CS 2451
Database Systems:
Relational Algebra & Relational Calculus

http://www.seas.gwu.edu/~bhagiweb/cs2541
Spring 2020
Instructor: Dr. Bhagi Narahari

1

Codd's Relational Algebra (RA)

- Data is stored as a set of relations
  - Relations implemented as tables
  - Tuple in a relation is a row in the table
  - Attribute (from domain) in relation is column in table
- RA = A set of mathematical operators that compose, modify, and combine tuples within different relations
- Relations are sets.
  - Mapping: maps elements in one set to another
- Relational algebra operations operate on relations and produce relations ("closure")

  f: Relation → Relation

  f: Relation x Relation → Relation

2

Preliminaries

- A query is applied to *relation instances*, and the result of a query is also a relation instance.
  - *Schemas* of input relations for a query are fixed (but query will run regardless of instance!)
  - The schema for the *result* of a given query is also fixed -- Determined by definition of query language constructs.
- **Notation:** Positional (R[0]) vs. named-field notation (R.name):
  - Positional notation easier for formal definitions, named-field notation more readable.

    We will use named field notation R.name
  - Both used in SQL

3

Relational Algebra

- ***query language*** used to update and retrieve data that is stored in a data model.
- ***Relational algebra*** is a set of relational operations for retrieving data.
  - Just like algebra with numbers, relational algebra consists of operands (which are relations) and a set of operators.
- Every relational operator takes as input one or more relations and produces a relation as output.
  - Closure property - input is relations, output is relations
  - Unary operations - operate on one relation
  - Binary operations - have two relations as input
- A sequence of relational algebra operators is called a ***relational algebra expression***.

4

Relational Algebra Operators

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* ($\times$) Allows us to combine two relations.
  - *Set-difference* ($-$) Tuples in relation 1, but not in relation 2.
  - *Union* ($\cup$) Tuples in relation. 1 or in relation. 2.
- Note: cross-product, set-difference, union are the set operations you have seen before
- Additional operations:
  - Intersection, *join*, assignment, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, *operations can be composed*! (Algebra is "closed".)

5

Relational Algebra Expression: Syntax

- RA operators operate on relations and produce relations – closed algebra
  - Defined recursively
- (Basis) basic expression consists of a relation in the schema or a constant relation
  - What is a constant relation ?
- (R) Let E_1 and E_2 be RA expressions, then

6

RA expressions..contd..

- ($E_1 \cup E_2$) is a RA expression
- ($E_1 - E_2$) is a RA expression
- ($E_1 \times E_2$) is a RA expression
- σ_P (E_1) is a RA expression

  Where P is a predicate (conditional statement)
- π_S(E_1) is a RA expression

  Where S is subset of the attributes in the schema
- ρ_R (E_1) is a RA expression
- Operations Can be composed
  - If R1, R2 are relations (sets), then R1 <op> R2 is also a relation (set)
  - Closed algebra – how is closure defined ??
- Operations are defined as Set operations
  - Input is a set, output is a set

    SQL allows duplicated, RA does not
- Above definition can be used to define syntax and construct a parser

7

Operator Precedence

- Just like mathematical operators, the relational operators have precedence.
- The precedence of operators from highest to lowest is:
  - unary operators - σ, Π, ρ
  - Cartesian product and joins - X, $\bowtie$, division
  - intersection
  - union and set difference
- Parentheses can be used to changed the order of operations.
- It is a good idea to ***always*** use parentheses around the argument for both unary and binary operators.

8

Data Instance for Mini-Banner Example

STUDENT

sid	name
1	Jill
2	Matt
3	Jack
4	Maury

Takes

sid	exp-grade	cid
1	A	550-0103
1	A	700-1003
3	A	700-1003
3	C	500-0103
4	C	500-0103

COURSE

cid	subj	sem
550-0103	DB	F03
700-1003	Math	S03
501-0103	Arch	F03

PROFESSOR

fid	name
1	Narahari
2	Youssef
8	Choi

Teaches

fid	cid
1	550-0103
2	700-1003
8	501-0103

9

Projection Operator

- We want to query the database and fetch only some column/attribute from the relation
- Example: We want student name only from students table

$$\Pi_{name}(Student)$$

10

Projection Operation Formal Definition

- Given a list of column names α and a relation R, π_α (R) extracts the columns in α from the relation.
- The projection operation on relation R with output attributes $A_1,\ldots,A_m$ is denoted by $\Pi_{A_1,\ldots,A_m}(R)$.

$$\Pi_{A_1,\ldots,A_m}(R)=\{t[A_1,\ldots, A_m] \mid t\in R\}$$

where

- R is a relation, t is a tuple variable
- $\{A_1,\ldots,A_m\}$ is a subset of the attributes of R over which the projection will be performed.
- Order of $A_1,\ldots, A_m$ is significant in the result.
- Cardinality of $\Pi_{A_1,\ldots,A_m}(R)$ is not necessarily the same as R because of duplicate removal.

