MySQL学习笔记

# 数据查询（DQL）

|  |  |  |
| --- | --- | --- |
| 查询方式 | 语法 | 特点 |
| 分组查询 | select 查询列表 from 表  【where 筛选条件】  group by 分组的字段； | 分组函数查询的字段必须是group by后出现的字段；  having是对分组后的结果集筛选； |
| 子查询 | select后面：  仅仅支持标量子查询  from后面：  支持表子查询  where或having后面：★  标量子查询（单行） √  列子查询 （多行） √  行子查询  exists后面（相关子查询）  表子查询 |  |
| 联合查询 | 查询语句1 UNION 查询语句2 UNION …… | 要求多条查询语句的查询列数是一致的，但顺序可以不同  查询结果默认去重，使用UNION ALL 可以包含重复项 |

|  |  |
| --- | --- |
| 连接查询 | 特点 |
| 内连接 |  |
| 左外连接 |  |
| 右外连接 |  |
| 全外连接 | mysql不支持： |

# 数据增删改（DML）

|  |  |  |
| --- | --- | --- |
| 类型 | 语法 | 特点 |
| 插入 | INSERT INTO 表名(列名,…)  VALUES (对应值,……), (对应值,……)…;  或  INSERT INTO 表名 SET 列名=值，……; | 方式一支持插入多行,方式二不支持；  方式一支持子查询，方式二不支持；  推荐使用方式一； |
| 修改 | UPDATE 表名  SET 列=新值,列=新值,...  WHERE 筛选条件;  或  UPDATE 表1 别名  INNER|LEFT|RIGHT JOIN 表2 别名  ON 连接条件  SET 列=值,...  WHERE 筛选条件; | 方式一为单表修改，方式二为多表修改； |
| 删除 | DELETE FROM 表名 WHERE 筛选条件；  或  TRUNCATE TABLE 表名; | 1.delete 可以加where 条件，truncate不能加  2.truncate效率较高  3.表中若有自增长列，用delete删除后，再插入数据，自增长列的值从断点开始，而truncate删除后，再插入数据，自增长列的值从1开始。  4.truncate删除没有返回值，delete删除有返回值  5.truncate删除不能回滚，delete删除可以回滚. |

# 库和表（DDL）

|  |  |  |
| --- | --- | --- |
| 操作 | 库 | 表 |
| 创建 | CREATE DATABASE 库名; | 创建:CREATE TABLE 表名(列名 类型(长度) 约束,…...);  复制表结构：CREATE TABLE 新表名LIKE 表名;  复制部分表结构及部分数据：  CREATE TABLE copy3  SELECT id,name FROM author WHERE nation='中国'; |
| 修改 | RENAME DATABASE 旧库名 TO 新库名;  ALTER DATABASE 库名CHARACTER SET gbk; | 表中列的操作：  ALTER TABLE 表名 ADD|MODIFY|DROP COLUMN 列名 [类型 约束];  ALTER TABLE 表名 CHANGE COLUMN 旧列名 新列名 类型;  修改表名：ALTER TABLE 旧表名RENAME TO 新表名; |
| 删除 | DROP DATABASE IF EXISTS 库名; | DROP TABLE IF EXISTS 表名; |
| 其他 | 查看所有：SHOW DATABASES;  查看当前：SELECT DATABASE();  使用：USE 库名; | 查看表结构：DESC 表名;  查看所有：SHOW TABLES;  查看表信息：SHOW CREATE TABLE 表名; |

# 数据类型

|  |  |  |
| --- | --- | --- |
| 分类 | 类型 | 特点 |
| 数值型 | 整型：  tinyint 1B、smallint 2B、bigint 8B mediumint 3B、int/integer 4B、  小数：  float(M,D)、double(M,D)、dec(M,D)/decimal(M,D) | 默认是有符号，unsigned设置无符号；  插入数值超出范围,会报out of range异常，并插入临界值；  长度代表显示的最大宽度；  ZEROFILL：用0填充不够的宽度，并会设置其为无符号；  M:总位数，D:小数位数；  dec/decimal默认为dec(10,0)； |
| 字符型 | char、varchar、text、blob | char(M),M默认为1，固定长度，效率高；  varchar(M),M不能省略，可变长度，效率低； |
| 日期型 | date/time/year/datetime/timestamp | datetime 8B，不受时区影响，timestamp 4B，受时区影响； |
| 其他 | ENUM、SET、 |  |

