mysql day03 课堂笔记

1、查询每一个员工的所在部门名称？要求显示员工名和部门名。

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+---------+---------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+--------+-----------+------+------------+---------+---------+--------+
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-20 | 1600.00 |  300.00 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00 |     30 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-28 | 1250.00 | 1400.00 |     30 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850.00 |    NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1987-04-19 | 3000.00 |    NULL |     20 |
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30 |
|  7876 | ADAMS  | CLERK     | 7788 | 1987-05-23 | 1100.00 |    NULL |     20 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-03 |  950.00 |    NULL |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 |
+-------+--------+-----------+------+------------+---------+---------+--------+

mysql> select * from dept;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
```

从 emp 表中取 ename，从 dept 表中取 dname，没有条件限制最终查询结果是？

```
ENAME                DNAME
--------------------------------------
SMITH                ACCOUNTING    无效记录
SMITH                RESEARCH      有效记录
SMITH                SALES             无效记录
SMITH                OPERATIONS    无效记录

ALLEN                ACCOUNTING
ALLEN                RESEARCH
ALLEN                SALES
ALLEN                OPERATIONS
```

.....
56 条记录。

加个条件是为了达到 4 选 1，也是为了数据的有效性。

```
select
    e.ename,d.dname
from
    emp e
join
    dept d
on
    e.deptno = d.deptno;
```

加条件只是为了避免笛卡尔积现象，只是为了查询出有效的组合记录。
匹配的次数一次都没有少，还是 56 次。

2、insert 语句可以一次插入多条记录吗？【掌握】

可以的！

```
mysql> desc t_user;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| id          | int(11)     | YES  |     | NULL    |       |
| name        | varchar(32) | YES  |     | NULL    |       |
| birth       | date        | YES  |     | NULL    |       |
| create_time | datetime    | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

一次可以插入多条记录：

```
    insert into t_user(id,name,birth,create_time) values
    (1,'zs','1980-10-11',now()),
    (2,'lisi','1981-10-11',now()),
    (3,'wangwu','1982-10-11',now());
```

语法：insert into t_user(字段名 1,字段名 2) values(),(),(),();

```
mysql> select * from t_user;
+------+--------+------------+---------------------+
| id   | name   | birth      | create_time         |
+------+--------+------------+---------------------+
|    1 | zs     | 1980-10-11 | 2020-03-19 09:37:01 |
|    2 | lisi   | 1981-10-11 | 2020-03-19 09:37:01 |
|    3 | wangwu | 1982-10-11 | 2020-03-19 09:37:01 |
+------+--------+------------+---------------------+
```

3、快速创建表？【了解内容】

```
mysql> create table emp2 as select * from emp;
```

原理：
将一个查询结果当做一张表新建!!!!!
这个可以完成表的快速复制!!!!
表创建出来，同时表中的数据也存在了!!!

```
create table mytable as select empno,ename from emp where job = 'MANAGER';
```

4、将查询结果插入到一张表当中？insert 相关的!!!【了解内容】

```
create table dept_bak as select * from dept;
mysql> select * from dept_bak;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+

insert into dept_bak select * from dept; //很少用!

mysql> select * from dept_bak;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
```

5、快速删除表中的数据？【truncate 比较重要，必须掌握】

```
//删除 dept_bak 表中的数据
delete from dept_bak; //这种删除数据的方式比较慢。

mysql> select * from dept_bak;
Empty set (0.00 sec)
```

delete 语句删除数据的原理？（delete 属于 DML 语句!!!）

表中的数据被删除了，但是这个数据在硬盘上的真实存储空间不会被释放!!!

