Tutorial: Constraints and DDL Query

> Designed by ZHU Yueming. A small part of descriptions of basic concepts in this tutorial are borrowed from the Stephane Faroult's Slide and Wikipedia.
>
> Modify it to simple database design by ZHU Yueming in 2022.1.10 and 2022.1.22

Experimental Objective

- Understand basic constraints in database design
- Understand basic DDL (Data Definition Language) language

Part 1. Constraints

Constraints are declarative rules that the DBMS apply to ensure the integrity of data. DBMS will check constraints every time new data is added, changed or deleted, to prevent any inconsistency. Any operation that violates a constraint fails and returns an error.

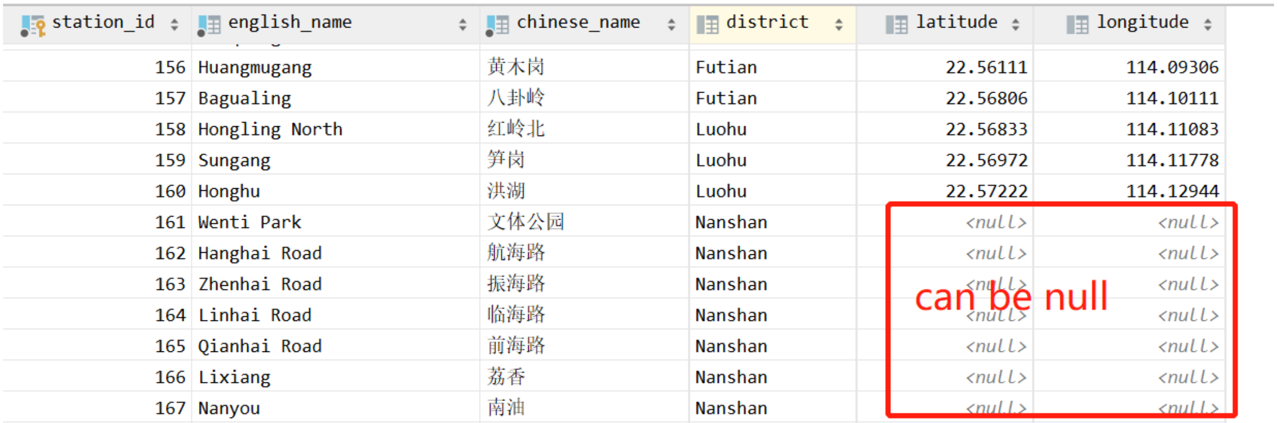
1. NOT NULL

If you want one column cannot be null, you can indicate the column by `not null`.

Sample DDL language.

```
create table if not exists stations
(
  station_id integer not null
    constraint stations_pkey
      primary key,
  english_name varchar(80) not null
    constraint stations_uq_1
      unique,
  chinese_name varchar(10) not null
    constraint stations_uq_2
      unique,
  district varchar(20),
  latitude double precision,
  longitude double precision
);
```

Sample graph:



station_id	english_name	chinese_name	district	latitude	longitude
156	Huangmugang	黄木岗	Futian	22.56111	114.09306
157	Bagualing	八卦岭	Futian	22.56806	114.10111
158	Hongling North	红岭北	Luohu	22.56833	114.11083
159	Sungang	笋岗	Luohu	22.56972	114.11778
160	Honghu	洪湖	Luohu	22.57222	114.12944
161	Wenti Park	文体公园	Nanshan	<null>	<null>
162	Hanghai Road	航海路	Nanshan	<null>	<null>
163	Zhenhai Road	振海路	Nanshan	<null>	<null>
164	Linhai Road	临海路	Nanshan	<null>	<null>
165	Qianhai Road	前海路	Nanshan	<null>	<null>
166	Lixiang	荔香	Nanshan	<null>	<null>
167	Nanyou	南油	Nanshan	<null>	<null>

