CSC343 Lab1

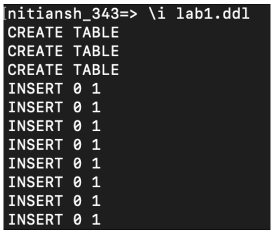
Yunkai Fan 1005088584

Tianshu Ni 1005682673

Jan. 28, 2022

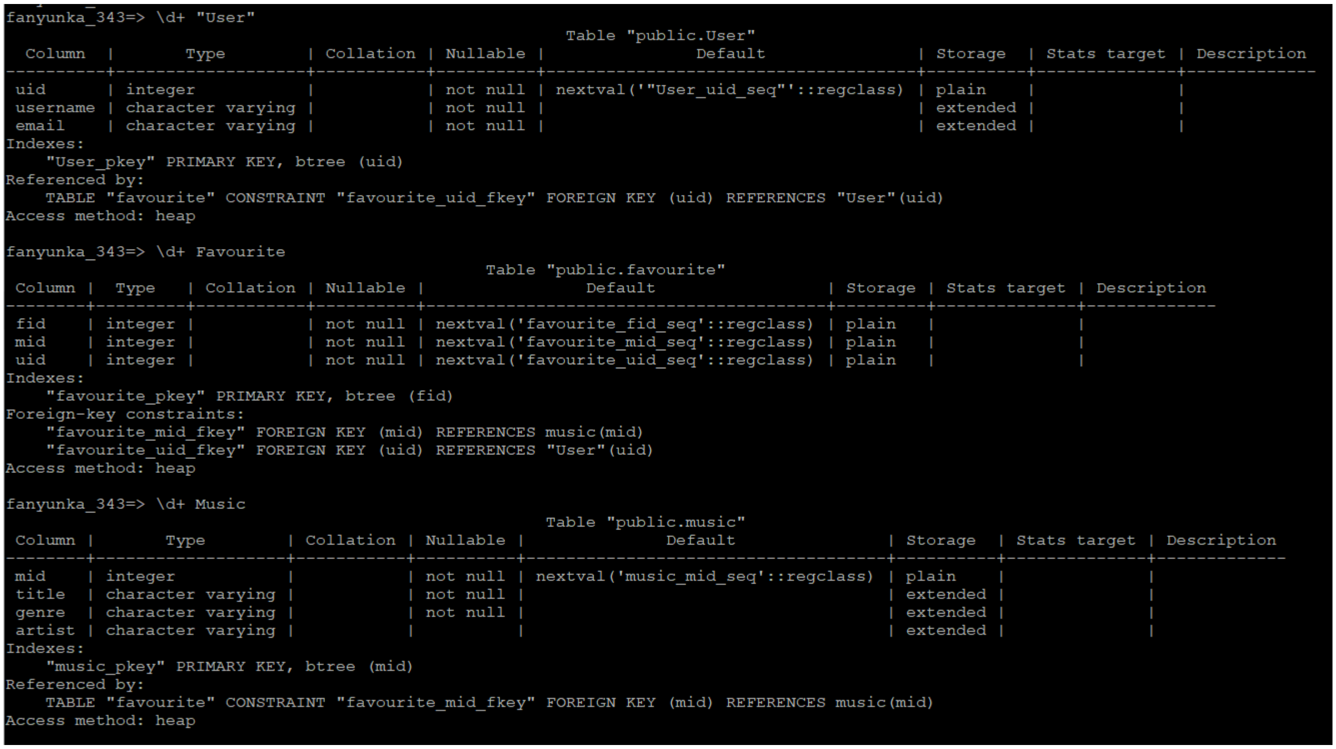
Setup

First import lab1.ddl, show Running Scripts of lab1.ddl.



