# MySQL高级

## 一、基本概念

**DB**(DataBase)：数据库，存储数据的容器。

**DBMS**(DataBase Management System)：数据库管理系统，又称为数据库软件或数据库产品，用于创建或管理DB。常见的数据库管理系统：MySqL、Oracle、DB2等。

**SQL**：结构化查询语言，用于和数据库通信的语言，不是某个数据库软件特有的，而是几乎所有的主流数据库软件通用的语言。

**数据库存储数据特点**

①数据存放到表中，然后表再放到数据库中。

②每个库中可以有多张表，每张表具有唯一的表名用来标识自己。

③表中有一个或多个列，列又称为“字段”，相当于java中“属性”。

④表中的每一行数据，相当于java中“对象”。

**MySQL优点**：

①开源、免费、成本低；②体积小、便于安装；③性能高；

## 二、其他

MySQL中的my.ini文件是MySQL在Windows下的配置文件，my.conf文件是MySQL在Linux下的配置文件。

### 1.查询补充

**查询常量**：select 常量值; 注意：字符型和日期型的常量值必须用单引号引起来，数值型不需要。

**查询函数**：select 函数名(实参列表);

**查询表达式**：如select 100/134;

**select中的加号**：

①select 数值+数值;直接运算。

**②**select 字符+数值;先试图将字符转换成数值，如果转换成功，则继续运算；否则转换成0，再做运算。

③select null+值;结果都为null。

select concat(字符1，字符2….);拼接字符串。

ifnull()函数：判断某字段或表达式是否为null，如果为null 返回指定的值，否则返回原本的值。如ifnull(empno,1)如果empno为null的话则返回1，否则返回原本的值。

