Relational Algebra and SQL

Relational Query Languages

- Languages for describing queries on a relational database
- *Structured Query Language* (SQL)
  - Predominant application-level query language
  - Declarative
- *Relational Algebra*
  - Intermediate language used within DBMS
  - Procedural

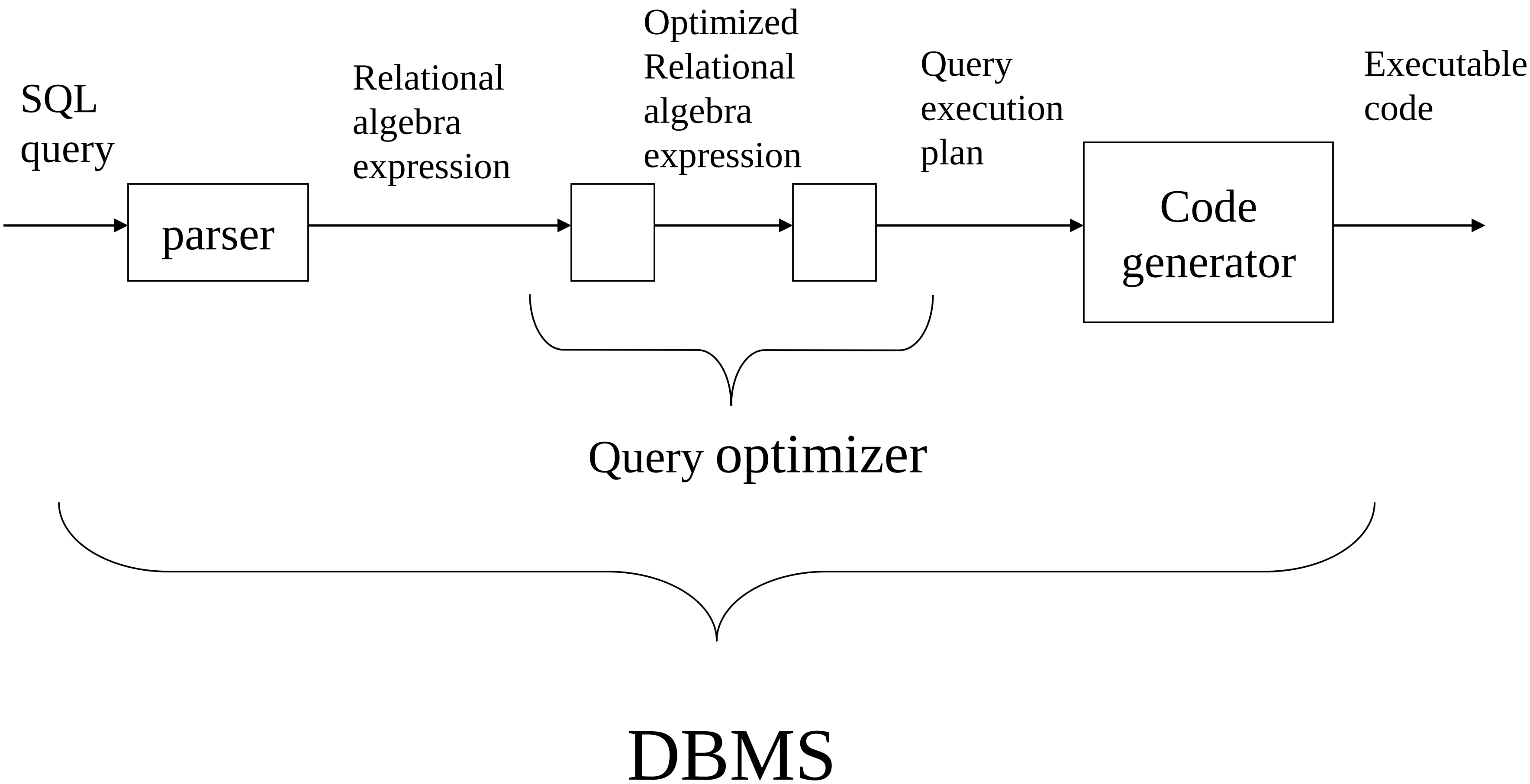
What is an Algebra?