11

Projection Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\Pi_{eno,ename}$(Emp)

eno	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	R. Davis
E8	J. Jones

12

Recap- Projection, Π_α

- Given a list of column names α and a relation R, π_α(R) extracts the columns in α from the relation. Output is relation…so removes duplicates!
- Example: find sid and grade from enrollment table

Takes

sid	exp-grade	cid
1	A	550-0103
1	A	700-1003
3	A	700-1003
3	C	500-0103
4	C	500-0103

$\Pi_{\text{sid,exp-grade}}$ (Takes)

sid	exp-grade
1	A
3	A
3	C
4	C

Note: duplicate elimination. In contrast, SQL returns by default a multiset and duplicates must be explicitly removed.

13

Selection Operation

- We want to query table and fetch only the rows that satisfy come condition
  - Example: Students whose grade = B
- The ***selection operation*** σ (sigma) is a unary operation that takes in a relation as input and returns a new relation as output that contains a subset of the tuples of the input relation.
  - output relation has the same number of columns as the input relation, but may have less rows.
- To determine which tuples are in the output, the selection operation has a specified condition, called a ***predicate***, that tuples must satisfy to be in the output.
  - The predicate is similar to a condition in an `if` statement.
- Selection σ_θ R takes a relation R and extracts those rows from it that satisfy the condition θ

14

Select Operation σ

- Notation: $\sigma_{predicate}(Relation)$
- Relation: can be table or result of another query
- Predicate:
  1. Simple predicate
     - Attribute1 = attribute2
     - Attribute = constant (also >, <, >=, <=, <>)
  2. Complex predicate
     - Predicate AND predicate
     - Predicate OR predicate
     - NOT predicate
- Idea: select rows from a Relation based on a predicate

15

Select Operation σ

- The selection operation on relation *R* with predicate *F* is denoted by $\sigma_F(R)$.

$$\sigma_F(R)=\{t \mid t \in R \text{ and } F(t) \text{ is true}\}$$

where
- *R* is a relation, *t* is a tuple variable
- *F* is a formula (predicate) consisting of
  operands that are constants or attributes
  comparison operators: $<, >, =, \neq, \leq, \geq$
  logical operators:AND, OR, NOT

- Similarity between formal definition of operators in RA and the syntax of Relational Calculus

16

Complex Predicate Conditions

- Conditions are built up from boolean-valued operations on the field names.
  - exp-grade <> “A”, name = “Jill”, STUDENT.sid=Takes.sid
- RA allows comparison predicate on attributes
  - =, not=, >, <, >=, <=
- Larger predicates can be formed using logical connectives – *or* (∨) and *and* (∧) and *not* (˥)
- Selection predicate can include comparison between attributes
- We don't lose any expressive power if we don't have complex predicates in the language, but they are convenient and useful in practice.

17

Selection Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\sigma_{title\ =\ 'EE'}(\text{Emp})$

eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

$\sigma_{salary\ >\ 35000\ \vee\ title\ =\ 'PR'}(\text{Emp})$

eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Logic operators: $\wedge$ AND, $\vee$ OR, $\neg$ NOT

18

Union ∪ (same as set union operation)

- If two relations have the same structure (Database terminology: are union-compatible. Programming language terminology: have the same type) we can perform set operations.
- `All persons who are students or faculty`

STUDENT

sid	name
1	Darby
2	Matt
3	Dan
4	Maury

FACULTY

fid	name
1	Darby
12	Youssef
18	Choi

STUDENT ∪ FACULTY

id	name
1	Darby
2	Matt
3	Dan
4	Marty
12	Youssef
18	Choi

19

Union Example

Emp

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

WorksOn

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) \cup \Pi_{eno}(\text{WorksOn})$

eno
E1
E2
E3
E4
E5
E6
E7
E8

20

Type Matching for Set operations

- Same number of attributes
- Same type of attributes
  - Each position must match domain

    Real systems sometimes allow sub-types: CHAR(2) and CHAR(20)

21

Set Difference

- ***Set difference*** is a binary operation that takes two relations *R* and *S* as input and produces an output relation that contains all the tuples of *R* that are not in *S*.
- General form:
- $R - S = \{t \mid t \in R \text{ and } t \notin S\}$
- where *R* and *S* are relations, *t* is a tuple variable.
- Note that:
  - R – S ≠ S – R
  - R and S must be union compatible.