# 约束

|  |  |
| --- | --- |
| 约束类型 | 特点 |
| 非空约束：  NOT NULL | 添加/删除：ALTER TABLE 表名 MODIFY COLUMN 列名 类型[NOT NULL]; |
| 默认约束：  DEFAULT | 添加/删除：ALTER TABLE 表名 MODIFY COLUMN 列名 类型[DEFAULT 默认值]; |
| 唯一约束：  UNIQUE | 添加：ALTER TABLE 表名 MODIFY COLUMN 列名 类型 UNIQUE;  或：ALTER TABLE 表名ADD UNIQUE(列名);  删除：ALTER TABLE 表名DROP INDEX 约束列;  可以多个字段组合成一个唯一约束； |
| 检查约束：  CHECK | MYSQL不支持检查约束；  语法：gender CHAR(1) CHECK(gender=’male’ OR gender=’female’) |
| 主键约束：  PRIMARY KEY | 添加：ALTER TABLE 表名 MODIFY COLUMN 列名 类型 PRIMARY KEY;  或：ALTER TABLE 表名ADD PRIMARY KEY(列名);  删除：ALTER TABLE 表名 DROP PRIMARY KEY;  唯一+非空，一张表只有一个主键约束，但可以有多个非空约束；  可以多个字段组合成一个主键；  删除主键约束前，要先删除自增长,否则无法删除； |
| 外键约束：  FOREIGN KEY | 要求主表中的关联列必须是一个键（主键、唯一）；  添加：ALTER TABLE 表名ADD CONSTRAINT 外键名 FOREIN KEY (外键列) REFERENCES 主表名(关联列);  删除：ALTER TABLE 表名 DROP FOREIGN KEY 外键名; |
| 自增约束：  AUTO\_INCREMENT | 添加/删除：ALTER TABLE 表名 MODIFY COLUMN 列名 类型[AUTO\_INCREMENT];  一张表只能有一个自增长列，并且该列必须定义了约束（主键、唯一、外键）；  只能在数值型中使用，一般配合主键使用； |
| 总结对比 | 列级约束：直接在字段类型后添加，支持默认、非空、主键、唯一；  表级约束：在所有字段后添加，使用CONSTRAINT，支持：主键、外键、唯一；  一般把外键写在表级约束中，其他写在列级约束；  添加列级约束格式：alter table 表名 modify column 字段名 字段类型 新约束;  添加表级约束格式：alter table 表名 add [constraint 约束名] 约束类型(字段名) 【外键的引用】; |

# 事务（TCL）

|  |  |
| --- | --- |
| ACID属性 | 解释 |
| 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。 |
| 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。 |
| 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。 |
| 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 |

|  |  |
| --- | --- |
| 并发问题 | 解释 |
| 脏读 | 对于两个事务 T1, T2, T1 读取了已经被 T2 更新但还没有被提交的字段之后, 若 T2 回滚, T1读取的内容就是临时且无效的. |
| 不可重复读 | 对于两个事务T1, T2, T1 读取了一个字段, 然后 T2 更新了该字段之后, T1再次读取同一个字段, 值就不同了. |
| 幻读 | 对于两个事务T1, T2, T1 从一个表中读取了一个字段, 然后 T2 在该表中插入了一些新的行之后, 如果 T1 再次读取同一个表, 就会多出几行. |

|  |  |
| --- | --- |
| mysql事务隔离级别 | 解释 |
| READ UNCOMMITTED  读未提交 |  |
| READ COMMITTED  读已提交 |  |
| REPEATABLE READ  可重复读 |  |
| SERIALIZABLE  串行化 |  |

innoDB才支持事务，事务只针对DML，对DDL无效；

事务演示：

SET autocommit=0;

START TRANSACTION;

DELETE FROM account WHERE id=25;

SAVEPOINT a;#设置保存点

DELETE FROM account WHERE id=28;