can be null

2. UNIQUE

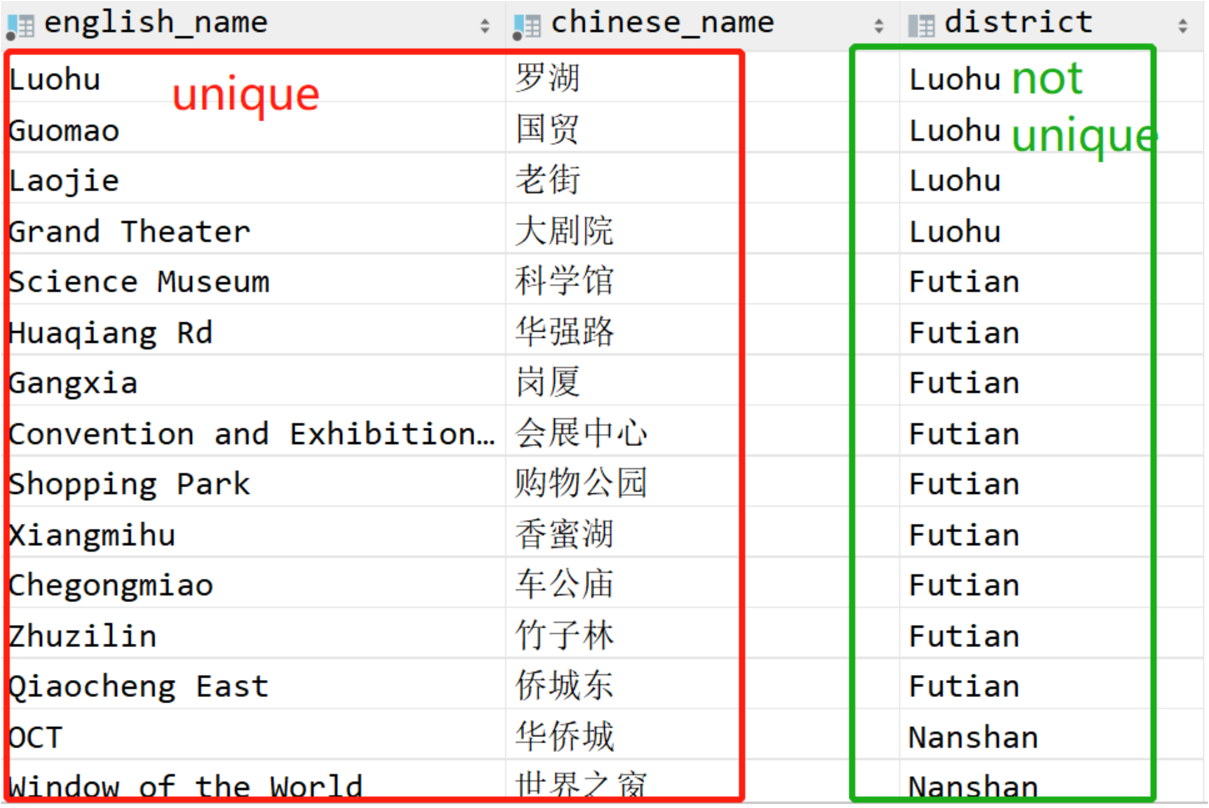
Each value in the specified column(s) is unique.

- Unique for one column

  Sample DDL language.

```
create table if not exists stations
(
  station_id integer not null
    constraint stations_pkey
      primary key,
  english_name varchar(80) not null
    constraint stations_uq_1
      unique,
  chinese_name varchar(10) not null
    constraint stations_uq_2
      unique,
  district varchar(20),
  latitude double precision,
  longitude double precision
);
```

  Sample graph:



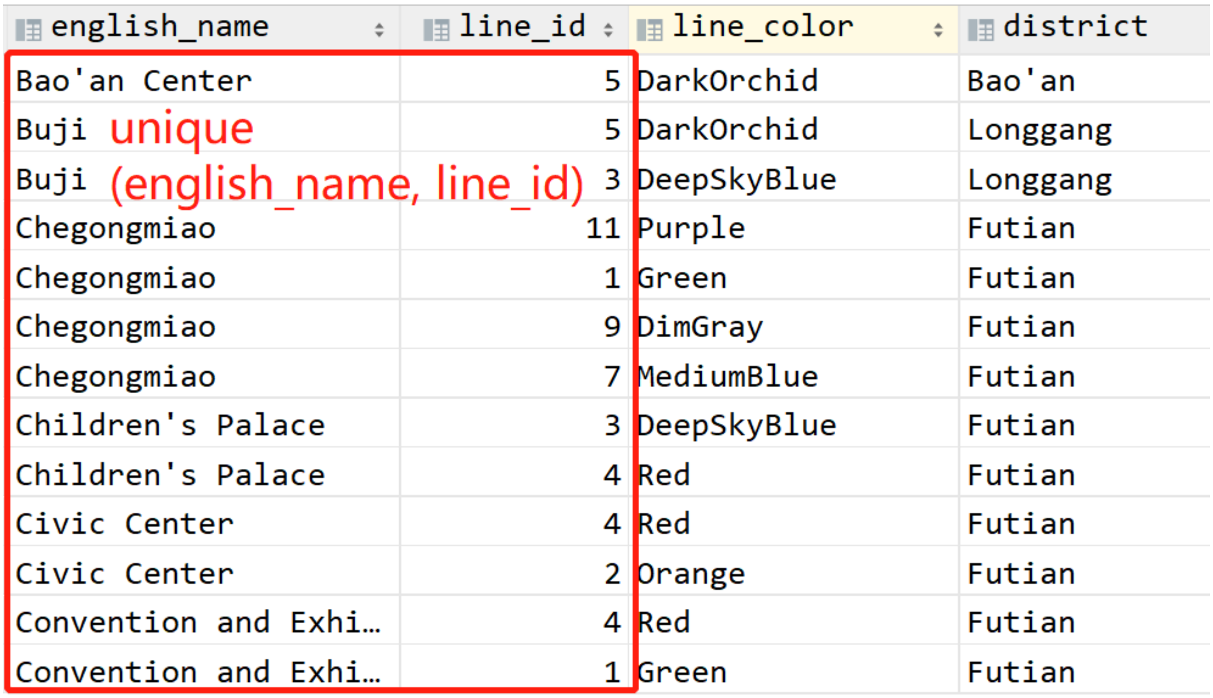
english_name	chinese_name	district
Luohu	罗湖	Luohu
Guomao	国贸	Luohu
Laojie	老街	Luohu
Grand Theater	大剧院	Luohu
Science Museum	科学馆	Futian
Huaqiang Rd	华强路	Futian
Gangxia	岗厦	Futian
Convention and Exhibition…	会展中心	Futian
Shopping Park	购物公园	Futian
Xiangmihu	香蜜湖	Futian
Chegongmiao	车公庙	Futian
Zhuzilin	竹子林	Futian
Qiaocheng East	侨城东	Futian
OCT	华侨城	Nanshan
Window of the World	世界之窗	Nanshan

- Unique for multiple columns

  Sample DDL Language

```
create table test_unique (
    id serial primary key,
    english_name varchar(80) not null,
    line_id integer not null,
    line_color varchar(20) not null,
    district varchar(20),
    constraint uq unique (english_name, line_id)
);
```

Sample graph:



unique (english_name, line_id)

english_name	line_id	line_color	district
Bao'an Center	5	DarkOrchid	Bao'an
Buji	5	DarkOrchid	Longgang
Buji	3	DeepSkyBlue	Longgang
Chegongmiao	11	Purple	Futian
Chegongmiao	1	Green	Futian
Chegongmiao	9	DimGray	Futian
Chegongmiao	7	MediumBlue	Futian
Children's Palace	3	DeepSkyBlue	Futian
Children's Palace	4	Red	Futian
Civic Center	4	Red	Futian
Civic Center	2	Orange	Futian
Convention and Exhi…	4	Red	Futian
Convention and Exhi…	1	Green	Futian

