

# MySQL-Day02必须掌握

## 外键

### 原理

让当前表字段的值在另一张表的范围内去选择

### 使用规则

- 1、数据类型要一致
- 2、主表被参考字段必须为KEY的一种 : PRI

### 级联动作

- 1、`cascade` : 删除 更新同步(被参考字段)
- 2、`restrict`(默认) : 不让主表删除 更新
- 3、`set null` : 删除 更新, 从表该字段值设置为NULL

## 嵌套查询（子查询）

### 定义

把内层的查询结果作为外层查询的条件

## 多表查询

### 笛卡尔积

多表查询不加where条件, 一张表的每条记录分别和另一张表的所有记录分别匹配一遍

## 连接查询

### 分类

- 1、内连接 (表1 `inner join` 表2 `on` 条件)
- 2、外连接 (表1 `left|right join` 表2 `on` 条件)
  - 1、左连接 : 以左表为主显示查询结果
  - 2、右连接 : 以右表为主显示查询结果

### 语法

```
select 表名.字段名 from 表1 inner join 表2 on 条件;
```

## 锁

1、目的：解决客户端并发访问的冲突问题

## 2、锁分类

- 1、锁类型：读锁 写锁
- 2、锁粒度：行级锁(InnoDB) 表级锁(MyISAM)

## 数据导入

### 方式一（使用source命令）

```
mysql> source /home/tarena/xxx.sql
```

### 方式二（使用load命令）

- 1、将导入文件拷贝到数据库搜索路径中  
`show variables like 'secure%';`
- 2、在数据库中创建对应的表
- 3、执行数据导入语句

## 索引

### 定义

对数据库表中一列或多列的值进行排序的一种结构(BTree)

### 优点

加快数据的检索速度

### 缺点

- 1、占用实际物理存储空间
- 2、索引需动态维护，消耗资源，降低数据的维护速度

### 分类及约束

- 1、普通索引（MUL）：无约束
- 2、唯一索引（UNI）：字段值不允许重复，但可为NULL
- 3、主键（PRI）：字段值不允许重复，不可为NULL
- 4、外键：让当前表字段的值在另一张表的范围内选择

## MySQL-Day03笔记

## 存储引擎

### 定义

处理表的处理器

## 基本操作

- 1、查看所有存储引擎  
`mysql> show engines;`
- 2、查看已有表的存储引擎  
`mysql> show create table 表名;`
- 3、创建表指定  
`create table 表名(... )engine=MyISAM,charset=utf8,auto_increment=10000;`
- 4、已有表指定  
`alter table 表名 engine=InnoDB;`

## ==常用存储引擎及特点==

- InnoDB

- 1、支持行级锁
- 2、支持外键、事务、事务回滚
- 3、表字段和索引同存储在一个文件中
  - 1、表名.frm : 表结构
  - 2、表名.ibd : 表记录及索引文件

- MyISAM

- 1、支持表级锁
- 2、表字段和索引分开存储
  - 1、表名.frm : 表结构
  - 2、表名.MYI : 索引文件(my index)
  - 3、表名.MYD : 表记录(my data)

- MEMORY

- 1、表记录存储在内存中，效率高
- 2、服务或主机重启，表记录清除

## 如何选择存储引擎

- 1、执行查操作多的表用 MyISAM(使用InnoDB浪费资源)
- 2、执行写操作多的表用 InnoDB
- 3、临时表 : MEMORY

# MySQL的用户账户管理

## 开启MySQL远程连接

更改配置文件，重启服务！

- 1、sudo su
- 2、cd /etc/mysql/mysql.conf.d
- 3、cp mysqld.cnf mysqld.cnf.bak
- 4、vi mysqld.cnf #找到44行左右,加 # 注释  
#bind-address = 127.0.0.1  
[mysqld]  
character\_set\_server = utf8
- 5、保存退出
- 6、service mysql restart

vi使用：按a ->编辑文件 ->ESC ->shift+: ->wq

## 添加授权用户

- 1、用root用户登录mysql  
mysql -uroot -p123456
- 2、授权  
grant 权限列表 on 库.表 to "用户名"@"%" identified by "密码" with grant option;
- 3、刷新权限  
flush privileges;