- A language based on operators and a domain of values
- Operators map values taken from the domain into other domain values
- Hence, an expression involving operators and arguments produces a value in the domain
- When the domain is a set of all relations (and the operators are as described later), we get the *relational algebra*
- We refer to the expression as a *query* and the value produced as the *query result*

Relational Algebra

- *Domain*: set of relations
- *Basic operators*: select, project, union, set difference, Cartesian product
- *Derived operators*: set intersection, division, join
- *Procedural*: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

Relational Algebra in a DBMS



Schema for Student Registration System

Student (*Id*, *Name*, *Addr*, *Status*)
Professor (*Id*, *Name*, *DeptId*)
Course (*DeptId*, *CrsCode*, *CrsName*, *Descr*)
Transcript (*StudId*, *CrsCode*, *Semester*, *Grade*)
Teaching (*ProfId*, *CrsCode*, *Semester*)
Department (*DeptId*, *Name*)

Relational Algebra

- Five operators:
  - Selection: σ
  - Projection: Π
  - Cartesian Product: $\times$
  - Union: $\cup$
  - Difference: -
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural, equi-join, theta join, semi-join)
  - Renaming: ρ

Set Operators

- Relation is a set of tuples, so set operations should apply: $\cap$, $\cup$, – (set difference)
- Result of combining two relations with a set operator is a relation => all its elements must be tuples having same structure
- Hence, scope of set operations limited to *union compatible relations*

Select Operator

- Produce table containing subset of rows of argument table satisfying condition

$$\sigma_{condition}\ relation$$

- Example:

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\sigma_{Hobby=\text{'stamps'}}(\text{Person})$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
9876	Bart	5 Pine St	stamps

1.Select (σ)

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$

SSN	Name	Salary
1234545	John	200000
5423341	Smith	600000
4352342	Fred	500000

- Examples
  - $\sigma_{\text{Salary} > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	600000
4352342	Fred	500000

  - $\sigma_{\text{name} = \text{“Smith”}}$ (Employee)

SSN	Name	Salary
5423341	Smith	600000

The condition c can be =, <, ≤, >, ≥, <>

Selection Condition

- Operators: $<, \leq, \geq, >, =, \neq$
- Conditions:
  - <*attribute*> *operator* <*constant*>
  - <*attribute*> *operator* <*attribute*>
  - <*condition*> AND <*condition*>
  - <*condition*> OR <*condition*>
  - NOT <*condition*>

Select - Examples

- $\sigma_{\mathit{Id}>3000 \text{ Or } \mathit{Hobby}=\text{'hiking'}}$ (Person)
- $\sigma_{\mathit{Id}>3000 \text{ AND } \mathit{Id} <3999}$ (Person)
- $\sigma_{\text{NOT}(\mathit{Hobby}=\text{'hiking'})}$ (Person)
- $\sigma_{\mathit{Hobby}\neq\text{'hiking'}}$ (Person)

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

2. Project (Π)

Produces table containing subset of columns of argument table

$$\Pi_{\textit{attribute list}}(\textit{relation})$$

Example 1:

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\Pi_{Name,Hobby}$**(Person)**

Name	*Hobby*
John	stamps
John	coins
Mary	hiking
Bart	stamps

Project Operator

- *Example 2:*

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\Pi_{Name,Address}$(Person)

Name	*Address*
John	123 Main
Mary	7 Lake Dr
Bart	5 Pine St

Result is a table (no duplicates)

Expressions

$\Pi_{Id,\ Name}\ (\sigma_{Hobby='stamps'\ \mathrm{OR}\ Hobby='coins'}\ (\mathrm{Person})\ )$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

Person

Id	*Name*
1123	John
9876	Bart

Result

3. Union Operation (∪)

- Fetch data from two relations(tables) or temporary relation(result of another operation).
- Relations(tables) specified should have same number of attributes(columns) and compatible.
- Duplicate tuples are eliminated from the result.

StdID	Name	StdCourse
5423341	Krishna	CS101
4352342	John	EC201

R1

Syntax: $R1 \cup R2$

where R1 and R2 are relations.

