# Relational Database Model

## Basic Structure

- Relation/Table
    - Named
    - No repeating fields
- Columns
    - named attributes
    - must be atomic values
    - values valid within a domain
- Rows
    - also called tupules
    - similar to record
    - must have primary key
- Keys
    - Super key: attribute(s) that identify a tupule. always at least one. relation can have more than one
    - Candidate key: min set of attributes that uniquely identify a tupule. minimal superkey. may be more than one.
    - Primary key: one per relation only. choosen candidate key

## Examples

Employee (Emp-ID, Emp-Name, Emp-Birthdate, Emp-Address, Emp-Salary)

- 🟪 -Super key
- 🟦 -Candidate key
- 🟥 -Primary key

Employee_Project (Emp-ID, Project-ID, Emp-Title-Proj, Hours-Worked)

## Foreign key

- used to reference another relation

Terminology

Domain - set of atomic valid values of one or more attribute.
may be specified as a data type

Atomic Values - invisible data values

Null Value - designates a missing value

Degree - number of attributes (columns) in a relation

Cardinality - number of rows in a relation

Intention - schema of a relation

Extension - data (tupules) in a relation

Characteristics of a Relation

1) ordering of rows w/in a relation
no particular order. unordered set

2) ordering of attributes w/in a relation
no particular order as long as attribute and value
is maintained

Relational Constraints

Domain/Integrity Constraints
specify the valid values of each attribute

Entity Integrity Constraints
states that no attribute of a primary key can contain a null value

Referential Integrity Constraint
a foreign key can contain a valid value of the primary key
in the home relation OR contain a NULL value

Relational Operators

Update operators - insert, delete, modify

Retrieval operators - relational algebra, relational calculus, SQL

SQL (structured query language)

Comprehensive database language

data definition language ← DDL, DML, view definition,

DML → data manipulation language

DML → SELECT, INSERT, UPDATE, DELETE

DDL → CREATE, DROP, ALTER

# Normalization to 3NF

Redundancy
  when attribute values are repeated unnecessarily

Anomolies
  caused by redundant info. must find all copies of info in order to prevent inconsitencies when updating
  Deletion Anomoly
    occurs when data is lost during a deletion that we still went
  Insertion Anomoly
    occurs when we cannot insert some info into a row because of a violation

Functional Dependency
  constraints in the data that depend upon whether or not two tupules agree on certain components
  Person (SSN, Age, Gender)

$$FD = \{ SSN \rightarrow Age,\ SSN \rightarrow Gender,\ Age \nrightarrow Gender \}$$

Since two people of the same age can be of different genders

Normal values
  1NF (first Normal Form)
    all values are atomic
  2NF (Second Normal Form)
    in 2NF if it's in 1NF and each ~~[illegible]~~ non-prime attributes are fully dependent on PK
  3NF (Third Normal Form)
    in 3NF if it's in 2NF and none of its non-prime attributes are "transitively dependent" on it's key
      ↳ np attribute is "—" upon the pk of a relation if there is also a ~~[illegible]~~ np attribute that functionally determines the attribute

prime attribute - attribute that appears in a key of a relation
non-prime attribute - attribute that doesn't appear in a key of a relation

# SQL

Create Table Syntax          name of attribute in table to be identified

```
    CREATE TABLE <table_name> (
        <attribute> <type>        can be integer, float/real, decimal(I,J),
            [not null][unique][pk])        char(N), varchar(N).
        {, <attribute> <type>
            [not null][unique][pk]}
        [, <pk constraint>]
        [{, <foreign key constraint>}]);
```

Examples

```
  CREATE TABLE person(
       ssn char(9) primary key,
       fname char(10) not null,
       lname char(10) not null,
       phone# char(10));
  CREATE TABLE student (
        ssn char(9),
        classification char(6),
        gpa decimal(4,3),
        total_hours integer,
          primary key (ssn),
          foreign key (ssn) references person (ssn));
```

ALTER TABLE syntax

```
  ALTER TABLE <table_name> add (
       <attribute> <type>
       {, <attribute> <type>});
  ALTER TABLE <table_name> modify (
       <attribute> ~~<type>~~ <new_length>
       {, <attribute> <new_length>});
```

Examples

```
    alter table person add (
      birthdate  char(8));
    alter table section modify (
      title   char(25),
       description   char(50));
```

DROP TABLE Syntax

```
    DROP TABLE <table_name>
         [cascade constraints];
```

Example

```
    DROP TABLE person;
         cascade constraints;
```

INSERT Syntax

```
    insert into <table_name>
          values (<value_list>);
    insert into <table_name>
         (<attribute_list>)
         values (<value_list>);
    insert into <table_name>
        select * from <another_table_name>;
```

Examples

```
   insert into student
       values ('298344', 'senior', 3.294, 110);
```

Delete syntax

```
   delete from <table_name>
       [where <condition>];
```

UPDATE syntax example

```
   update student
       set classification = 'senior'
       where total_hours > 90;
```