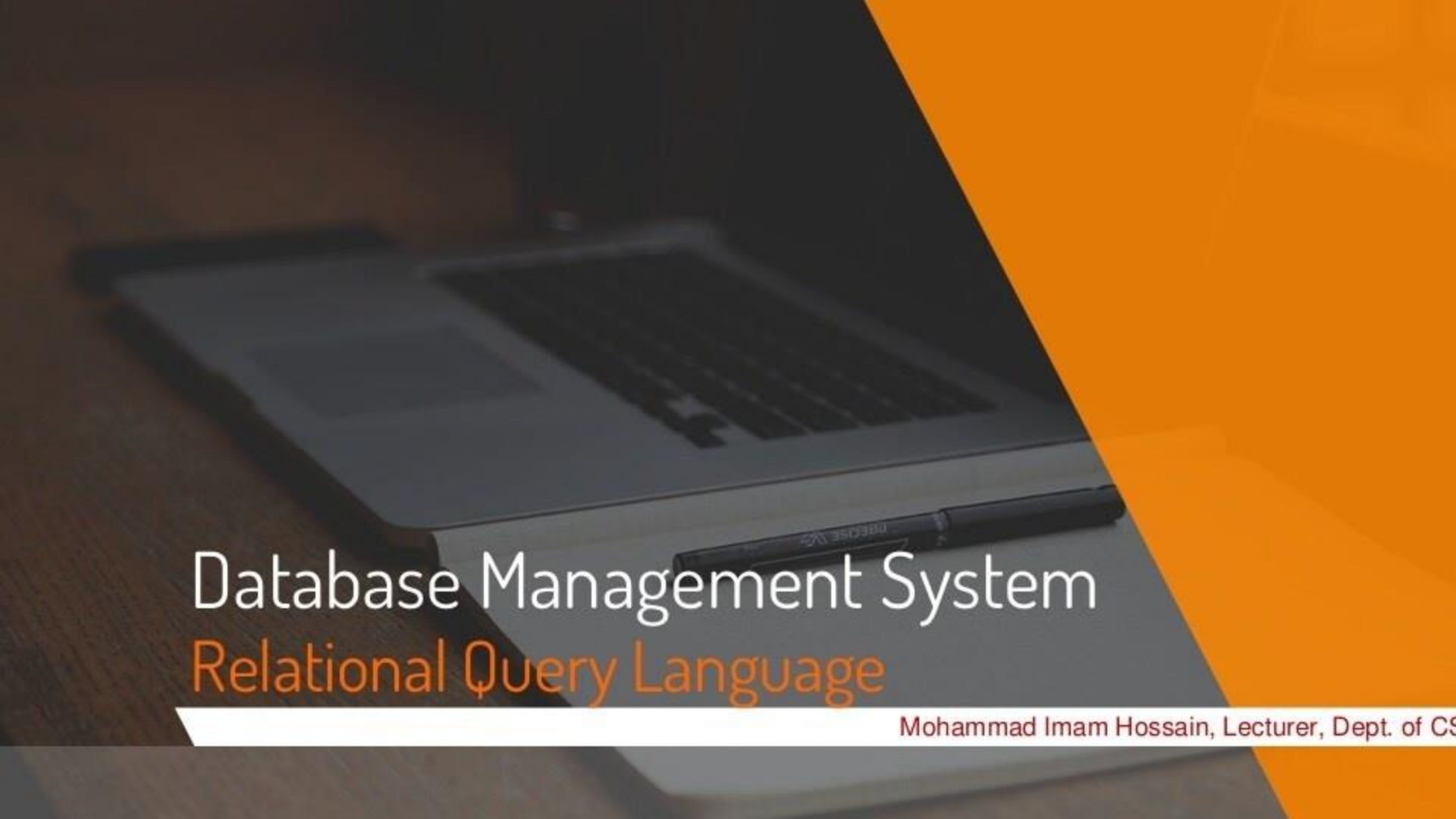
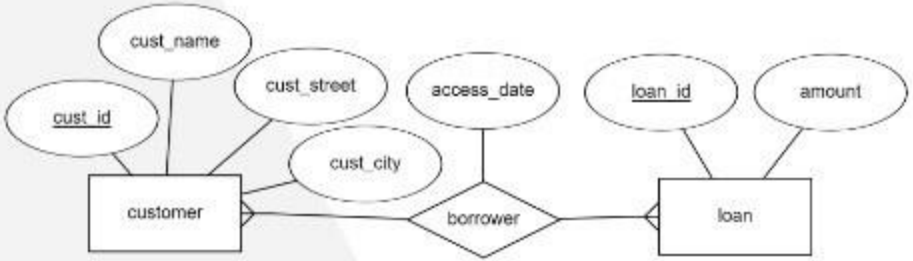