3. PRIMARY KEY

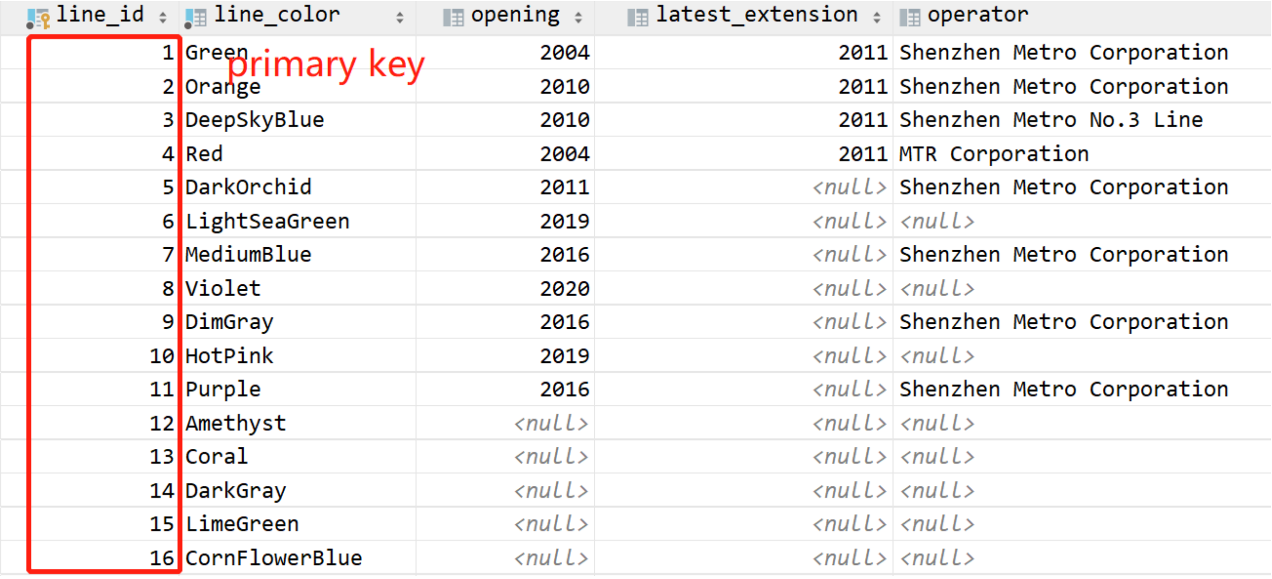
Primary key specifies the main key for the table, which is:

- Mandatory (the additional NOT NULL doesn't hurt but is redundant)
- Unique (no duplicates allowed in the column)

Sample DDL Language

```
create table if not exists lines
(
  line_id integer not null
      primary key,
  line_color varchar(20) not null
      unique,
  opening integer,
  latest_extension integer,
  operator varchar(30)
);
```

Sample graph:



primary key

line_id	line_color	opening	latest_extension	operator
1	Green	2004	2011	Shenzhen Metro Corporation
2	Orange	2010	2011	Shenzhen Metro Corporation
3	DeepSkyBlue	2010	2011	Shenzhen Metro No.3 Line
4	Red	2004	2011	MTR Corporation
5	DarkOrchid	2011	<null>	Shenzhen Metro Corporation
6	LightSeaGreen	2019	<null>	<null>
7	MediumBlue	2016	<null>	Shenzhen Metro Corporation
8	Violet	2020	<null>	<null>
9	DimGray	2016	<null>	Shenzhen Metro Corporation
10	HotPink	2019	<null>	<null>
11	Purple	2016	<null>	Shenzhen Metro Corporation
12	Amethyst	<null>	<null>	<null>
13	Coral	<null>	<null>	<null>
14	DarkGray	<null>	<null>	<null>
15	LimeGreen	<null>	<null>	<null>
16	CornFlowerBlue	<null>	<null>	<null>

4. FOREIGN KEY

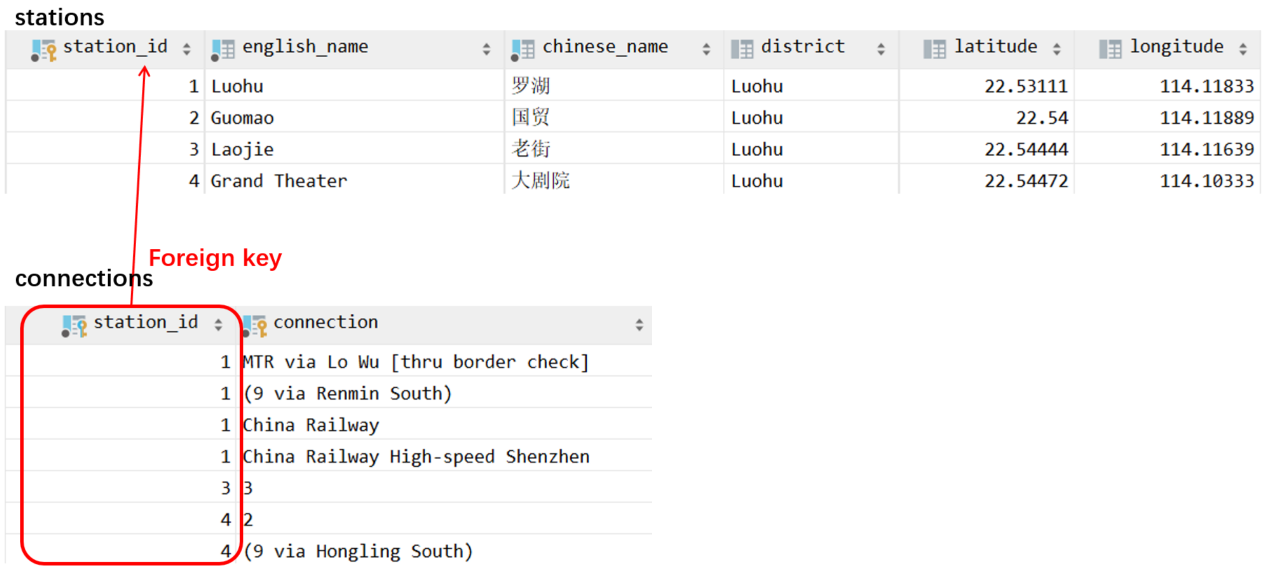
Foreign key indicates that the column must reference a key (Only primary keys and columns declared as UNIQUE) of another table.

