### SQL语句

# insert playerprop(userid,help,behelp) values (1,1,1); 插入数据

# alter table 原表名 rename to 新表名

# alter table 表名 change 要修改的字段名 新字段名 新字段的数据类型

# alter table 表名 modify 要修改的字段名 新字段名 新字段的数据类型

# alter table playerprop character set utf8mb4 COLLATE utf8mb4\_general\_ci; 修改表的编码格式

# select distinct name数据去重

# offset：偏移量，和limit一起用做到分页的效果。select \* from test1 limit 5 offset 1;

# union：俩表数据去重。union all输出所有。select \* from test1 union select \* from test2;

# group by：分组查询和去重查询（distinct）。可以跟（with rollup 总统计），SELECT coalesce(name, '总金额'),name, SUM(money) as money FROM test GROUP BY name WITH ROLLUP;

# SELECT @@tx\_isolation;查看mysql事务隔离级别。

# SELECT count(TABLE\_NAME) FROM information\_schema.TABLES WHERE TABLE\_SCHEMA='dbname';查看某个数据库有多少表

# insert时如果数据重复如何用update

insert into user values(0,"001"),(1,"112") on duplicate key update name=values(name);

# Explain查看sql语句执行

# 字符串拼接

1. concat(a,b,c...) 如果a=null则拼接结果为null。
2. concat(split,a,b,c...) 如果a为null跳过，如果aplit=null则拼接结果为null。

### 知识点

# 关系型数据库的优势：

保持数据的一致性（事务处理）

由于以标准化为前提，数据更新的开销很小（相同的字段基本上都只有一处）

可以进行Join等复杂查询

其中能够保持数据的一致性是关系型数据库的最大优势。

# 关系型数据库的不足：

1. 不擅长的处理大量数据的写入处理
2. 为有数据更新的表做索引或表结构（schema）变更
3. 字段不固定时应用
4. 对简单查询需要快速返回结果的处理。

# 索引有什么作用

1. 对数据库表中一列或多列的值进行排序的一种结构
2. 第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。
3. 第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。
4. 第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。
5. 第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。
6. 第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

# MyISAM数据库引擎。

1. 只支持表级锁，在操作表的时候会自动给表加锁。也可以通过lock table的命令锁表，模拟事务，但是消耗非常大。

# 事务的四个隔离级别。

1. 脏读：事务未提交但是在修改前后造成其他事务的脏读。
2. 不可重复读：发生修改的事务提交前后造成其他事务的脏读。
3. 幻读：增加了表的数据，试一个事务读取前后有误差

select @@tx\_isolation,@@global.tx\_isolation;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| isolation level | 数据一致性 | 脏读 | 不可重复读 | 幻读 | 特点 |
| read uncommitted | 不读物理损坏的数据 | 是 | 是 | 是 | 不能解决任何问题 |
| read committed | 语句级 | 否 | 是 | 是 | oracle默认 |
| repeatable read | 事务级 | 否 | 否 | 是 | mysql默认 |
| serializable | 事务级 | 否 | 否 | 否 | 锁表 |

# 事务的七个传输方式

1. PROPAGATION\_REQUIRED（required）如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中。
2. PROPAGATION\_SUPPORTS (supports)如果有一个事务的话,它就在这个事务中运行。反之则否。
3. PROPAGATION\_REQUIRES\_NEW(requires\_new)总是开启一个新的事务。如果一个事务已经存在，则将这个存在的事务挂起。
4. PROPAGATION\_NOT\_SUPPORTED(not\_supported)总是非事务地执行，并挂起任何存在的事务。
5. PROPAGATION\_MANDATORY(mandatory)表示当前方法必须在一个事务中运行，如果没有事务，将抛出异常
6. PROPAGATION\_NEVER(never)总是非事务地执行，如果存在一个活动事务，则抛出异常
7. PROPAGATION\_NESTED(nested)表示如果当前方法正有一个事务在运行中,则该方法应该运行在一个嵌套事务中 ,被嵌套的事务可以独立于被封装的事务中进行提交或者回滚。如果封装事务存在,并且外层事务抛出异常回滚，那么内层事务必须回滚,反之,内层事务并不影响外层事务。
8. readOnly：事务期间无法看到其他事务已经提交的数据。

# 事务的ACID属性：

1. 原子性（Atomicity）原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。
2. 一致性（Consistency）：事务必须使数据库从一个一致性状态变换到另外一个一致性状态。
3. 隔离性（Isolation）：事务的隔离性是指一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。
4. 持久性（Durability）：持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来的其他操作和数据库故障不应该对其有任何影响。

# Alert,modify,change

1. Alert:删除列等。
2. Modify:细微操作：改类型等（不能重命名外和change一样）。
3. Change:大操作。

# 变量

1. 局部变量:作用域(只在begin/end代码块生效),赋值(declare name type default value//set name=value//select的查询结果into给变量).
2. 用户变量:作用域(服务重启,变量失效),赋值(),使用(),例子().
3. 会话变量:作用域(连接退出,变量失效),赋值(set @name=value//set @@global.name=value),使用(@name),例子(SELECT (@name:=@name+1//”:”在select时候必须加上),id FROM TABLE).
4. 全局变量:作用域(服务重启,变量初始化,可设置)),赋值(set global name=value),使用(),例子().

