sudo service mongod stop

sudo service mysql start

然后：

A screenshot of a computer screen

Description automatically generated

Password: Dsci-551

**Lecture 5**

A diagram of a company

Description automatically generated

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.**

A screen shot of a black and white screen

Description automatically generated

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.**

A black and white screen with white text

Description automatically generated

**\*FYI:**

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

it generates values with NULL.

A black screen with white text

Description automatically generated

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.**

A screenshot of a computer code

Description automatically generated

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.**

A close-up of a number

Description automatically generated

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.**

A white text with black lines

Description automatically generated with medium confidence

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.**

A screenshot of a computer code

Description automatically generated

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。

A grey diamond with black text

Description automatically generated

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

A close up of a computer code

Description automatically generated

创建空白database，并导入提供的sql文件。

A black background with white text

Description automatically generated

A screenshot of a computer error

Description automatically generated

看看有几个table

A black screen with white text

Description automatically generated

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**

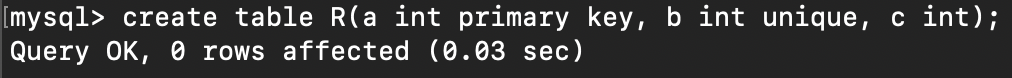
建一个新的database Constraints

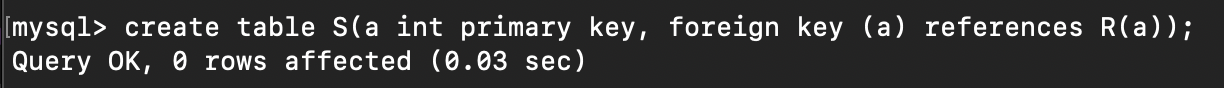
A black background with white text

Description automatically generated

建两个table，R和S

S有foreign key，连R的primary key





**一、Foreign-key, or referential-integrity**

往里面分别放两个entry进去

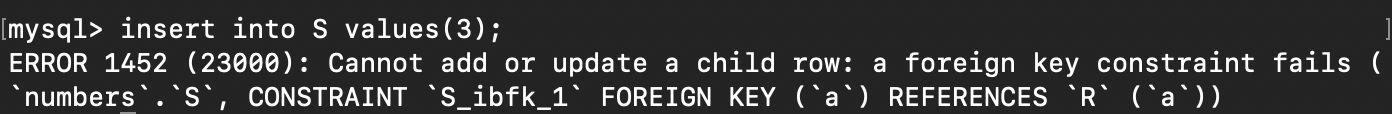
A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

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。



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。

A black background with white text

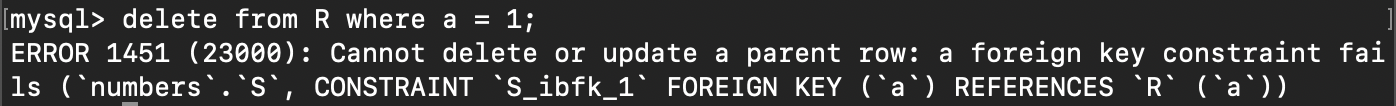
Description automatically generated

总结：

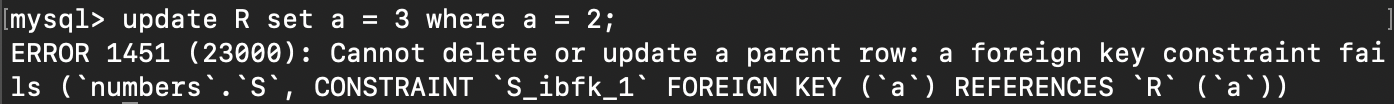
Parent table在不同情况下要不要进行constraints检测：

Insert: n delete: y update: y

Delete:



Update:



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



A black background with white text

Description automatically generated

* **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：

A black background with white text

Description automatically generated

A screen shot of a computer code

Description automatically generated

Update

A black background with white text

Description automatically generated

Update 之后：

A screenshot of a computer program

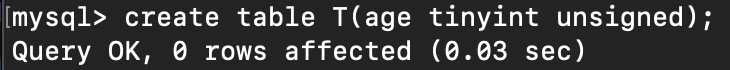
Description automatically generated

Delete:

A screenshot of a computer program

Description automatically generated

\* tinyint unsigned



A black screen with white text

Description automatically generated

Check 关键词

A black background with white text

Description automatically generated

**二、Value-based constraints**

**Views**

A screenshot of a computer

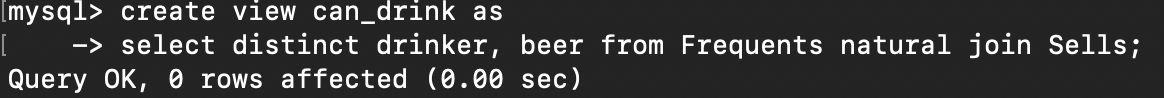
Description automatically generated

发现有重复的，加上distinct关键词

A screenshot of a computer

Description automatically generated

为了后续继续读取can\_drink这个table，用 view：

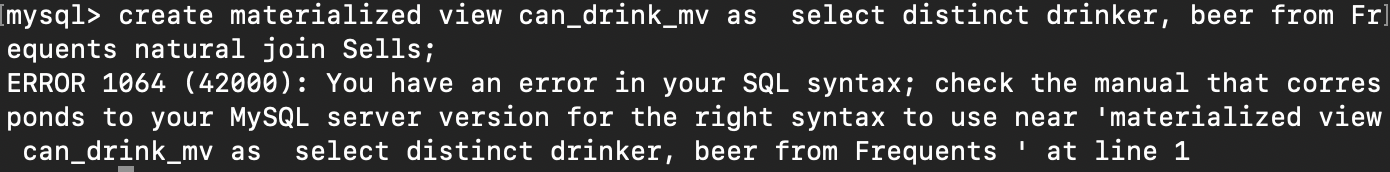


这个can\_drink table实际上不是一个真的table，sql只是一个query存下来了而已：

A black screen with white lines

Description automatically generated

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



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



A screenshot of a computer

Description automatically generated