Relational Algebra

Instructor: Wang-Chiew Tan

Reference:
A First Course in Database Systems,
3rd edition, Chapter 2.4 – 2.6

What is a Data Model?

- A *data model* is a mathematical formalism that consists of two parts:
  1. A notation for describing data and mathematical objects for representing data.
  2. A set of operations for manipulating data.

- The relational data model
- What is the associated query language for the relational data model?

Two different query languages

- Codd proposed two different query languages for the relational data model.
  - Relational Algebra
    - Queries are expressed as a sequence of operations on relations.
    - Procedural language.
  - Relational Calculus
    - Queries are expressed as formulas of first-order logic.
    - Declarative language.
- **Codd's theorem**: The Relational Algebra query language has the same *expressive power* as the Relational Calculus query language.

Procedural vs. Declarative Languages

- Procedural program:
  - The program is specified as a sequence of operations to obtain the desired the outcome. I.e., *how* the outcome is to be obtained.
  - E.g., Java, C, …
- Declarative program:
  - The program specifies *what* is the expected outcome, and not *how* the outcome is to be obtained.
  - E.g., Scheme, Ocaml, …

SQL – Structured Query Language

- Is SQL a procedural or a declarative language?

- SQL – pronounced as "sequel"
- Principle language used to describe and manipulate data stored in relational database systems.
- Neither relational algebra nor relational calculus proposed by Codd.

Desiderata of a good database query language

1. Physical database independence.
   – One should be able to formulate a query without understanding the mechanics of the physical layer.
2. Highly expressive.
   – One should be able to formulate interesting queries with the language.
3. Can be efficiently executed.
   – One should be able to compute the answers to the query fast.

- Physical data independence is achieved by most query languages today.
- Increased expressiveness often comes at the expense of lower performance.

Relation Algebra

- Relational Algebra: a query language for manipulating data in the relational data model.
- Not used directly as a query language.
- Internally, an SQL query is translated into a relational algebra expression.
  - Manipulation, analysis, and optimization are performed based on the relational algebra expression.

Relation Algebra Operators

- Queries in relational algebra are composed using basic operations or functions.
  - Selection (σ)
  - Projection (π)
  - Set-theoretic operations:
    - Union (∪)
    - Set-difference (-)
    - Cross-product (x)
    - Intersection (∩)
  - Renaming (ρ)
  - Natural Join (⋈), theta-join ($\bowtie_\theta$)
  - Division (/)

Compositionality

- Each operator is either a unary or binary operator.
- A complex expression is built from basic ones.
- We assume set semantics for relational algebra queries. Duplicates are always removed.

Selection: $\sigma_{condition}(R)$

- Unary operation
  - Input: A relation with schema $R(A_1, \ldots, A_n)$.
  - Output: A relation with attributes $A_1, \ldots, A_n$.
  - Meaning: Takes a relation R and extracts only rows from R that satisfy the *condition*.
  - Condition is a logical conjunction (using AND, OR, NOT) of atomic expressions of the form:

    <expr> <op> <expr>

    where <expr> is an attribute name, a constant, a string, and op is one of (=,≥, ≥, <, >, <>)
    - E.g., “age > 20 OR height < 6”,
    - “name LIKE “Anne%” AND salary > 200000”
    - NOT (age > 20 AND salary < 100000)
- Output is always a set (no duplicates).

Example

- $\sigma_{\text{rating} > 6}$ (Hotels)

Hotels

name	address	rating	capacity
~~Windsor~~	~~54th ave~~	~~6.0~~	~~135~~
Astoria	5th ave	8.0	231
BestInn	45th st	6.7	28
~~ELogde~~	~~39 W st~~	~~5.6~~	~~45~~
~~ELodge~~	~~2nd E st~~	~~6.0~~	~~40~~

name	address	rating	capacity
Astoria	5th ave	8.0	231
BestInn	45th st	6.7	28

Example

- $\sigma_{\text{rating} > 6 \text{ AND capacity} > 50}$ (Hotel)

name	address	rating	capacity
Windsor	54th ave	6.0	135
Astoria	5th ave	8.0	231
BestInn	45th st	6.7	28
ELogde	39 W st	5.6	45
ELodge	2nd E st	6.0	40

- Is $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C1 \text{ AND } C2}(R)$?
- Prove or give a counter-example.

- Is $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R))$?
- Prove or give a counter-example.

name	address	rating	capacity
Astoria	5th ave	8.0	231

Projection: $\pi_{<attribute\ list>}(R)$

- Unary operation
  - Input : A relation with schema $R(A_1, \ldots, A_n)$.
  - Output: relation has attributes according to *attribute list.*
  - Meaning: For every tuple in relation R, output only the attributes stated in *attribute list.*
- Eliminate duplicates.