is null：用于判断null值。

分组函数：max、min、avg、sum、count都可以搭配distinct使用。如select sum(distinct 字段) from 表名。

count(字段)：统计表中该字段非空的个数。

count(\*)和count(1)：会统计表中的所有的记录数，包含字段为null 的记录。

在InnoDB存储引擎中，count(\*)和count(1)的执行效率要高于count(字段)。在MyISAM引擎中，count(\*)的执行效率最高。

注意在连接查询时，n个表进行连接查询必须要包含n-1个连接条件。

**truncate和delete删除数据的区别**：

DDL语句是直接提交的不能回滚。DML语句是可以回滚的，不是自动提交的。

truncate是DDL语句，truncate是清空表格数据，不能加条件即不能跟着where。一旦执行了truncate不可以回滚。执行起来比delete快。

delete是DML语句。delete是删除表格中指定的内容的数据，后面可跟where条件，执行之后可以回滚。

drop是将整个表都删除，包括删除表结构。

**MySQL中常见的约束**：

Primary key：主键约束，该字段的值不可重复且不能为空。

not null：非空约束，该字段的值不能为null，必填。

unique：唯一约束，该字段的值不可重复。

default：默认约束，该字段不插入数据时，它的值有默认值。

foreign key：外键约束，该字段的值引用了另一个表中的主键值。

**主键约束和唯一约束的区别**：

相同点：都具有唯一性。

不同点：一个表只能有一个主键，但是可以有多个唯一约束。主键的值不能为null，但是唯一约束可以为null，且只能有一个。

### 2.事务

**事务**：事物是作为单个逻辑工作单元执行的一系列操作，这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行。

**事务的四个属性**：

①**原子性**（Atomicity）：事务是一个完整的操作，不能对它进行再分割。

②**一致性**（Consistency）：当事务完成时，数据必须处于一致状态。

③**隔离性**（Isolation）：对数据进行修改的所有并发事务是彼此隔离的。

④**永久性（**Durability）：事务完成后，它对数据库的修改被永久保持。

**隐式事务**：没有明显的开启和结束，本身就是一条事务可以自动提交，比如insert、update、delete。

**显示事务**：有明显的开启和结束事务。要设置setAutoCommit，commit、rolleback。

**并发事务**：多个事务同时操作同一个数据库的相同数据时。

**并发产生的问题**：

**脏读**：一个事务A读取了另一个B事务修改的数据，但是B修改的数据还没提交到数据库中。解决脏读办法，设置数据库隔离级别为Read Commited。

**不可重复读**：比如事务A读取了两次数据，然后在这两次的读取过程中事务B修改了数据，因此事务A两次读取到的数据结果不一致。解决办法，将隔离级别设置为Repeatable read。

**幻读**： 比如事务A读取了两次数据，在这两次的读取过程中，事务B新增了行数据，因此事务A这两次读取的数据集合不一致。解决办法，将事务隔离级别设置为Serializable。

### 3.四种隔离级别：

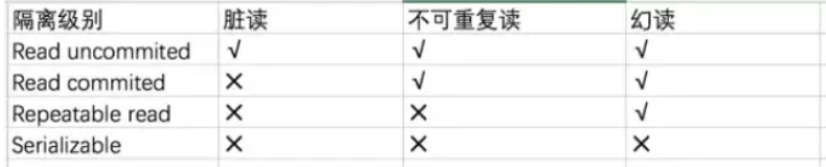
**Read Uncommited**：读未提交的内容。就读取未提交的数据，也被称之为脏读(Dirty Read)。这种隔离级别**可能产生脏读、不可重复读、幻读问题**。

**Read Commited**：读取已经提交的。大多数数据库系统的默认隔离级别(但不是MySQL默认的)。即一个事务只能看已经提交的数据。这种隔离级别**可能产生不可重复读、幻读问题**。

**Repeatable Read**：可重复读。这是**MySQL默认的隔离级别**。这种**可能产生幻读**问题。

**Serializable**：可串行化。这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之，它是在每个读的数据行上加上共享锁。在这个级别，可能导致大量的超时现象和锁竞争。这种不会产生三种问题。

在MySQL中实现了这四种隔离级别，可能产生如下问题：



**四种隔离级别由高到低**：**Serializable、Repeatable read、read commited、read uncommited**。

### 4.日志记录

**错误日志**：log-error，记录严重的警告和错误信息，每次启动和关闭的详细信息等。MySQL5.7默认是开启的。

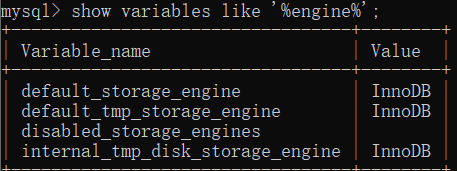
**查询日志**：默认是关闭的，用来记录查询的sql语句，若开启则会降低mysql的整体性能。

**数据文件**：在window中，数据文件存放在MySQL的data文件夹下。frm文件存放的是表结构，myd文件存放表的数据，myi文件，存放表的索引。

### 5.存储引擎

查看存储引擎命令：show engines;

查看当前数据库的存储引擎：show variables like ‘%engine%’;



