Relational Algebra

Some slide content courtesy of Raghu Ramakrishnan

Codd's Relational Algebra

- A set of mathematical operators that compose, modify, and combine tuples within different relations
- Relational algebra operations operate on relations and produce relations ("closure")

  f: Relation → Relation   f: Relation x Relation → Relation

Relational Algebra

- The Relational Algebra is used to define the ways in which relations (tables) can be operated to manipulate their data.
- It is used as the basis of SQL for relational databases, and illustrates the basic operations required of any DML.
- This Algebra is composed of Unary operations (involving a single table) and Binary operations (involving multiple tables).

Codd's Logical Operations: The Relational Algebra

- Six basic operations:
  - Projection $\pi_{\overline{\alpha}}(R)$
  - Selection $\sigma_{\theta}(R)$
  - Union $R_1 \cup R_2$
  - Difference $R_1 - R_2$
  - Product $R_1 \times R_2$
  - (Rename) $\rho_{\overline{\alpha} \rightarrow \overline{\beta}}(R)$
- And some other useful ones:
  - Join $R_1 \bowtie_{\theta} R_2$
  - Semijoin $R_1 \ltimes_{\theta} R_2$
  - Intersection $R_1 \cap R_2$
  - Division $R_1 \div R_2$

Unary Operations

Selection
Projection
Rename

Selection

- The selection or σ operation selects rows from a table that satisfy a ***condition***:

$$\sigma_{<\,condition\,>} < tablename >$$

- Example: $\sigma_{\text{course} = \text{'CM'}}$ Students

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

→

student#	*name*	*course*
200	Dave	CM
300	Bob	CM

Projection

- The projection or π operation selects a list of columns from a table.

$$\pi_{<\,column\ list\,>} <\, tablename \,>$$

- Example: $\pi_{\text{student\#, name}}$ Students

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

→

student#	*name*
100	Fred
200	Dave
300	Bob

Selection / Projection

- Selection and Projection are usually combined:

$$\pi_{\text{student\#, name}} (\sigma_{\text{course = 'CM'}} \text{Students})$$

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

→

student#	*name*
200	Dave
300	Bob

Rename, $\rho_{\overline{\alpha}\rightarrow\overline{\beta}}$

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

    $\rho_{\alpha\rightarrow\beta}(x)$ Takes the relation with schema $\overline{\alpha}$
    Returns a relation with the attribute list $\overline{\beta}$

Binary Operations

Cartesian Product
Theta Join
Inner Join
Natural Join
Outer Joins
Semi Joins

Cartesian Product

- Concatenation of every row in the first relation (R) with every row in the second relation (S):

  R X S

Cartesian Product - Example

Students

student#	name	course
100	Fred	PH
200	Dave	CM
300	Bob	CM

Courses

course#	name
PH	Pharmacy
CM	Computing

Students X Courses =

student#	Students.name	course	course#	Courses.name
100	Fred	PH	PH	Pharmacy
100	Fred	PH	CM	Computing
200	Dave	CM	PH	Pharmacy
200	Dave	CM	CM	Computing
300	Bob	CM	PH	Pharmacy
300	Bob	CM	CM	Computing

Theta Join

- A Cartesian product with a condition applied:

$$R \bowtie_{\text{<condition>}} S$$

Theta Join - Example

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

Courses

course#	*name*
PH	Pharmacy
CM	Computing

Students $\bowtie_{student\# = 200}$ Courses

student#	*Students.name*	*course*	*course#*	*Courses.name*
200	Dave	CM	PH	Pharmacy
200	Dave	CM	CM	Computing

Inner Join (Equijoin)

- A Theta join where the <condition> is the match (=) of the primary and foreign keys.

$$R \bowtie_{<R.primary\_key\, =\, S.foreign\_key>} S$$

Inner Join - Example

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

Courses

course#	*name*
PH	Pharmacy
CM	Computing

Students $\bowtie_{\textit{course = course\#}}$ Courses

student#	*Students.name*	*course*	*course#*	*Courses.name*
100	Fred	PH	PH	Pharmacy
200	Dave	CM	CM	Computing
300	Bob	CM	CM	Computing

Natural Join

- Inner join produces redundant data (in the previous example: course and course#). To get rid of this duplication:

  $\pi_{< student\#,\ Students.name,\ course,\ Courses.name >}$

  (Students $\bowtie_{<course\ =\ course\#>}$ Courses)

Or

R1= Students $\bowtie_{<course\ =\ course\#>}$ Courses

R2= $\pi_{< student\#,\ Students.name,\ course,\ Courses.name >}$ R1

The result is called the natural join of Students and Courses

Natural Join - Example

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
300	Bob	CM

Courses

course#	*name*
PH	Pharmacy
CM	Computing

R1= Students $\bowtie_{<course\ =\ course\#>}$ Courses

R2= $\pi_{<\ student\#,\ Students.name,\ course,\ Courses.name\ >}$ R1

student#	*Students.name*	*course*	*Courses.name*
100	Fred	PH	Pharmacy
200	Dave	CM	Computing
300	Bob	CM	Computing

Outer Joins

- Inner join + rows of one table which do not satisfy the <condition>.

