SECD2523 DATABASE

FROM
RELATIONAL ALGEBRA TO SQL

Week 4

Innovating Solutions

RELATIONAL ALGEBRA I & II

σ	Select
п	Projection
∪	Union
∩	Intersection
–	Difference
×	Product
⋈	Join

Fundamental operations to retrieve and manipulate tuples in a relation.

- Based on set algebra (unordered lists with no duplicates).

Each operator takes one or more relations as its inputs and outputs a new relation.

- We can “chain” operators together to create more complex operations.

RELATIONAL ALGEBRA I: SELECTION

Choose a subset of the tuples from a relation that satisfies a selection predicate.

- Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.
- Can combine multiple predicates using conjunctions / disjunctions.

Syntax: $\sigma_{predicate}(R)$

SELECT * FROM R
WHERE a_id='a2' AND
b_id>102;

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a2	103
a3	104

$\sigma_{a\_id='a2'}(R)$

a_id	b_id
a2	102
a2	103

$\sigma_{a\_id='a2' \wedge b\_id>102}(R)$

a_id	b_id
a2	103

www.utm.my

RELATIONAL ALGEBRA I : PROJECTION

Generate a relation with tuples that contains only the specified attributes.

- Rearrange attributes' ordering.
- Remove unwanted attributes.
- Manipulate values to create derived attributes.

Syntax: $\Pi_{A1,A2,\ldots,An}(R)$

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a2	103
a3	104

```
SELECT b_id-100, a_id
  FROM R WHERE a_id =
  'a2';
```

$\Pi_{b\_id-100,a\_id}(\sigma_{a\_id='a2'}(R))$

b_id-100	a_id
2	a2
3	a2

www.utm.my

RELATIONAL ALGEBRA I : UNION

Generate a relation that contains all tuples that appear in either only one or both input relations.

Syntax: (R ∪ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

```
(SELECT * FROM R)
UNION
(SELECT * FROM S) ;
```

(R ∪ S)

a_id	b_id
a1	101
a2	102
a3	103
a4	104
a5	105

RELATIONAL ALGEBRA I : INTERSECTION

Generate a relation that contains only the tuples that appear in both of the input relations.

Syntax: **(R ∩ S)**

(**SELECT * FROM** R)
INTERSECT
(**SELECT * FROM** S) ;

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R ∩ S)

a_id	b_id
a3	103

www.utm.my

RELATIONAL ALGEBRA I : SET DIFFERENCE

Generate a relation that contains only the tuples that appear in the first and not the second of the input relations.

Syntax: (R – S)

(**SELECT * FROM** R)
EXCEPT
(**SELECT * FROM** S) ;

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R – S)

a_id	b_id
a1	101
a2	102

RELATIONAL ALGEBRA I : CARTESIAN PRODUCT

Generate a relation that contains all possible combinations of tuples from the input relations.

Syntax: **(R × S)**

SELECT * FROM R **CROSS JOIN** S;

SELECT * FROM R, S;

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R × S)

R.a_id	R.b_id	S.a_id	S.b_id
a1	101	a3	103
a1	101	a4	104
a1	101	a5	105
a2	102	a3	103
a2	102	a4	104
a2	102	a5	105
a3	103	a3	103
a3	103	a4	104
a3	103	a5	105

RELATIONAL ALGEBRA II : JOIN

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

Syntax: (R ⋈ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id,val)

a_id	b_id	val
a3	103	XXX
a4	104	YYY
a5	105	ZZZ

(R ⋈ S)

a_id	b_id	val
a3	103	XXX

RELATIONAL ALGEBRA II : JOIN

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

Syntax: (R ⋈ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id,val)

a_id	b_id	val
a3	103	XXX
a4	104	YYY
a5	105	ZZZ

(R ⋈ S)

a_id	b_id	val
a3	103	XXX

R.a_id	R.b_id	S.a_id	S.b_id	S.val
a3	103	a3	103	XXX

RELATIONAL ALGEBRA II : JOIN

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

Syntax: **(R $\bowtie$ S)**

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id,val)

a_id	b_id	val
a3	103	XXX
a4	104	YYY
a5	105	ZZZ

(R $\bowtie$ S)

a_id	b_id	val
a3	103	XXX

R.a_id	R.b_id	~~S.a_id~~	~~S.b_id~~	S.val
a3	103	~~a3~~	~~103~~	XXX

www.utm.my

RELATIONAL ALGEBRA II : JOIN

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

(R ⋈ S)

a_id	b_id	val
a3	103	XXX

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id,val)

a_id	b_id	val
a3	103	XXX
a4	104	YYY
a5	105	ZZZ

```
SELECT * FROM R NATURAL JOIN S;

SELECT * FROM R JOIN S USING (a_id, b_id);

SELECT * FROM R JOIN S
  ON R.a_id = S.a_id AND R.b_id = S.b_id;
```

www.utm.my

AGGREGATION: $\mathfrak{I}$AL(R)

- **Identify the attributes in AL that are not aggregate functions:** These attributes will form the basis for grouping the data. They will be included in the `SELECT` clause and the `GROUP BY` clause.
- **Identify the aggregate functions in AL:** These will be applied to specific columns to produce a single summary value for each group. Common aggregate functions include `COUNT()`, `SUM()`, `AVG()`, `MIN()`, and `MAX()`.

Example:

```
SELECT MIN(HireDate) FROM Employees;
```

```
SELECT Department, AVG(Salary)FROM
Employees GROUP BY Department;
```

GROUPING : GA$\mathfrak{I}$AL(R)

DIVISION : R÷S

Relational division (R ÷ S) in relational algebra, which identifies tuples in R that are associated with all tuples in S, does not have a direct, single operator equivalent in SQL.

SQL does not have a direct division operator, but it can be implemented using combinations of `NOT EXISTS`, `EXCEPT` (or `MINUS`), `COUNT`, and `GROUP BY`.

Here's an example using the `NOT EXISTS` and `EXCEPT` approach, which is a common way to simulate relational division.

```
SELECT DISTINCT S.StudentID, S.StudentName FROM Students AS S

WHERE NOT EXISTS

Subquery 1: Find courses NOT taken by the current student

SELECT C.CourseID  FROM Courses AS C

EXCEPT

SELECT E.CourseID

FROM Enrollments AS E

WHERE E.StudentID = S.StudentID);
```

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Innovating Solutions
Menginovasi Penyelesaian