这种删除缺点是：删除效率比较低。

这种删除优点是：支持回滚，后悔了可以再恢复数据!!!

truncate 语句删除数据的原理？

这种删除效率比较高，表被一次截断，物理删除。

这种删除缺点：不支持回滚。

这种删除优点：快速。

用法：truncate table dept_bak; （这种操作属于 DDL 操作。）

大表非常大，上亿条记录？？？？

删除的时候，使用 delete，也许需要执行 1 个小时才能删除完！效率较低。

可以选择使用 truncate 删除表中的数据。只需要不到 1 秒钟的时间就删除结束。效率较高。

但是使用 truncate 之前，必须仔细询问客户是否真的要删除，并警告删除之后不可恢复！

truncate 是删除表中的数据，表还在！

删除表操作？

drop table 表名; // 这不是删除表中的数据，这是把表删除。

6、对表结构的增删改？

什么是对表结构的修改？

添加一个字段，删除一个字段，修改一个字段!!!

对表结构的修改需要使用：alter
属于 DDL 语句

DDL 包括：create drop alter

第一：在实际的开发中，需求一旦确定之后，表一旦设计好之后，很少的
进行表结构的修改。因为开发进行中的时候，修改表结构，成本比较高。
修改表的结构，对应的 java 代码就需要进行大量的修改。成本是比较高的。
这个责任应该由设计人员来承担！

第二：由于修改表结构的操作很少，所以我们不需要掌握，如果有一天
真的要修改表结构，你可以使用工具!!!!

修改表结构的操作是不需要写到 java 程序中的。实际上也不是 java 程序员的范畴。

7、约束（非常重要，五颗星*****）

7.1、什么是约束？

约束对应的英语单词：constraint
在创建表的时候，我们可以给表中的字段加上一些约束，来保证这个表中数据的完整性、有效性!!!

约束的作用就是为了保证：表中的数据有效!!

7.2、约束包括哪些？

非空约束：not null
唯一性约束: unique
主键约束: primary key （简称 PK）
外键约束：foreign key（简称 FK）
检查约束：check（mysql 不支持，oracle 支持）

我们这里重点学习四个约束：
- not null
- unique
- primary key
- foreign key

7.3、非空约束：not null

非空约束 not null 约束的字段不能为 NULL。

```
drop table if exists t_vip;
create table t_vip(
    id int,
    name varchar(255) not null  // not null 只有列级约束，没有表级约束!
);
insert into t_vip(id,name) values(1,'zhangsan');
insert into t_vip(id,name) values(2,'lisi');

insert into t_vip(id) values(3);
ERROR 1364 (HY000): Field 'name' doesn't have a default value
```

小插曲：
- xxxx.sql 这种文件被称为 sql 脚本文件。
- sql 脚本文件中编写了大量的 sql 语句。
- 我们执行 sql 脚本文件的时候，该文件中所有的 sql 语句会全部执行！
- 批量的执行 SQL 语句，可以使用 sql 脚本文件。
- 在 mysql 当中怎么执行 sql 脚本呢？

```
mysql> source D:\course\03-MySQL\document\vip.sql
```

你在实际的工作中，第一天到了公司，项目经理会给你一个 xxx.sql 文件，你执行这个脚本文件，你电脑上的数据库数据就有了！

7.4、唯一性约束: unique

唯一性约束 unique 约束的字段不能重复，但是可以为 NULL。

```
drop table if exists t_vip;
create table t_vip(
    id int,
    name varchar(255) unique,
    email varchar(255)
);
insert into t_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');
insert into t_vip(id,name,email) values(2,'lisi','lisi@123.com');
insert into t_vip(id,name,email) values(3,'wangwu','wangwu@123.com');
select * from t_vip;

insert into t_vip(id,name,email) values(4,'wangwu','wangwu@sina.com');
ERROR 1062 (23000): Duplicate entry 'wangwu' for key 'name'

insert into t_vip(id) values(4);
insert into t_vip(id) values(5);
+------+----------+------------------+
| id   | name     | email            |
+------+----------+------------------+
|    1 | zhangsan | zhangsan@123.com |
|    2 | lisi     | lisi@123.com     |
|    3 | wangwu   | wangwu@123.com   |
|    4 | NULL     | NULL             |
|    5 | NULL     | NULL             |
+------+----------+------------------+
```

name 字段虽然被 unique 约束了，但是可以为 NULL。

新需求：name 和 email 两个字段联合起来具有唯一性!!!!

```
drop table if exists t_vip;
create table t_vip(
    id int,
    name varchar(255) unique,  // 约束直接添加到列后面的，叫做列级约束。
    email varchar(255) unique
);
```