**常用存储引擎**：

**InnoDB**：MySQL默认的存储引擎；支持事务控制；有外键；使用的是行锁，操作时只锁一行，不对其他行有影响，适合高并发的操作；不仅缓存索引还缓存真实数据，对内存要求较高。这个更**关注事务**。

**MyISAM**：不支持外键；不支持事务；表锁，操作一条记录也会将整个表锁主，不适合高并发操作；缓存时，只缓存索引，不缓存真实数据。这个**更关注性能**。

**MySQL的架构，由上到下**：连接层🡪服务层🡪引擎层🡪存储层。

### 6.索引优化

性能下降，查询时执行时间长，等待时间长，原因可能是查询语句写的烂、索引失效、关联查询使用太多的表、服务器中参数没有设置好。

**索引**：**索引是一种数据结构，可理解为是排好序的快速查找数据结构**。相当于表的目录，可以通过索引快速找出在某个列中有一特定值的行。

不使用索引，MySQL必须从第一条记录开始读完整个表，直到找出相关的数据，表越大，查询数据所花费的时间就越多。

**使用索引的优缺点**：

优点：可提高对表中数据的查询速度；通过索引列对数据进行排序，降低数据排序的成本。

缺点：创建和维护索引需要耗费时间；会降低表的增删改效率，因为每次增删改，索引需要进行动态维护；索引也需要占空间。

**使用原则**：

**①**对经常作为查询条件的字段应该创建索引，对经常增删改的表应该避免创建过多的索引。

**②**对数据量小的表最好不用使用索引，因为数据量小可能查询全部数据花费的时间比遍历索引所需的时间短，因此索引将起不到优化作用。

**③**一个字段的值不同的较少时，不要使用索引，比如sex的值为男和女两种。一个字段的值不同的较多可以使用索引。

**④**where条件里用不到的字段不创建索引。

**⑤**查询中排序的字段可以创建索引，排序字段若通过索引访问将大大提高排序速度。

**索引分类**：**单列索引、组合索引、全文索引**等。

MySQL的**索引结构**主要有：Btree索引、Hash索引、full-text全文索引、R-Tree索引。

**单列索引分为：普通索引、唯一索引、主键索引**。

**①普通索引**：允许在定义索引的列中插入重复值和空值。一般在出现查询条件或者排序条件中使用。

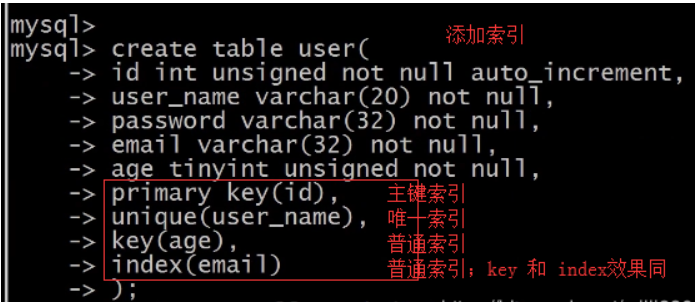
**直接创建**：**create index 索引名 on 表名(字段名);**

例子：create index idx\_user\_name on user(name);

**修改表结构创建**：**alter table 表名 add index 索引名 on (字段名)**；

例子：alter table user add index idx\_user\_name on(name);

**创建表的时候创建索引**：



**删除索引**：**drop index 索引名 on 表名**；或者**alter table 表名 drop index 索引名**;

例子：drop index idx\_user\_name on user;

**查看索引**：**show index from 表名**;

例子：show index from user;

**②唯一索引**：索引列的值必须唯一，允许有空值，如果是组合索引，列值的组合必须唯一。

**直接创建**：**create unique index 索引名 on 表名(字段名);**

例子：create unique index idx\_user\_name on user(name)；

修改表结构创建唯一索引和创建表时创建唯一索引的方式和创建普通索引的类似，只不过添加了一个unique。

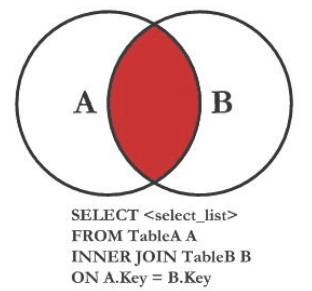
**③主键索引**：主键索引是一种特殊的唯一索引，一个表只能有一个主键，不允许有空值。一般是在创建表的时候指定主键，主键默认就是主键索引。

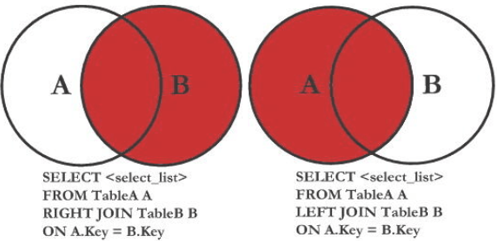
**组合索引**：使用多个字段创建的索引，只有在查询条件中使用了创建索引时的第一个字段，索引才会被使用。

