Relational Data Model

Relational Table requires Name and Schema. **Database** is a collection of Relations tables. Record, tuple is a single row. Schema is the name of relation plus set of attributes of the relation. Relation instance is a set of tuples for a given relation. Cardinality is # of tuples. Degree is # of attributes. **DBMS**is a software designed to maintain and support databases.

Constraints

- Candidate key is a set of attributes that uniquely defines a record in table, minimal such set set of attributes (every attribute is necessary)
- Superkey - set of attributes that uniquely defines a record in table
- Primary key - candidate key chosen to be used as primary key (can have multiple candidate keys as one primary key)

DDL

Commands act upon the schema. CREATE, DROP, ALTER. To define relational table you need table name, attributes (name and type) and constraints.

```
CREATE TABLE <table name> (<attribute name> <type> <constraint list>)
DROP TABLE <table name>
ALTER TABLE <table name> ADD <attribute name> <type> <constraint list>
ALTER TABLE <table name> DROP <attribute name>
```

Put UNIQUE after attribute name to make it a candidate key. Put PRIMARY KEY after attribute name to make it a primary key. Put FOREIGN KEY after attribute name to make it a foreign key.

DML

Commands act upon the data. INSERT, DELETE, UPDATE, SELECT.

```
SELECT <attribute list> FROM <table list> WHERE <condition>
UPDATE <table list> SET <attribute list> WHERE <condition>
DELETE FROM <table list> WHERE <condition> -- just with table name deletes all rows
INSERT INTO <table list> VALUES <value list>
```

Relational Algebra

Doing these removes duplicates since these are sets. Try not to look through entire table when solving problems.

- Selection - $\sigma_{\text{C}}(R)$ - returns rows that satisfy C
- Projection - $\pi_{\text{attributes}}(R)$ - returns columns that are in attribute list, no duplicates
- Cartesian Product - $R \times S$ - returns all combinations of rows (match every row in R with every row in S)
- Rename - $\rho_{\text{new}}(R)$ - used to do self joins, once original renamed they are forgotten for the duration of the operation
- Duplication Elimination - $\delta(R)$ - removes duplicates, enables us to between set operations and bag operations
- Sort - $\tau_{\text{C}}(R)$ - sorts rows of R based on C, $F = Desc(A\%B), B, Desc(C)$

Joins

- Theta Join (Equi-Join) - $R \bowtie_{\mathrm{C}} S$ - returns all combinations of rows that satisfy C, compare every combination **Keep columns since there is no projection**, $R \bowtie_{\Theta} S = \sigma_{\Theta}(R \times S)$
- Natural Join - $R \bowtie S$ - returns all combinations of rows that match on common attributes, **removes one set of common attributes from the final relation**
- Left/Right Semi Join - $R \ltimes \rtimes S$ - only attributes of one relation are kept, projection on all elements of one relation

Set Operations

Only apply these when R and S have the same schema. These are bag operations.

- Union - $R \cup S$ - combine rows of R and S, **remove duplicates**
- Set Difference - $R - S$ - keep rows that are unique to R
- Intersection - $R \cap S$ - keep rows that are in both R and S

Types

- **Numeric Types**
  - **Integer Types**
    - TINYINT
    - SMALLINT
    - MEDIUMINT
    - **INT**
    - BIGINT
  - **Floating Point Types**
    - **FLOAT**
    - **DOUBLE(P, D)**
    - **DECIMAL**
- **String Types**
  - **Character Types**
    - **CHAR(N)** $\longrightarrow$ **Fixed Length**
    - **VARCHAR(N)** $\longrightarrow$ **Variable Length**
    - TINYTEXT
    - **TEXT** $\longrightarrow$ for storing large amounts of text
    - MEDIUMTEXT
    - LONGTEXT
- **Date and Time Types**
  - **Date Types**
    - DATE
    - DATETIME
    - TIMESTAMP
    - TIME
    - YEAR