Module - 3

3.2

Overview of SQL

SQL has established itself as the standard relational database language.

The SQL language has many parts:

- **Data-definition language** (DDL). The SQL DDL provides commands for defining relation schemas, deleting relations, and modifying relation schemas.

- **Data-manipulation language** (DML). The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- **Integrity**. The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.
- **View definition**. The SQL DDL includes commands for defining views.
- **Transaction control**. SQL includes commands for specifying the beginning and end points of transactions.
- **Embedded SQL** and **dynamic SQL**. Embedded and dynamic SQL define how SQL statements can be embedded within general-purpose programming languages, such as C, C++, and Java.
- **Authorization**. The SQL DDL includes commands for specifying access rights to relations and views.

In other words, it is a method of combining the computing power of a programming language and the data manipulation capabilities of SQL. Embedded SQL statements are SQL statements written inline with the program source code.

Data Definition Language:

1) The set of relations / tables in a database are specified using a data-definition language (DDL). The ~~set~~ of SQL DDL allows specification of not only a set of relations / tables, but also information about each relation, including:

a) The schema for each relation.
b) The types of values associated with each attribute. (data-type)
c) The integrity constraints.
d) The set of indices to be maintained for each relation.
e) The security & authorization information for each relation.
f) The physical storage structure of each relation on disk.

2) DDL commands:

a) CREATE command: We define a relation using CREATE ~~command~~. TABLE command.

Syntax & example:

```
create table department
    ( dept-name    varchar(20)
      building     varchar(15)
      budget       numeric(12,2)
      primary key (dept_name));
```

b) DROP TABLE command: To remove a relation from an SQL database, we use the ~~database~~ 'drop table' command. The 'drop table' command deletes all information about the dropped relation from the database.

The command:

```
drop table r;
```

r → relation

The whole relation r is deleted.

TRUNCATE

c) DELETE Command:
The relation r is retained, but the all the tuples in the relation is deleted.
Tuples can be re-inserted after 'truncate' command, but not after 'drop table' command.

d) ALTER command: We use the 'ALTER TABLE' command to add attributes to an existing relation / table.
(Attributes ≡ a column)

All values of tuples for this newly added attribute is NULL.

ALTER can also be used for dropping a single attribute, but this operation may not be supported by some database systems.

i) Syntax for adding:

Attribute name ↑ , Data type ↗

alter table r add A D;

ii) Syntax for dropping:

alter table r drop A;

iii) The data type for an attribute column using alter:
Syntax:
alter table r MODIFY COLUMN A D;

iv) NOT NULL constraint can also be added using ALTER on an attribute / column.

Syntax:

Alter table r Modify A D NOT NULL;

v) Unique constraint can also be added using ALTER (will see later)

Syntax:

```
ALTER TABLE x ADD CONSTRAINT c UNIQUE(col1, col2, ...);
```

(c: Constraint)

vi) CHECK can also be added using ALTER:

Syntax:

```
ALTER TABLE x ADD CONSTRAINT c CHECK(condition);
```

vii) Primary key can also be added using ALTER.

Syntax:

```
ALTER TABLE x ADD CONSTRAINT pk PRIMARY KEY(col1, col2, ...);
```

Domain Constraints

1) Basic Types (Data types): (Domain)

i) char(n): A fixed length character string with user specified length n. The full form 'character', can be used instead.

ii) Varchar(n): A variable length character string with user specified maximum length n. The full form, character varying, is equivalent.

iii) int: An integer (a finite subset of integers that is machine dependent). The full form, integer, is equivalent.

iv) smallint: A small integer (a machine-dependent subset of the integer type).

v) numeric (p, d) : A fixed-point number with user specific precision. The number consists of p digits (plus a sign), & d of the p digits are to the right of the decimal point. Thus, numeric (3,1) allows 44.5 to be stored exactly but neither 444.5 nor 0.32 can be stored exactly in a field in this type.

vi) real, double precision : Floating-point & double-precision floating point numbers with machine-dependent precision.

vii) float (n) : A floating-point number with precision of at least n digits.

Each type may even include NULL values.

Two 'char' values can be added directly if they are equal in length & indirectly (by adding spaces to the smaller value) if they are of unequal lengths.
However, adding 'varchar' and 'char' is possible because of the extra spaces in 'char'.
Thus, using 'varchar' is recommended over 'char'.

2) SQL domain constraints:

a) SQL NOT NULL constraint:
To prevent NULL value to be entered into a column 'NOT NULL' constraint is used.

This constraint can be implemented with CREATE as well as ALTER commands.

b) SQL CHECK constraint:
CHECK constraint restricts the values that are accepted by one or more columns.

Meaning, it checks whether the condition is true or not for a value that is added in the column, and accordingly returns TRUE or FALSE.

Eg:
student_age int checks (student age > 17)

c) SQL Unique key constraint:

SQL unique key constraint is used to ensure that no duplicate values entered in the specific columns, the data must be unique for all records for particular column which contains unique constraint.

* The syntax we saw earlier for CHECK:

ALTER TABLE r ADD CONSTRAINT c CHECK (cond.);

CHECK (cond.) is the constraint we have added to the attribute r.
The name of the constraint is c.