这张表这样创建是不符合我以上“新需求”的。

这样创建表示：name 具有唯一性，email 具有唯一性。各自唯一。

以下这样的数据是符合我“新需求”的。

但如果采用以上方式创建表的话，肯定创建失败，因为'zhangsan'和'zhangsan'重复了。

```
insert into t_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');
insert into t_vip(id,name,email) values(2,'zhangsan','zhangsan@sina.com');
```

怎么创建这样的表，才能符合新需求呢？

```
drop table if exists t_vip;
create table t_vip(
    id int,
```

```
                name varchar(255),
                email varchar(255),
                unique(name,email) // 约束没有添加在列的后面，这种约束被称为表级约束。
        );
        insert                  into                  t_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');
        insert                  into                  t_vip(id,name,email) values(2,'zhangsan','zhangsan@sina.com');
        select * from t_vip;

        name 和 email 两个字段联合起来唯一!!!
        insert                  into                  t_vip(id,name,email) values(3,'zhangsan','zhangsan@sina.com');
        ERROR 1062 (23000): Duplicate entry 'zhangsan-zhangsan@sina.com' for key 'name'
```

什么时候使用表级约束呢？

需要给多个字段联合起来添加某一个约束的时候，需要使用表级约束。

unique 和 not null 可以联合吗？

```
drop table if exists t_vip;
create table t_vip(
    id int,
    name varchar(255) not null unique
);

mysql> desc t_vip;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | int(11)      | YES  |     | NULL    |       |
| name  | varchar(255) | NO   | PRI | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

在 mysql 当中，如果一个字段同时被 not null 和 unique 约束的话，
该字段自动变成主键字段。（注意：oracle 中不一样！）

```
insert into t_vip(id,name) values(1,'zhangsan');

insert into t_vip(id,name) values(2,'zhangsan'); //错误了：name 不能重复

insert into t_vip(id) values(2); //错误了：name 不能为 NULL。
```

7.5、主键约束（primary key，简称 PK）非常重要五颗星*****

主键约束的相关术语？

主键约束：就是一种约束。
主键字段：该字段上添加了主键约束，这样的字段叫做：主键字段
主键值：主键字段中的每一个值都叫做：主键值。

什么是主键？有啥用？
主键值是每一行记录的唯一标识。
主键值是每一行记录的身份证号!!!

记住：任何一张表都应该有主键，没有主键，表无效!!

主键的特征：not null + unique（主键值不能是NULL，同时也不能重复!）

怎么给一张表添加主键约束呢？

```
drop table if exists t_vip;
// 1个字段做主键，叫做：单一主键
create table t_vip(
    id int primary key,  //列级约束
    name varchar(255)
);
insert into t_vip(id,name) values(1,'zhangsan');
insert into t_vip(id,name) values(2,'lisi');

//错误：不能重复
insert into t_vip(id,name) values(2,'wangwu');
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'

//错误：不能为NULL
insert into t_vip(name) values('zhaoliu');
ERROR 1364 (HY000): Field 'id' doesn't have a default value
```

可以这样添加主键吗，使用表级约束？

```
drop table if exists t_vip;
create table t_vip(
    id int,
    name varchar(255),
    primary key(id)  // 表级约束
);
insert into t_vip(id,name) values(1,'zhangsan');

//错误
insert into t_vip(id,name) values(1,'lisi');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

表级约束主要是给多个字段联合起来添加约束？

```
drop table if exists t_vip;
// id和name联合起来做主键：复合主键!!!!
create table t_vip(
```

```
    id int,
    name varchar(255),
    email varchar(255),
    primary key(id,name)
);
insert into t_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');
insert into t_vip(id,name,email) values(1,'lisi','lisi@123.com');

//错误：不能重复
insert into t_vip(id,name,email) values(1,'lisi','lisi@123.com');
ERROR 1062 (23000): Duplicate entry '1-lisi' for key 'PRIMARY'
```

在实际开发中不建议使用：复合主键。建议使用单一主键！

因为主键值存在的意义就是这行记录的身份证号，只要意义达到即可，单一主键可以做到。

复合主键比较复杂，不建议使用!!!