- Constraints are used to prevent actions that break the connection between tables
- Constraints also prevent illegal data from being inserted into the column.

Sample DDL Language: we add a foreign key constraint `station_id` in table `connections` , which references the primary key `station_id` in table `stations`

```
create table connections
(
    station_id integer      not null
        constraint connections_fk
            references stations(station_id),
    connection varchar(100) not null,
    constraint connections_pk
        primary key (station_id, connection)
);
```

Sample graph:



stations

station_id	english_name	chinese_name	district	latitude	longitude
1	Luohu	罗湖	Luohu	22.53111	114.11833
2	Guomao	国贸	Luohu	22.54	114.11889
3	Laojie	老街	Luohu	22.54444	114.11639
4	Grand Theater	大剧院	Luohu	22.54472	114.10333

connections

station_id	connection
1	MTR via Lo Wu [thru border check]
1	(9 via Renmin South)
1	China Railway
1	China Railway High-speed Shenzhen
3	3
4	2
4	(9 via Hongling South)

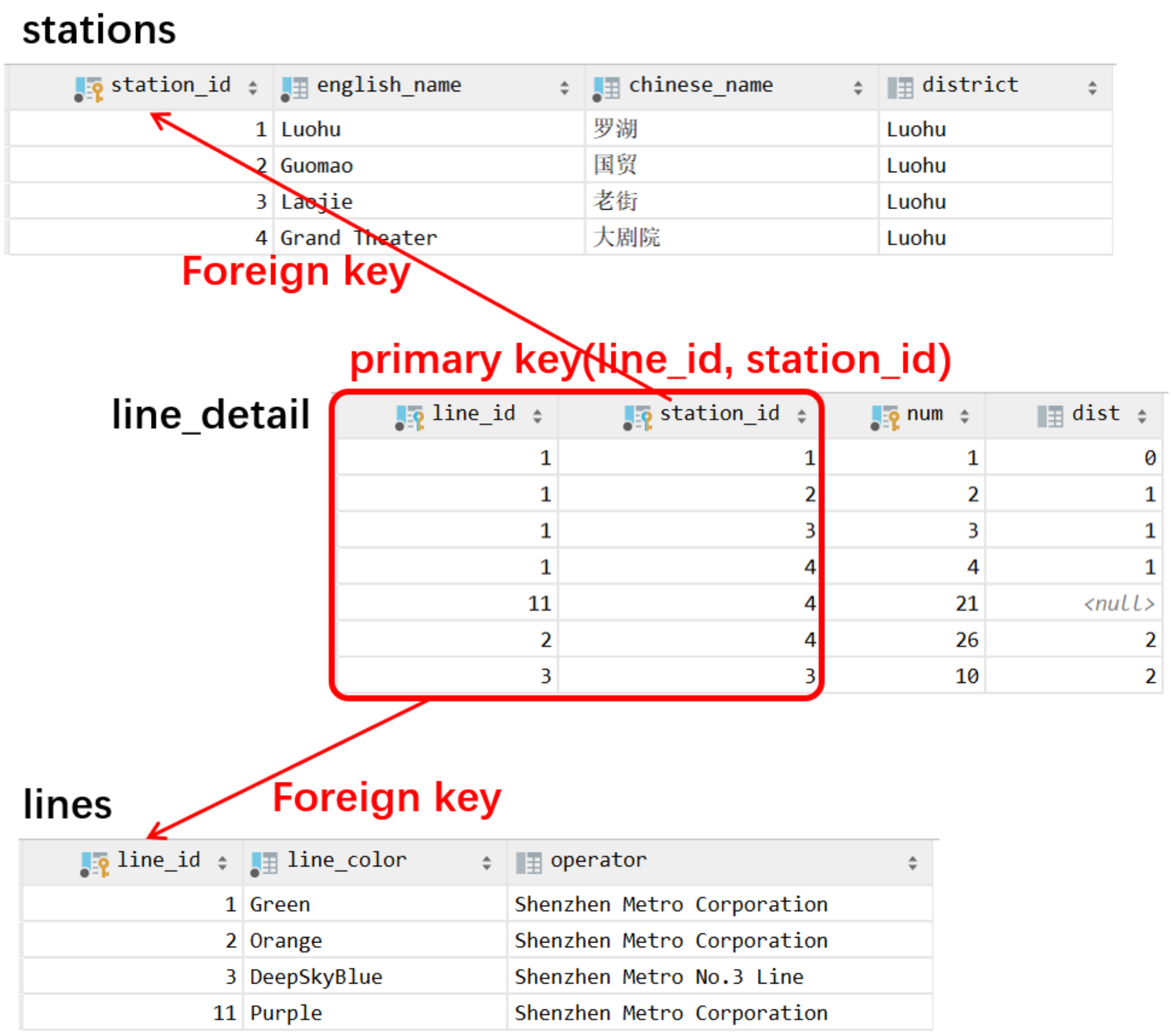
In this case, we cannot insert any data in connections if the inserted station_id is not appeared in the column station_id in stations table because of the foreign key. For example

```
insert into connections (station_id, connection) values (10086,'CSE');
```