Referential Integrity:

1) SQL supports a number of different ~~integrity constraints: data types:~~ integrity constraints:

a) Primary key $(A_{j_1}, A_{j_2}, A_{j_3}, \ldots A_{j_m})$: The primary key specific states that attributes $(A_{j_1}, A_{j_2}, \ldots A_{j_m})$ form the primary key for the relation / table. (meaning, they individually are composite keys) The primary keys are required to be non-null & unique.

b) Foreign key: $(A_{K_1}, A_{K_2}, A_{K_3}, \ldots A_{K_m})$ References s:

The foreign key specification says that the values of the attributes $(A_{K_1}, A_{K_2}, \ldots A_{K_m})$ for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation s.

Database systems like Postgresql & mySQL require an alternate syntax:

foreign key (dept_name) references department (dept_name)

where the referenced attribute in the referenced table are listed explicitly.

c) not null : The 'not null' constraint on an attribute specifies that null value is not allowed for that attribute (column).

2) Referential Integrity:

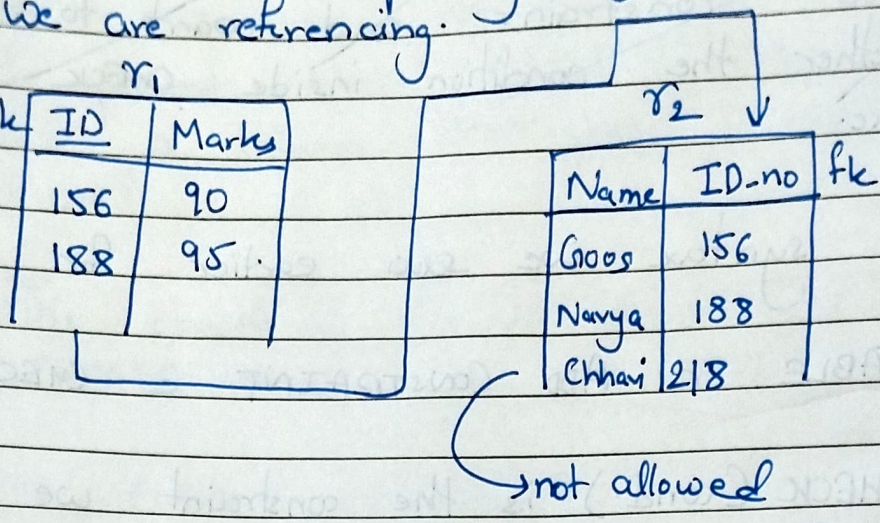
Referential Integrity Constraints are used when one attribute from a relation refers to another attribute, from the referred relation. Basically, whenever foreign keys are used, referential integrity constraints should be added.

These constraints are:

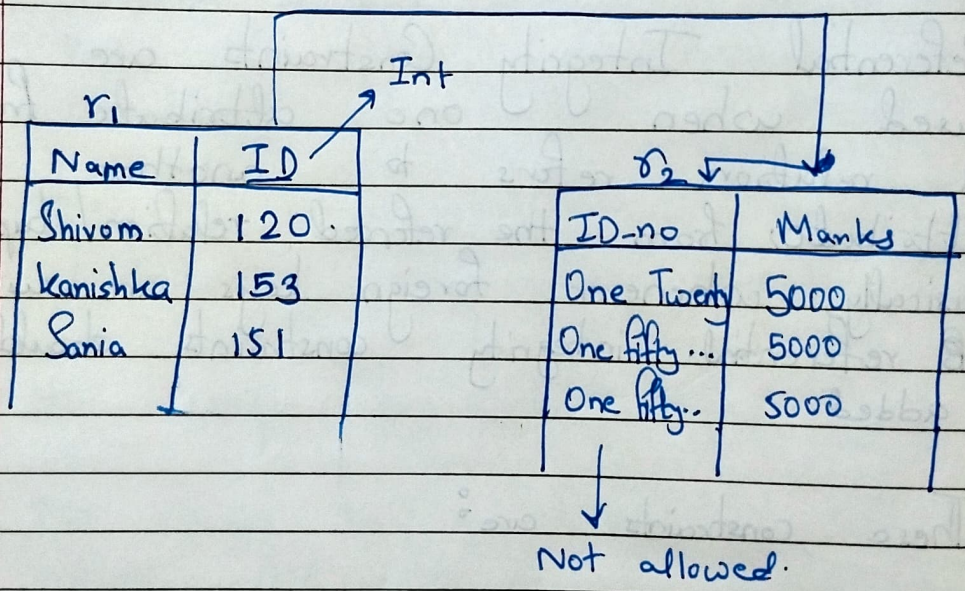
a) The Attribute that we select as the foreign key should be a primary key or a unique key, meaning it has unique value for all tuples.

→ You cannot refer to something that does not exist.

b) The foreign keys can only be a subset of primary key values, that we are referencing.



c) The data types of the foreign key & primary key should be same or at least compatible.



** Note: One thing I learned while researching is that it is not necessary that the foreign key comes from a different relation everytime.
A relation can refer to itself ~~while~~ using foreign keys. Same constraints hold ~~for it~~

for it

Example:

Students

ID	Name	Project partner
151	~~Goos~~ Sania	153
153	Kanishka	151
156	Goos	—

Foreign keys can have null values, in cases like these.