CSCC43 Tutorial #1

Relational Algebra

Andrew Leung

Administration

Email: andrewyk.leung@mail.utoronto.ca

Notes and Recordings: On Quercus

There will be no tutorials next week (holiday)!

Today’s material can be found on Quercus under the ‘Tutorial Week 2’ page

Schema 1

branch(branch_name, branch_city, assets)
customer(ID, customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(ID, loan_number)
account(account_number, branch_name, balance)
depositor(ID, account_number)

Consider the bank database above. Assume that branch names and customer IDs uniquely identify branches and customers, but loans and accounts can be associated with more than one customer.

branch(branch_name, branch_city, assets)
customer(ID, customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(ID, loan_number)
account(account_number, branch_name, balance)
depositor(ID, account_number)

Schema 1

1. What are the appropriate primary keys?

- branch: branch_name
- customer: ID
- loan: loan_number
- borrower: ID, loan_number
- account: account_number
- depositor: ID, account_number

Note: We allow customers to have more than one account and more than one loan. This is why both ID and loan_number are keys for the borrower to uniquely identify the borrower tuple. Similarly, both ID and account_number are keys for the depositor to uniquely identify depositor tuple.

Schema 1

branch(branch_name, branch_city, assets)
customer(ID, customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(ID, loan_number)
account(account_number, branch_name, balance)
depositor(ID, account_number)

2. Given your choice of primary keys, identify appropriate foreign keys.

- loan: branch_name referencing branch
- borrower: ID referencing customer, loan_number referencing loan
- account: branch_name referencing branch
- depositor: ID referencing customer, account_number referencing account

Schema 1

branch(branch_name, branch_city, assets)
customer(ID, customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(ID, loan_number)
account(account_number, branch_name, balance)
depositor(ID, account_number)

3. Give an expression in relational algebra to find each loan number with a loan amount greater than $10000.

$$\Pi_{loan\_number}(\sigma_{amount > 10000}(loan))$$

Schema 2

Consider the employee database of with the appropriate primary keys underlined

employee(emp_ID, person_name)

company(company_name, rank)

roster(emp_ID, company_name, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

Schema 2

employee(<u>emp_ID</u>, person_name)

company(<u>company_name</u>, rank)

roster(<u>emp_ID</u>, <u>company_name</u>, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

1. Find all the IDs of all employees with the name “Rahul” or with the name “Emma”

$$\Pi_{emp\_id}\left(\sigma_{person\_name = \text{`Rahul'} \,\vee\, person\_name = \text{`Emma'}}\left(employee\right)\right)$$

Schema 2

employee(emp_ID, person_name)

company(company_name, rank)

roster(emp_ID, company_name, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

2. Find the name of each employee who lives in city Miami

$$\Pi_{person\text{-}name}\left(\sigma_{city='Miami'}(roster) \bowtie employee\right)$$

Schema 2

employee(<u>emp_ID</u>, person_name)

company(<u>company_name</u>, rank)

roster(<u>emp_ID</u>, <u>company_name</u>, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

3. Find the name of each employee whose salary is greater than $100000.

$$\Pi_{person\_name}\left(\sigma_{salary > 100000}\left(roster \bowtie employee\right)\right)$$

Schema 2

employee(emp_ID, person_name)

company(company_name, rank)

roster(emp_ID, company_name, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

4. Find the ID and names of each employee who lives in Miami and whose salary is greater than $100000.

$$\Pi_{emp\_id, person\_name}\left(\sigma_{city = 'Miami' \wedge salary > 100000}\left(roster \bowtie employee\right)\right)$$

Schema 2

employee(<u>emp_ID</u>, person_name)

company(<u>company_name</u>, rank)

roster(<u>emp_ID</u>, <u>company_name</u>, salary, city)

roster[emp_ID] ⊆ employee[emp_ID]

roster[company_name] ⊆ company[company_name]

5. Find the names of companies that have a rank of at least 5 and are in Miami.

$$\Pi_{company\_name}\left(\sigma_{city = \text{'Miami'} \wedge rank \geq 5}\left(company \bowtie roster\right)\right)$$

Schema 3

Suppliers(<u>sID</u>, sName, address)
Parts(<u>pID</u>, pName, colour)
Catalog(<u>sID</u>, <u>pID</u>, price)

Catalog[sID] ⊆ Suppliers[sID]
Catalog[pID] ⊆ Parts[pID]

Solve the following queries using only select, project, cartesian product, and natural join.

Schema 3

Suppliers(<u>sID</u>, sName, address)

Parts(<u>pID</u>, pName, colour)

Catalog(<u>sID</u>, <u>pID</u>, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

1. If sID is a key for the Suppliers relation, could it be a key for the Catalog relation?

- No.
- Keys are relative to a particular relation, just because it is a key in one relation doesn't mean it is in one another.
- It is not a key for Catalog mainly because we want to be able to list multiple parts by one supplier in our catalog.

Schema 3

Suppliers(sID, sName, address)

Parts(pID, pName, colour)

Catalog(sID, pID, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

2. Find the names of all red parts.

$$\pi_{pName}\left(\sigma_{colour='red'}\left(Parts\right)\right)$$

Schema 3

Suppliers(<u>sID</u>, sName, address)

Parts(<u>pID</u>, pName, colour)

Catalog(<u>sID</u>, <u>pID</u>, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

3. Find the sIDs of all suppliers who supply a part that is red or green.

$$\pi_{sID}\left(\left(\sigma_{\substack{Colour='Red' \vee \\ Colour='Green'}} Parts\right) \bowtie Catalog\right)$$

Schema 3

Suppliers(sID, sName, address)

Parts(pID, pName, colour)

Catalog(sID, pID, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

4. Find all prices for parts that are red or green.

$$\Pi_{price}\left(\left(\sigma_{colour='red' \vee colour='green'} Parts\right) \bowtie Catalog\right)$$

Schema 3

Suppliers(sID, sName, address)

Parts(pID, pName, colour)

Catalog(sID, pID, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

5. Find the names of all suppliers who supply a part that is red or green.

$$\pi_{sName}\left(\left(\pi_{sID}\left(\left(\sigma_{colour='red' \vee colour='green'} Parts\right) \bowtie (Catalog)\right)\right) \bowtie Suppliers\right)$$

Schema 3

pID	colour
	red
	blue
	green
	yellow

Suppliers(<u>sID</u>, sName, address)

Parts(<u>pID</u>, pName, colour)

Catalog(<u>sID</u>, <u>pID</u>, price)

Catalog[sID] ⊆ Suppliers[sID]

Catalog[pID] ⊆ Parts[pID]

$\pi_{sID}(Catalog \bowtie ((\pi_{pID}(\sigma_{colour='red'}\ Part\_Colours)) \cap (\pi_{pID}(\sigma_{colour='green'}\ Part\_Colours))))$

6. Find the sIDs of all suppliers who supply a part that is red *and* green.

- It is not possible for a part to be red *and* green.
- Each tuple has only one colour, and each part has only one tuple (since pID is a key), so no part can be recorded as both red and green.

$\pi_{colour}(\sigma_{pID=1}(Parts \bowtie Part\_Colours))$

Part_Colours(<u>pID</u>, <u>colour</u>)

Part_Colours[pID] ⊆ Parts[pID]