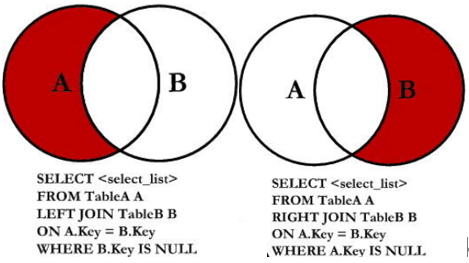
例子：create index ind\_user\_name on user(name,age,num)；

**全文索引**：索引类型为fullText，一般在CHAR、VARCHAR、TEXT类型字段上使用全文索引。全文索引的目的是为了解决模糊查询效率低下的问题出现的。在全文搜索中，LIKE查询的数据如果量比较大，效率低下。

### 7.七种Joins

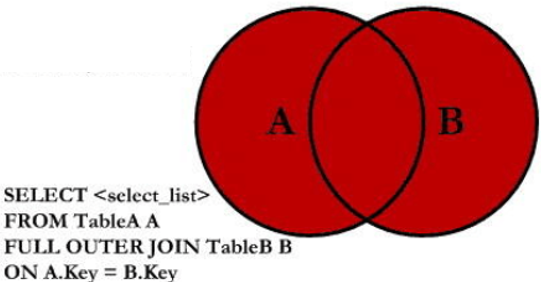






**注意下面两个在MySQL中是不支持的**。

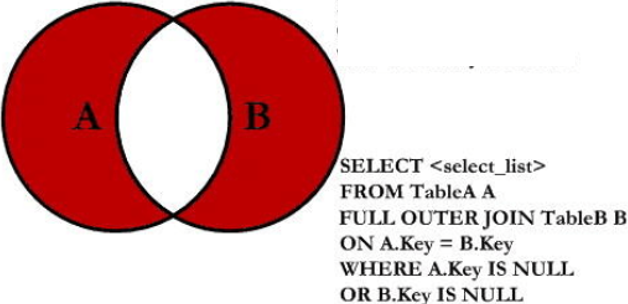
**Outer Join**



在MySQL中，可以使用如下SQL获得Outer Join的结果：

select \* from A left join B on A.key =B.key **union** select \* from A right join B on A.key=B.key;

**Outer Excluding Join**

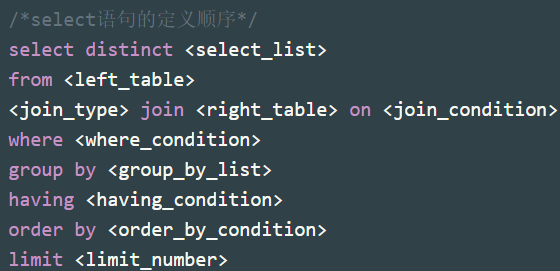


在MySQL中，可以使用如下SQL获得Outer Join的结果：

select \* from A left join B on A.key =B.key where B.key is null **union** select \* from A right join B on A.key=B.key where A.key is null;

### 8.select语句的定义和执行顺序

**select语句的定义顺序：**



**select语句的执行顺序：**



**即执行select语句是先从from开始的。**

### 9.distinct、union、union all区别

distinct可以去除重复的数据，一般用在查询字段的前面，比如select distinct name….。不建议使用，因为效率低。

union 和union all可以用来合并两个或多个select语句的结果集， 这些select语句查询的结果集必须列的数据类型一致、顺序和数量一致。其中，union合并之后可以去除重复的数据，同时也会根据字段的顺序进行排序。union all则不会去除重复数据，而是直接将查询结果连接起来。 在执行效率上，union all要比union更快。

