Relational Model

Relational Model Concepts

- The relational Model of Data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

Relational Model Concepts

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper: "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970.

The above paper caused a major revolution in the field of Database management and earned Ted Codd the coveted ACM Turing Award.

INFORMAL DEFINITIONS

- RELATION: A table of values
  - A relation may be thought of as a **set of rows**.
  - A relation may alternately be thought of as a **set of columns**.
  - Each row represents a fact that corresponds to a real-world **entity** or **relationship**.
  - Each row has a value of an item or set of items that uniquely identifies that row in the table.
  - Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
  - Each column typically is called by its column name or column header or attribute name.

FORMAL DEFINITIONS

- A **Relation** may be defined in multiple ways.
- The **Schema** of a Relation: R (A1, A2, ...An)

  Relation schema R is defined over **attributes** A1, A2, ...An

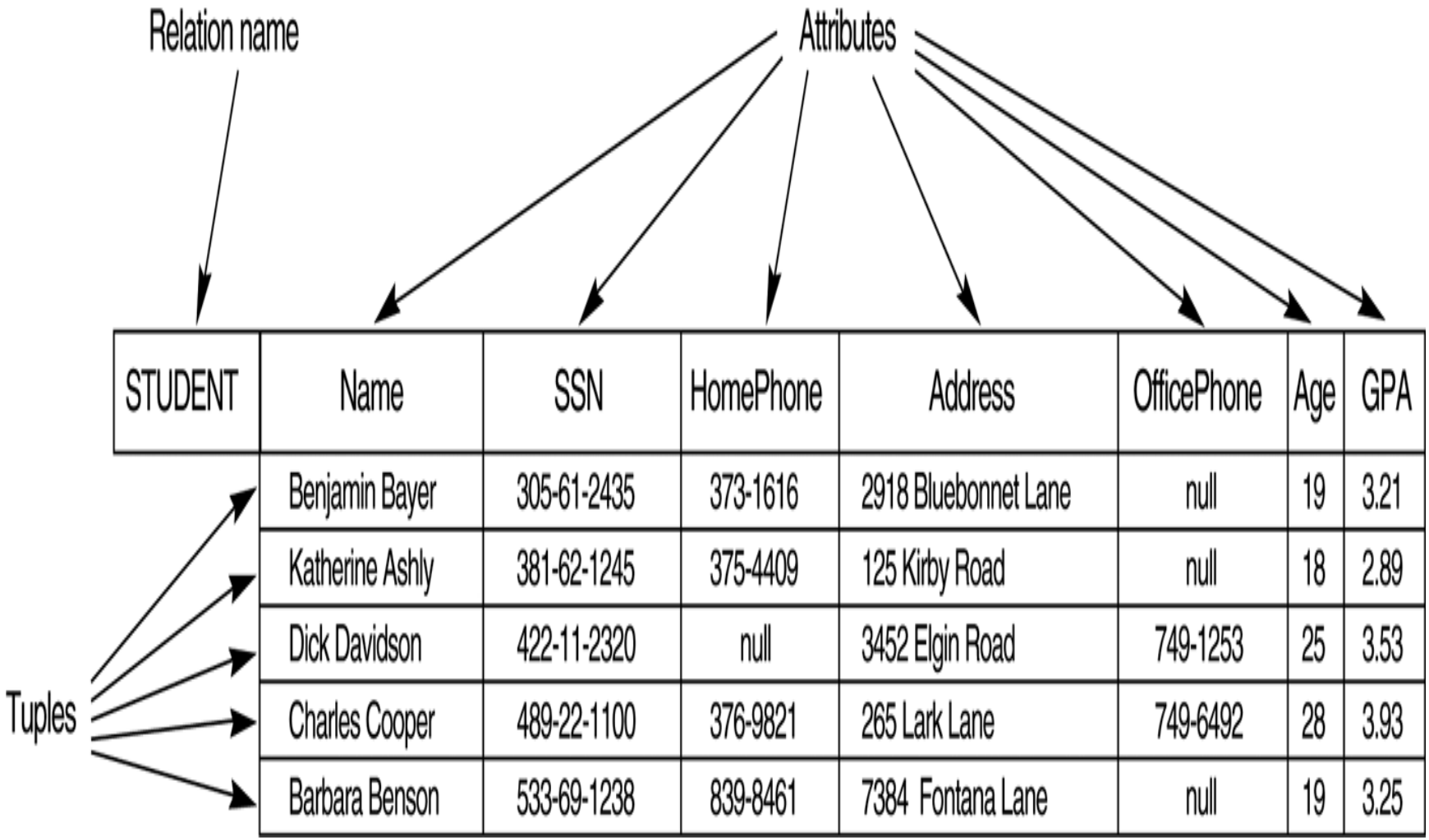
  For Example -

  CUSTOMER (Cust-id, Cust-name, Address, Phone#)