## 权限列表

all privileges 、select 、insert ... ..  
库.表： \*.\* 代表所有库的所有表

## 示例

- 1、添加授权用户work,密码123,对所有库的所有表有所有权限  
mysql>grant all privileges on \*.\* to 'work'@"%" identified by '123' with grant option;  
mysql>flush privileges;
- 2、添加用户duty,对db2库中所有表有所有权限

# ==事务和事务回滚==

## 事务定义

一件事从开始发生到结束的过程

## 作用

确保数据的一致性、准确性、有效性

## 事务操作

- 1、开启事务  
mysql>**begin**; # 方法1  
mysql>**start transaction**; # 方法2
- 2、开始执行事务中的1条或者n条SQL命令
- 3、终止事务  
mysql>**commit**; # 事务中SQL命令都执行成功, 提交到数据库, 结束!  
mysql>**rollback**; # 有SQL命令执行失败, 回滚到初始状态, 结束!

## ==事务四大特性 (ACID) ==

- 1、原子性 (atomicity)

一个事务必须视为一个不可分割的最小工作单元, 整个事务中的所有操作要么全部提交成功, 要么全部失败回滚, 对于一个事务来说, 不可能只执行其中的一部分操作

- 2、一致性 (consistency)

数据库总是从一个一致性的状态转换到另一个一致性的状态

- 3、隔离性 (isolation)

一个事务所做的修改在最终提交以前, 对其他事务是不可见的

- 4、持久性 (durability)

一旦事务提交, 则其所做的修改就会永久保存到数据库中。此时即使系统崩溃, 修改的数据也不会丢失

## 注意

- 1、事务只针对于表记录操作 (增删改) 有效, 对于库和表的操作无效
- 2、事务一旦提交结束, 对数据库中数据的更改是永久性的

# E-R模型(Entry-Relationship)

## 定义

E-R模型即 实体-关系 数据模型, 用于数据库设计  
用简单的图(E-R图)反映了现实世界中存在的事物或数据以及他们之间的关系

## 实体、属性、关系

- 实体

- 1、描述客观事物的概念
- 2、表示方法 : 矩形框
- 3、示例 : 一个人、一本书、一杯咖啡、一个学生

- 属性

- 1、实体具有的某种特性
- 2、表示方法：椭圆形
- 3、示例  
学生属性：学号、姓名、年龄、性别、专业 ...  
感受属性：悲伤、喜悦、刺激、愤怒 ...

- ==关系（重要）==

- 1、实体之间的联系
- 2、一对一关联(1:1)：老公对老婆  
A中的一个实体,B中只能有一个实体与其发生关联  
B中的一个实体,A中只能有一个实体与其发生关联
- 3、一对多关联(1:n)：父亲对孩子  
A中的一个实体,B中有多个实体与其发生关联  
B中的一个实体,A中只能有一个与其发生关联
- 4、多对多关联(m:n)：兄弟姐妹对兄弟姐妹、学生对课程  
A中的一个实体,B中有多个实体与其发生关联  
B中的一个实体,A中有多个实体与其发生关联

## ER图的绘制

矩形框代表实体,菱形框代表关系,椭圆形代表属性

- 课堂示例（老师研究课题）

- 1、实体：教师、课题
- 2、属性  
教师：教师代码、姓名、职称  
课题：课题号、课题名
- 3、关系  
多对多 (m:n)  
# 一个老师可以选择多个课题，一个课题也可以被多个老师选

- 练习

设计一个学生选课系统的E-R图

- 1、实体：学生、课程、老师
- 2、属性
- 3、关系  
学生 选择 课程 (m:n)  
课程 任课 老师 (1:n)

==关系映射实现（重要）==

- ```
1:1实现 --> 主外键关联, 外键字段添加唯一索引
表t1 : id int primary key,
      1
表t2 : t2_id int unique,
      foreign key(t2_id) references t1(id)
      1
1:n实现 --> 主外键关联
表t1 : id int primary key,
      1
表t2 : t2_id int,
```

```
foreign key(t2_id) references t1(id)
1
1
m:n实现(借助中间表):
t1 : t1_id
t2 : t2_id
```