ROLLBACK TO a;#回滚到保存点

# 视图

|  |  |
| --- | --- |
| 操作 | 特点 |
| 创建 | 语法：CREATE VIEW 视图名 AS 查询语句； |
| 修改 | 语法：  ALTER VIEW 视图名 AS 查询语句；  或CREATE OR REPLACE VIEW 视图名 AS 查询语句； |
| 查看 | 语法：DESC 视图名;  或SHOW CREATE VIEW 视图名; |
| 删除 | DROP VIEW 视图名,…; |
| 视图内容操作 | 视图内容的操作同表中记录的操作一致；修改视图中的内容可能会影响到原表数据；  一般不操作内容； |
| 总结 | 不占用实际的数据物理空间（保留一些逻辑）； |

# 索引

|  |  |
| --- | --- |
| 操作 | 特点 |
| 创建 | 语法：CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名 ON 表名（列名）;  ALTER TABLE 表名 ADD [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名 ON （列名）; |
| 删除 | 语法：DROP INDEX 索引名 ON 表名；  ALTER TABLE 表名 DROP INDEX 索引名 |
| 查看 | 语法：SHOW INDEX FROM 表名；  SHOW CREATE TABLE 表名； |
| 总结 | 主键自动创建唯一索引； |

|  |  |  |
| --- | --- | --- |
| 索引类型 | 普通索引 |  |
| 唯一索引 |  |
| 复合索引 |  |
|  |  |
| 索引结构 | Btree索引 |  |
| Hash索引 |  |
| FullText索引 |  |
| Rtree索引 |  |

# 变量

|  |  |  |
| --- | --- | --- |
| 变量 | 分类 | 特点 |
| 系统变量 | 全局  会话 | 查看所有：SHOW GLOBAL|SESSION VARIABLES;  查看：SHOW GLOBAL|SESSION VARIABLES LIKE ‘%char%’;或SELECT @@GLOBAL|SESSION.变量名；  修改：SET GLOBAL|SESSION 变量名=值；或SET @@GLOBAL|SESSION.变量名=值；  全局变量对所有会话有效，但重启后恢复默认值；局部变量只对当前会话有效； |
| 自定义变量 | 用户变量 | 初始化/修改：SET @变量名=值；或SET @变量名:=值；或SELECT @变量名:=值；  查看：SELECT @变量名; |
| 局部变量 | 声明：DECLARE 变量名 类型；或 DECLARE 变量名 类型 DEFAULT 值；  赋值：SET 变量名=值；或SET 变量名:=值；或SELECT @变量名:=值；  查看：SELECT 变量名;  只能在BEGIN。。。END中的第一句； |

# 存储过程

|  |  |
| --- | --- |
| 操作 | 特点 |
| 创建 | CREATE PROCEDURE 存储过程名(IN|OUT|INOUT 参数名 参数类型，。。。)  BEGIN  存储过程体（一组合法的SQL语句）  END |
| 调用 | CALL 存储过程名（实参列表）； |
| 删除 | DROP PROCEDURE 存储过程名； |
| 查看 | SHOW CREATE PROCEDURE 存储过程名； |

# 函数

|  |  |
| --- | --- |
| 操作 | 特点 |
| 创建 | CREATE FUNCTION 函数名(参数列表) RETURNS 返回类型  BEGIN  函数体；RETURN 值；  END |
| 调用 | SELECT 函数名（实参列表） |
| 删除 | DROP FUNCTION 函数名； |
| 查看 | SHOW CREATE FUNCTION 函数名； |

