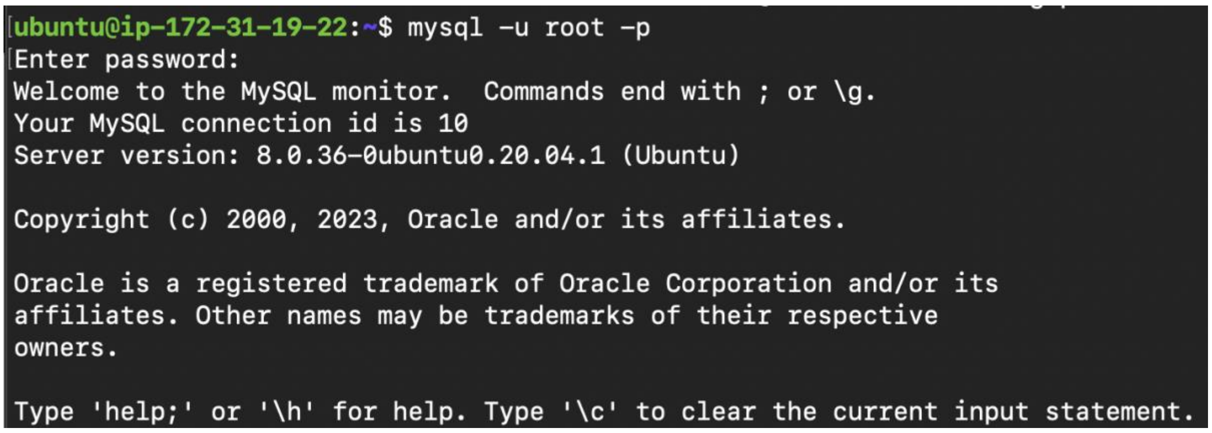
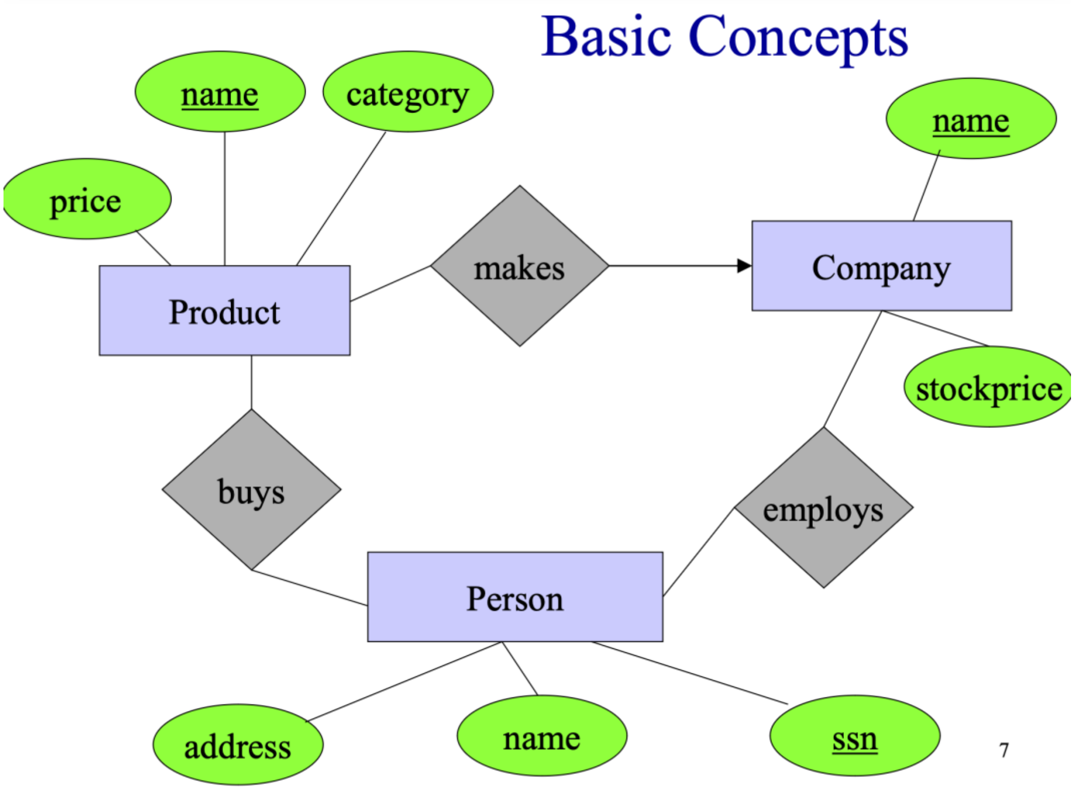
sudo service mongod stop
sudo service mysql start
然后：



```
ubuntu@ip-172-31-19-22:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.36-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Password: Dsci-551

Lecture 5



Now lets write sql based on this ER diagram.

Commands:

mysql> create database dsci551
mysql> show databases;
mysql> use dsci551;

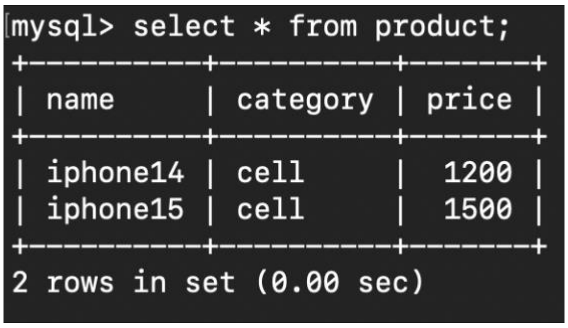
mysql> show tables

mysql> create table product(name varchar(20) primary key, category varchar(20), price int);

mysql> insert into product values('iphone15', 'cell', 1500);

mysql> insert into product(category, price, name) values('cell', 1200, 'iphone14');

now lets see what we have now.



```
mysql> select * from product;
+----------+----------+-------+
| name     | category | price |
+----------+----------+-------+
| iphone14 | cell     |  1200 |
| iphone15 | cell     |  1500 |
+----------+----------+-------+
2 rows in set (0.00 sec)
```

mysql> insert into product values('t450s', 'laptop', 2000);

mysql> create table person(ssn int primary key, name varchar(20), addr varchar(20),
    -> unique(name, addr));

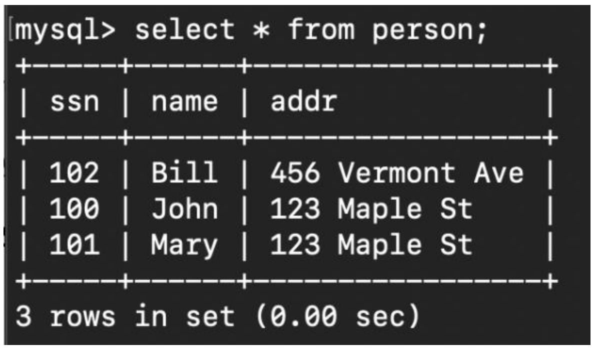
Here, person has 2 keys: one primary key and one unique key.

mysql> insert into person values(100, 'John', '123 Maple St');

mysql> insert into person values(101, 'Mary', '123 Maple St');

mysql> insert into person values(102, 'Bill', '456 Vermont Ave');

now lets see what we have now.

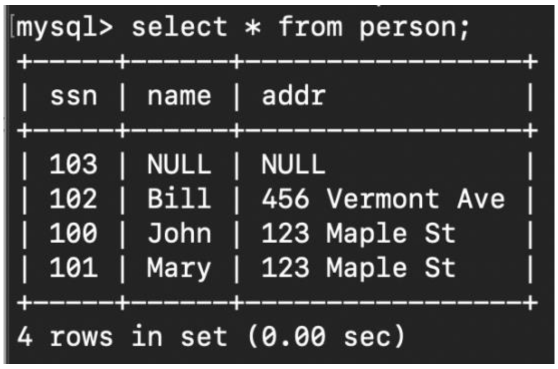


```
mysql> select * from person;
+-----+------+-----------------+
| ssn | name | addr            |
+-----+------+-----------------+
| 102 | Bill | 456 Vermont Ave |
| 100 | John | 123 Maple St    |
| 101 | Mary | 123 Maple St    |
+-----+------+-----------------+
3 rows in set (0.00 sec)
```

***FYI:**

mysql> insert into person (ssn) values(103);

it generates values with NULL.