22

Difference –

- Another set operator. Example:

STUDENT

sid	name
1	Sarah
2	Matt
3	Dan
4	Maury

FACULTY

sid	name
1	Sarah
12	Youssef
18	Choi

STUDENT – FACULTY

sid	name
2	Matt
3	Dan
4	Maury

23

Set Difference Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) - \Pi_{eno}(\text{WorksOn})$

eno
E4
E8

Question 1: What is the meaning of this query?

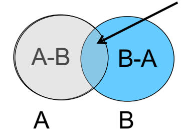
24

Intersection??

- People who are both Students and Faculty ??
- Do we need an Intersection operator?

Deriving Intersection using Union and Difference

Intersection: as with set operations, derivable from difference



$A \cap B$
$\equiv (A - (A\text{-}B)) = (B - (B\text{-}A))$

Operators that ‘combine’ relations

- Thus far, only operated on a single relation
- How to connect two relations ?
  - To find name of students taking a specific course with cid, we need to look at both students and Takes (enrolled) tables
- Operator(s) that produce a relation (set of tuples) after combining two different relations
- Set theory provides us with the cartesian product operator (between two sets; but can be applied to product of any number of sets – to get a k-tuple)

Product X

- “Join” is a generic term for a variety of operations that connect two relations. The basic operation is the cartesian product, R $\times$ S, *which concatenates every tuple in R with every tuple in S*. Example:

STUDENT

sid	name
1	Jill
2	Matt

SCHOOL

school
UPenn
GWU

STUDENT x SCHOOL

sid	name	school
1	Jill	UPenn
2	Matt	UPenn
1	Jill	GWU
2	Matt	GWU

Cartesian Product Example

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000

Emp × *Proj*

eno	ename	title	salary	pno	pname	budget
E1	J. Doe	EE	30000	P1	Instruments	150000
E2	M. Smith	SA	50000	P1	Instruments	150000
E3	A. Lee	ME	40000	P1	Instruments	150000
E4	J. Miller	PR	20000	P1	Instruments	150000
E1	J. Doe	EE	30000	P2	DB Develop	135000
E2	M. Smith	SA	50000	P2	DB Develop	135000
E3	A. Lee	ME	40000	P2	DB Develop	135000
E4	J. Miller	PR	20000	P2	DB Develop	135000
E1	J. Doe	EE	30000	P3	CAD/CAM	250000
E2	M. Smith	SA	50000	P3	CAD/CAM	250000
E3	A. Lee	ME	40000	P3	CAD/CAM	250000
E4	J. Miller	PR	20000	P3	CAD/CAM	250000

29

Product/Join

- Tuple in $R_1 \times R_2$ constructed by associating a tuple from R_1 with every tuple in R_2
- If R_1 has n_1 tuples and R_2 has n_2 tuples how many does $R_1 \times R_2$ have ?
- 'Same' attribute can appear in both tables ?
  - *This is the "link" between the two tables*
- Rather than write Product followed by Selection predicate, some RA versions give us **join** operators that perform multiple operations
  - Theta join: product followed by selection operator
    Equijoin when selection is an equality predicate
  - Natural Join…in addition, apply projection operator
  - Outer join, etc. etc.

30

Types of Joins

- The θ-Join is a general join in that it allows any expression in the condition *F*. However, there are more specialized joins that are frequently used.
- A ***equijoin*** is theta-join that only contains the equality operator (=) in formula *F*.
  - e.g. WorksOn $\bowtie_{\textit{WorksOn.pno = Proj.pno}}$ Proj
- A ***natural join*** over two relations *R* and *S* denoted by *R* ⋈ *S* is the equijoin of *R* and *S* over a set of attributes common to both *R* and *S*.
  - It removes the "extra copies" of the join attributes.
  - The attributes must have the same name in both relations.