**单表上有重复记录，去重只能使用distinct。合并多个查询的结果集，若不考虑去除重复数据则使用union all，若要考虑去重则使用union**。

### 10. 性能分析

**MySQL常见瓶颈**

CPU：CPU饱和时一般发生在数据装入内存或从磁盘中读取数据的时候。

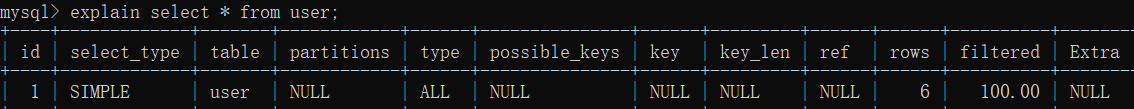
IO：磁盘I/O瓶颈发生在装入数据远大于内存容量的时候。

服务器硬件的性能瓶颈。

### 11.explain关键字

**Explain关键字**：使用该关键字可以模拟优化器执行SQL查询语句，从而知道MySQL是如何处理SQL语句，从而可以分析查询语句或者表结构的性能瓶颈。

**语法**：explain + SQL语句。



使用explain关键字执行SQL语句之后，可以知道表的读取顺序、数据读取操作的操作类型、哪些索引可以使用、哪些索引被实际使用、表之间的引用、每张表有多少行被优化器查询。

使用explain执行SQL语句之后，各关键子的含义：

要能说出**id、select\_type、type、possible\_keys、key、rows**等。

#### ⑴id

**id**：这是select查询的序列号，包含一组数字，指查询中执行select子句或者操作表的顺序。**值有三种情况**：**id相同，执行顺序由上至下**；**id值不同，如果是子查询，id的序号会递增，id值越大优先级越高，越先被执行**；**id值有相同的也有不同的，其中id值相同可认为是一组，从上往下顺序执行，id值不同，id值越大优先级越高，越先被执行**。

#### ⑵select\_type

**select\_type**：指查询类型，主要用于区别普通查询、联合查询、子查询等，常用的值有simple、primary、subquery、derived、union、union result。

**simple**：指简单的select查询，查询中不包含子查询或union。

**primary**：当查询中包含任何复杂的子部分，则最外层查询则被标记为primary。

**subquery**：在select或者where列表中包含了子查询，则该查询类型被是subquery。

**derived**：在from列表中包含的子查询被标记为derived(衍生),MySQL会递归执行这些子查询，然后把结果放到临时表中。

**union**：若第二个select出现在union之后，则该select语句标记为union，若union包含在from子句的子查询中，外层select将被标记为derived。**union result**：从union表中获取结果的select语句。

#### ⑶table

**table**：查询的表。

#### ⑷type

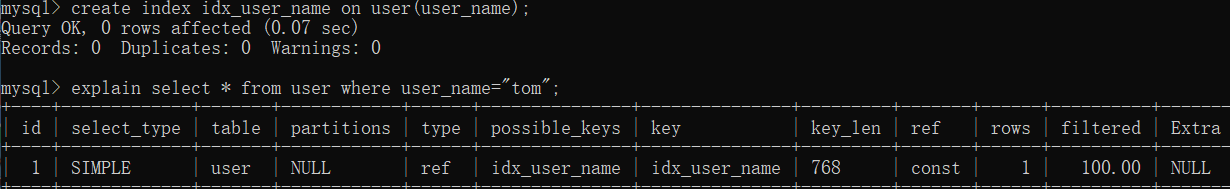
**type**：显示查询使用了何种类型。有八种值，为all、index、range、ref、eq\_ref、const、system、null。从最好到最后依次为：sytem>const>eq\_ref>ref>range>index>all。