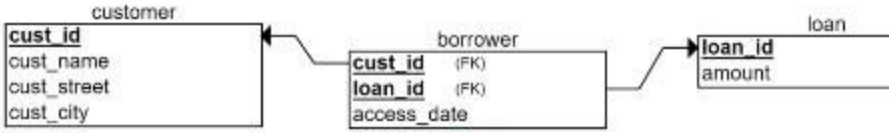
Database Management System

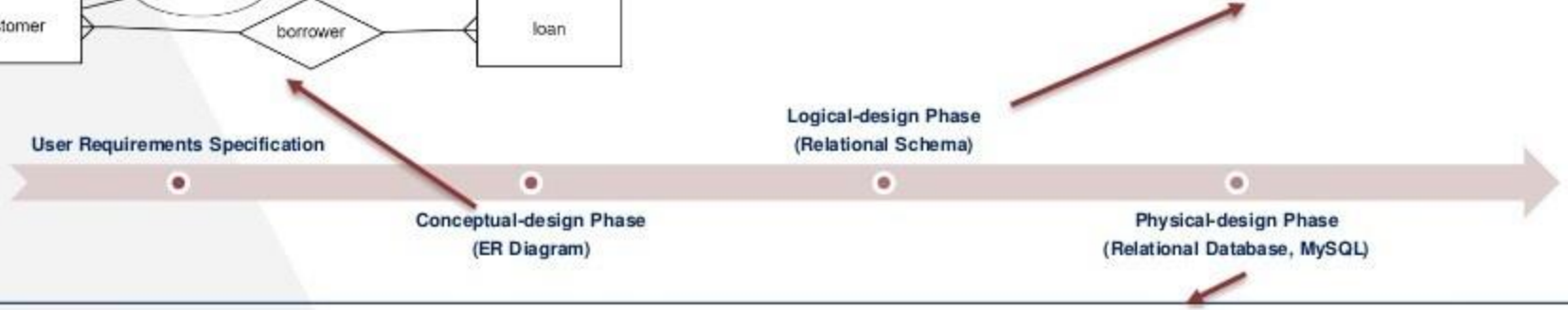
Relational Query Language

Mohammad Imam Hossain, Lecturer, Dept. of CS

Design Phases





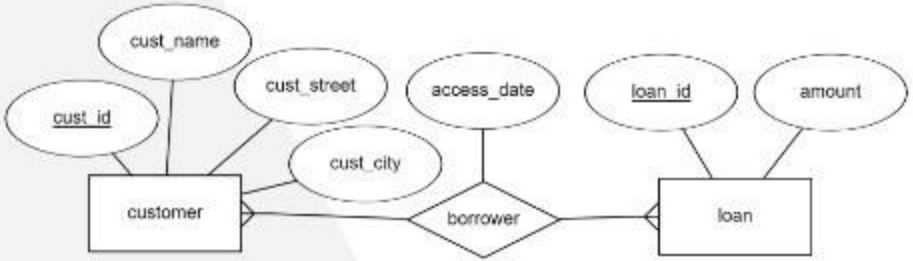


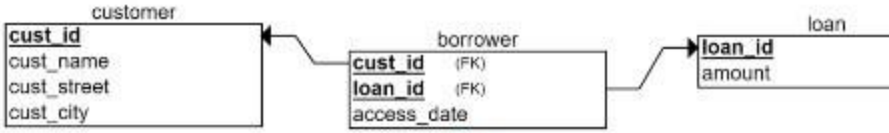

```sql
CREATE TABLE customer
(
  cust_id INT NOT NULL,
  cust_name VARCHAR(30) NOT NULL,
  cust_street VARCHAR(100) NOT NULL,
  cust_city VARCHAR(100) NOT NULL,
  PRIMARY KEY (cust_id)
);
```