$\prod_{StdiD,Name,StdCourse}(R1) \cup \prod_{StdiD,Name,StdCourse}(R2)$

StdID	Name	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

R2

StdID	Name	StdCourse
4352342	John	EC201
4352500	Smita	CS200
5423341	Krishna	CS101

4. Difference Operation (-)

- Attribute name in relation R1 has to match with attribute name in R2.
- Both Relations should be compatible
- Resultant relation will have tuples present in one relation and not present in the second relation.

Syntax: R1 - R2

where R1 and R2 are relations.

$\prod_{Stdid,Name,StdCourse}(R1) - \prod_{Stdid,Name,Stdcourse}(R2)$

StdID	Name	StdCourse
5423341	Krishna	CS101
4352342	John	EC201

R1

StdID	Name	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

R2

StdID	Name	StdCourse
4352342	John	EC201

4. Intersection Operation (∩)

- Attribute name in relation R1 has to match with attribute name in R2.
- Both Relations should be compatible
- Resultant relation will have tuples present in both relations

.

Syntax: R1 - R2

where R1 and R2 are relations.

$$\prod_{\text{Stdid, Name,StdCourse}}(R1) \ \cap \ \prod_{\text{Stdid,Name,StdCourse}}(R2)$$

StdID	Name	StdCourse
5423341	Krishna	CS101
4352342	John	EC201

R1

StdID	Name	StdCourse
5423341	Krishna	CS101

StdID	Name	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

R2

5. Cartesian Product (X)

- If *R1* and *R2* are two relations, $R1 \times R2$ is the set of all concatenated tuples <*x*,*y*>, where *x* is a tuple in *R1* and *y* is a tuple in *R2*
  - (*R1* and *R2* need not be union compatible)

- $R1 \times R2$ is <u>expensive to compute</u>:

R1

a	*b*
x1	x2
x3	x4

R2

c	*d*
y1	y2
y3	y4

$R1 \times R2$

a	*b*	*c*	*d*
x1	x2	y1	y2
x1	x2	y3	y4
x3	x4	y1	y2
x3	x4	y3	y4

Cartesian Product

- Concatenation of every tuple of one relation with every tuple of a second relations.
- Relation A (having *m* tuples) and relation B (having *n* tuples) has *m* times *n* tuples.
- Denoted A X B or A TIMES B.

R1

StdID	Name	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

Notation: R1 × R2

Example: Student × Credit

R2

Id	Credit	Dep
5423341	4	CS
4352300	3	CE

Result : R1 × R2

StdID	Name	Stdcourse	Id	Credit	Dep
5423341	Krishna	CS101	5423341	4	CS
5423341	Krishna	CS101	4352300	3	CE
4352500	Smita	CS200	5423341	4	CS
4352500	Smita	CS200	4352300	3	CE

$\sigma_{Credit\ =\ ‘4’}\ (R1\ \ X\ \ R2)$

6. Rename (ρ)

Changes the schema, not the instance

Notation: $\rho_{B1,\ldots,Bn}(R)$

Example:

Old Schema:

Student(StdID,Name,StdCourse)

– $\rho_{id,\ StudentName}(Student)$

New schema:

Student(Id, StudentName)

StdID	Name	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

id	StudentName	Stdcourse
5423341	Krishna	CS101
4352500	Smita	CS200

Rename (ρ)

- Result of expression evaluation is a relation
- Attributes of relation must have distinct names. This is not guaranteed with Cartesian product
  - e.g., suppose in previous example *a* and *c* have the same name
- Renaming operator tidies this up. To assign the names $A_1, A_2, \ldots A_n$ to the attributes of the *n* column relation produced by expression *expr* use

$$expr\ [A_1, A_2, \ldots A_n]$$

Join and Rename

- **Problem**: *R1* and *R2* might have attributes with the same name – in which case the Cartesian product is not Possible
- **Solution**:
  - Rename attributes prior to forming the product and use new names in *join-condition*.
  - Common attribute names are qualified with relation names in the result of the join

Relational Algebra and SQL

Which of the two expression is more easy to understand?

$$\pi_{CrsName}\ \sigma_{C\_CrsCode=T\_CrsCode\ AND\ Sem=\text{'S2000'}}$$
$$(Course\ [C\_CrsCode,\ DeptId,\ CrsName,\ Desc]$$
$$\times\ Teaching\ [ProfId,\ T\_CrsCode,\ Sem])$$

equivalent to:

```
SELECT  C.CrsName
FROM  Course C, Teaching T
WHERE  C.CrsCode=T.CrsCode AND T.Sem='S2000'
```

SELECT is simple evaluation algorithm .