一个表中主键约束能加两个吗？

```
drop table if exists t_vip;
create table t_vip(
    id int primary key,
    name varchar(255) primary key
);
ERROR 1068 (42000): Multiple primary key defined
```

结论：一张表，主键约束只能添加 1 个。（主键只能有 1 个。）

主键值建议使用：

int
bigint
char
等类型。

不建议使用：varchar 来做主键。主键值一般都是数字，一般都是定长的！

主键除了：单一主键和复合主键之外，还可以这样进行分类？

自然主键：主键值是一个自然数，和业务没关系。

业务主键：主键值和业务紧密关联，例如拿银行卡账号做主键值。这就是业务主键！

在实际开发中使用业务主键多，还是使用自然主键多一些？

自然主键使用比较多，因为主键只要做到不重复就行，不需要有意义。

业务主键不好，因为主键一旦和业务挂钩，那么当业务发生变动的时候，可能会影响到主键值，所以业务主键不建议使用。尽量使用自然主键。

在 mysql 当中，有一种机制，可以帮助我们自动维护一个主键值？

```
drop table if exists t_vip;
create table t_vip(
    id int primary key auto_increment, //auto_increment 表示自增，从 1 开
```

始，以 1 递增！

```
        name varchar(255)
);
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
insert into t_vip(name) values('zhangsan');
select * from t_vip;

+----+----------+
| id | name     |
+----+----------+
|  1 | zhangsan |
|  2 | zhangsan |
|  3 | zhangsan |
|  4 | zhangsan |
|  5 | zhangsan |
|  6 | zhangsan |
|  7 | zhangsan |
|  8 | zhangsan |
+----+----------+
```

7.6、外键约束（foreign key，简称 FK）非常重要五颗星*****

外键约束涉及到的相关术语：

外键约束：一种约束（foreign key）

外键字段：该字段上添加了外键约束

外键值：外键字段当中的每一个值。

业务背景：

请设计数据库表，来描述“班级和学生”的信息？

第一种方案：班级和学生存储在一张表中？？？

t_student

no(pk)　　name　　classno　　classname

1　　jack　　100　　北京市大兴区亦庄镇第二中学高三 1 班

2　　lucy　　100　　北京市大兴区亦庄镇第二中学高三 1 班

3 lilei 100 北京市大兴区亦庄镇第二中学高三1班
4 hanmeimei 100 北京市大兴区亦庄镇第二中学高三1班
5 zhangsan 101 北京市大兴区亦庄镇第二中学高三2班
6 lisi 101 北京市大兴区亦庄镇第二中学高三2班
7 wangwu 101 北京市大兴区亦庄镇第二中学高三2班
8 zhaoliu 101 北京市大兴区亦庄镇第二中学高三2班

分析以上方案的缺点：

数据冗余，空间浪费!!!!

这个设计是比较失败的!

第二种方案：班级一张表、学生一张表？？

t_class 班级表

classno(pk)	classname
100	北京市大兴区亦庄镇第二中学高三1班
101	北京市大兴区亦庄镇第二中学高三1班

t_student 学生表

no(pk)	name	cno(FK引用t_class这张表的classno)
1	jack	100
2	lucy	100
3	lilei	100
4	hanmeimei	100
5	zhangsan	101
6	lisi	101
7	wangwu	101
8	zhaoliu	101

当cno字段没有任何约束的时候，可能会导致数据无效。可能出现一个102，但是102班级不存在。

所以为了保证cno字段中的值都是100和101，需要给cno字段添加外键约束。

那么：cno字段就是外键字段。cno字段中的每一个值都是外键值。

注意：

t_class是父表

t_student是子表

删除表的顺序？

先删子，再删父。

创建表的顺序？
先创建父，再创建子。

删除数据的顺序？
先删子，再删父。

插入数据的顺序？
先插入父，再插入子。

思考：子表中的外键引用的父表中的某个字段，被引用的这个字段必须是主键吗？
不一定是主键，但至少具有 unique 约束。

测试：外键可以为 NULL 吗？
外键值可以为 NULL。

8、存储引擎（了解内容）

8.1、什么是存储引擎，有什么用呢？
存储引擎是 MySQL 中特有的一个术语，其它数据库中没有。（Oracle 中有，但是不叫这个名字）
存储引擎这个名字高端大气上档次。
实际上存储引擎是一个表存储/组织数据的方式。
不同的存储引擎，表存储数据的方式不同。