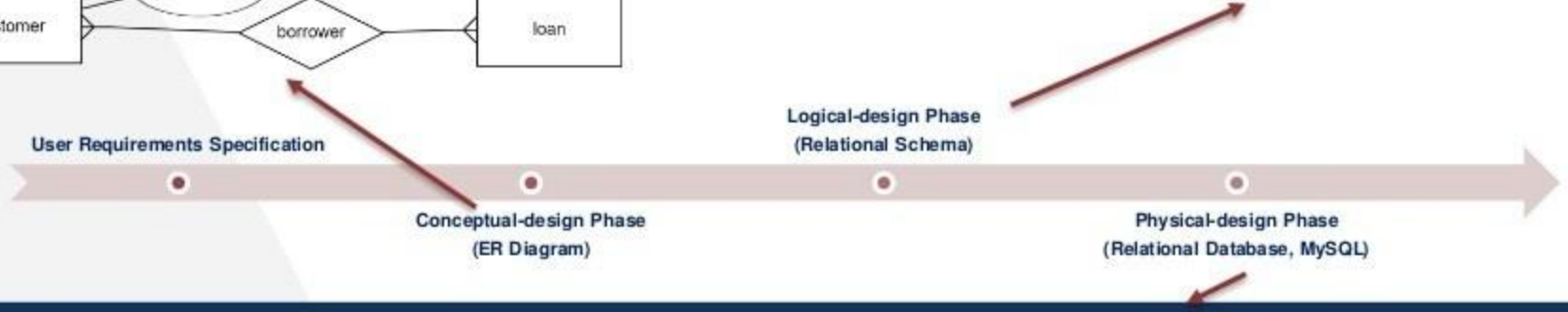
```sql
CREATE TABLE borrower
(
  access_date DATE NOT NULL,
  cust_id INT NOT NULL,
  loan_id INT NOT NULL,
  PRIMARY KEY (cust_id, loan_id),
  FOREIGN KEY (cust_id) REFERENCES customer(cust_id),
  FOREIGN KEY (loan_id) REFERENCES loan(loan_id)
);
```

```sql
CREATE TABLE loan
(
  loan_id INT NOT NULL,
  amount NUMERIC NOT NU
  PRIMARY KEY (loan_id)
);
```

Structured Query Language



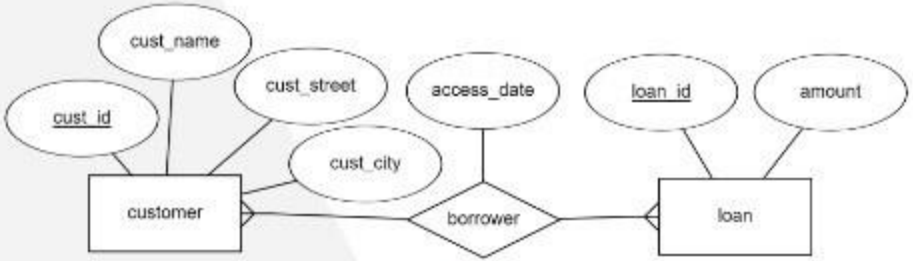




Structured Query Language (SQL)

- Domain-specific programming language
- Highly targeted language for talking to databases

Relational Query Language



Relational Query Language

- Query language
  – allows manipulation and retrieval of data from a datab
- Uses relational algebra to communicate with the databa

Structured Query Language (SQL)