```
mysql> select * from person;
+-----+------+-----------------+
| ssn | name | addr            |
+-----+------+-----------------+
| 103 | NULL | NULL            |
| 102 | Bill | 456 Vermont Ave |
| 100 | John | 123 Maple St    |
| 101 | Mary | 123 Maple St    |
+-----+------+-----------------+
4 rows in set (0.00 sec)
```

Now create table for ‘buys’ relation.
mysql> create table buys(ssn int, name varchar(20), primary key (ssn, name));

Here, we state primary key separately, not right after attibs has been declared.
Also, it has two primary key because person and product has m-m relation.

mysql> insert into buys values(100, 'iphone15');

mysql> insert into buys values(100, 'iphone14');

mysql> insert into buys values(101, 't450s');

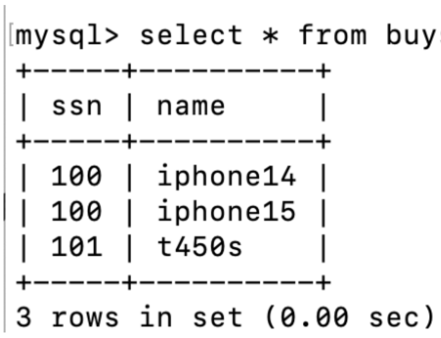
mysql> insert into buys values(101, 'iphone13');

但是没有iPhone13这个product诶。我们需要foreign key。

mysql> delete from buys where name = 'iphone13';

先删掉。

now lets see what we have now.



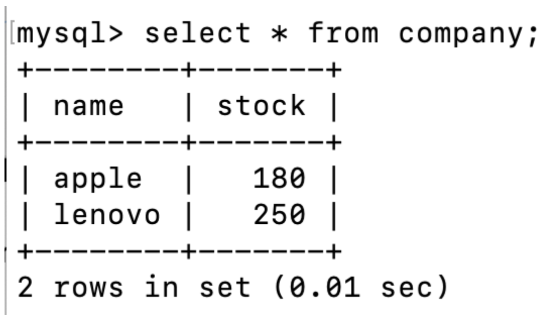
```
mysql> select * from buys;
+-----+----------+
| ssn | name     |
+-----+----------+
| 100 | iphone14 |
| 100 | iphone15 |
| 101 | t450s    |
+-----+----------+
3 rows in set (0.00 sec)
```

mysql> create table company (name varchar(20), stock int, primary key (name));

mysql> insert into company values('apple', 180);

mysql> insert into company values('lenovo', 250);

now lets see what we have now.



```
mysql> select * from company;
+--------+-------+
| name   | stock |
+--------+-------+
| apple  |   180 |
| lenovo |   250 |
+--------+-------+
2 rows in set (0.01 sec)
```

mysql> create table makes (pname varchar(20), cmake varchar(20), primary key (pname));

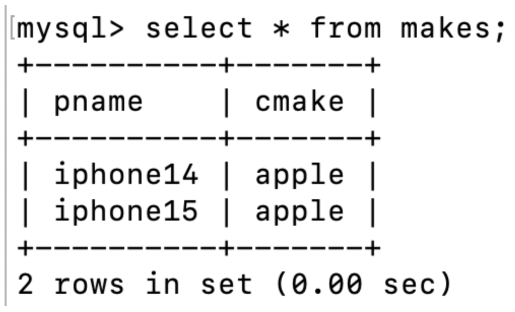
mysql> insert into makes values('iphone14', 'apple');

mysql> insert into makes values('iphone15', 'apple');

clarification:

we cannot add another iphone14 or iphone15 into the table because a product is make by at most one company. That also explains why pname alone is the primary key – because itself is already unique.

now lets see what we have now.



```
mysql> select * from makes;
+----------+-------+
| pname    | cmake |
+----------+-------+
| iphone14 | apple |
| iphone15 | apple |
+----------+-------+
2 rows in set (0.00 sec)
```

mysql> create table employs (ssn int, cname varchar(20), primary key (ssn, cname));

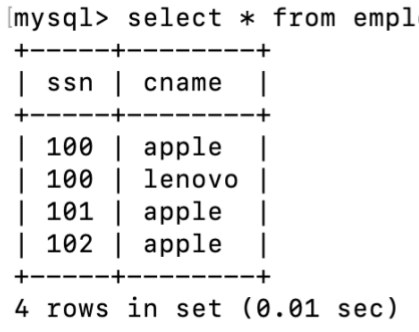
mysql> insert into employs values(100, 'apple');

mysql> insert into employs values(100, 'lenovo');

mysql> insert into employs values(101, 'apple');

mysql> insert into employs values(102, 'apple');

now lets see what we have now.



```
mysql> select * from employs;
+-----+--------+
| ssn | cname  |
+-----+--------+
| 100 | apple  |
| 100 | lenovo |
| 101 | apple  |
| 102 | apple  |
+-----+--------+
4 rows in set (0.01 sec)
```

Now lets think about something new. Do we really need table for relations? Can we put makes table and company table into one?

```
mysql> create table company_makes(cname varchar(20), stock int, pname varchar(20),
    -> primary key(pname));

mysql> insert into company_makes values('apple', 180, 'iphone14');

mysql> insert into company_makes values('apple', 180, 'iphone15');
```

we can notice that the stock price has been repeatedly appearing for many times unnecessarily. So this is not a good practice.

```
mysql> drop table company_makes;
```

now let's try it on a different side.

```
mysql> create table product_makes(pname varchar(20), category varchar(20), price int,
cname varchar(20), primary key (pname));
```

this is good. So if we want to combine a entity table and relation table together, combine the relation table with the multiple side entity.

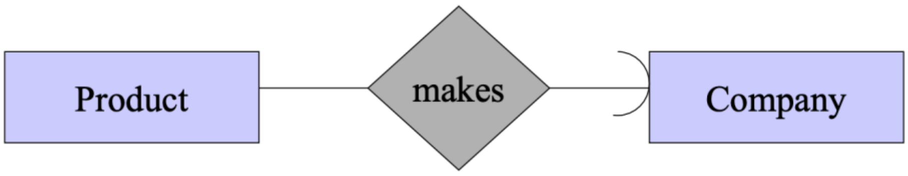
But still there's an issue. Let's try inserting the values.

```
mysql> insert into product_makes values('iphone14', 'cell',  1200, 'apple');

mysql> insert into product_makes values('iphone15', 'cell',  1500, 'apple');

