Relational Algebra:

Selection (σ):
Selects tuples that satify certain conditions.
<u>Operators:</u> −, =, ≠, ≥, <, >, ≤ <u>Connectors:</u> *and*, *or*, *not*

$\sigma_{\text{sentiment > 0.9}}$(Chirp) — Selects Chirps with sentiment less than 0.9.
$\sigma_{\text{last\_name = 'Trump'}}$(Bird) — Selects Birds whose last name is Trump.

Projection (π):
Projects a subset of a table's columns.

$\pi_{\text{btag, first\_name}}$(Bird) — Projects the tag and first name of all Birds

***Cross-Product (×):**
Combines two relations with every possible combination of tuples.

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R CROSS S

A	1	A	1
A	1	C	2
A	1	D	3
A	1	E	4
B	2	A	1
B	2	C	2
B	2	D	3
B	2	E	4
D	3	A	1
D	3	C	2
D	3	D	3
D	3	E	4

F	4	A	1
F	4	C	2
F	4	D	3
F	4	E	4
E	5	A	1
E	5	C	2
E	5	D	3
E	5	E	4

***Difference (−):**
Selects tuples that are present in one relation but not the other.

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

***Union (∪):**
Selects tuples that are present in both relations.

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R UNION S

A	1
B	2
C	2
D	3
E	5
F	4
E	4

Natural Join (⋈):
Combines two relations by finding a common attribute between them

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee ⋈ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

Conditional Join ($\bowtie_c$):
Combines two relations similar to cross product but with a condition

Car

CarModel	CarPrice
CarA	20,000
CarB	30,000
CarC	50,000

Boat

BoatModel	BoatPrice
Boat1	10,000
Boat2	40,000
Boat3	60,000

$Car \underset{CarPrice \geq BoatPrice}{\bowtie} Boat$

CarModel	CarPrice	BoatModel	BoatPrice
CarA	20,000	Boat1	10,000
CarB	30,000	Boat1	10,000
CarC	50,000	Boat1	10,000
CarC	50,000	Boat2	40,000

Division (÷):
Reduces a relation by performing the opposite of a cartesian product

Completed

Student	Task
Fred	Database1
Fred	Database2
Fred	Compiler1
Eugene	Database1
Eugene	Compiler1
Sarah	Database1
Sarah	Database2

DBProject

Task
Database1
Database2

Completed
÷
DBProject

Student
Fred
Sarah

*Must be union compatible: 1) Same number of columns 2) Corresponding columns are of the same variable type

Relational Calculus [Examples]:

```
Sailors(sid, sname, rating, age),  Reserves(sid, bid, date),  Boats(bid, bname, color)
```

1. Find sailors with a rating > 7
   $\{s \mid s \in Sailors \wedge s.rating > 7\}$
2. Find names of sailors who've reserved a red boat
   $\{t(sname) \mid \exists s \in Sailors(t.sname = s.sname \wedge \exists r \in Reserves(r.sid = s.sid \wedge \exists b \in Boats(b.bid = r.bid \wedge b.color = 'red')))\}$
3. Find the names of sailors who've reserved all "Interlake" boats
   $\{t(sname) \mid \exists s \in Sailors(t.sname = s.sname \wedge \forall b \in Boats(b.bname = 'Interlake' \rightarrow (\exists r \in Reserves(r.sid = s.sid \wedge b.bid = r.bid))))\}$

MySQL:

Queries:

SELECT
- **DISTINCT**
- T.attr, T.attr **as** attribute
- **COUNT**(*)
- **MAX**(T.attr), **MIN**(T.attr), **AVG**(T.attr)

FROM
- Table T
- Table2 T2

WHERE
- =, !=, >, <, >=, <=
- [NOT] **IN**
- [NOT] **EXISTS**
- **ANY**
- **ALL**
- **LIKE** _ (1 char), % (n chars)

GROUP BY
- attribute

HAVING
- condition on grouping

Insertions:

```
INSERT INTO table_name (field1, field2, ...fieldN)
                       VALUES
                       (value1, value2, ...valueN);
```

Deletions:

```
DELETE FROM table_name [WHERE clause]
```

Views:

```
CREATE VIEW view_name(attr1, attr2, ...) AS
        SELECT []
        FROM []
        WHERE []
```

Procedures: **CALL** Procedure(params)

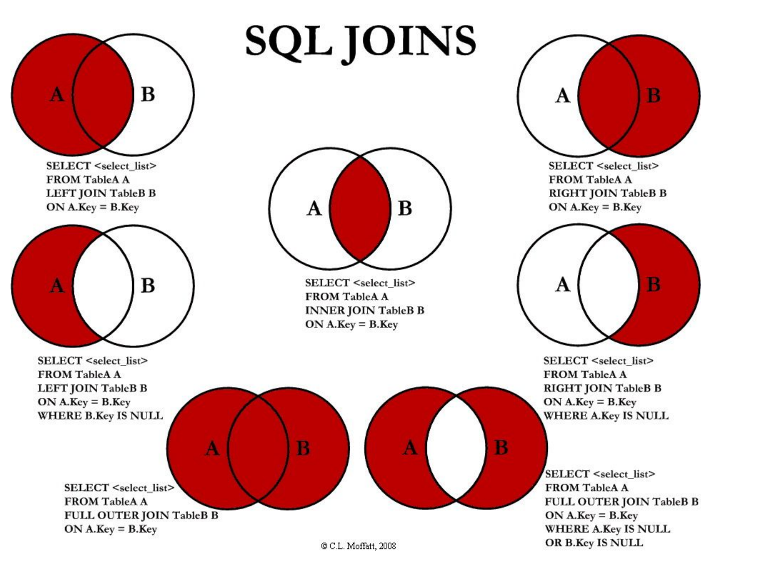
```
CREATE PROCEDURE NewChirp(
    new_btag VARCHAR(30),
    content VARCHAR(255))
BEGIN
    DECLARE new_cno INT(11);
    SET new_cno = (SELECT MAX(cno)+1 FROM Chirp
                   WHERE btag = new_btag);
    INSERT INTO Chirp(btag, cno, content)
    VALUES (new_btag, new_cno, content);
END
```

Alter Table:

```
DROP COLUMN col_name
DROP PRIMARY KEY
DROP FOREIGN KEY fk_name
```

Update Table:

```
UPDATE [LOW_PRIORITY] [IGNORE] table
SET column1 = expression1,
    column2 = expression2,
    ...
[WHERE conditions]
[ORDER BY expression [ASC | DESC]]
[LIMIT number_rows]
```



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```