COMP9311: Database Systems

SQL

(textbook: chapters 6 and 7)

Term 3 2021

Week 3 Relational Algebra and SQL

By Helen Paik, CSE UNSW

Disclaimer: the course materials are sourced from

- previous offerings of COMP9311 and COMP3311
- Prof. Werner Nutt on Introduction to Database Systems (http://www.inf.unibz.it/~nutt/Teaching/IDBs1011/)

SQL syntax overview

An SQL query consists of a sequence of clauses:

```
SELECT    projectionList
FROM      relations/joins
WHERE     condition
GROUP BY  groupingAttributes
HAVING    groupCondition
```

FROM, WHERE, GROUP BY, HAVING clauses are optional.

Result of query: a relation, typically displayed as a table.

Result could be just one tuple with one attribute (i.e. one value) or even empty

Relational Algebra: Principles

Atoms are relations

Operators are defined for arbitrary instances of a relation

The following two results have to be defined for each operator:

- **result schema**
- **result instance**

Set theoretic operators

- union "$\cup$", intersection "$\cap$", difference "\"

Renaming operator ρ

Removal operators

- projection π, selection σ

Combination operators

- Cartesian product "$\times$", joins "$\bowtie$"

Extended operators

- duplicate elimination, grouping, aggregation, sorting, outer joins, etc.

- *"Equivalent" to SQL query language … Relational Algebra concepts reappear in SQL*
- *Used inside a DBMS, to express query plans*

Set Operators

Observations:

Instances of relations are sets
➔ we can form **unions**, **intersections**, and **differences**

Set algebra operators can only be applied to relations with identical attributes,

- same number of attributes
- same attribute names
- same domains

- (i.e., set operation compatibility)

Union (∪)

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student ∪ Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4
s4	Maurer	2

UNSW SYDNEY

Intersection ($\cap$)

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student $\cap$ Master-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4

Difference (\)

CS-Student

Studno	Name	Year
s1	Egger	5
s3	Rossi	4
s4	Maurer	2

Master-Student

Studno	Name	Year
s1	Egger	5
s2	Neri	5
s3	Rossi	4

CS-Student \ Master-Student

Studno	Name	Year
s4	Maurer	2

Set difference, formally:

$$B \setminus A = \{x \in B \mid x \notin A\}.$$

Renaming ρ

- The renaming operator ρ (reads 'rho') changes the name of relation schema (both for relation name and relation attributes)
- It changes the schema, but only within a query
- ρ_{x}(E) where E is the relation name and x is the new name for E, usually a shorter name
  - ρ_{FC}(Father-Child)
- $\rho_{a/b}$(E) where E is the relation name, a, b are attribute names, b is an attribute of E
  - $\rho_{parent/father}$(Father-Child)

Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

ρ_{FC}(Father-Child)

FC

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

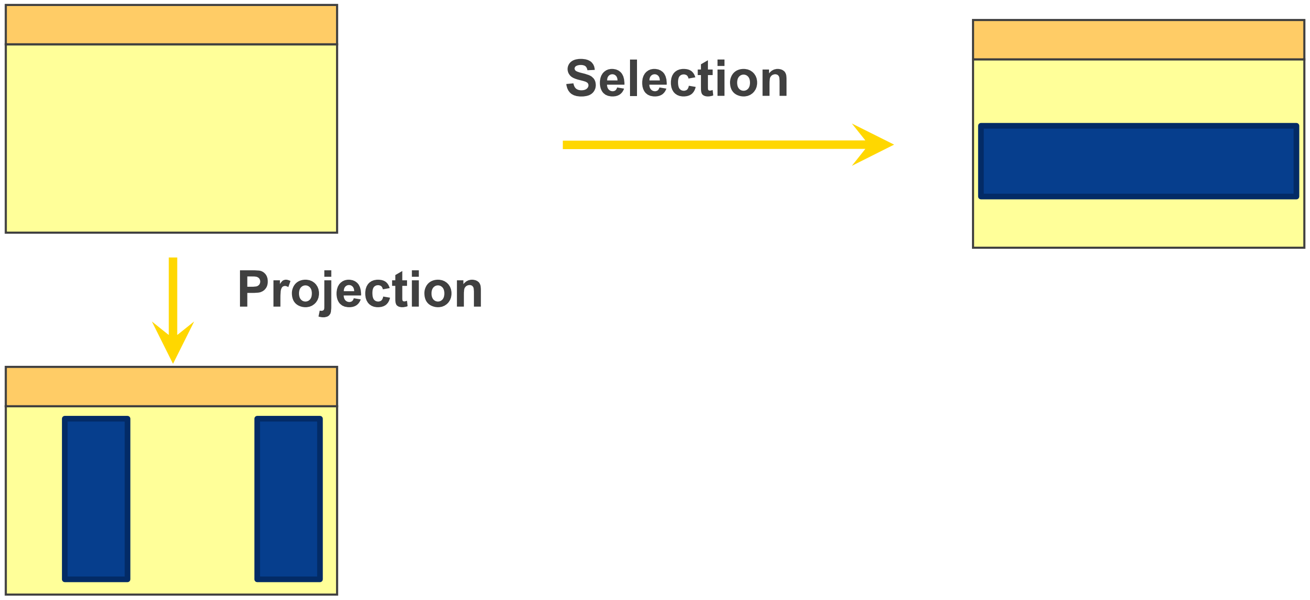
$\rho_{parent/father}$(Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Projection and Selection