mysql> insert into product_makes values('t450s', 'laptop',  2000, null);
```

A ceveate:
We are forced to add a null value to the product which we do not know what company it's made by. If we make them into separate tables, we don't have to do that.

Another case is merging two tables that has m-m relations. Conclusion: not a good idea.

但如果我们按照下面这个ER diagram来做database的话，这个cname不能为空。



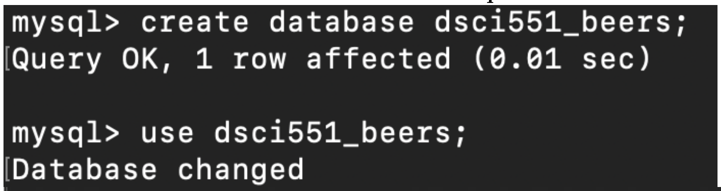
```
mysql> create table product_one(pname varchar(20), category varchar(20), price int, cname
varchar(20) not null, primary key (pname));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into product_one values('t450s', 'laptop', 2000, null);
ERROR 1048 (23000): Column 'cname' cannot be null
```

Lecture 6

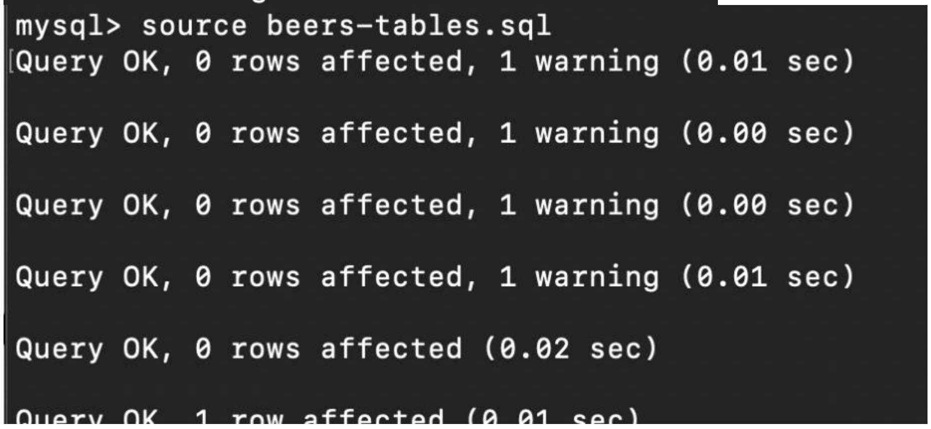
上传文件到ec2

```
(base) fionazhang@Fionas-MacBook-Air-2 ~ % sftp  -i
"dsci551-sp24.pem" ubuntu@ec2-3-137-155-242.us-east-
2.compute.amazonaws.com
Connected to ec2-3-137-155-242.us-east-
2.compute.amazonaws.com.
sftp> put beers-tables.sql
Uploading beers-tables.sql to /home/ubuntu/beers-
tables.sql
```

创建空白database，并导入提供的sql文件。

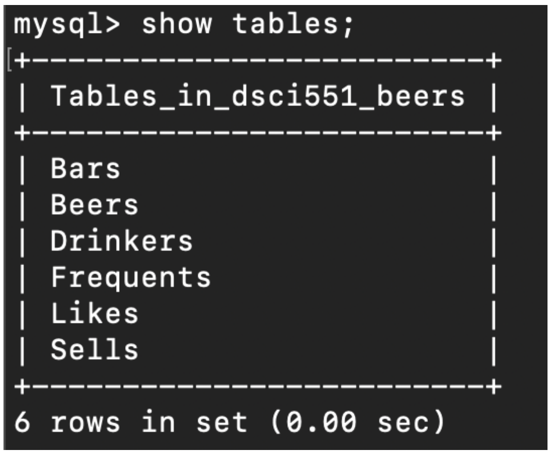


```
mysql> create database dsci551_beers;
Query OK, 1 row affected (0.01 sec)

mysql> use dsci551_beers;
Database changed
```



```
mysql> source beers-tables.sql
Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 1 row affected (0.01 sec)
```

看看有几个table



```
mysql> show tables;
+-------------------------+
| Tables_in_dsci551_beers |
+-------------------------+
| Bars                    |
| Beers                   |
| Drinkers                |
| Frequents               |
| Likes                   |
| Sells                   |
+-------------------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from Beers;

+------------+----------------+
| name       | manf            |
+------------+----------------+
| Bud        | Anheuser-Busch |
| Bud Lite   | Anheuser-Busch |
| Budweiser  | Heineken       |
| Michelob   | Anheuser-Busch |
| Summerbrew | Pete's         |
+------------+----------------+
5 rows in set (0.00 sec)
```

单个table的单个关键词filter：Selection

```
mysql> select name from Beers where manf = "Pete\'s";
+------------+
| name       |
+------------+
| Summerbrew |
+------------+
1 row in set (0.00 sec)

mysql> select name from Beers where name like 'bud%';
+-----------+
| name      |
+-----------+
| Bud       |
| Bud Lite  |
| Budweiser |
+-----------+
3 rows in set (0.00 sec)

mysql> select name from Beers where name like 'Bud%';
+-----------+
| name      |
+-----------+
| Bud       |
| Bud Lite  |
| Budweiser |
+-----------+
3 rows in set (0.00 sec)

mysql> select name from Beers where name like '%e';
+----------+
| name     |
+----------+
| Bud Lite |
+----------+
1 row in set (0.00 sec)

mysql> select name from Beers where name like '%e_';
+------------+
| name       |
+------------+
| Budweiser  |
| Summerbrew |
+------------+
2 rows in set (0.00 sec)

mysql> select name from Beers where name not like '%e';
+------------+
| name       |
+------------+
| Bud        |
| Budweiser  |
| Michelob   |
| Summerbrew |
+------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from Sells;
+------------+------------+-------+
| bar        | beer       | price |
+------------+------------+-------+
| Bob's bar  | Bud        |     3 |
| Bob's bar  | Summerbrew |     3 |
| Joe's bar  | Bud        |     3 |
| Joe's bar  | Bud Lite   |     3 |
| Joe's bar  | Michelob   |     3 |
| Joe's bar  | Summerbrew |     4 |
| Mary's bar | Bud        |  NULL |
| Mary's bar | Bud Lite   |     3 |
| Mary's bar | Budweiser  |     2 |
+------------+------------+-------+
9 rows in set (0.00 sec)

mysql> select * from Sells where price > 3 or price <= 3 or price is null;
+------------+------------+-------+
| bar        | beer       | price |
+------------+------------+-------+
| Bob's bar  | Bud        |     3 |
| Bob's bar  | Summerbrew |     3 |
| Joe's bar  | Bud        |     3 |
| Joe's bar  | Bud Lite   |     3 |
| Joe's bar  | Michelob   |     3 |
| Joe's bar  | Summerbrew |     4 |
| Mary's bar | Bud        |  NULL |
| Mary's bar | Bud Lite   |     3 |
| Mary's bar | Budweiser  |     2 |
+------------+------------+-------+
9 rows in set (0.00 sec)

mysql> select * from Sells where price is not null;
+------------+------------+-------+
| bar        | beer       | price |
+------------+------------+-------+
| Bob's bar  | Bud        |     3 |
| Bob's bar  | Summerbrew |     3 |
| Joe's bar  | Bud        |     3 |
| Joe's bar  | Bud Lite   |     3 |
| Joe's bar  | Michelob   |     3 |
| Joe's bar  | Summerbrew |     4 |
| Mary's bar | Bud Lite   |     3 |
| Mary's bar | Budweiser  |     2 |
+------------+------------+-------+
8 rows in set (0.00 sec)

mysql> select price, price from Sells;
+-------+-------+
| price | price |
+-------+-------+
|     3 |     3 |
|     3 |     3 |
|     3 |     3 |
|     3 |     3 |
|     3 |     3 |
|     4 |     4 |
|  NULL |  NULL |
|     3 |     3 |
|     2 |     2 |
+-------+-------+
9 rows in set (0.00 sec)
```

Alias:

```
mysql> select price as p1, price as p2 from Sells;
+------+------+
| p1   | p2   |
+------+------+
|    3 |    3 |
|    3 |    3 |
|    3 |    3 |
|    3 |    3 |
|    3 |    3 |
|    4 |    4 |
| NULL | NULL |
|    3 |    3 |
|    2 |    2 |
+------+------+
9 rows in set (0.00 sec)

mysql> select beer from Sells where price > 3;
+------------+
| beer       |
+------------+
| Summerbrew |
+------------+
1 row in set (0.00 sec)
```

Relational Algebra

Pandas:

(procedural)

```
Sells[Sells.price > 3][['beers']]
```

伪代码:

```
For s in Sells:
    if s.price > 3:
        output s.beer
```

```
mysql> select * from Likes;
+----------+------------+
| drinker  | beer       |
+----------+------------+
| Bill     | Bud        |
| Jennifer | Bud        |
| Steve    | Bud        |
| Steve    | Bud Lite   |
| Steve    | Michelob   |
| Steve    | Summerbrew |
+----------+------------+
6 rows in set (0.00 sec)

mysql> select drinker from Likes;
+----------+
| drinker  |
+----------+
| Bill     |
| Jennifer |
| Steve    |
| Steve    |
| Steve    |
| Steve    |
+----------+
6 rows in set (0.00 sec)

mysql> select distinct drinker from Likes;
+----------+
| drinker  |
+----------+
| Bill     |
| Jennifer |
| Steve    |
+----------+
3 rows in set (0.00 sec)
```

```
mysql> select distinct drinker from Likes order by drinker;
+----------+
| drinker  |
+----------+
| Bill     |
| Jennifer |
| Steve    |
+----------+
3 rows in set (0.00 sec)


mysql> select distinct drinker from Likes order by drinker desc;
+----------+
| drinker  |
+----------+
| Steve    |
| Jennifer |
| Bill     |
+----------+
3 rows in set (0.00 sec)

mysql> select distinct drinker from Likes order by drinker limit 1;
+---------+
| drinker |
+---------+
| Bill    |
+---------+
1 row in set (0.00 sec)

mysql> select distinct drinker from Likes order by drinker limit 1 offset 0;
+---------+
| drinker |
+---------+
| Bill    |
+---------+
1 row in set (0.00 sec)

mysql> select distinct drinker from Likes order by drinker limit 1 offset 2;
+---------+
| drinker |
+---------+
| Steve   |
+---------+
1 row in set (0.00 sec)

mysql> select distinct drinker from Likes order by drinker limit 2, 1;
+---------+
| drinker |
+---------+
| Steve   |
+---------+
1 row in set (0.00 sec)

mysql> select distinct drinker from Likes order by drinker limit 1, 2;
+----------+
| drinker  |
+----------+
| Jennifer |
| Steve    |
+----------+
2 rows in set (0.00 sec)
```

limit 1, 2 means limit 2 offset 1

```
mysql> select concat(beer, ' ', price) from Sells;
+--------------------------+
| concat(beer, ' ', price) |
+--------------------------+
| Bud 3                    |
| Summerbrew 3             |
| Bud 3                    |
| Bud Lite 3              |
| Michelob 3              |
| Summerbrew 4            |
| NULL                    |
| Bud Lite 3              |
| Budweiser 2             |
+-------------------------+
9 rows in set (0.00 sec)
```

SubQuery:

Task:

Find the manf of the beer that Jennifer likes

Step1. Find beers like by Jennifer

Step2. Find manf of those beers found in step1

Step1.

```
mysql> select beer from Likes where drinker = "Jennifer";
+------+
| beer |
+------+
| Bud  |
+------+
1 row in set (0.00 sec)
```

Step2.

```
mysql> select manf from Beers where name = "Bud";
+----------------+
| manf           |
+----------------+
| Anheuser-Busch |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> select manf from Beers where name = (select beer from Likes where drinker = "Jennifer");
+----------------+
| manf           |
+----------------+
| Anheuser-Busch |
+----------------+
1 row in set (0.00 sec)
```

伪代码：

```
For l in Likes:
    if b.drinker = “Jennifer”:
        for b in Beers:
            if b.name == l.beer:
                output b.manf
```

Task 2:

Find the manf of the beer that Steve likes

伪代码：

```
for l in Likes:
    if l.drinker == “Steve”:
        add l.beer to result

result = {‘Bud’, ‘Bud Lite’, ...}

for b in Beers:
    if (b.name in result):
        output b.manf
```

```
mysql> select manf from Beers where name in (select beer from Likes where drinker = "Steve");
+----------------+
| manf           |
+----------------+
| Anheuser-Busch |
| Anheuser-Busch |
| Anheuser-Busch |
| Pete's         |
+----------------+
4 rows in set (0.00 sec)
```

Here if we use “=” instead of “in”, errors will arise.

下面我们看看 <, >, >=, <=, all, any的用法

```
mysql> select price from Sells where price >= all (select price from Sells);
Empty set (0.00 sec)
```

是空的因为没有任何

```
mysql> # for s in Sells;
mysql> #   if s.price >= all {3, 3, ..., 4, null, 3, 2}
mysql> #       output s.price
mysql>
mysql>
mysql> select price from Sells;
+-------+
| price |
+-------+
|     3 |
|     3 |
|     3 |
|     3 |
|     3 |
|     4 |
|  NULL |
|     3 |
|     2 |
+-------+
9 rows in set (0.00 sec)
```

```
mysql> select price from Sells where price >= all (select price from Sells where price is not null);
+-------+
| price |
+-------+
|     4 |
+-------+
1 row in set (0.00 sec)

mysql> select price from Sells where price > any (select price from Sells);
+-------+
| price |
+-------+
|     3 |
|     3 |
|     3 |
|     3 |
|     3 |
|     4 |
|     3 |
+-------+
7 rows in set (0.01 sec)
```

Task:

find drinkers who likes only a single beer

伪代码：

```
for l1 in Likes:
        repli = false
        for l2 in Likes:
            if (l2.drinker == l1.drinker and l2.beer != l1.beer):
            repli = true
            break
        if (!repli):
            output l1.drinker
```

```
mysql> select drinker from Likes l1 where not exists (select beer from Likes l2 where l2.drinker =
l1.drinker and l2.beer != l1.beer);
+----------+
| drinker  |
+----------+
| Bill     |
| Jennifer |
+----------+
2 rows in set (0.00 sec)
```

Summary:

a (not) in (Q)

a (!) = (Q)

price >= all (Q), price >= any (Q)

 (not) exists (Q)

  Means: attrib not in subquery

a in (Q) <==> a = any (Q)

a not in (Q) <==> a != all (Q)

Task2:

Find beers that nobody likes

```
mysql> select name from Beers where name not in (select beer from Likes);
+-----------+
| name      |
+-----------+
| Budweiser |
+-----------+
1 row in set (0.00 sec)
```

Subquery is O(1) because (select beer from Likes) will always give the same result.

注意 not in <==> != all, 而不是not in <==> != any. 请看VCR：

```
mysql> select name from Beers where name != all (select beer from Likes);
```

```
+-----------+
| name      |
+-----------+
| Budweiser |
+-----------+
1 row in set (0.00 sec)

mysql> select name from Beers where name != any (select beer from Likes);
+------------+
| name       |
+------------+
| Bud        |
| Bud Lite   |
| Budweiser  |
| Michelob   |
| Summerbrew |
+------------+
5 rows in set (0.00 sec)
```

now lets write JOIN

```
mysql> select * from Sells;
+------------+------------+-------+
| bar        | beer       | price |
+------------+------------+-------+
| Bob's bar  | Bud        |     3 |
| Bob's bar  | Summerbrew |     3 |
| Joe's bar  | Bud        |     3 |
| Joe's bar  | Bud Lite   |     3 |
| Joe's bar  | Michelob   |     3 |
| Joe's bar  | Summerbrew |     4 |
| Mary's bar | Bud        |  NULL |
| Mary's bar | Bud Lite   |     3 |
| Mary's bar | Budweiser  |     2 |
+------------+------------+-------+
9 rows in set (0.00 sec)
```

这三个都是一个意思：

```
mysql> select name from Beers join Likes on name = beer;

mysql> select name from Beers join Likes on Beers.name = Likes.beer;

mysql> select name from Beers b join Likes l on b.name = l.beer;
+------------+
| name       |
+------------+
| Bud        |
| Bud        |
| Bud        |
| Bud Lite   |
| Michelob   |
| Summerbrew |
+------------+
6 rows in set (0.00 sec)
```

伪代码：

```
for b in Beers:
    for l in Likes:
        if b.name = l.beer:
            output b.name
```

```
mysql> select distinct name from Beers b join Likes l on b.name = l.beer;
+------------+
| name       |
+------------+
| Bud        |
| Bud Lite   |
| Michelob   |
| Summerbrew |
+------------+
4 rows in set (0.00 sec)
```