- <u>Left Outer Join:</u> R ⟕ <R.primary_key = S.foreign_key> S

All rows from R are retained and unmatched rows of S are padded with NULL

- <u>Right Outer Join:</u> R ⟖ <R.primary_key = S.foreign_key> S

All rows from S are retained and unmatched rows of R are padded with NULL

Left Outer Join - Example

Students

student#	name	course
100	Fred	PH
200	Dave	CM
400	Peter	EN

Courses

course#	name
PH	Pharmacy
CM	Computing
CH	Chemistry

Students ⟕ <course = course#> Courses

student#	Students.name	course	course#	Courses.name
100	Fred	PH	PH	Pharmacy
200	Dave	CM	CM	Computing
400	Peter	EN	NULL	NULL

Right Outer Join - Example

Students

student#	*name*	*course*
100	Fred	PH
200	Dave	CM
400	Peter	EN

Courses

course#	*name*
PH	Pharmacy
CM	Computing
CH	Chemistry

Students ⟖ <course = course#> Courses

student#	*Students.name*	*course*	*course#*	*Courses.name*
100	Fred	PH	PH	Pharmacy
200	Dave	CM	CM	Computing
NULL	NULL	NULL	CH	Chemistry

Combination of Unary and Join Operations

Students

student#	*name*	*address*	*course*
100	Fred	San Jose	PH
200	Dave	Fremont	CM
300	Bob	San Jose	CM

Courses

course#	*name*
PH	Pharmacy
CM	Computing

Show the names of students (from San Jose) and the names of their courses

R1= Students $\bowtie$ <course=course#> Courses

R2= σ <address="San Jose"> R1

R3= π <Students.name, Course.name> R2

Students.name	Courses.name
Fred	Pharmacy
Bob	Computing

Set Operations

Union

Intersection

Difference

Union

- Takes the set of rows in each table and combines them, eliminating duplicates
- Participating relations must be compatible, ie have the same number of columns, and the same column names, domains, and data types

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

$R \cup S$

A	B
a1	b1
a2	b2
a3	b3

Intersection

- Takes the set of rows that are common to each relation
- Participating relations must be compatible

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

$R \cap S$

A	B
a2	b2

Difference

- Takes the set of rows in the first relation but not the second
- Participating relations must be <u>compatible</u>

R

A	B
a1	b1
a2	b2

S

A	B
a2	b2
a3	b3

R - S

A	B
a1	b1

Relational Algebra

Operations written in SQL

Unary Operations

Selection

$\sigma_{course\ =\ 'Computing'}$ Students

In SQL:

Select *

From Students

Where course = 'Computing';

Projection

$\pi_{student\#,\ name}$ Students

In SQL:

Select student#, name

From Students;

Selection & Projection

$\pi_{student\#,\ name}(\sigma_{course\ =\ 'Computing'}$ Students)

In SQL:

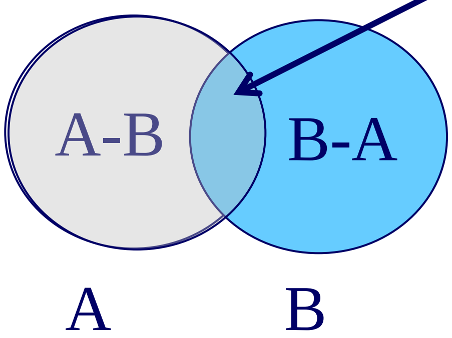
Select student#, name

From students

Where course = 'Computing';

Deriving Intersection

Intersection: as with set operations, derivable from difference



$$A \cap B$$
$$\equiv (A \cup B) - (A - B) - (B - A)$$
$$\equiv B - (B - A) \equiv A - (A - B)$$

Binary Operations/Joins

Cartesian Product: Students X Courses

In SQL:

Select *

From Students, Courses;

Theta Join: Students ⋈ <student# =200>

Courses

In SQL:

Select *

From Students, Courses

Where student# = 200;

Binary Operations/Joins

Inner Join (Equijoin): Students $\bowtie_{<course=course\#>}$ Courses

In SQL:

Select *

From Students, Courses

Where course=course#;

Natural Join:

R1= Students $\bowtie_{<course = course\#>}$ Courses

R2= $\pi_{< student\#, Students.name, course, Courses.name >}$ R1

In SQL:

Select student#, Students.name, course, Courses.name

From Students, Courses

Where course=course#;

Outer Joins

Left Outer Join

Students ⟕ <course = course#>
Courses

In SQL:

Select *

From Students, Courses

Where course = course#(+)