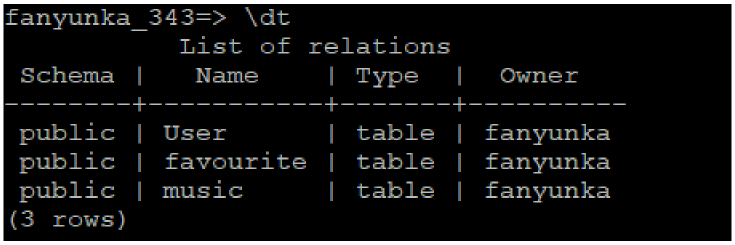
```
nitiansh_343=> \i lab1.ddl
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

We could observe the structure of all of the tables:



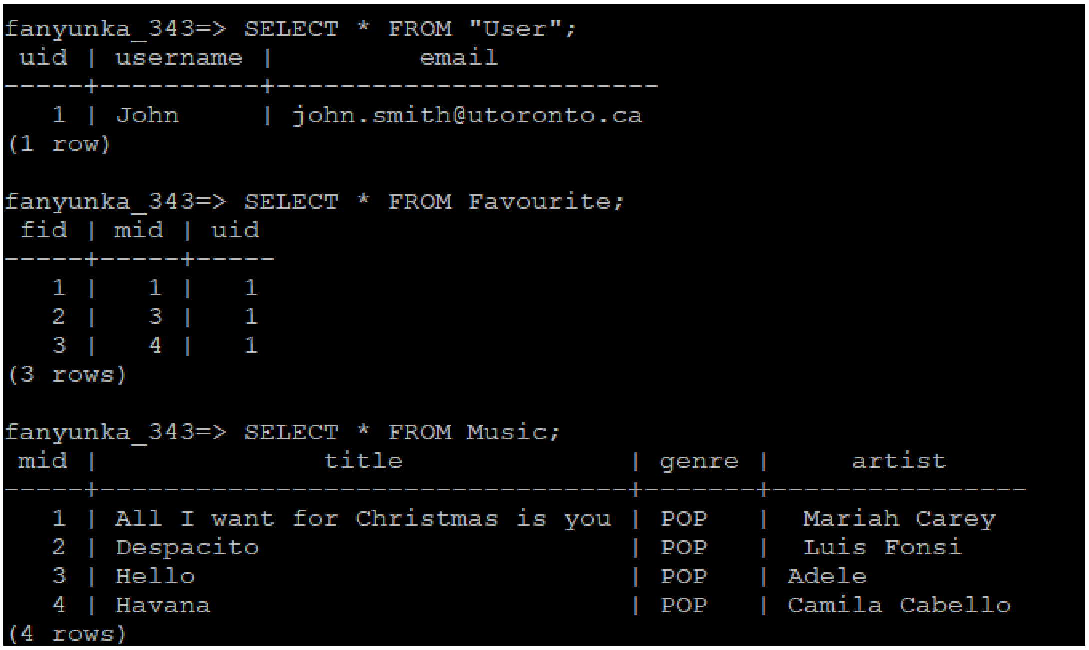
```
fanyunka_343=> \d+ "User"
                                                     Table "public.User"
  Column  |       Type        | Collation | Nullable |               Default               | Storage  | Stats target | Description
----------+-------------------+-----------+----------+-------------------------------------+----------+--------------+-------------
 uid      | integer           |           | not null | nextval('"User_uid_seq"'::regclass) | plain    |              |
 username | character varying |           | not null |                                     | extended |              |
 email    | character varying |           | not null |                                     | extended |              |
Indexes:
    "User_pkey" PRIMARY KEY, btree (uid)
Referenced by:
    TABLE "favourite" CONSTRAINT "favourite_uid_fkey" FOREIGN KEY (uid) REFERENCES "User"(uid)
Access method: heap

fanyunka_343=> \d+ Favourite
                                             Table "public.favourite"
 Column |  Type   | Collation | Nullable |                Default                 | Storage | Stats target | Description
--------+---------+-----------+----------+----------------------------------------+---------+--------------+-------------
 fid    | integer |           | not null | nextval('favourite_fid_seq'::regclass) | plain   |              |
 mid    | integer |           | not null | nextval('favourite_mid_seq'::regclass) | plain   |              |
 uid    | integer |           | not null | nextval('favourite_uid_seq'::regclass) | plain   |              |
Indexes:
    "favourite_pkey" PRIMARY KEY, btree (fid)
Foreign-key constraints:
    "favourite_mid_fkey" FOREIGN KEY (mid) REFERENCES music(mid)
    "favourite_uid_fkey" FOREIGN KEY (uid) REFERENCES "User"(uid)
Access method: heap