```
mysql> # O(m+n) assume likes has m rows, beers has n rows
mysql>
mysql> # subquery: O(m+n) assume likes has m rows, beers has n rows
mysql> # join: O(m*n) assume likes has m rows, beers has n rows
mysql>
mysql> # select * from Beers, Likes;
mysql> # it will have 30 lines. Doing cartesian product.
```

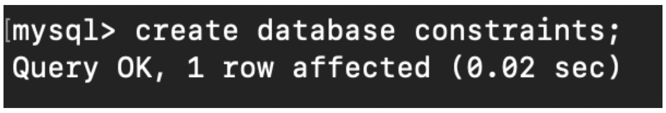
```
mysql> select * from Frequents;
+----------+------------+
| drinker  | bar        |
+----------+------------+
| Steve    | Bob's bar  |
| David    | Joe's bar  |
| Jennifer | Joe's bar  |
| Steve    | Joe's bar  |
| Bill     | Mary's bar |
+----------+------------+
5 rows in set (0.00 sec)

mysql> select distinct f.drinker from Frequents f join Likes l on f.drinker = l.drinker;
+----------+
| drinker  |
+----------+
| Bill     |
| Jennifer |
| Steve    |
+----------+
3 rows in set (0.00 sec)

mysql> select * from Frequents f join Likes l on f.drinker = l.drinker;
+----------+------------+----------+------------+
| drinker  | bar        | drinker  | beer       |
+----------+------------+----------+------------+
| Bill     | Mary's bar | Bill     | Bud        |
| Jennifer | Joe's bar  | Jennifer | Bud        |
| Steve    | Bob's bar  | Steve    | Bud        |
| Steve    | Joe's bar  | Steve    | Bud        |
| Steve    | Bob's bar  | Steve    | Bud Lite   |
| Steve    | Joe's bar  | Steve    | Bud Lite   |
| Steve    | Bob's bar  | Steve    | Michelob   |
| Steve    | Joe's bar  | Steve    | Michelob   |
| Steve    | Bob's bar  | Steve    | Summerbrew |
| Steve    | Joe's bar  | Steve    | Summerbrew |
+----------+------------+----------+------------+
10 rows in set (0.00 sec)

mysql> select * from Frequents f natural join Likes l where f.drinker = l.drinker
;
+----------+------------+------------+
| drinker  | bar        | beer       |
+----------+------------+------------+
| Bill     | Mary's bar | Bud        |
| Jennifer | Joe's bar  | Bud        |
| Steve    | Bob's bar  | Bud        |
| Steve    | Joe's bar  | Bud        |
| Steve    | Bob's bar  | Bud Lite   |
| Steve    | Joe's bar  | Bud Lite   |
| Steve    | Bob's bar  | Michelob   |
| Steve    | Joe's bar  | Michelob   |
| Steve    | Bob's bar  | Summerbrew |
| Steve    | Joe's bar  | Summerbrew |
+----------+------------+------------+
10 rows in set (0.00 sec)
```

等效于

```
mysql> select f.drinker, f.bar, l.beer from Frequents f join Likes l on f.drinker = l.drinker;
```

Week 8 Constraints

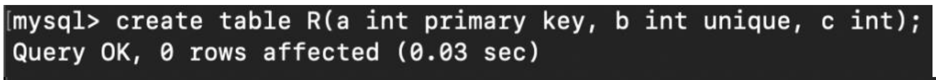
建一个新的database Constraints



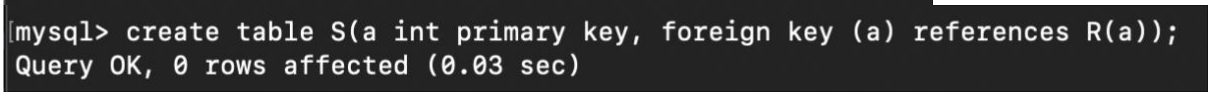
```
mysql> create database constraints;
Query OK, 1 row affected (0.02 sec)
```

建两个table，R和S

S有foreign key，连R的primary key



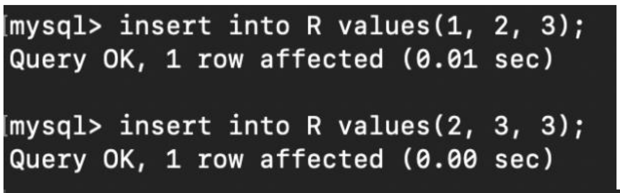
```
mysql> create table R(a int primary key, b int unique, c int);
Query OK, 0 rows affected (0.03 sec)
```



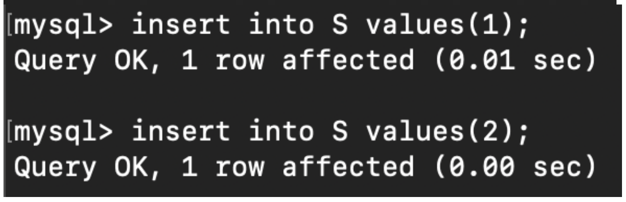
```
mysql> create table S(a int primary key, foreign key (a) references R(a));
Query OK, 0 rows affected (0.03 sec)
```

一、Foreign-key, or referential-integrity

往里面分别放两个 entry进去

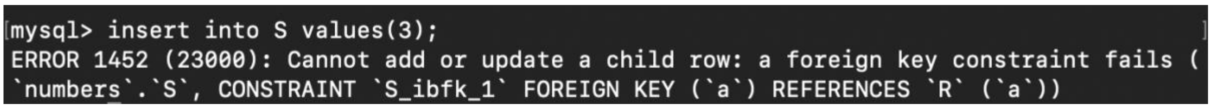


```
mysql> insert into R values(1, 2, 3);
Query OK, 1 row affected (0.01 sec)

mysql> insert into R values(2, 3, 3);
Query OK, 1 row affected (0.00 sec)
```



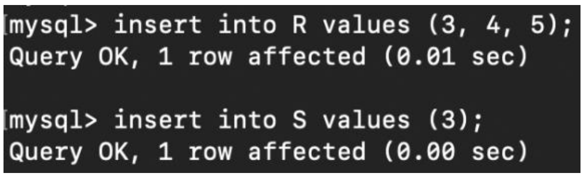
```
mysql> insert into S values(1);
Query OK, 1 row affected (0.01 sec)

mysql> insert into S values(2);
Query OK, 1 row affected (0.00 sec)
```

想往S放第三个entry的时候就不行了，因为R里面没有a = 3这个primary key。



```
mysql> insert into S values(3);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (
`numbers`.`S`, CONSTRAINT `S_ibfk_1` FOREIGN KEY (`a`) REFERENCES `R` (`a`))
```

在R里面建了第三个entry之后，再往S里面加数据，就不报错了。



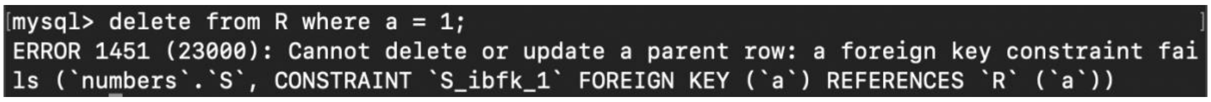
```
mysql> insert into R values (3, 4, 5);
Query OK, 1 row affected (0.01 sec)

mysql> insert into S values (3);
Query OK, 1 row affected (0.00 sec)
```

总结：

Parent table在不同情况下要不要进行constraints检测：

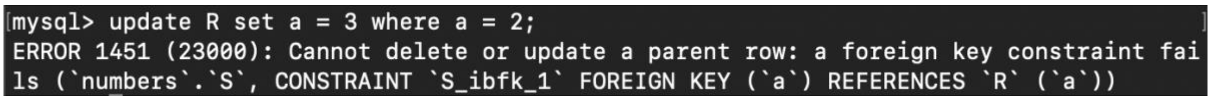
Insert: n　　　　　　　delete: y　　　　　　　update: y

Delete:



```
mysql> delete from R where a = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fai
ls (`numbers`.`S`, CONSTRAINT `S_ibfk_1` FOREIGN KEY (`a`) REFERENCES `R` (`a`))
```

Update:



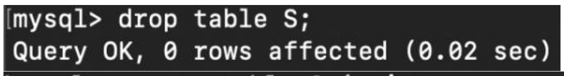
```
mysql> update R set a = 3 where a = 2;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fai
ls (`numbers`.`S`, CONSTRAINT `S_ibfk_1` FOREIGN KEY (`a`) REFERENCES `R` (`a`))
```

总结：

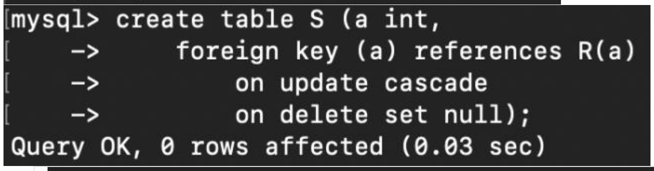
child table在不同情况下要不要进行constraints检测：

Insert: y   delete: n   update: y

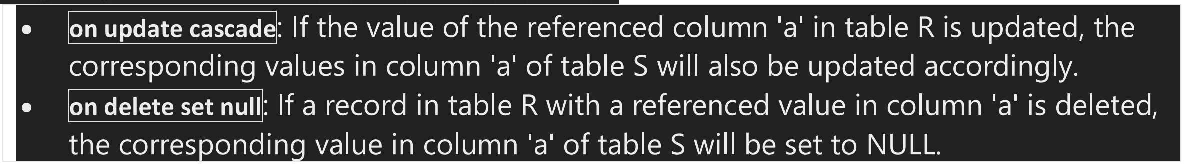
下面重新建一个新的table S



```
mysql> drop table S;
Query OK, 0 rows affected (0.02 sec)
```

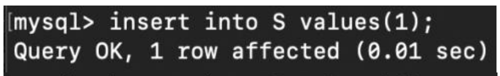


```
mysql> create table S (a int,
    ->     foreign key (a) references R(a)
    ->         on update cascade
    ->         on delete set null);
Query OK, 0 rows affected (0.03 sec)
```

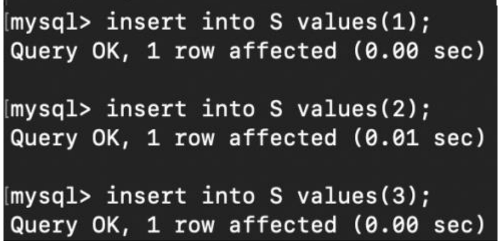


- **on update cascade**: If the value of the referenced column 'a' in table R is updated, the corresponding values in column 'a' of table S will also be updated accordingly.
- **on delete set null**: If a record in table R with a referenced value in column 'a' is deleted, the corresponding value in column 'a' of table S will be set to NULL.

往 S 里面加 4 条数据:

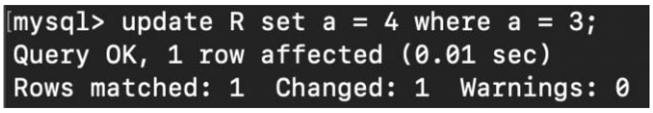


```
mysql> insert into S values(1);
Query OK, 1 row affected (0.01 sec)
```



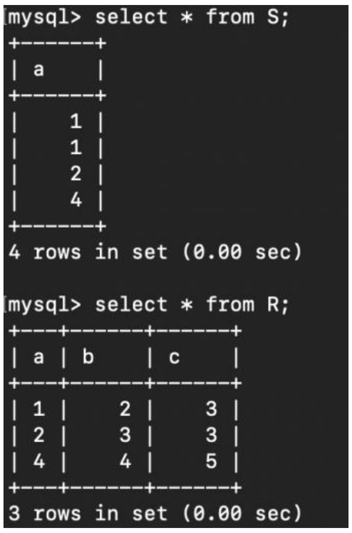
```
mysql> insert into S values(1);
Query OK, 1 row affected (0.00 sec)

mysql> insert into S values(2);
Query OK, 1 row affected (0.01 sec)

mysql> insert into S values(3);
Query OK, 1 row affected (0.00 sec)
```

Update



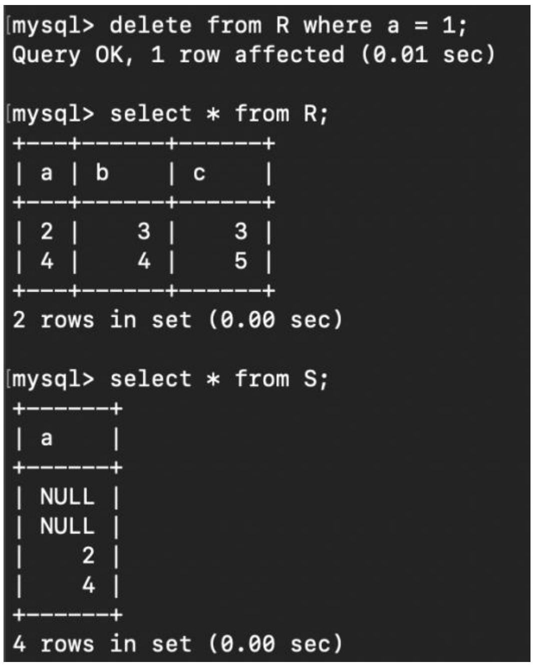
```
mysql> update R set a = 4 where a = 3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Update 之后:



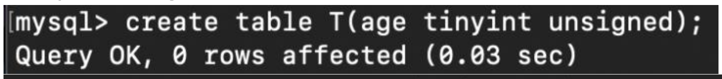
```
mysql> select * from S;
+------+
| a    |
+------+
|    1 |
|    1 |
|    2 |
|    4 |
+------+
4 rows in set (0.00 sec)

mysql> select * from R;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 1 |    2 |    3 |
| 2 |    3 |    3 |
| 4 |    4 |    5 |
+---+------+------+
3 rows in set (0.00 sec)
```

Delete:

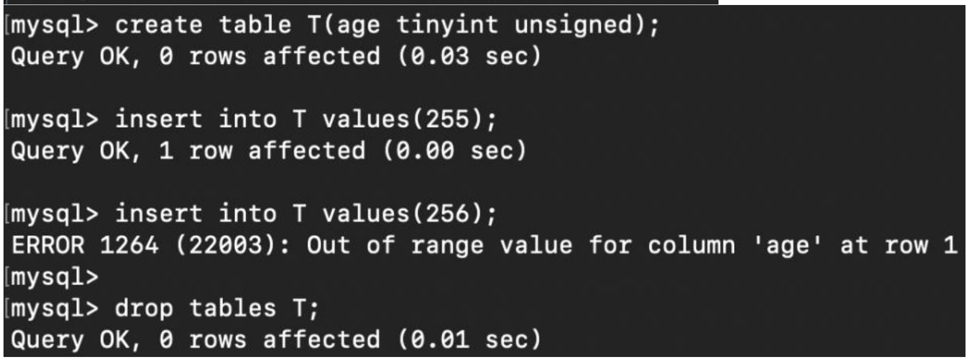


```
mysql> delete from R where a = 1;
Query OK, 1 row affected (0.01 sec)

mysql> select * from R;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |    3 |    3 |
| 4 |    4 |    5 |
+---+------+------+
2 rows in set (0.00 sec)

mysql> select * from S;
+------+
| a    |
+------+
| NULL |
| NULL |
|    2 |
|    4 |
+------+
4 rows in set (0.00 sec)
```