Example

- $\pi_{name,\ address}$ (Hotels)

name	address
Windsor	54^{th} ave
Astoria	5^{th} ave
BestInn	45^{th} st
ELogde	39 W st
ELodge	2nd E st

- Suppose name and address form the key of Hotels relation, is the cardinality of the output relation the same as the cardinality of Hotels? Why?

Example

- π_{name} (Hotel)

name
Windsor
Astoria
BestInn
ELodge

- Note that there are no duplicates.

Set Union: R ∪ S

- Binary operator.
  - Input: two relations R and S which must be union-compatible.
    - They have the same set of arity, i.e., same number of columns.
    - The ith column of R has the same type as the ith column of S, for every column i.
    - Note that field names are not used in defining union-compatibility though we can also think that R and S is union-compatible if they having the same type (a set of record type).
  - Output: a relation that has the same type as R (or S).
  - Meaning: the output consists of the set of all tuples in R and S.

Example

- Dell_Desktops ∪ IBM_Desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

All tuples in R occurs in R ∪ S.
All tuples in S occurs in R ∪ S.
R ∪ S contains tuples that either occur in R or S (or both).

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux
30G	1.2Ghz	Windows

Example

- Dell_Desktops ∪ IBM_Desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

R ∪ S = S ∪ R (commutativity)
R ∪ (S ∪ T) = (R ∪ S) ∪ T (associativity)

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux
30G	1.2Ghz	Windows

Set Difference: R - S

- Binary operator.
  - Input: two relations R and S which must be union-compatible.
  - Output: a relation with the same type as R (or S).
  - Meaning: output consists of all tuples in R and not in S.

Example

- Dell_Desktops - IBM_Desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

Dell_Desktops – IBM_Desktops

Harddisk	Speed	OS
30G	1.0Ghz	Windows
20G	750Mhz	Linux

Example

- IBM_Desktops – Dell_Desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

IBM_Desktops – Dell_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows

Is it commutative?
Is it associative?

Product: R x S

- Binary operator.
  - Input: two relations R and S. R is a relation schema: R(A_1, …, A_{k1}), S(B_1, …, B_{k2}).
  - Output: a relation of arity $k_1 + k_2$.
  - Meaning:

    R x S = { (a_1, …, a_{k1}, b_1, …, b_{k2}) | (a_1, …, a_{k1}) ∈ R and (b_1, …, b_{k2}) ∈ S) }.

Example

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

S

D	E
d_1	e_1
d_2	e_2
d_3	e_3

R x S

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_1	b_1	c_1	d_3	e_3
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2
a_2	b_2	c_2	d_3	e_3

Is it commutative?
Is it associative?
Is it distributive? I.e,
Is $Rx(S \cup T) = (RxS) \cup (RxT)$?

Example

- What happens if R and S contain common attributes? e.g., R(A,B,C) and S(A,E).

A.1	B	C	A.2	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_1	b_1	c_1	d_3	e_3
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2
a_2	b_2	c_2	d_3	e_3

Derived Operators

- So far, we have learnt:
  - Selection
  - Projection
  - Union
  - Difference
  - Cross Product
- Some operators can be derived by composing the operators we have learnt so far:
  - Set Intersection
  - Natural Join, Theta Join, Semi join
  - Quotient

Set Intersection: R∩S

- Find all desktops sold by Dell and IBM.
- Dell_desktops∩IBM_desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

Harddisk	Speed	OS
20G	500Mhz	Windows

Theta-Join: R $\bowtie_\theta$ S

- Binary operator.
  - Input: $R(A_1, \ldots, A_m)$, $S(B_1, \ldots, B_n)$.
  - Output: A relation that consists of all attributes $A_1, \ldots, A_m$ and all attributes $B_1, \ldots, B_n$. Identical attribtues in R and S are disambiguated with the relation names.
  - Meaning: $\sigma_\theta(R \times S)$.
- The θ-Join selects those tuples from R x S that satisfy the condition θ.
  - Compute R x S. Keep only those tuples in R x S that satisfy θ.
- If θ always evaluates to true, then $\sigma_\theta(R \times S) = R \times S$.