The result would be:

```
[23503] ERROR: insert or update on table "connections" violates foreign key
constraint "connections_fk" 详细: Key (station_id)=(10086) is not present in
table "stations".
```

Another sample graph below represents one table can have multiple foreign keys.



stations

station_id	english_name	chinese_name	district
1	Luohu	罗湖	Luohu
2	Guomao	国贸	Luohu
3	Laojie	老街	Luohu
4	Grand Theater	大剧院	Luohu

line_detail

line_id	station_id	num	dist
1	1	1	0
1	2	2	1
1	3	3	1
1	4	4	1
11	4	21	<null>
2	4	26	2
3	3	10	2

lines

line_id	line_color	operator
1	Green	Shenzhen Metro Corporation
2	Orange	Shenzhen Metro Corporation
3	DeepSkyBlue	Shenzhen Metro No.3 Line
11	Purple	Shenzhen Metro Corporation

Part 2. Sample data definition language （DDL）

In the database cs307 you have been created last week, do following exercises step by step.

```
create database cs307 encoding='utf8';
```

1. Create Table Example

- First table represents ticket.

```
create table if not exists ticket
(
    id           serial primary key,
    train_number varchar(10) not null unique,
    depart_city  varchar     not null,
    arrival_city varchar     not null,
    depart_time  time,
    price        numeric check (price > 0)
);
```

- Second table represents the customer.

```
create table customer
(
    id           serial,
    username     varchar(10) not null unique,
    password     varchar,
    phone_number varchar(11)
);
```

- The relationship between ticket and customer is many-to-many, in this case it is usually create a relation table if we want to make a connection of those two tables. In the relation table, usually contains a primary key id, and two foreign key id related to those two tables responstively.

```
create table booking_record
(
    id          serial primary key,
    ticket_id   integer,
    customer_id     integer,
    date        date not null,
    depart_city varchar,
    arrive_city varchar,
    constraint fk_ticket foreign key (ticket_id) references ticket(id)
);
```

2. Alter Table Example

1) add not null constraint

General Syntax:

```
ALTER TABLE [ IF EXISTS ] [ ONLY ] table_name
    ALTER [ COLUMN ] column_name { SET | DROP } NOT NULL
```

Example:

```
alter table customer alter password set not null;
```

2) add primary key

General Syntax:

```
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

Example:

```
alter table customer add constraint pk_customer_id primary key (id);
```

3) add foreign key

General Syntax:

```
ALTER TABLE table_name
ADD CONSTRAINT ForeignKey FOREIGN KEY (column) REFERENCES table2 (column);
```

Example:

```
alter table booking_record add constraint fk_customer foreign key (customer_id)
references customer(id);
```

4) change data type of column

General Syntax:

```
ALTER TABLE table_name ALTER COLUMN column_name TYPE datatype;
```

Example:

```
alter table customer alter column phone_number type varchar;
```

5) add one column

General Syntax:

```
ALTER TABLE table_name ADD COLUMN column_name TYPE datatype;
```

Example:

```
alter table ticket add column seat_type varchar(2);
```

6) drop one column

General Syntax:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Example:

```
alter table booking_record drop column depart_city;
alter table booking_record drop column arrive_city;
```

Analysis from the database design, why those two columns need drop?

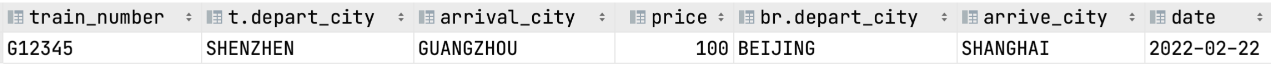
`depart_city` and `arrive_city` are duplicate column, because the relation table do not need store the columns that declared in its referenced table.

Suppose insert a row in those three table respectively:

```
insert into ticket(train_number, depart_city, arrival_city, depart_time, price)
values ('G12345', 'SHENZHEN', 'GUANGZHOU', '153000', 100);
insert into customer (username, password, phone_number)
values ('A', '123456', 12345678901);
insert into booking_record(ticket_id, customer_id, date, depart_city,
arrive_city)
VALUES (1, 1, '2022/2/22', 'BEIJING', 'SHANGHAI');
```

If we do following query, we would find two different arrive city and depart city.

```
select t.train_number,
       t.depart_city,
       t.arrival_city,
       t.price,
       br.depart_city,
       br.arrive_city,
       br.date
from booking_record br
         join ticket t
              on br.ticket_id = t.id;
```



train_number	t.depart_city	arrival_city	price	br.depart_city	arrive_city	date
G12345	SHENZHEN	GUANGZHOU	100	BEIJING	SHANGHAI	2022-02-22

7) drop constraints

General Syntax:

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Example:

```
alter table booking_record drop constraint unique_columns;
alter table booking_record add constraint unique_columns unique
(customer_id,ticket_id,date);
```

8) rename

General Syntax:

```
ALTER TABLE table_name RENAME TO new_name;
ALTER TABLE table_name RENAME COLUMN column_name TO new_column
```

Example:

```
alter table booking_record rename column date to booking_date;
```

9). add check constraints

General Syntax:

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name CHECK (CONDITION);
```

Example:

```
alter table customer add constraint check_phone check (
length(phone_number)>=11 );
```

3. Drop Table Example

- Firstly, you drop the table, which has been referenced by an foreign key of other table. Try the query below

  ```
  drop table customer;
  ```

  The result would be:

  ```
  [2BP01] ERROR: cannot drop table customer because other objects depend on
  it Detail: constraint fk_customer on table booking_record depends on table
  customer Hint: Use DROP ... CASCADE to drop the dependent objects too.
  ```

- Solution 1. Drop table customer and all its foreign key constraints in other tables.

  ```
  drop table customer cascade;
  ```

- Solution 2. Drop the all other tables which have foreign key constraint that related to the current table first, and then drop current table.

  ```
  drop table booking_record;
  drop table customer;
  ```

4. Check the constraints:

Use the query below to check the constraints:

```
select tc.constraint_name, tc.constraint_type, tc.table_name
from information_schema.table_constraints tc
where tc.constraint_schema="current_schema"();
```

Part 3. Exercise

Design a simple database which contains four tables as follows:

- student (name, student_number(unique), department, gender)
- department (name, location, website)

- course (name, course_number, department, credit)
- course_selected(student_id, course_id, semester, course_status)

Other requirements describe as follows: The relationship between Student and Course is many-to-many. The relationship between Course and Department is many-to-one. The relationship between Student and Department is many-to-one.

Hints: How to represent many-to-one relationship between two tables? You can reference the following queries:

Those queries represents the relationship between department and student, and we can find that one student has one department while one department contains many student. In this case, we add a foreign key on student, which is an id of the primary column in department table.

```
create table department(
    id serial primary key ,
    name varchar not null,
    location varchar,
    website varchar
);
```

```
create table student(
    id serial primary key,
    name varchar not null,
    student_number varchar unique not null,
    department_id integer,
    gender varchar(2) not null,
    constraint fk_stu_department foreign key (department_id) references
department(id)
);
```

Please design a simple database that can match all requirements above by DDL language in postgres.