# 主键KEY和索引INDEX

1. 主键和索引都是键，主键是逻辑键，索引是物理键。主键只是避免一张表里有相同的数据，索引则是真实存在数据库的用于快速查找。但是主键只能有一个且非空，索引不限制。
2. primary key 主键，unique key唯一键，foreign key外键。mysql建key也包含了index的意义；
3. Index的创建会在innodb表空间以一个类似目录的结构存储。
4. 在理论上是不能将MySQL的key和index划等号的，他们不是一回事，但在实际使用中，他们基本没有区别

# 主键

|  |  |
| --- | --- |
| 名字 | 使用 |
| primary key | 建表：column\_name column\_type primary key  primary key(column\_name)  增加：alter table table\_name add primary key(column\_name)  删除：alter table table\_name drop primary key |
| unique key | 建表：column\_name column\_type unique //(默认别名为column\_name)  unique key name(column\_name,...)  增加：alter table table\_name add unique key name(column\_name,...)  删除：alter table table\_name drop index name |
| foreign key | 建表：foreign key (name1) references table2(name2)  增加：alter table table\_name add constraint name foreign key(column\_name) references table\_name2(column\_name2);  删除：alter table table\_name drop foreign key name; |

# 索引

(非)聚集索引，索引顺序和物理顺序是否一致。

|  |  |
| --- | --- |
| 名字 | 使用 |
| 普通索引 | 添加：alter table table\_name add index/key name(...); |
| 唯一索引 | 添加：alter table table\_name add unique index/key name(...); |
| 主键索引 | 即主键操作 |
| 外键索引 | 即外键操作 |
| 全文索引 | 添加：alter table table\_name add fulltext index/key name(...);使用： where match(column\_name) against (“\*” in boolean mode);//in boolean mode用于使用通配符 |
| 哈希索引 | 添加：create index name using hash on table\_name(column\_name); |
| 组合索引 | 以上所有key为多个就是组合索引 |

1. 普通索引：最基本的索引，没有任何限制
2. 唯一索引：必须唯一允许有空值
3. 主键索引：唯一且不为空
4. 全文索引：5.6版本之前仅用于MyISAM,类似like但是效率更高where match(column) against('xx')，
5. 组合索引：最左优先规则
6. 哈希索引：key name(column) using hash

# 优先级

1. and优先级高于or
2. where>group by>having>order by

# in和exists

1. in:in()语句只会执行一次，它查出in()所有的数据并且缓存起来,在内存里进行双层循环判断..(内数据少可用)
2. exists:执行总次数为外层数据量,等价于,内外层拆为一条语句执行.(外数据少可用)

# For update

1. 是一种行级锁，其他用户不能执行更新操作，也可以通过加锁的方式共享更新
2. 释放锁的条件：commit、退出数据库、程序停止运行

# 自增长id

# mysql缓存

1. 查看缓存机制设置：SHOW VARIABLES LIKE '%query\_cache%';

have\_query\_cache:

query\_cache\_limit:缓存的最大内存，如果超出，则增加 Qcache\_not\_cached的值，并删除查询结果

query\_cache\_min\_res\_unit:分配内存块时的的最小单位大小。

query\_cache\_size:缓存使用内存大小，单位字节，必须是1024倍否则不准确，太小甚至0无法使用

query\_cache\_type:是否打开缓存，0/OFF关闭，1/ON总是打开（SQL\_NO\_CACHE关闭），2/DEMAND按需使用缓存（SQL\_CACHE打开）

query\_cache\_wlock\_invalidate:表被锁住是否返回缓存数据。

### 常用命令

1. 命令事务

开启：begin/start transaction，回滚rollback，提交commit。

### MVCC

# 概念

1. 多版本控制（Multiversion Concurrency Control）: 只有写写之间相互阻塞。在内部实现中，InnoDB通过undo log保存每条数据的多个版本，并且能够找回数据历史版本提供给用户读，每个事务读到的数据版本可能是不一样的。在同一个事务中，用户只能看到该事务创建快照之前已经提交的修改和该事务本身做的修改。

# 隐藏字段