31

(Theta) Join, $\bowtie_\theta$: A Combination of Product and Selection

- Example: Find students (id and name) and courses they took with grades and cid

$$\sigma_{\text{STUDENT.sid=Takes.sid}}\,(\text{STUDENT} \times \text{Takes}) =$$

$$(\text{STUDENT} \bowtie_{\text{STUDENT.sid=Takes.sid}} \text{Takes})$$

Join condition

sid:1	name	sid:2	exp-grade	cid
1	Jill	1	A	550-0103
1	Jill	1	A	700-1003
3	Alex	3	A	700-1003
3	Alex	3	C	500-0103
4	Maury	4	C	500-0103

32

θ -Join Example

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie_{dur*10000 > budget}$ Proj

eno	pno	resp	dur	P.pno	pname	budget
E2	P1	Analyst	24	P1	Instruments	150000
E2	P1	Analyst	24	P2	DB Develop	135000
E3	P4	Engineer	48	P1	Instruments	150000
E3	P4	Engineer	48	P2	DB Develop	135000
E3	P4	Engineer	48	P3	CAD/CAM	250000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P1	Instruments	150000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P1	Instruments	150000
E6	P4	Manager	48	P2	DB Develop	135000
E6	P4	Manager	48	P3	CAD/CAM	250000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P1	Instruments	150000
E7	P3	Engineer	36	P2	DB Develop	135000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P1	Instruments	150000
E7	P4	Engineer	23	P2	DB Develop	135000

33

Natural Joins

- Example: Find students (id and name) and courses they took with grades and cid

$\sigma_{STUDENT.sid=Takes.sid}$ (STUDENT x Takes) =

(STUDENT $\bowtie_{STUDENT.sid=Takes.sid}$ Takes)

sid:1 and sid:2 are duplicate information….
Do we need two columns?
Why not project only one of them ?

sid:1	name	sid:2	exp-grade	cid
1	Jill	1	A	550-0103
1	Jill	1	A	700-1003
3	Alex	3	A	700-1003
3	Alex	3	C	500-0103
4	Maury	4	C	500-0103

34

“Natural” Join, ⋈

- A common join to do is an equality join of two relations on commonly named fields, and to leave one copy of those fields in the resulting relation. Example:

STUDENT ⋈ Takes =

$(\Pi_{sid:1, name, exp\text{-}grade, cid}$ (STUDENT $\bowtie_{STUDENT.sid=Takes.sid}$ Takes))

sid	name	exp-grade	cid
1	Jill	A	550-0103
1	Jill	A	700-1003
3	Nick	A	700-1003
3	Nick	C	500-0103
4	Sina	F	500-0103

What if all the field names are the same in the two relations?
What if the field names are all disjoint?

35

Equijoin Example

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie_{WorksOn.pno = Proj.pno}$ Proj

eno	pno	resp	dur	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000

36

Natural join Example

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie$ Proj

eno	pno	resp	dur	pname	budget
E1	P1	Manager	12	Instruments	150000
E2	P1	Analyst	24	Instruments	150000
E2	P2	Analyst	6	DB Develop	135000
E3	P4	Engineer	48	Maintenance	310000
E5	P2	Manager	24	DB Develop	135000
E6	P4	Manager	48	Maintenance	310000
E7	P3	Engineer	36	CAD/CAM	250000
E7	P4	Engineer	23	Maintenance	310000

Natural join is performed by comparing *pno* in both relations.

37

Combining Operations….assignment operator

- Relational algebra operations can be combined in one expression by nesting them:

$$\Pi_{eno,pno,dur}(\sigma_{ename='J.\ Doe'}(Emp) \bowtie \sigma_{dur>16}(WorksOn))$$

  - Return the eno, pno, and duration for employee 'J. Doe' when he has worked on a project for more than 16 months.
- Operations also can be combined by using temporary relation variables to hold intermediate results.
  - We will use the **assignment operator** $\leftarrow$ for indicating that the result of an operation is assigned to a temporary relation.

$$\text{empdoe} \leftarrow \sigma_{ename='J.\ Doe'}(\text{Emp})$$
$$\text{wodur} \leftarrow \sigma_{dur>16}(\text{WorksOn})$$
$$\text{empwo} \leftarrow \text{empdoe} \bowtie \text{wodur}$$
$$\text{result} \leftarrow \Pi_{eno,pno,dur}(\text{empwo})$$

38

Referencing same relation twice..

