================ **Relational Algebra** ================

THETA JOIN:
$R_1 \bowtie_C R_2 = \sigma_C(R_1 \times R_2)$

UNION, DIFFERENCE, INTERSECT Operators:

- Schemas must be the same
- No duplicates remain

INTERSECT:
$R \cap S = R - (R - S)$

DIVISION:
$R/S = \pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$
R/S is the largest relation T such that $T \times S \subseteq R$

===================== **SQL** =====================

General statement:
SELECT attributes, aggregates
FROM relations(tables)
WHERE conditions
GROUP BY attributes
HAVING conditions on aggregates
ORDER BY attributes, aggregates

Evaluation order:
FROM → WHERE → GROUP BY → HAVING → ORDER BY →SELECT

Set Operators:
INTERSECT, UNION, EXCEPT

- Follow set semantics and remove duplicates
- To keep duplicates: UNION ALL, INSERSECT ALL, EXCEPT ALL

Set membership

- IN, NOT IN

Set comparison operator

- ALL, < SOME, = SOME,... etc.

Subqueries

- in WHERE clause
  - Considered as a regular relation
  - One-attribute one-tuple relation -> use like a 'value'
- In FROM clause
  - Considered as a regular relation
  - Must be renamed to a new table name

Aggregates

- Sum, Count, Avg, Min, Max, …

INSERT, DELETE, UPDATE

- Insertion: INSERT INTO <Relation> <Tuples>
- Deletion: DELETE FROM <R> WHERE <Condition>
- Update: Update R
  SET A_1 = V_1, A_2 = V_2, …, A_n = V_n
  WHERE <Condition>

================ **Database Integrity** ================

Key Constraints
    CREATE TABLE <name> (
    …,
    PRIMARY KEY(dept, cnum, sec),
    UNIQUE(dept, cnum, instructor))

Referential Integrity (Foreign Key)

- E.A references S.A
  - E.A: referencing attribute / foreign key
  - S.A: referenced attribute
- CREATE TABLE <name> (
  …,
  sid INTEGER REFERENCES Student(sid),
  FOREIGN KEY (dept, cnum, sec) REFERENCES Class(dept, cnum, sec)
- Referenced attributes must be PRIMARY KEY or UNIQUE
- RI Violation
  - Default: not allowed
    - System rejects the statement
    - Always insert/update S first
  - ON DELETE/UPDATE SET NULL/SET DEFAULT/CASCADE
    - Added on Referencing attributes declaration
    - SET NULL/SET DEFAULT
      - Change E.A value to NULL or default value
    - CASCADE
      - On deletion of S: delete referencing tuples in E
      - On update of S.A: change E.A to the new S.A

Check Constraints

- CRATE TABLE Enroll (
  dept CHAR(2), cnum INT, unit INT,
  title VARCHAR(50),
  CHECK (cnum < 600 AND unit < 10))

Triggers
CREATE Trigger <name>
    <event>
    <referencing clause>
    WHEN (<condition>)
    <action>

<event>

- BEFORE | AFTER INSERT/DELETE/UPDATE [OF A1, A2, …, An] ON R

<referencing clause>

- REFERENCING OLD | NEW TABLE | ROW AS <var>, …
- FOR EACH ROW|STATEMENT

<action>

- Any SQL statement

================ **Views and Authorization** ================

Views
    CREATE VIEW <name> AS
    <Query>

Authorization

- GRANT <privileges> ON <R> TO <user> [WITH GRANT OPTION]
- REVOKE <privileges> ON <R> FROM <user> [CASCADE | RESTRICT]

================ **Misc** ================

Logical Implies in SQL
$p \rightarrow q \equiv \neg p \lor q$

================ Files and Disks ================

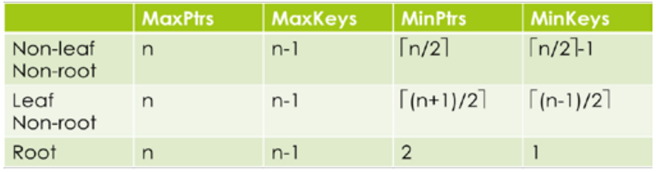
Access time = (seek time) + (rotational delay) + (transfer time)

- Seek time: time to find the target track
  - Typical average seek time: 10 ms
- Rotational delay: time to rotate to the target sector
  - For 6000 RPM, average rotational delay=0.5*(1min/6000)=0.5*60sec/6000=5 ms
- Transfer Time
  - Time to read one block
  - For example, 6000 RPM, 1000 sector/track, 1KB/sector
    - Read a track, rotate a circle: 1min/6000 = 10 ms/track
    - Read one sector(block): (10ms/track) / (1000sector/track) = **0.01ms/sector**
    - Transfer rate: 1KB/(0.01ms/sector)=**100MB/s**

================ B+ Trees ================

Insertion:

- Leaf node overflow
  - The first key of the new node is ***copied*** to the parent
- Non-leaf node overflow
  - The middle key is ***moved*** to the new parent



	MaxPtrs	MaxKeys	MinPtrs	MinKeys
Non-leaf Non-root	n	n-1	$\lceil n/2 \rceil$	$\lceil n/2 \rceil$-1
Leaf Non-root	n	n-1	$\lceil (n+1)/2 \rceil$	$\lceil (n-1)/2 \rceil$
Root	n	n-1	2	1