* tinyint unsigned

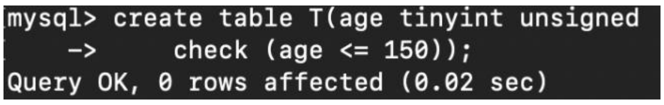


```
mysql> create table T(age tinyint unsigned);
Query OK, 0 rows affected (0.03 sec)
```



```
mysql> create table T(age tinyint unsigned);
Query OK, 0 rows affected (0.03 sec)

mysql> insert into T values(255);
Query OK, 1 row affected (0.00 sec)

mysql> insert into T values(256);
ERROR 1264 (22003): Out of range value for column 'age' at row 1
mysql>
mysql> drop tables T;
Query OK, 0 rows affected (0.01 sec)
```

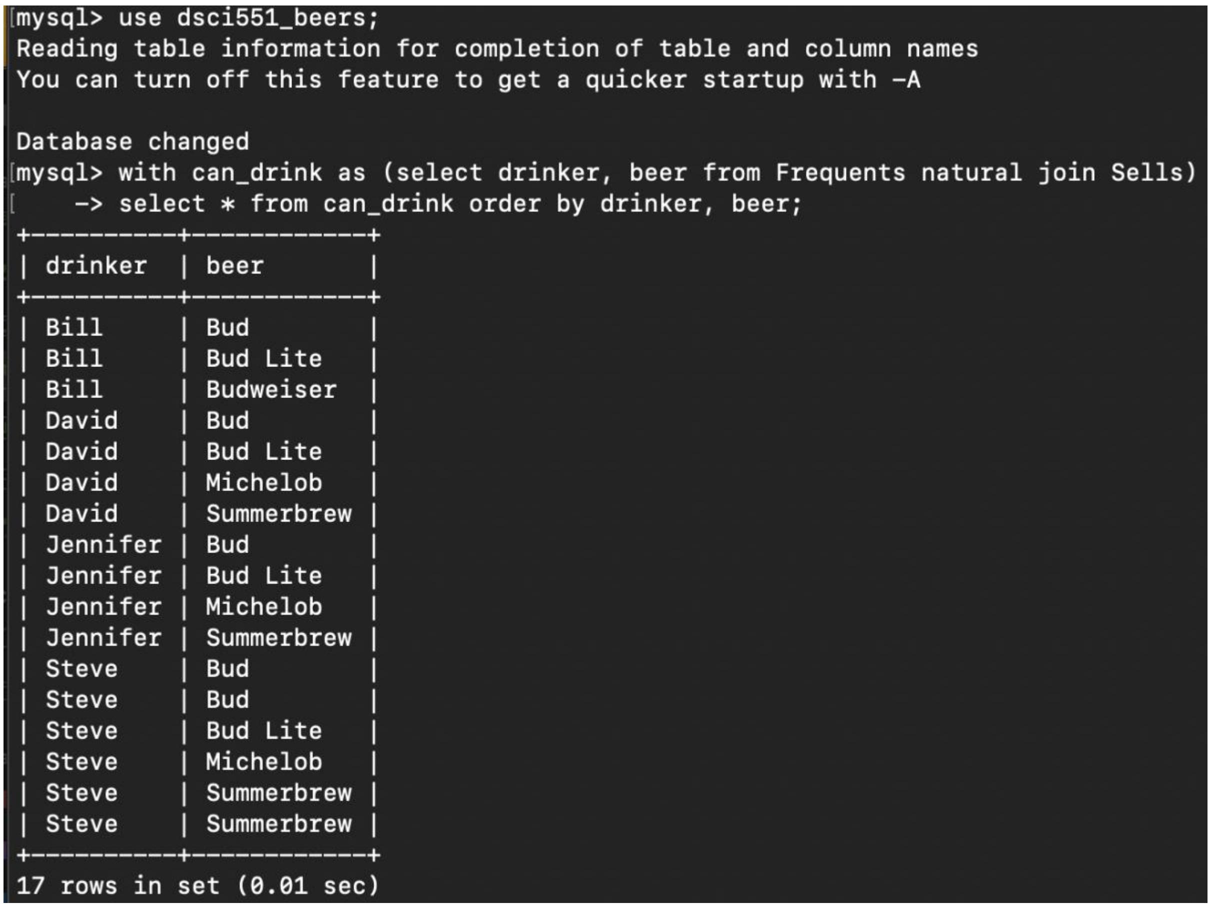
Check 关键词



```
mysql> create table T(age tinyint unsigned
    ->     check (age <= 150));
Query OK, 0 rows affected (0.02 sec)
```

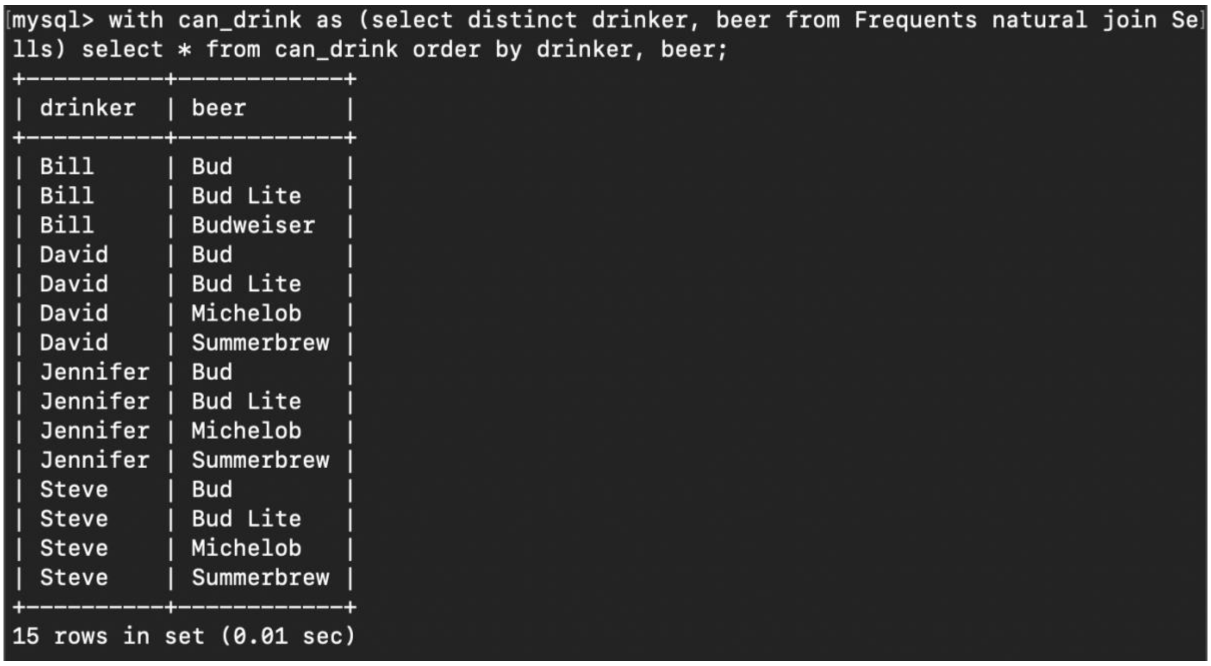
二、Value-based constraints

Views



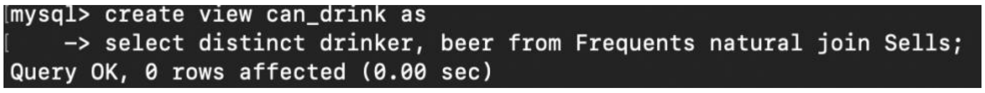
```
mysql> use dsci551_beers;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> with can_drink as (select drinker, beer from Frequents natural join Sells)
    -> select * from can_drink order by drinker, beer;
+----------+------------+
| drinker  | beer       |
+----------+------------+
| Bill     | Bud        |
| Bill     | Bud Lite   |
| Bill     | Budweiser  |
| David    | Bud        |
| David    | Bud Lite   |
| David    | Michelob   |
| David    | Summerbrew |
| Jennifer | Bud        |
| Jennifer | Bud Lite   |
| Jennifer | Michelob   |
| Jennifer | Summerbrew |
| Steve    | Bud        |
| Steve    | Bud        |
| Steve    | Bud Lite   |
| Steve    | Michelob   |
| Steve    | Summerbrew |
| Steve    | Summerbrew |
+----------+------------+
17 rows in set (0.01 sec)
```