- Domain-specific programming language
- Highly targeted language for talking to databases

Relational Algebra

Relational Algebra

- Relational Algebra is a procedural language consisting of a set of operations that take one or two relations as input produce a new relation as their result.
- Six basic operations:
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: $-$
  - Cartesian product: $\times$
  - rename: ρ
- Additional operations:
  - set intersection: $\cap$
  - division: $\div$
  - assignment: $\leftarrow$
  - aggregate: $\mathcal{G}$
  - Natural join: $\bowtie$
  - Theta join: $\bowtie_\theta$
  - Outer join: $\supset\bowtie$ $(left)$, $\bowtie\subset(right)$, $\supset\bowtie\subset$ $(full)$

project Operation

project Operation

- Notation: $\prod_{A_1, A_2, \ldots, A_k}(r)$
- $A_1, A_2, \ldots, A_k$ are attribute names
- r is a relation name
- Duplicate rows are removed from the result, since relations are sets

A	B	C	D
p	p	10	70
p	q	50	70
q	q	12	30
q	q	25	15

Relation, r

```
SELECT    A, C
FROM      r
```

MySQL

$$\prod_{A,C}(r)$$

A	C
p	10
p	50
q	12
q	25

output

select Operation

select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as: $\sigma_p(r) = \{t \mid t \in r \; and \; p(t)\}$
- Operators: $=, \neq, >, \geq, <, \leq, \wedge (and), \vee (or), \neg (not)$

A	B	C	D
p	p	10	70
p	q	50	70
q	q	12	30
q	q	25	15

Relation, r

```
SELECT    *
FROM      r
WHERE     A=B AND D>20
```

MySQL

$$\sigma_{A=B \wedge D>20}(r)$$

A	B	C	D
p	p	10	7
q	q	12	3

output

select Operation

select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as: $\sigma_p(r) = \{t \mid t \in r \; and \; p(t)\}$
- Operators: $=, \neq, >, \geq, <, \leq, \wedge (and), \vee (or), \neg (not)$

A	B	C	D
p	p	10	70
p	q	50	70
q	q	12	30
q	q	25	15

Relation, r

```
SELECT    A, C, D
FROM      r
WHERE     A=B AND D>20
```

MySQL

$$\prod_{A,C,D} (\sigma_{A=B \wedge D>20}(r))$$

A	C	D
p	10	70
q	12	30

output

rename Operation

rename Operation

- Notation: $\rho_{X(A_1,A_2,\ldots,A_n)}(E)$
- It returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$

A	B	C	D
p	p	10	70
p	q	50	70
q	q	12	30
q	q	25	15

Relation, r

```
SELECT    A AS P,
          C AS Q,
          D AS R

FROM      r AS r1
WHERE     A=B AND D>20
```

MySQL

$$\rho_{r_1(P,Q,R)}(\prod_{A,C,D}(\sigma_{A=B \wedge D>20}(r)))$$

P	Q	R
p	10	70
q	12	30

Relation, r_1

output

Cartesian-product Operation

Cartesian-product Operation (Cross Join)

- Notation: $r \times s$
- Defined as: $r \times s = \{ tq \mid t \in r \; and \; q \in s \}$
- Assume that attributes of r and s are disjoint.
- If attributes of r and s are not disjoint, then renaming must be used.

A	B
p	1
q	2

Relation, r

C	D	E
p	10	a
q	10	a
q	20	b
r	10	b

Relation, s

```
SELECT    *
FROM      r
          JOIN
          s
```

MySQL

$r \times s$

A	B	C	D
p	1	p	10
p	1	q	10
p	1	q	20
p	1	r	10
q	2	p	10
q	2	q	10
q	2	q	20
q	2	r	10

output

Cartesian-product Operation

Cartesian-product Operation (Cross Join)

- Notation: $r \times s$
- Defined as: $r \times s = \{ tq \mid t \in r \; and \; q \in s \}$
- Assume that attributes of r and s are disjoint.
- If attributes of r and s are not disjoint, then renaming must be used.