  Here, CUSTOMER is a relation defined over the four attributes Cust-id, Cust-name, Address, Phone#, each of which has a **domain** or a set of valid values. For example, the domain of Cust-id is 6 digit numbers.

DEFINITION SUMMARY

Informal Terms		Formal Terms
Table		Relation
Column		Attribute/Domain
Row		Tuple
Values in a column		Domain
Table Definition		Schema of a Relation



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

CHARACTERISTICS OF RELATIONS

- **Ordering of tuples in a relation r(R)**: The tuples are *not* considered to be ordered, even though they appear to be in the tabular form.
- **Ordering of attributes in a relation schema R** (and of values within each tuple): We will consider the attributes in $R(A_1, A_2, ..., A_n)$ and the values in $t=<v_1, v_2, ..., v_n>$ to be *ordered* .
  (However, a more general *alternative definition* of relation does not require this ordering).
- **Values in a tuple**: All values are considered *atomic* (indivisible). A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:
  1. **Key** constraints
  2. **Entity integrity** constraints
  3. **Referential integrity** constraints

DBMS Keys

- A key is an attribute (also known as column or field) or a combination of attribute that is used to identify records. Sometimes we might have to retrieve data from more than one table, in those cases we require to join tables with the help of keys. The purpose of the key is to bind data together across tables without repeating all of the data in every table.

  The various types of key with e.g. in SQL are mentioned below, (For examples let suppose we have an Employee Table with attributes 'ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')

For examples let suppose we have an Employee Table with attributes:

Employee ('ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')

The various types of key with e.g. in SQL are mentioned below:

1. Super Keys
2. Candidate Keys
3. Primary Keys
4. Foreign Keys (Referential Integrity)
5. Alternate Keys
6. Secondary Keys

Employee ('ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')

(I) Super Key – An attribute or a combination of attribute that is used to identify the records uniquely is known as Super Key. A table can have many Super Keys.
E.g. of Super Key
1 ID
2 ID, Name
3 ID, Address
4 ID, Department_ID
5 ID, Salary
6 Name, Address
7 Name, Address, Department_ID ……

- So on as any combination which can identify the records uniquely will be a Super Key.

Employee ('ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')

- **(II) Candidate Key** – It can be defined as minimal Super Key or irreducible Super Key. **In other words an attribute or a combination of attribute that identifies the record uniquely but none of its proper subsets can identify the records uniquely.**
  E.g. of Candidate Key
  1 ID
  2 Name, Address

- For above table we have only two Candidate Keys (i.e. **Irreducible Super Key**) used to identify the records from the table uniquely. ID Key can identify the record uniquely and similarly combination of Name and Address can identify the record uniquely, but neither Name nor Address can be used to identify the records uniquely as it might be possible that we have two employees with similar name or two employees from the same house.

- **(III) Primary Key** – **A Candidate Key that is used by the database designer for unique identification of each row in a table is known as Primary Key.** A Primary Key can consist of one or more attributes of a table.
  E.g. of Primary Key - Database designer can use one of the Candidate Key as a Primary Key. In this case we have "ID" and "Name, Address" as Candidate Key, we will consider "ID" Key as a Primary Key as the other key is the combination of more than one attribute.

Primary Key Syntax

```
CREATE TABLE STUDENT( ROLL_NO   INT  PRIMARY KEY);

CREATE TABLE STUDENT( ROLL_NO   INT, NAME CHAR(20),
PRIMARY KEY (ROLL_NO) );

CREATE TABLE STUDENT(NAME CHAR(20), ADDRESS CHAR(25),
CONSTRAINT PRI_K PRIMARY KEY (NAME, ADDRESS) );

CREATE TABLE STUDENT(ROLL_NO   INT, NAME CHAR(20),
CONSTRAINT PRI_K PRIMARY KEY (ROLL_NO) );

ALTER TABLE STUDENT ADD PRIMARY KEY (ROLL_NO);

ALTER TABLE STUDENT ADD CONSTRAINT PRI_K PRIMARY KEY
(ROLL_NO);

ALTER TABLE STUDENT DROP CONSTRAINT PRI_K;
```

- **(IV) Foreign Key** – **A foreign key is an attribute or combination of attribute in one base table that points to the candidate key (generally it is the primary key) of another table.** The purpose of the foreign key is to ensure **referential integrity** of the data i.e. only values that are supposed to appear in the database are permitted.