发现有重复的，加上 distinct 关键词



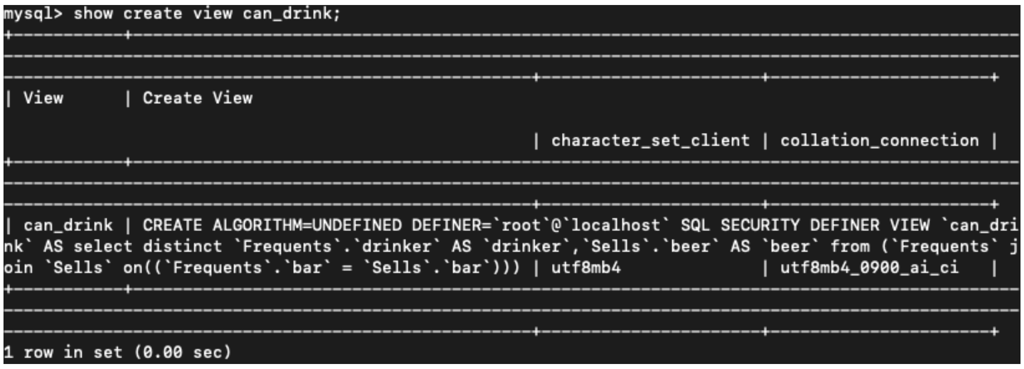
```
mysql> with can_drink as (select distinct drinker, beer from Frequents natural join Se
lls) select * from can_drink order by drinker, beer;
+----------+------------+
| drinker  | beer       |
+----------+------------+
| Bill     | Bud        |
| Bill     | Bud Lite   |
| Bill     | Budweiser  |
| David    | Bud        |
| David    | Bud Lite   |
| David    | Michelob   |
| David    | Summerbrew |
| Jennifer | Bud        |
| Jennifer | Bud Lite   |
| Jennifer | Michelob   |
| Jennifer | Summerbrew |
| Steve    | Bud        |
| Steve    | Bud Lite   |
| Steve    | Michelob   |
| Steve    | Summerbrew |
+----------+------------+
15 rows in set (0.01 sec)
```

为了后续继续读取 can_drink 这个 table，用 view：



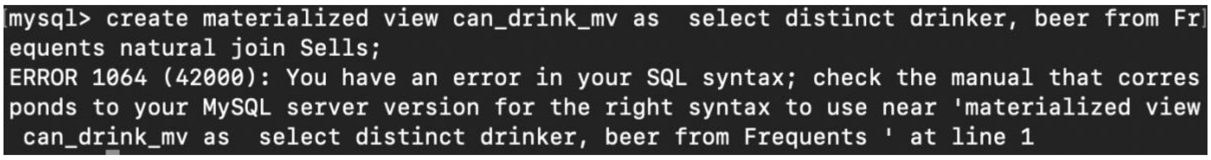
```
[mysql> create view can_drink as
[    -> select distinct drinker, beer from Frequents natural join Sells;
Query OK, 0 rows affected (0.00 sec)
```

这个 can_drink table 实际上不是一个真的 table，sql 只是一个 query 存下来了而已：



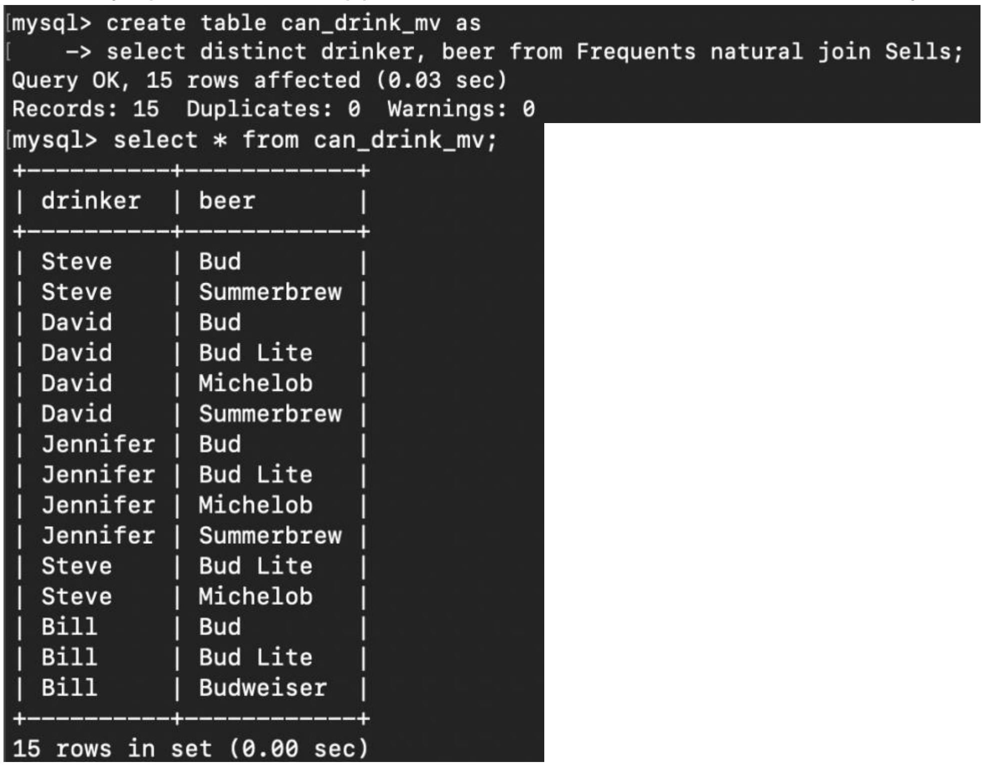
```
mysql> show create view can_drink;
+-----------+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
------------------------------------------------------+----------------------+----------------------+
| View      | Create View

                                                      | character_set_client | collation_connection |
+-----------+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
------------------------------------------------------+----------------------+----------------------+
| can_drink | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `can_dri
nk` AS select distinct `Frequents`.`drinker` AS `drinker`,`Sells`.`beer` AS `beer` from (`Frequents` j
oin `Sells` on((`Frequents`.`bar` = `Sells`.`bar`))) | utf8mb4              | utf8mb4_0900_ai_ci   |
+-----------+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
------------------------------------------------------+----------------------+----------------------+
1 row in set (0.00 sec)
```

In oracle, we can create a materialized view. Like this:



```
[mysql> create materialized view can_drink_mv as  select distinct drinker, beer from Fr]
equents natural join Sells;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corres
ponds to your MySQL server version for the right syntax to use near 'materialized view
 can_drink_mv as  select distinct drinker, beer from Frequents ' at line 1
```

But in mysql it does not support this. But we can create a similar one. By stating create table:



```
[mysql> create table can_drink_mv as
[    -> select distinct drinker, beer from Frequents natural join Sells;
Query OK, 15 rows affected (0.03 sec)
Records: 15  Duplicates: 0  Warnings: 0
[mysql> select * from can_drink_mv;
+----------+------------+
| drinker  | beer       |
+----------+------------+
| Steve    | Bud        |
| Steve    | Summerbrew |
| David    | Bud        |
| David    | Bud Lite   |
| David    | Michelob   |
| David    | Summerbrew |
| Jennifer | Bud        |
| Jennifer | Bud Lite   |
| Jennifer | Michelob   |
| Jennifer | Summerbrew |
| Steve    | Bud Lite   |
| Steve    | Michelob   |
| Bill     | Bud        |
| Bill     | Bud Lite   |
| Bill     | Budweiser  |
+----------+------------+
15 rows in set (0.00 sec)
```