Right Outer Join

Students ⟖ <course = course#> Courses

In SQL:

Select *

From Students, Courses

Where course(+) = course#

Combination of Unary and Join Operations

R1= Students $\bowtie$ <course=course#> Courses

R2= σ <address=“San Jose”> R1

R3= π <Students.name, Course.name> R2

<u>In SQL:</u>

Select Students.name, Courses.name

From Students, Courses

Where course=course#

AND address=“San Jose”;

Set Operations

Union: R ∪ S

In SQL:

Select * From R

Union

Select * From S;

Intersection: R ∩ S

In SQL:

Select * From R

Intersect

Select * From S;

Difference: R - S

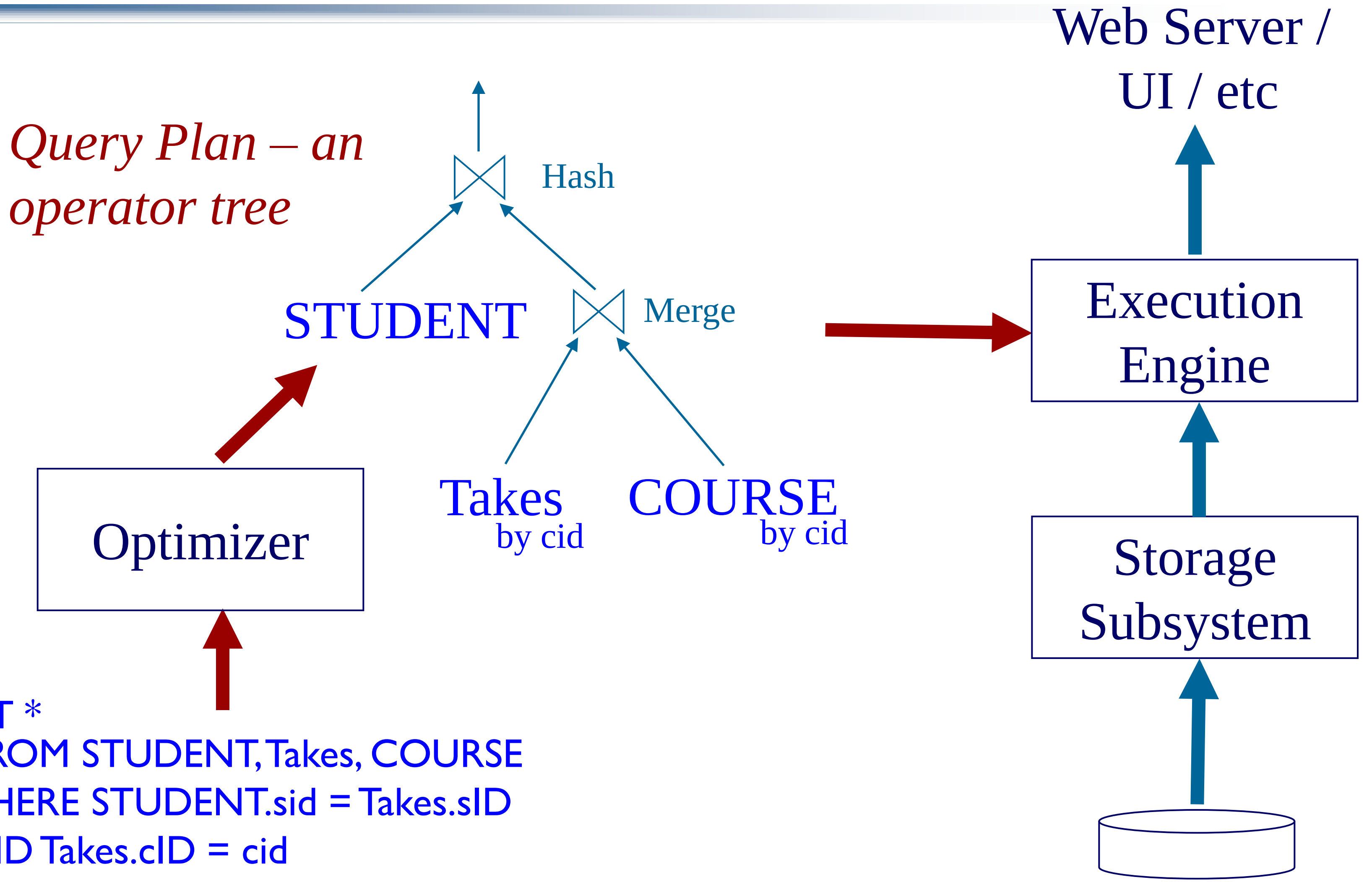
In SQL:

Select * From R

Minus

Select * From S;

The Big Picture: SQL to Algebra to Query Plan to Web Page




```
SELECT *
  FROM STUDENT, Takes, COURSE
 WHERE STUDENT.sid = Takes.sID
 AND Takes.cID = cid
```