A	B
p	1
q	2

Relation, r

C	D	E
p	10	a
q	10	a
q	20	b
r	10	b

Relation, s

```
SELECT    *
FROM      r
          JOIN
          s
WHERE     A=C
```

MySQL

$$\sigma_{A=C}(r \times s)$$

A	B	C	D
p	1	p	10
~~p~~	~~1~~	~~q~~	~~10~~
~~p~~	~~1~~	~~q~~	~~20~~
~~p~~	~~1~~	~~r~~	~~10~~
~~q~~	~~2~~	~~p~~	~~10~~
q	2	q	10
q	2	q	20
~~q~~	~~2~~	~~r~~	~~10~~

output

Natural Join Operation

Natural Join Operation

- Notation: $r \bowtie s$
- Matches all the common column values.

A	B	C	D
p	1	p	a
q	2	r	a
r	4	q	b
p	1	r	a
s	2	q	b

Relation, r

B	D	E
1	a	p
3	a	q
1	a	r
2	b	s
3	b	t

Relation, s

```
SELECT    *
FROM      r
          NATURAL JOIN
          s
```

MySQL

$r \bowtie s$

A	B	C	D
p	1	p	a
p	1	r	a
p	1	p	a
p	1	r	a
s	2	q	b

output

Theta Join Operation

Theta Join Operation (Inner Join)

- Notation: $r \bowtie_\theta s$
- Matches the θ condition.

A	B
p	1
q	2

Relation, *r*

C	D	E
p	10	a
q	10	a
q	20	b
r	10	b

Relation, *s*

```
SELECT    *
FROM      r
          JOIN
          s
          ON r.A=s.C
```

MySQL

$$r \bowtie_{r.A=s.C} s$$

A	B	C	D
p	1	p	10
q	2	q	10
q	2	q	20

output

Outer Join Operation

Outer Join Operation

- Avoids loss of information.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result the join.
- Uses *null* values.

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation, *loan*

customer_name	l_no
Jones	L-170
Smith	L-230
Hayes	L-155

Relation, *borrower*

Outer Join Operation – Left Outer Join

Left Outer Join Operation

- Notation: $\supset\!\bowtie$

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation, *loan*

customer_name	l_no
Jones	L-170
Smith	L-230
Hayes	L-155

Relation, *borrower*

```
SELECT    *
FROM      loan
          LEFT JOIN
          borrower
          ON
          loan.loan_number=borrower.l_no
```

MySQL

$$loan \supset\!\bowtie_{(loan.loan\_number=borrower.l\_no)} borrower$$

loan_number	branch_name	amount	customer_name	l_n
L-170	Downtown	3000	Jones	L-17
L-230	Redwood	4000	Smith	L-23
L-260	Perryridge	1700	null	nul

output

Outer Join Operation – Right Outer Join

Right Outer Join Operation

- Notation: ⟖

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation, *loan*

customer_name	l_no
Jones	L-170
Smith	L-230
Hayes	L-155

Relation, *borrower*

```
SELECT    *
FROM      loan
          RIGHT JOIN
          borrower
          ON
          loan.loan_number=borrower.l_no
```

MySQL

$$loan \text{ ⟖}_{(loan.loan\_number=borrower.l\_no)} borrower$$

loan_number	branch_name	amount	customer_name	l_n
L-170	Downtown	3000	Jones	L-17
L-230	Redwood	4000	Smith	L-23
null	null	null	Hayes	L-15

output

Outer Join Operation – Full Outer Join

Full Outer Join Operation

- Notation: $\supset\!\bowtie\!\subset$

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation, *loan*

customer_name	l_no
Jones	L-170
Smith	L-230
Hayes	L-155

Relation, *borrower*

```
SELECT      *
FROM        loan
            FULL OUTER JOIN
            borrower
            ON
            loan.loan_number=borrower.l_no
```