8.2、怎么给表添加/指定“存储引擎”呢？
```
show create table t_student;
```

可以在建表的时候给表指定存储引擎。
```
CREATE TABLE `t_student` (
  `no` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `cno` int(11) DEFAULT NULL,
  PRIMARY KEY (`no`),
  KEY `cno` (`cno`),
   CONSTRAINT `t_student_ibfk_1` FOREIGN KEY (`cno`) REFERENCES `t_class` (`classno`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8
```

在建表的时候可以在最后小括号的")"的右边使用：
ENGINE 来指定存储引擎。
CHARSET 来指定这张表的字符编码方式。

结论：
mysql 默认的存储引擎是：InnoDB
mysql 默认的字符编码方式是：utf8

建表时指定存储引擎，以及字符编码方式。

```
create table t_product(
    id int primary key,
    name varchar(255)
)engine=InnoDB default charset=gbk;
```

8.3、怎么查看 mysql 支持哪些存储引擎呢？

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.5.36    |
+-----------+
```

命令： show engines \G

```
*************************** 1. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: CSV
```

```
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 9. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
```

mysql 支持九大存储引擎，当前 5.5.36 支持 8 个。版本不同支持情况不同。

8.4、关于 mysql 常用的存储引擎介绍一下

MyISAM 存储引擎？

它管理的表具有以下特征：

使用三个文件表示每个表：

格式文件 — 存储表结构的定义（mytable.frm）

数据文件 — 存储表行的内容（mytable.MYD）

索引文件 — 存储表上索引（mytable.MYI）：索引是一本书的目录，缩小扫描范围，提高查询效率的一种机制。

可被转换为压缩、只读表来节省空间

提示一下：
对于一张表来说，只要是主键，
或者加有 unique 约束的字段上会自动创建索引。

MyISAM 存储引擎特点：
可被转换为压缩、只读表来节省空间
这是这种存储引擎的优势!!!!

MyISAM 不支持事务机制，安全性低。

InnoDB 存储引擎？
这是 mysql 默认的存储引擎，同时也是一个重量级的存储引擎。
InnoDB 支持事务，支持数据库崩溃后自动恢复机制。
InnoDB 存储引擎最主要的特点是：非常安全。

它管理的表具有下列主要特征：
- 每个 InnoDB 表在数据库目录中以.frm 格式文件表示
- InnoDB 表空间 tablespace 被用于存储表的内容（表空间是一个逻辑名称。表空间存储数据+索引。）

- 提供一组用来记录事务性活动的日志文件
- 用 COMMIT(提交)、SAVEPOINT 及 ROLLBACK(回滚)支持事务处理
- 提供全 ACID 兼容
- 在 MySQL 服务器崩溃后提供自动恢复
- 多版本（MVCC）和行级锁定
- 支持外键及引用的完整性，包括级联删除和更新

InnoDB 最大的特点就是支持事务：
以保证数据的安全。效率不是很高，并且也不能压缩，不能转换为只读，
不能很好的节省存储空间。

MEMORY 存储引擎？
使用 MEMORY 存储引擎的表，其数据存储在内存中，且行的长度固定，
这两个特点使得 MEMORY 存储引擎非常快。

MEMORY 存储引擎管理的表具有下列特征：
- 在数据库目录内，每个表均以.frm 格式的文件表示。
- 表数据及索引被存储在内存中。（目的就是快，查询快!）
- 表级锁机制。
- 不能包含 TEXT 或 BLOB 字段。

MEMORY 存储引擎以前被称为 HEAP 引擎。

MEMORY 引擎优点：查询效率是最高的。不需要和硬盘交互。
MEMORY 引擎缺点：不安全，关机之后数据消失。因为数据和索引都是在内存当中。

9、事务（重点：五颗星*****，必须理解，必须掌握）