E.g. of Foreign Key – Let consider we have another table i.e. Department Table with Attributes "Department_ID", "Department_Name", "Manager_ID", "Location_ID" with Department_ID as an Primary Key. Now the Department_ID attribute of Employee Table (dependent or child table) can be defined as the Foreign Key as it can reference to the Department_ID attribute of the Departments table (the referenced or parent table), a Foreign Key value must match an existing value in the parent table or be NULL.

Foreign Key

- Record can't be inserted into a detail table if corresponding records in the master table do not exist.
- Record of the master table cannot be deleted if corresponding records in the detail table actually exists.

```
CREATE TABLE ACCOUNTS( ROLL_NO   INT CONSTRAINT FK
FOREIGN KEY REFERENCES STUDENT, DUE_AMOUNT INT);
```

```
CREATE TABLE ACCOUNTS( ROLL_NO   INT, DUE_AMOUNT
INT, CONSTRAINT FK FOREIGN KEY (ROLL_NO) REFERENCES
STUDENT(ROLL_NO) );
```

Master Table

Roll_No	Name
101	AA
202	BB
105	CC

Detail Table

Roll_No	Due_Amount
202	2000
101	1000
300	2000

Foreign Key

- If the ON DELETE CASCADE option is set, a DELETE operation in master table will trigger a DELETE operation for corresponding records in all detail tables.
- If the ON DELETE SET NULL option is set, a DELETE operation in master table will set the value set by the foreign key of the detail tables to NULL.

```
CREATE TABLE ACCOUNTS( ROLL_NO  INT, DUE_AMOUNT
INT, CONSTRAINT FK FOREIGN KEY (ROLL_NO)
REFERENCES STUDENT(ROLL_NO) ON DELETE CASCADE);
```

```
CREATE TABLE ACCOUNTS( ROLL_NO  INT, DUE_AMOUNT
INT, CONSTRAINT FK FOREIGN KEY (ROLL_NO)
REFERENCES STUDENT(ROLL_NO) ON DELETE SET NULL);
```

- **(V) Alternate Key** – Alternate Key can be any of the Candidate Keys except for the Primary Key.
  E.g. of Alternate Key is "Name, Address" as it is the only other Candidate Key which is not a Primary Key.

- **(VI) Secondary Key** – The attributes that are not even the Super Key but can be still used for identification of records (not unique) are known as Secondary Key.
  E.g. of Secondary Key can be Name, Address, Salary, Department_ID etc. as they can identify the records but they might not be unique.

CONSTRAINTS

- Constraints within a database are rules which control values allowed in columns and also enforce the integrity between columns and tables
- Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. Several types of constraints can be created in RDBMS.

Types of constraints

- NOT NULL
- UNIQUE
- DEFAULT
- CHECK
- Key Constraints – PRIMARY KEY, FOREIGN KEY (Referential integrity constraint)
- Domain constraints