**system**：表只有一行记录(等于系统表)。

**const**：表通过索引一次就找到了，const用于比较primary key或者unique索引。因为只匹配一行数据，所以很快。如将主键作为where条件，MySQL就能将该查询转换为一个常量。

**eq\_ref**：唯一索引扫描，对于每个索引键，表种只有一条记录与之相匹配。常用于主键或者唯一索引扫描。

**ref**：非唯一性索引扫描，返回匹配某个单独值的所有行，它可能会找到多个符合条件的行。



**range**：只检索给定范围的行，使用一个索引来选择行。key列显示使用了哪个索引。一般就是在where语句中出现了between、<、>、in等的查询。这种扫描比全表扫描要好，因为它只需要开始于索引的某一点，而结束于另一点，不用扫描全部索引。

**index**：full index scan，index与all的区别为index类型只遍历索引树，通常比all快，因为索引文件通常比数据文件小。也就是说all和index都是读全表，但index是从索引中读取的，而all则是从硬盘中读取的。例子：explain select id from user;

**all**：full table scan，将遍历全表以找到匹配的行。

一般保证查询至少达到range级别，最后能达到ref。

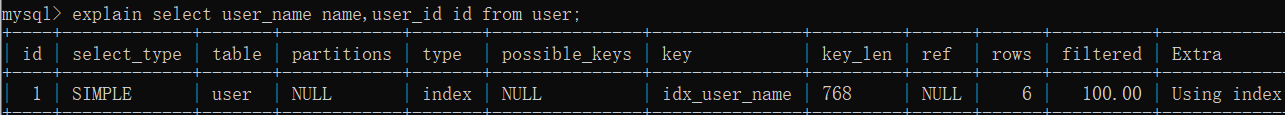
#### ⑸possible\_keys

**possible\_keys**：显示可能应用在这张表中的索引，一个或多个,即理论上应该用到的索引。查询设计到的字段若存在索引，则该索引将被列出，但不一定被实际使用。

#### ⑹key

**key**：实际使用的索引。若值为null，则没有使用索引，要么没建索引，要么建的索引失效了。查询中若使用了覆盖索引，则该索引仅出现在key列表中。

有时会出现possible\_keys的值为null，即理论上没有用到索引，但是key的值不为null，即实际上使用了索引。



#### ⑺key\_len

**key\_len**：表索引中使用的字节数，可通过该列计算查询中使用的索引的长度。在不损失精确性的情况下，长度越短越好。key\_len显示的值为索引字段的最大可能长度，并非实际使用长度，即key\_len是根据表定义计算的，而不是通过表内检索出的。

#### ⑻ref

**ref**：显示索引的哪一列被使用了，若可能的话，是一个常数。哪些列或常量被用于查找索引列上的值。

#### ⑼rows

**rows**：根据表统计信息及索引选用情况，大致估算出找到所需的记录所需读取的行数。rows的值越小越好。

#### ⑽extra

**extra**：包含不适合在其他列中显示但十分重要的额外信息。extra的值常用的如下：

**Using filesort**：mysql会对数据使用一个外部的索引排序，而不是按照表内的索引顺序进行读取。即MySQL无法利用索引完成的排序操作成为“文件排序”。若出现这个则尽量优化一下。

**Using temporary**：使用了临时表保存中间结果，MySQL在对查询结果排序时使用了临时表。常见于排序order by和分组查询group by中。若出现这个则尽量优化一下。

**Using where**：使用到了where过滤。

**Using index**：表示相应的select操作中使用了覆盖索引(Covering Index)，避免访问了表的数据行，效率不错。若同时出现了Using where，表明索引被用来执行索引键值的查找。 若没有同时出现Using where，表明索引用来读取数据而非执行查找动作。

**覆盖索引的理解**：就是select的数据列只用从索引中就能获取，不必读取数据行。MySQL可以利用索引返回select列表中的字段，而不必根据索引再次读取数据文件。

若需要使用覆盖索引，一定要注意select列表中只需取出需要的列，不可select \*，因为若将索引字段一起做索引会等导致索引文件过大，查询性能下降。

**Using join buffer**：使用了连接缓存。

**impossible where**：where子句的值为false。