Derived Operation: Join

The expression :

$\sigma_{join\text{-}condition}\ (R1 \times R2)$

where *join-condition* is a *conjunction* of terms:

A_i *oper* B

A_i is an attribute of $R1$, B_i is an attribute of $R2$, and *oper* is one of $=, \ <, >, \geq \neq, \leq,$
is referred to as the (theta) join of $R1$ and $R2$ and denoted:

$R1\ _{join\text{-}condition}\ R2$

where *join-condition* is represented by $\bowtie$ or |X|

Theta Join

A join that involves a predicate

$R1 \;|\times|_{\theta}\; R2 \;=\; \sigma_{\theta}(R1 \times R2)$

- Here θ can be any condition

Theta Join

A (*general* or *theta*) *join* of *R1* and *R2* is the expression

$$R1 \bowtie_{c} R2$$

where *join-condition* *c* is a *conjunction* of terms:

$$A_i \ \ oper \ B_i$$

Where A_i is an attribute of *R1*; B_i is an attribute of *R2* ; and *oper* is one of =, <, >, ≥ ≠, ≤.

Q: Any difference between join condition and selection condition?

The meaning is:

$$\sigma_c (R1 \times R2)$$

Where join-condition c becomes a select condition c except for possible renamings of attributes

Theta Join – Example

Output the names of all employees that earn more than their managers.

$$\Pi_{\text{Employee.}\mathit{Name}}(\text{Employee} \bowtie_{\mathit{MngrId}=\mathit{Id} \text{ AND } \mathit{Salary}>\mathit{Salary}} \text{Manager})$$

The join yields a table with attributes:

Employee.*Name*, Employee.*Id*, Employee.*Salary*,Employee.MngId, Manager.*Name*, Manager.*Id*, Manager.*Salary*

Natural Join

- Special case of equijoin:
  - join condition equates *all* and *only* those attributes with the same name (condition doesn't have to be explicitly stated)
  - duplicate columns eliminated from the result

Transcript (*StudId*, *CrsCode*, *Sem*, *Grade*)
Teaching (*ProfId*, *CrsCode*, *Sem*)

$$\text{Transcript} \bowtie \text{Teaching} =$$
$$\pi_{StudId,\ Transcript.CrsCode,\ Transcript.Sem,\ Grade,\ ProfId}\big(\text{Transcript} \bowtie_{CrsCode=CrsCode \text{ AND } Sem=\text{Sem}} \text{Teaching}\big)$$

Natural Join (con't)

Notation :

$$R1 \bowtie R2 = \pi_{attr\text{-}list}\,(\sigma_{join\text{-}cond}\,(R1 \times R2)\,)$$

where

attr-list = *attributes* (*R1*) $\cup$ *attributes* (*R2*)

(duplicates are eliminated) and *join-cond* has the form:

$A_1 = A_1$ AND … AND $A_n = A_n$

where

$\{A_1 \ldots A_n\} = attributes(R1) \cap attributes(R2)$

Natural Join

Notation: R1 $|\times|$ R2

Meaning: R1 $|\times|$ R2 = $\Pi_A(\sigma_C(R1 \times R2))$

Where:

- The selection σ_C checks equality of all common attributes
- The projection eliminates the duplicate common attributes

Natural Join

- R1=

A	B
X	Y
X	Z
Y	Z
Z	V

R2=

B	C
Z	U
V	W
Z	V

- R1 |×| R2=

A	B	C
X	Z	U
X	Z	V
Y	Z	U
Y	Z	V
Z	V	W

Natural Join : Examples

- Given the schemas R1(A, B, C, D), R2(A, C, E), what is the schema of R |×| S ?

- Given R1(A, B, C), R2(D, E), what is R1 |×| R2 ?

- Given R1(A, B), R2(A, B), what is R1 |×| R2 ?