- We want to find the student IDs of students who have the same name
- Ambiguity in the statement
  - Student.name = student.name ??
- easiest approach is to create a copy of Student with a different name

39

and lastly….
Rename Operator , $\rho_\alpha(R)$

- The rename operator can be expressed several ways:
  - One definition:

ρ_α(x) Takes the relation x and returns a copy of the relation with the name α
General Def: can rename attribute list with new names β

  - Rename isn't all that useful, except if you join a relation with itself
  - **ρ_{Person} (STUDENT) =** copy of STUDENT with table name Person
  - **Find pairs of student IDs who have the same name:**

$$\Pi_{\text{STUDENT.sid, Person.sid}}(\text{STUDENT} \bowtie_{\text{STUDENT.name=Person.name}} (\rho_{\text{Person}}(\text{STUDENT})))$$

40

Rename Operator

- General definition allows renaming specific attributes
  - $\rho_{X(C,D)}$ **(R (A,B))**

    Relation R renamed to X

    Fields A,B in R are now renamed to C,D in X

  - $\rho_{Person(idnum,\ who)}$ **(STUDENT (sid, name))**
- Find pairs of student IDs who have the same name: ?
- $\Pi_{\text{Student.sid, Person.idnum}}$ (Student $\bowtie_{\text{Student.name=Person.who}}$ ($\rho_{\text{Person(idnum,who)}}$ (Student))
- Note: not necessary to rename the attributes …below will also work:
- $\Pi_{\text{Student.sid, Person.sid}}$ (Student $\bowtie_{\text{Student.name=Person.sid}}$ (ρ_{Person} (Student))

41

Rename Operation

- Renaming can be applied when assigning a result:

$$\text{result(EmployeeNum, ProjectNum, Duration)} \leftarrow \Pi_{eno,pno,dur}(empwo)$$

- Or by using the rename operator ρ (rho):

$$\rho_{\text{result(EmployeeName, ProjectNum, Duration)}}(empwo)$$

42

Completeness of Relational Algebra Operators

- It has been shown that the relational operators $\{\sigma, \Pi, \times, \cup, -\}$ form a complete set of operators.
  - That is, any of the other operators can be derived from a combination of these 5 basic operators.
- Examples:
  - Intersection - $R \cap S \equiv R \cup S - ((R - S) \cup (S - R))$
  - We have also seen how a join is a combination of a Cartesian product followed by a selection.

43

Other Relational Algebra Operators

- There are other relational algebra operators that we will not discuss. Most notably, we often need ***aggregate operations*** that compute functions on the data.
- For example, given the current operators, we cannot answer the query:
  - What is the total number of students enrolled in a course
  - What are the total number of employees in department 5 ?
- We will see how to answer these queries when we study SQL.

44

How to write a RA query ?

- Find out which tables you need to access
  - Compute $\times$ of these tables
- What are the conditions/predicates you need to apply ?
  - Determines what select σ operators you need to apply
- What attributes/columns are needed in result
  - Determines what project π operators you need

Project (Select (Product))

45

Modifying the Database

- Need to insert, delete, update tuples in the database
- What is insert ?
  - Add a new tuple to existing set = Union
- What is delete ?
  - Remove a tuple from existing set = Set difference
- How to update attribute to new value ?
  - Need new operator: δ

46

Schema of Bank DB

- Customer (<u>CustID</u>, Name, street,city,zip)
  - Customer ID, Name, and Address info: street, city, zip
- Deposit (Cust<u>ID, Acct-num</u>, balance,Branch-name)
  - Customer ID, Account number, Balance in account, name of branch where account is held
- Loan (Cust<u>ID, Loan-num</u>, Amount, Branch-name)
  - Customer ID, loan number, amount of loan…
- Branch (<u>Branch-name</u>, assets, Branch-city)

47

Modifying Database

- Delete all accounts of Customer with CustID=3333
  - Deposit $\leftarrow$ Deposit – (tuples of CustID 3333)
- Insert tuple (4444, Downtown, 1000,1234)
  - Deposit $\leftarrow$ Deposit $\cup$ (4444,Downtown,1000,1234)
- Update: $\delta_{A \leftarrow E}(R)$
  - Update attribute A to E for tuples in relation R
  - $\delta_{balance \leftarrow 1.05*balance}$(Deposit) : updates balances
  - Can also specify selection condition on Deposit
    Update balances only for customers with CustID=1234

48