Two "orthogonal" operators

- Selection:
  - horizontal decomposition
- Projection:
  - vertical decomposition



Projection (π)

General form: $\pi_{A1,\dots,Ak}(R)$

where R is a relation and $A_1,\dots,A_k$ are attributes of R.

Result:

- Schema: $(A_1,\dots,A_k)$
- Instance: the set of all subtuples $t[A_1,\dots,A_k]$ where $t \in R$

Intuition: “removes” all attributes that are not in projection list

Projection: Example

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$\pi_{tutor}(\text{STUDENT})$ =

tutor
bush
kahn
goble
zobel

Note:

- *result relations don't have a name*
- *If duplicates?*

UNSW SYDNEY

Selection: Example

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$\sigma_{name='bloggs'}(STUDENT)$ =

STUDENT

studno	name	hons	tut or	year
s4	bloggs	ca	goble	1

Note:

- *result relation has a name*

Selection conditions

STUDENT

<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$$\sigma_{((hons='cs') \text{ or } (hons = 'ca')) \text{ and } (tutor='goble')}(STUDENT) \quad =$$

STUDENT

studno	name	hons	tut or	year
s3	smiths	cs	goble	2
s4	bloggs	ca	goble	1

Operators Can Be Nested

Who is the tutor of the student named “Bloggs”?

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$$\pi_{tutor}\left(\ \sigma_{name=\text{‘bloggs’}}\ (STUDENT)\ \right)$$

=

tutor
goble

UNSW SYDNEY

Cartesian Product (X)

General form:

where R and S are arbitrary relations $R \times S$

Result:

- Schema: (A1,…,Am,B1,…,Bn), where (A1,…,Am) is the schema of R and (B1,…,Bn) is the schema of S.

  (If A is an attribute of both, R and S, then $R \times S$ contains the disambiguated attributes R.A and S.A.)

- Instance: the set of all concatenated tuples (t,s) where $t \in R$ and $s \in S$

“Where are the Tutors of Students?”

To answer the query

“For each student, identified by name and student number, return the name of the tutor and their office number”

we have to

- combine tuples from Student and Staff
- that satisfy “`Student.tutor=Staff.lecturer`”
- and keep the attributes `studno`, `name`, (`tutor` or `lecturer`), `and roomno.`

In relational algebra:

STAFF

lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$$\pi_{studno,name,lecturer,roomno}(\sigma_{tutor=lecturer}(Student \times Staff))$$

The part $\sigma_{tutor=lecturer}(Student \times Staff)$ is a “<u>join</u>”.

Example: Student Marks in Courses

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

ENROL

stud no	course no	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

"For each student, show the courses in which they are enrolled and their marks"

First,

$R \leftarrow \sigma_{\text{Student.studno= Enrol.studno}}(\text{Student} \times \text{Enrol})$,

then

$\text{Result} \leftarrow \pi_{\text{studno,name, …,exam\_mark}}(R)$

$\pi_{\text{studno,name, …,exam\_mark}}(\sigma_{\text{Student.studno= Enrol.studno}}(\text{Student} \times \text{Enrol}))$

Join (⋈)

- The most used operator in the relational algebra.