## ==多对多实现==

- 老师研究课题

表1、老师表

表2、课题表

问题？如何实现老师和课程之间的多对多映射关系？

中间表：

- 后续

1、每个老师都在研究什么课题？

2、郭小闹在研究什么课题？

## ==MySQL调优==

### 存储引擎优化

- 1、读操作多：MyISAM
- 2、写操作多：InnoDB

### 索引优化

在 select、where、order by 常涉及到的字段建立索引

### SQL语句优化

- 1、单条查询最后添加 `LIMIT 1`，停止全表扫描
- 2、where子句中不使用 `!=`，否则放弃索引全表扫描
- 3、尽量避免 `NULL` 值判断，否则放弃索引全表扫描  
优化前：`select number from t1 where number is null;`  
优化后：`select number from t1 where number=0;`  
# 在number列上设置默认值0,确保number列无NULL值
- 4、尽量避免 `or` 连接条件，否则放弃索引全表扫描  
优化前：`select id from t1 where id=10 or id=20;`  
优化后：`select id from t1 where id=10 union all  
select id from t1 where id=20;`
- 5、模糊查询尽量避免使用前置 `%`，否则全表扫描  
`select name from t1 where name like "c%";`
- 6、尽量避免使用 `in` 和 `not in`，否则全表扫描  
优化前：`select id from t1 where id in(1,2,3,4);`  
优化后：`select id from t1 where id between 1 and 4;`
- 7、尽量避免使用 `select * ...`；用具体字段代替 `*`，不要返回用不到的任何字段

## 作业讲解

有一张文章评论表comment如下

| comment_id | article_id | user_id | date                |
|------------|------------|---------|---------------------|
| 1          | 10000      | 10000   | 2018-01-30 09:00:00 |
| 2          | 10001      | 10001   | ... ..              |
| 3          | 10002      | 10000   | ... ..              |
| 4          | 10003      | 10015   | ... ..              |
| 5          | 10004      | 10006   | ... ..              |
| 6          | 10025      | 10006   | ... ..              |
| 7          | 10009      | 10000   | ... ..              |

以上是一个应用的comment表格的一部分，请使用SQL语句找出在本站发表的所有评论数量最多的10位用户及评论数，并按评论数从高到低排序

备注：comment\_id为评论id

article\_id为被评论文章的id

user\_id 指用户id

2、把 /etc/passwd 文件的内容导入到数据库的表中

## 3、外键及查询题目

综述：两张表，一张顾客信息表customers，一张订单表orders

表1：顾客信息表，完成后插入3条表记录

c\_id 类型为整型，设置为主键，并设置为自增长属性  
c\_name 字符类型，变长，宽度为20  
c\_age 微小整型，取值范围为0~255(无符号)  
c\_sex 枚举类型，要求只能在('M','F')中选择一个值  
c\_city 字符类型，变长，宽度为20  
c\_salary 浮点类型，要求整数部分最大为10位，小数部分为2位



```

create table customers(
  c_id int primary key auto_increment,
  c_name varchar(20),
  c_age tinyint unsigned,
  c_sex enum('M', 'F'),
  c_city varchar(20),
  c_salary decimal(12,2)
)charset=utf8;
insert into customers values(1, 'Tom', 25, 'M', '上海', 10000), (2, 'Lucy', 23, 'F', '广州', 12000), (3, 'Jim', 22, 'M', '北京', 11000);

```

表2: 顾客订单表 (在表中插入5条记录)

```

o_id 整型
o_name 字符类型, 变长, 宽度为30
o_price 浮点类型, 整数最大为10位, 小数部分为2位
设置此表中的o_id字段为customers表中c_id字段的外键, 更新删除同步
insert into orders values(1, "iphone", 5288), (1, "ipad", 3299), (3, "mate9", 3688),
(2, "iwatch", 2222), (2, "r11", 4400);

```

```

create table orders(
  o_id int,
  o_name varchar(30),
  o_price decimal(12,2),
  foreign key(o_id) references customers(c_id) on delete cascade on update cascade
)charset=utf8;
insert into orders values(1, "iphone", 5288), (1, "ipad", 3299), (2, "iwatch", 2222),
(2, "r11", 4400);

```

增删改查题

- 1、返回customers表中, 工资大于4000元, 或者年龄小于29岁, 满足这样条件的前2条记录
- 2、把customers表中, 年龄大于等于25岁, 并且地址是北京或者上海, 这样的人的工资上调15%
- 3、把customers表中, 城市为北京的顾客, 按照工资降序排列, 并且只返回结果中的第一条记录
- 4、选择工资c\_salary最少的顾客的信息
- 5、找到工资大于5000的顾客都买过哪些产品的记录明细
- 6、删除外键限制
- 7、删除customers主键限制
- 8、增加customers主键限制c\_id