14	Downtown	1500
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300
Williams	L-17	L-17	Downtown	1000
Williams	L-17	L-23	Redwood	2000
Williams	L-17	L-93	Mianus	500

customer-name	borrower.loan-number	loan.loan-number	branch-name	amount
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500
Hayes	L-15	L-16	Perryridge	1300
Jackson	L-14	L-15	Perryridge	1500
Jackson	L-14	L-16	Perryridge	1300
Jones	L-17	L-15	Perryridge	1500
Jones	L-17	L-16	Perryridge	1300
Smith	L-11	L-15	Perryridge	1500
Smith	L-11	L-16	Perryridge	1300
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300

$\sigma_{branch\text{-}name = \text{"Perryridge"}}(borrower \times loan)$

Cartesian product with condition

– $(\sigma_{\text{borrower.}loan\_number=loan.loan\_number}(\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}(\text{borrowerXloan}))$ → will give all the attributes .

of cartesian product

– $\Pi_{customer\text{-}name}\,(\sigma_{\text{borrower.}loan\_number=loan.loan\_number}(\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}(\text{borrowerXloan}))$

customer-name
Adams
Hayes

E.G 2 id is primary key

In student table

Activity is primary key

In activity table

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Participants Table

ID*	Activity*
084	Tennis
084	Swimming
100	Squash
100	Swimming
182	Tennis
219	Golf
219	Swimming
219	Squash

Activities Table

Activity*	Cost
Golf	$47
Sailing	$50
Squash	$40
Swimming	$15
Tennis	$36

- Find the name students who has participate in activity swimming
  - (student X participant) show all the combination of records and fields it contains is
  - (student.student,student.ID,participant.activity,participant.id)

- (studentXparticipant)

Student.student	Student.ID	Participant.id	Participant.activity
John Smith	084	084	tennis
John Smith	084	084	swimming
John Smith	084	100	sqaush
John Smith	084	100	swimming
John Smith	084	182	tennis
John Smith	084	219	golf
John Smith	084	219	Swimming
John Smith	084	219	sqaush

- $\sigma_{\text{participant.activity="swimming"}}$ *(student x participant)*

student	studentID	Participant.id	Participant.activity
John smith	084	084	swimming
John smith	084	100	swimming
John smith	084	219	Swimming
Jane Blogge	100	084	swimming
Jane Blogge	100	100	swimming
Jane Blogge	100	219	Swimming
John smith	182	084	swimming
John smith	182	100	swimming
John smith	182	219	Swimming
Mark antony	219	084	swimming
Mark antony	219	100	swimming
Mark antony	219	219	Swimming

- $\sigma_{student.ID=participant.ID}(\sigma_{participant.activity="swimming"}(student\ x\ participant))$

student	studentID	Participant.id	Participant.activity
John smith	084	084	swimming
Jane Blogge	100	100	swimming
Mark antony	219	219	Swimming

- $\Pi_{student.student}(\sigma_{student.ID=participant.ID}(\sigma_{participant.activity="swimming"}(student\ x\ participant)))$

Rename Operation

– Allows us to give a name to the results of relational-algebra expressions.

– Allows us to refer to a relation by more than one name.

Notation: $\rho_X(E)$

– returns the expression *E* under the name *X*

– If a relational-algebra expression *E* has arity *n*, then $\rho_{X(A1, A2, \ldots, An)}(E)$

– returns the result of expression *E* under the name *X*, and with the

– attributes renamed to $A_1, A2, \ldots., An$.

- ρ_{stud_part} (studentxpraticipant)
- Stud_part is then name student x participant
- $\rho_{stud\_part(stud\_name,stud\_id,part\_id,part\_activity)}$ (stude
- ntxpraticipant)

E.G Find the largest account balance

1. Find the balance which are not largest
2. Take the difference between account and temporary relation created by first step

– Rename *account* relation as *d*

– The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}$$

$$(\sigma_{account.balance < d.balance}\ (account\ x\ \rho_d\ (account)))$$

– E.g find the name of all customer who live on the same street and in the same city as Smith

– Customer

City and street where the smith is living

$$Sresult \leftarrow \rho_{simth\text{-}add(street,city)}$$

$$(\pi_{customer\text{-}street,customer\text{-}city}(\sigma_{cusomter\text{-}name=\text{“smith”}}(customer))$$

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye

#

$$\sigma_{\text{customer.cusomter\_street=smith\_add.street ^ customer.customer\_city = smith\_add.city}} \text{(customer X sresult)}$$

$$\pi_{\text{customer\_street,customer\_city}}(\sigma_{\text{customer.cusomter\_name=smith\_add.street ^ customer.customer\_city = smith\_add.city}} \text{(customer X sresult)})$$