Example

Enrollment(esid, ecid, grade)

Course(cid, cname, instructor-name)

- Find the ids of all students who obtained grade A in Physics.
- Enrollment $\bowtie_{\text{ecid = cid AND grade='A' AND cname='Physics'}}$ Course
  - $\sigma_{\text{ecid = cid AND grade='A' AND cname='Physics'}}$ (Enrollment x Course)
- π_{esid}(Enrollment $\bowtie_{\text{ecid = cid AND grade='A' AND cname='Physics'}}$ Course)
  - $\pi_{\text{esid}}(\sigma_{\text{ecid = cid AND grade='A' AND cname='Physics}}$(Enrollment x Course))

Natural Join: R⋈S

- Very often, a query over two relations can be formulated with the natural join.
- Binary operator.
  - Input: Two relations R and S where { A_1, …, A_k } is the set of common attributes between R and S.
  - Output: A relation where its attributes are attr(R) U attr(S). In other words, the attributes consists of the attributes in R x S – {A_1, …, A_k}.

  - Meaning:
    $R \bowtie S = \pi_{(attr(R) \cup attr(S))}(\sigma_{R.A1=S.A1 \text{ AND } R.A2 = S.A2 \text{ AND } \ldots \text{ AND } R.Ak=S.Ak}(R \times S))$.

  - Compute R x S.
  - Keep only those tuples in R x S where R.A1=S.A1 AND R.A2 = S.A2 AND … AND R.Ak=S.Ak.
  - Project on the attributes of R U S for the resulting set of tuples.

Example

Enrollment(esid, cid, grade)

Course(cid, cname, instructor-name)

cid is the common attribute between the two relations.

- Want: Course-grade(esid, cid, cname, grade).
- $\pi_{(esid, cid, cname, grade)}$(Enrollment ⋈ Course).

- What happens when R and S have no common attributes?
- What happens when R and S have only common attributes?

Semi join: R ⋉ S

- Meaning: R ⋉ S = $\pi_{attr(R)}(R \bowtie S)$

- Find all courses with some enrollments: Course ⋉ Enrollment
- Find all faculty who is advising at least one student: Faculty ⋉ Student

Quotient or Division: R / S or R ÷ S

- Input: Two relations R and S where $attr(S) \subset attr(R)$ and attr(S) is non-empty.
- Output: a relation where its attributes are attr(R) – attr(S).

- Example: R(A,B,C,D), S(B,D).
- Meaning: $R/S = \{ (a, c) \mid \text{for all } (b,d) \in S, \text{ we have } (a,b,c,d) \in R \}$

- The quotient (or division) R / S is the relation consisting of all tuples $(a_1, \ldots, a_{r-s})$ such that for every tuple $(b_1, \ldots, b_s)$ in S, we have that $(a_1, \ldots, a_{r-s}, b_1, \ldots, b_s)$ is in R.

Example

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b1	c1
a1	b3	c3
a4	b2	c2
a3	b2	c2
a4	b1	c1

S

B	C
b1	c1
b2	c2

R / S

A
a1
a4

Example

Enrollment(esid, cid, grade)
Course(cid, cname, instructor-name)

- Find the esids of all students who are enrolled in all courses.

  Enrollment / π_{cid}(Course)

- Find the esids of all students who are enrolled in all courses taught by "Ullman".

  Enrollment / π_{cid} ($\sigma_{\text{instructor-name='Ullman'}}$ (Course))

Quotient (or Division) (cont'd)

- How can we express R/S with basic operators (select, project, cross product, union, difference) ?
- (to fill in)

More complex queries

- Relational operators can be composed to form more complex queries. We have already seen examples of this.

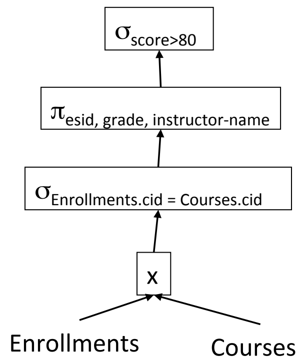
Enrollments(esid, cid, score)

Courses(cid, cname, instructor-name)

- Find student and instructor pairs where the student scored well (more than 80 pts) in a course taught by the instructor.
- $\sigma_{score>80}$ ($\pi_{esid,\ grade,\ instructor\text{-}name}$ (
  $\sigma_{Enrollments.cid\ =\ Courses.cid}$ (Enrollments x Courses)))