Key Constraint

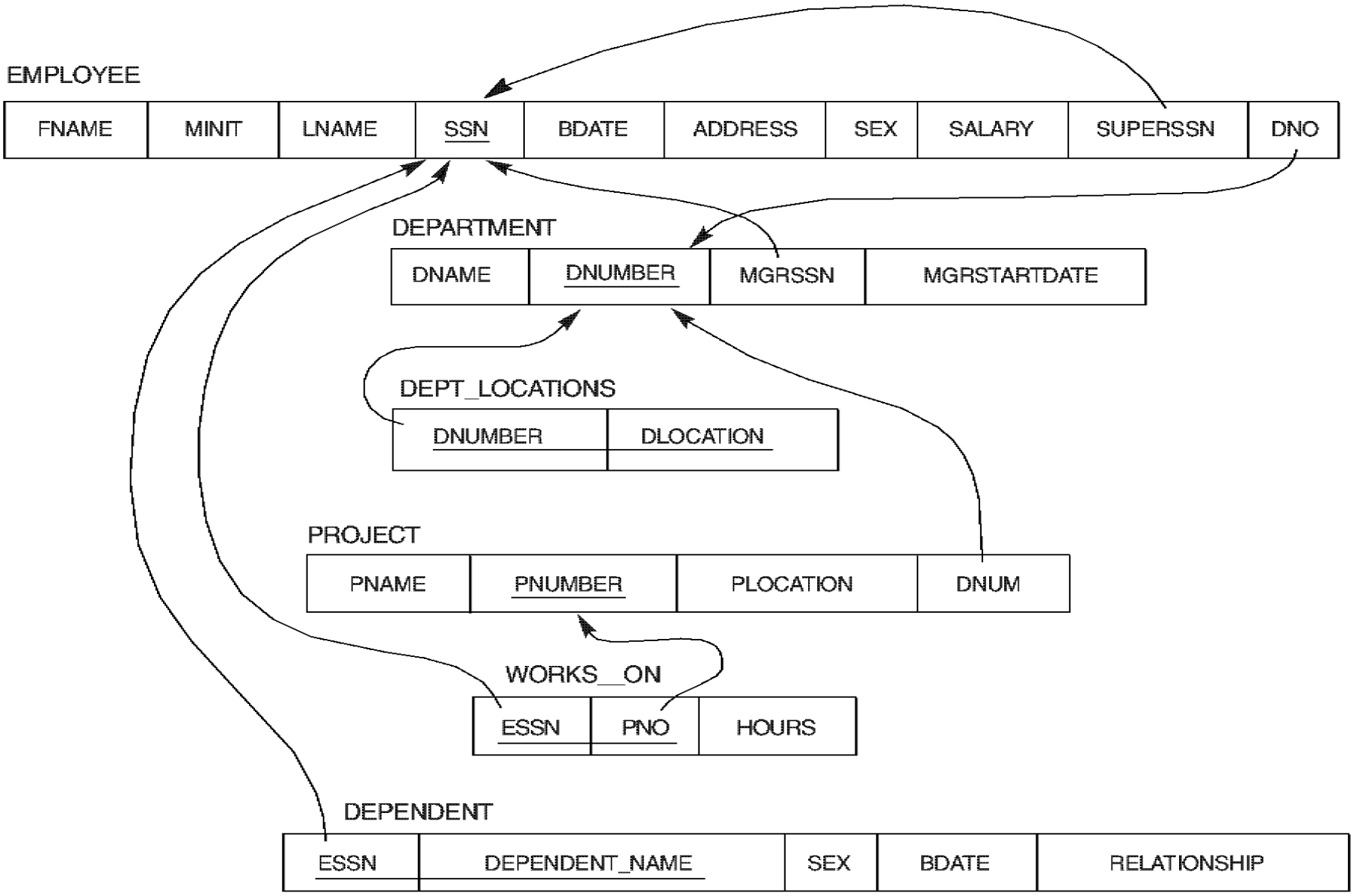
- PRIMARY KEY: Primary key uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL_NO field is marked as primary key, that means the ROLL_NO field cannot have duplicate and null values.

```
CREATE TABLE STUDENT(
ROLL_NO   INT  NOT NULL,
STU_NAME VARCHAR (35)  NOT NULL UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY (ROLL_NO) );
```

FOREIGN KEY (Referential integrity constraint):

- Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



NOT NULL

- NOT NULL constraint makes sure that a column does not hold NULL value. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.
- Example:

```
CREATE TABLE STUDENT(
ROLL_NO INT,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (235),
PRIMARY KEY (ROLL_NO) );
```

Unique Constraints

- UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

```
CREATE TABLE STUDENT(
ROLL_NO INT,
STU_NAME VARCHAR (35) NOT NULL,
MOBILE INT UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35),
PRIMARY KEY (ROLL_NO)
);
```

Default Constraints

- The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

```
CREATE TABLE STUDENT(
ROLL_NO   INT  NOT NULL,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
EXAM_FEE INT  DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
```

Check Constraints

- This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

```
CREATE TABLE STUDENT(
ROLL_NO   INT  NOT NULL CHECK(ROLL_NO >1000),
STU_NAME VARCHAR (35)  NOT NULL,
STU_AGE INT  NOT NULL CHECK(STU_AGE>=18)
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
```

Domain Constraints

- Each table has certain set of columns and each column allows a same type of data, based on its data type. The column does not accept values of any other data type.
  Domain constraints are **user defined data type** and we can define them like this:
- Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)