9.1、什么是事务？

一个事务其实就是一个完整的业务逻辑。
是一个最小的工作单元。不可再分。

什么是一个完整的业务逻辑？
假设转账，从 A 账户向 B 账户中转账 10000.
将 A 账户的钱减去 10000（update 语句）
将 B 账户的钱加上 10000（update 语句）
这就是一个完整的业务逻辑。

以上的操作是一个最小的工作单元，要么同时成功，要么同时失败，不可再分。
这两个 update 语句要求必须同时成功或者同时失败，这样才能保证钱是正确的。

9.2、只有 DML 语句才会有事务这一说，其它语句和事务无关!!!
insert
delete
update
只有以上的三个语句和事务有关系，其它都没有关系。

因为 只有以上的三个语句是数据库表中数据进行增、删、改的。
只要你的操作一旦涉及到数据的增、删、改，那么就一定要考虑安全问题。

数据安全第一位!!!

9.3、假设所有的业务，只要一条 DML 语句就能完成，还有必要存在事务机制吗？
正是因为做某件事的时候，需要多条 DML 语句共同联合起来才能完成，
所以需要事务的存在。如果任何一件复杂的事儿都能一条 DML 语句搞定，
那么事务则没有存在的价值了。

到底什么是事务呢？
说到底，说到本质上，一个事务其实就是多条 DML 语句同时成功，或者同时失败！

事务：就是批量的 DML 语句同时成功，或者同时失败！

9.4、事务是怎么做到多条 DML 语句同时成功和同时失败的呢？

InnoDB 存储引擎：提供一组用来记录事务性活动的日志文件

事务开启了：
insert
insert
insert
delete
update
update

update
事务结束了!

在事务的执行过程中，每一条 DML 的操作都会记录到“事务性活动的日志文件”中。
在事务的执行过程中，我们可以提交事务，也可以回滚事务。

提交事务？
清空事务性活动的日志文件，将数据全部彻底持久化到数据库表中。
提交事务标志着，事务的结束。并且是一种全部成功的结束。

回滚事务？
将之前所有的 DML 操作全部撤销，并且清空事务性活动的日志文件
回滚事务标志着，事务的结束。并且是一种全部失败的结束。

9.5、怎么提交事务，怎么回滚事务？
提交事务：commit; 语句
回滚事务：rollback; 语句（回滚永远都是只能回滚到上一次的提交点!）

事务对应的英语单词是：transaction

测试一下，在 mysql 当中默认的事务行为是怎样的？
mysql 默认情况下是支持自动提交事务的。（自动提交）
什么是自动提交？
每执行一条 DML 语句，则提交一次!

这种自动提交实际上是不符合我们的开发习惯，因为一个业务
通常是需要多条 DML 语句共同执行才能完成的，为了保证数据
的安全，必须要求同时成功之后再提交，所以不能执行一条
就提交一条。

怎么将 mysql 的自动提交机制关闭掉呢？
先执行这个命令：start transaction;

演示事务：
---------- 回 滚 事 务 ----------

```
mysql> use bjpowernode;
Database changed
mysql> select * from dept_bak;
Empty set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into dept_bak values(10,'abc', 'tj');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into dept_bak values(10,'abc', 'tj');
Query OK, 1 row affected (0.00 sec)

mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | tj   |
|     10 | abc   | tj   |
+--------+-------+------+
2 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from dept_bak;
Empty set (0.00 sec)
```

---------- 提 交 事 务 ----------

```
mysql> use bjpowernode;
Database changed
mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | bj   |
+--------+-------+------+
1 row in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from dept_bak;
```