More complex queries

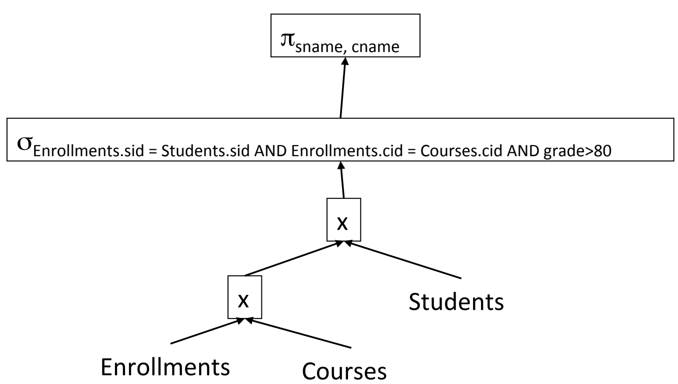
- Query tree or Operator tree



Tells us exactly the procedure to take in order to arrive at the answer.
Also known as execution plan.

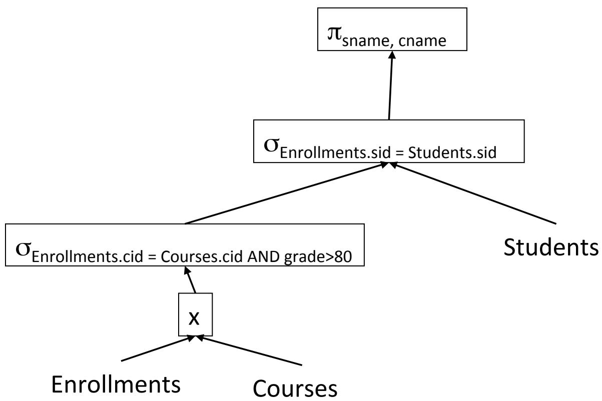
Another example

- Find the student and course names where the student scored well (more than 80 pts) in the course.



An alternative execution plan

- Find the student and course names where the student scored well (more than 80 pts) in the course.



$\sigma_{C1\ AND\ C2\ AND\ C3}$ (E x C x S) =
σ_{C1} ($\sigma_{C2\ AND\ C3}$ (E x C x S)) =
σ_{C1} ($\sigma_{C2\ AND\ C3}$ (E x C) x S)

Comparing execution plans

- Which is a better execution plan?
  - Intuitively, the second plan is better because it filters tuples before the cross product much more aggressively than the first plan.
  - Smaller intermediate tuples to manipulate.
  - An even “better” plan is to push the selection condition “grade > 80” all the way to the Enrollment relation.

Renaming: $\rho_{S(A1, \ldots, An)}$ (R)

- To specify the attributes of a relational expression.
- Input: a relation, a relation symbol R, and a set of attributes {B1, … ,Bn}
- Output: the same relation with name S and attributes A1, …, An.

- Meaning: rename relation R to S with attributes A1, …, An.

Example

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

S

C	D
d_1	e_1
d_2	e_2
d_3	e_3

R x $\rho_{T(X,D)}$ S

A	B	C	X	D
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_1	b_1	c_1	d_3	e_3
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2
a_2	b_2	c_2	d_3	e_3

Independence of Basic Operators

- Many interesting queries can be expressed using the five basic operators (σ , π , x , U , -).
- Can one of the five operators be derived by the other four operators?

Theorem (Codd)

The five basic operators are independent of each other. In other words, for each relational operator *o*, there is no relational algebra expression that is built from the rest that defines *o*.

Practice Homework 5

Sailors(sid, sname, rating, age) // sailor id, sailor name, rating, age
Boats(bid, bname, color) // boat id, boat name, color of boat
Reserves(sid, bid, day) // sailor id, boat id, date that sid reserved bid.

1. Find the names of sailors who reserved boat 103.
2. Find the colors of boats reserved by Lubber.
3. Find the names of sailors who reserved at least one boat.

Practice Homework 5 (cont'd)

Sailors(sid, sname, rating, age) // sailor id, sailor name, rating, age
Boats(bid, bname, color) // boat id, boat name, color of boat
Reserves(<u>sid, bid</u>, day) // sailor id, boat id, date that sid reserved bid.

4. Find the names of sailors whose age > 20 and have not reserved any boats.

5. Find the names of sailors who have reserved a red or a green boat.

6. Find the names of sailors who have reserved a red and a green boat.

Practice Homework 5 (cont'd)

Sailors(sid, sname, rating, age) // sailor id, sailor name, rating, age
Boats(bid, bname, color) // boat id, boat name, color of boat
Reserves(<u>sid, bid</u>, day) // sailor id, boat id, date that sid reserved bid.

7. Find the names of sailors who have reserved at least 2 different boats.

8. Find the names of sailors who have reserved exactly 2 different boats.

Express these queries in SQL.

END