Oracle Supported

$$loan \supset\!\bowtie\!\subset_{(loan.loan\_number = borrower.l\_no)} borrower$$

loan_number	branch_name	amount	customer_name	l_n
L-170	Downtown	3000	Jones	L-17
L-230	Redwood	4000	Smith	L-23
L-260	Perryridge	1700	null	nul
null	null	null	Hayes	L-15

output

Aggregate Operation

Aggregate Function

- Takes a collection of values and returns a single value as a result.
- Functions: *avg, min, max, sum, count*

Aggregate Operation

- Notation: $_{G_1, G_2, \ldots, G_n}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(\mathrm{E})$
- E is any relational-algebra function.
- Each F_i is an aggregate function.
- Each A_i is an attribute name.

A	B	C
p	p	10
p	q	50
q	q	12
q	q	25

Relation, *r*

```
SELECT    SUM(C)
FROM      r
```

MySQL

$\mathcal{G}_{SUM(C)}(\mathrm{r})$

SUM(C)
97

output

Aggregate Operation

branch_name	account_number	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

Relation, *account*

```
SELECT      branch_name,
            SUM(balance) AS sum_balance
FROM        account
GROUP BY    branch_name
```

MySQL

$$_{branch\_name}\mathcal{G}_{SUM(balance)\ AS\ sum\_balance}(\text{account})$$

branch_name	sum_balance
Perryridge	1300
Brighton	1500
Redwood	700

output

union Operation

union Operation

- **Notation:** $r \cup s$
- **Defined as:** $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

A	B
p	1
p	2
q	1

Relation, *r*

A	B
p	2
q	3

Relation, *s*

```
SELECT  *
FROM    r
UNION
SELECT  *
FROM    s
```

MySQL

$r \cup s$

A	B
p	1
p	2
q	1
q	3

output

set difference Operation

set difference Operation

- Notation: $r - s$
- Defined as: $r - s = \{t \mid t \in r \; and \; t \notin s\}$

A	B
p	1
p	2
q	1

Relation, *r*

A	B
p	2
q	3

Relation, *s*

```
SELECT  *
FROM    r
MINUS
SELECT  *
FROM    s
```

ORACLE Supported

$r - s$

A	B
p	1
q	1

output

set intersection Operation

set intersection Operation

- **Notation:** $r \cap s$
- **Defined as:** $r \cap s = \{t \mid t \in r \text{ and } t \in s\}$

A	B
p	1
p	2
q	1

Relation, *r*

A	B
p	2
q	3

Relation, *s*

```
SELECT    *
FROM      r
INTERSECT
SELECT    *
FROM      s
```

ORACLE Supported

$r \cap s$

A	B
p	2

output

Practices

A	B	C1	A1
1	2000	2	3
2	2500	1	1
3	2200	2	2
4	2000	2	2

Relation, t1

C	D
1	ios
2	web

Relation, t2

Convert to equivalent Relational Algebra:

```
SELECT    N.D, COUNT(M.A) AS cnt
FROM      t1 AS M
          JOIN
          t2 AS N
          ON M.C1=N.C
WHERE     N.C>1
GROUP BY  N.C, N.D
HAVING    cnt>2
```

Practices

A	B	C	D
100	David	Jones	20000
102	Loren	Ipsum	40000
103	Chris	Stanley	10000

Relation, r

C	D	E
Jones	20000	101
Stanley	10000	101
Ipsum	30000	103

Relation, s

B	E
David	101
Felix	101
Chris	100

Relation, t

Simulate all the following Relational Algebra Expression. You have to show each and every steps of the simulations:

- $_{t1.D}\mathcal{G}_{count(t2.D)}(\sigma_{t1.D<t2.D}(\rho_{t1}(r) \times \rho_{t2}(r)))$
- $\sigma_{E>100}((r \bowtie s) \cup (r \bowtie t))$
- $(s \rightarrowtail t) \cap (s \bowtie t)$

THANKS!

Any questions?

Email : imam@cse.uiu.ac.bd

References:

- Database System Concepts by S. Sudarshan, Henry F. Korth, Abraham Silberschatz