fanyunka_343=> \d+ Music
                                                   Table "public.music"
 Column |       Type        | Collation | Nullable |              Default               | Storage  | Stats target | Description
--------+-------------------+-----------+----------+------------------------------------+----------+--------------+-------------
 mid    | integer           |           | not null | nextval('music_mid_seq'::regclass) | plain    |              |
 title  | character varying |           | not null |                                    | extended |              |
 genre  | character varying |           | not null |                                    | extended |              |
 artist | character varying |           |          |                                    | extended |              |
Indexes:
    "music_pkey" PRIMARY KEY, btree (mid)
Referenced by:
    TABLE "favourite" CONSTRAINT "favourite_mid_fkey" FOREIGN KEY (mid) REFERENCES music(mid)
Access method: heap
```

Then we show all the tables in the database.



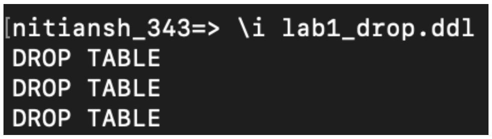
```
fanyunka_343=> \dt
           List of relations
 Schema |    Name    | Type  |  Owner
--------+-----------+-------+----------
 public | User       | table | fanyunka
 public | favourite  | table | fanyunka
 public | music      | table | fanyunka
(3 rows)
```

The following image also shows all of their records.



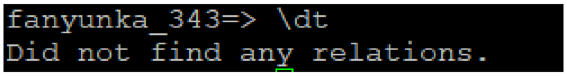
```
fanyunka_343=> SELECT * FROM "User";
 uid | username |          email
-----+----------+-----------------------
   1 | John     | john.smith@utoronto.ca
(1 row)

fanyunka_343=> SELECT * FROM Favourite;
 fid | mid | uid
-----+-----+-----
   1 |   1 |   1
   2 |   3 |   1
   3 |   4 |   1
(3 rows)

fanyunka_343=> SELECT * FROM Music;
 mid |              title               | genre |     artist
-----+---------------------------------+-------+----------------
   1 | All I want for Christmas is you | POP   |  Mariah Carey
   2 | Despacito                       | POP   |  Luis Fonsi
   3 | Hello                           | POP   | Adele
   4 | Havana                          | POP   | Camila Cabello
(4 rows)
```

After importing lab1_drop.ddl,



```
[nitiansh_343=> \i lab1_drop.ddl
DROP TABLE
DROP TABLE
DROP TABLE
```

there exists no relations in the database.



```
fanyunka_343=> \dt
Did not find any relations.
```

Database Independence

1. For the logical data independence of this database, when the records has been combined from duplicated records with different primary keys, the view into the database will remain the same.

2. For the physical data independence of this database, when the database has been moving from physical storage, which has moved from Michael's local machine to the cloud service, the tables and records will not be affected, there will be no need to rewrite records or repeat the procedure of remove duplicates. In addition, while applying the LZMA algorithm to the database, the content of the database is lossless, and it is only the size of the database being changed (compressed).

Keys and Referential Integrity

2. In this question, we will be making a few assumptions to the attributes of the tables from the database, and the following are the candidate keys:

    - **Department**: the department id **did** is unique, and is the candidate key.
    - **Employee**: the employee identification number **eid**, and social insurance number **sin** are unique, and are the candidate keys.
    - **Providers**: the provider id **pid**, employer identification number **ein** are unique, and are the candidate keys.
    - **Customer**: the customer id **cid** is unique, and is the candidate key.
    - **Product**: the product id **productid** is unique, and is the candidate key.
    - **Warehousing**: since each product can only be stocked once per day, the combination of **{data, productid}** is unique, and is the candidate key.
    - **Promotion**: the promotion id **promotionid** is unique, and is the candidate key.
    - **Sale**: the sale id **sid**, and the receipt number **receipt_num** are unique, and are the candidate keys.
    - **Receipt**: since each sale could contain multiple products, the combination of **{receipt_num, productid}** is unique, and is the candidate key.

1. All of the superkey of each relation (follwing assumptions from previous part):

    - **Department**, the super key is **{did}** or **{did, name}**

- **Employee**, the super key is any combination of attributes including **{eid}** or **{sin}**.
- **Providers**, the super key is any combination of attributes including **{pid}** or **{ein}**.
- **Customer**, the super key is any combination of attributes including **{cid}**.
- **Product**, the super key is any combination of attributes including **{productid}**.
- **Warehousing**, the super key is any combination of attributes including **{date, productid}**
- **Promotion**, the super key is any combination of attributes including **{promotionid}**.
- **Sale**, the super key is any combination of attributes including **{sid}** or **{receipt_num}**.
- **Receipt**, the super key is any combination of attributes including **{receipt_num, productid}**.

3. All of the foreign keys of each relation (follwing assumptions from previous parts):

    - **Employee**: the foreign key is **did** from Department.
    - **Product**: the foreign key is **promotionid** from Promotion.
    - **Warehousing**: the foreign key is **productid** from Products, **pid** from Providers, and **eid** from Employee.
    - **Sale**: the foreign key is **eid** from Employee, **receipt_num** from Recipt, and **cid** from Customer.
    - **Receipt**: the foreign key is **productid** from Product, and **sid** from Sale.