```
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | bj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
+--------+-------+------+
4 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | bj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
+--------+-------+------+
4 rows in set (0.00 sec)
```

9.6、事务包括 4 个特性？

A：原子性

说明事务是最小的工作单元。不可再分。

C：一致性

所有事务要求，在同一个事务当中，所有操作必须同时成功，或者同时失败，以保证数据的一致性。

I：隔离性

A 事务和 B 事务之间具有一定的隔离。

教室 A 和教室 B 之间有一道墙，这道墙就是隔离性。

A 事务在操作一张表的时候，另一个事务 B 也操作这张表会那样？？？

D：持久性

事务最终结束的一个保障。事务提交，就相当于将没有保存到硬盘上的数据保存到硬盘上！

9.7、重点研究一下事务的隔离性!!!

A 教室和 B 教室中间有一道墙，这道墙可以很厚，也可以很薄。这就是事务的隔离级别。这道墙越厚，表示隔离级别就越高。

事务和事务之间的隔离级别有哪些呢？4 个级别

读未提交：read uncommitted（最低的隔离级别）《没有提交就读到了》

什么是读未提交？

事务 A 可以读取到事务 B 未提交的数据。

这种隔离级别存在的问题就是：

脏读现象！(Dirty Read)

我们称读到了脏数据。

这种隔离级别一般都是理论上的，大多数的数据库隔离级别都是二档起步！

读已提交：read committed《提交之后才能读到》

什么是读已提交？

事务 A 只能读取到事务 B 提交之后的数据。

这种隔离级别解决了什么问题？

解决了脏读的现象。

这种隔离级别存在什么问题？

不可重复读取数据。

什么是不可重复读取数据呢？

在事务开启之后，第一次读到的数据是 3 条，当前事务还没有结束，可能第二次再读取的时候，读到的数据是 4 条，3 不等于 4 称为不可重复读取。

这种隔离级别是比较真实的数据，每一次读到的数据是绝对的真实。

oracle 数据库默认的隔离级别是：read committed

可重复读：repeatable read《提交之后也读不到，永远读取的都是刚开启事务时的数据》

什么是可重复读取？

事务 A 开启之后，不管是多久，每一次在事务 A 中读取到的数据都是一致的。即使事务 B 将数据已经修改，并且提交了，事务 A 读取到的数据还是没有发生改变，这就是可重复读。

可重复读解决了什么问题？

解决了不可重复读取数据。

可重复读存在的问题是什么？

可以会出现幻影读。

每一次读取到的数据都是幻象。不够真实！

早晨 9 点开始开启了事务，只要事务不结束，到晚上 9 点，读到的数据还是那样！

读到的是假象。不够绝对的真实。

mysql 中默认的事务隔离级别就是这个!!!!!!!!!!

序列化/串行化：serializable（最高的隔离级别）

这是最高隔离级别，效率最低。解决了所有的问题。

这种隔离级别表示事务排队，不能并发！

synchronized，线程同步（事务同步）

每一次读取到的数据都是最真实的，并且效率是最低的。

9.8、验证各种隔离级别

查看隔离级别：SELECT @@tx_isolation

```
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+
```

mysql 默认的隔离级别

被测试的表 t_user

验证：read uncommited

```
mysql> set global transaction isolation level read uncommitted;
```

事务 A	事务 B
use bjpowernode;	
	use bjpowernode;
start transaction;	
select * from t_user;	
	start transaction;
	insert into t_user values('zhangsan');
select * from t_user;	

验证：read commited

```
mysql> set global transaction isolation level read committed;
```

事务 A	事务 B
use bjpowernode;	
	use bjpowernode;
start transaction;	
	start transaction;
select * from t_user;	
	insert into t_user values('zhangsan');
select * from t_user;	
	commit;
select * from t_user;	

验证：repeatable read

```
mysql> set global transaction isolation level repeatable read;
```

事务 A	事务 B
use bjpowernode;	
	use bjpowernode;
start transaction;	
	start transaction;
select * from t_user;	
	insert into t_user values('lisi');
	insert into t_user values('wangwu');
	commit;
select * from t_user;	

验证：serializable

```
mysql> set global transaction isolation level serializable;
```

事务 A	事务 B
use bjpowernode;	
	use bjpowernode;
start transaction;	
	start transaction;
select * from t_user;	
insert into t_user values('abc');	
	select * from t_user;