  Allows us to establish connections among data in different relations, taking advantage of the ”data-based" nature of the relational model.

- Three main versions of the join:
  - "**natural**" join: takes attribute names into account;
  - "**theta**" join.
  - “**equi**” join (a special form of theta join)
  - all denoted by the symbol ⋈

Natural Join

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

ENROL

stud no	course no	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

Student ⋈ Enrol

- **Implicit** join based on **common** attributes
- The tuples in the resulting relation are obtained by combining tuples in the operands with equal values on the common attributes
- Common attributes appear once in the results

θ-Joins (read "Theta"-Joins), Equi-Joins

Theta-Join:

- *The most general form of JOIN …*
- Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.
- Theta join can use comparison operators and common attributes are not required.

$$\text{Student} \bowtie_{\text{student.year} < \text{enrol.labmark}} \text{Enrol}$$

- The results include the 'joined' attributes from both relations
- The attribute names do not have to match (but their domains have to be compatible)

Equi-Join:

A special form of Theta-join, and *the most common form of JOIN …*

with a **join** condition containing an equality operator (i.e., explicitly stating the joining attributes)

$$\text{Student} \bowtie_{\text{stuno}=\text{stuno}} \text{Enrol}$$

STUDENT

<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STAFF

<u>lecturer</u>	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

$$\text{Student} \bowtie_{\text{tutor=lecturer}} \text{Staff}$$

(equivalent to: $\sigma_{\text{tutor=lecturer}}(\text{Student} \times \text{Staff})$)

stud no	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s4	bloggs	ca	goble	1	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34
s6	peters	ca	kahn	3	kahn	IT206

Outer Join

An outer join extends those tuples with null values that would get lost by a join like natural join or equi join (a.k.a. inner joins)

The outer join comes in three versions

- left: keeps the tuples of the left argument, extending them with nulls if necessary
- right: ... of the right argument ...
- full: ... of both arguments ...

(Natural) Left Outer Join

Employee

Employee	Department
Brown	A
Jones	B
Smith	B

Department

Department	Head
B	Black
C	White

Employee $⟕_{\text{Left}}$ Department

Employee	Department	Head
Brown	A	null
Jones	B	Black
Smith	B	Black

(Natural) Right Outer Join

Employee

Employee	Department
Brown	A
Jones	B
Smith	B

Department

Department	Head
B	Black
C	White

Employee $\bowtie_{\text{Right}}$ Department

Employee	Department	Head
Jones	B	Black
Smith	B	Black
null	C	White

(Natural) Full Outer Join

Employee

Employee	Department
Brown	A
Jones	B
Smith	B

Department

Department	Head
B	Black
C	White

Employee $\bowtie_{\text{Full}}$ Department

Employee	Department	Head
Brown	A	null
Jones	B	Black
Smith	B	Black
null	C	White

Duplicate Elimination

Real DBMSs implement a version of relational algebra that operates on multisets ("bags") instead of sets.

(Which of these operators may return bags, even if the input consists of sets?)

For the bag version of relational algebra, there exists a duplicate elimination operator δ.

If R =

A	B
1	2
3	4
3	4
1	2

, then $\delta(R)$ =

A	B
1	2
3	4

Aggregation

Often, we want to retrieve aggregate values, like the “sum of salaries” of employees, or the “average age” of students.

This is achieved using aggregation functions, such as SUM, AVG, MIN, MAX, or COUNT.

Such functions are applied by the grouping and aggregation operator γ.

If R =

A	B
1	2
3	4
3	5
1	1

, then $\gamma_{SUM(A)}(R)$ =

SUM(A)
8

and $\gamma_{AVG(B)}(R)$ =

AVG(B)
3

Grouping and Aggregation

More often, we want to retrieve aggregate values for groups, like the “sum of employee salaries” per department, or the “average student age” per faculty.

As additional parameters, we give γ attributes that specify the criteria according to which the tuples of the argument are grouped.

E.g., the operator $\gamma_{A,SUM(B)}(R)$

- partitions the tuples of R in groups that agree on A,
- returns the sum of all B values for each group.

If R =

A	B
1	2
3	4
3	5
1	3

, then $\gamma_{A,SUM(B)}(R)$ =

A	SUM(B)
1	5
3	9