# 流程控制

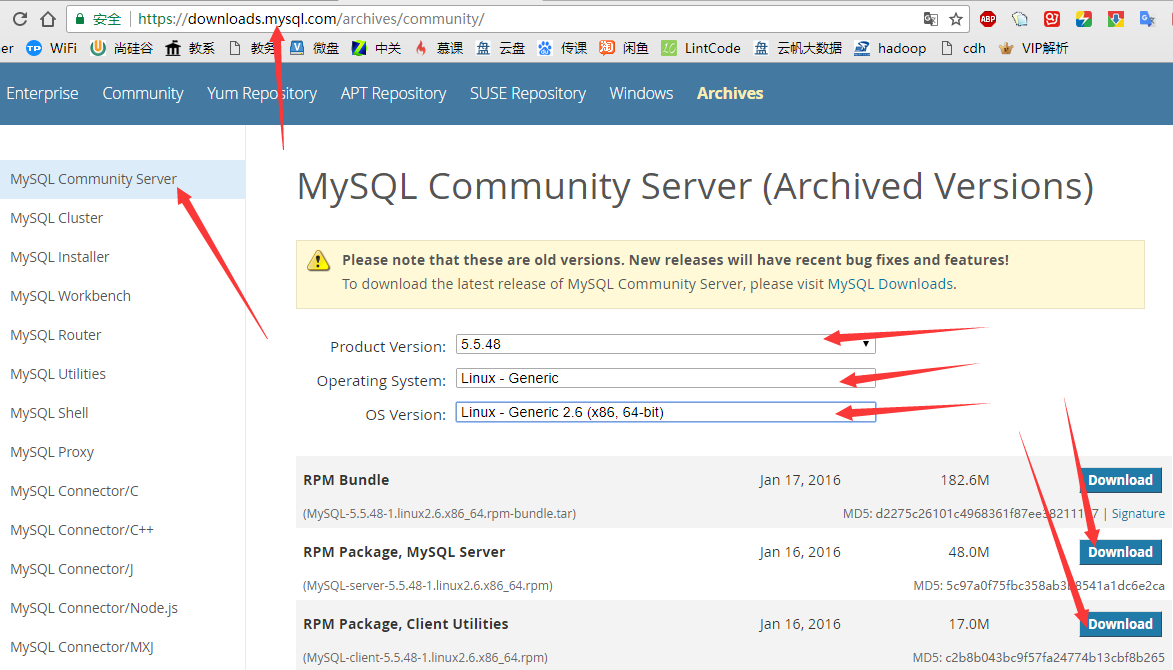
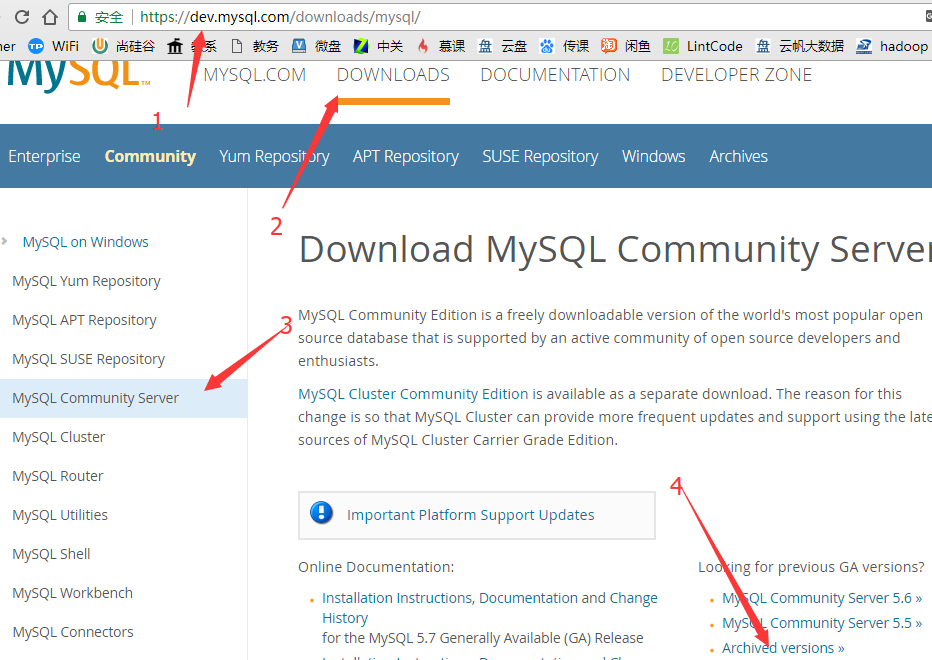
|  |  |  |
| --- | --- | --- |
| 结构 |  | 举例 |
| 分支 | IF语法：IF(判断，结果1，结果2);  CASE语法：  CASE [变量|表达式|字段]  WHEN 判断 THEN 结果1/语句1;  WHEN 判断 THEN 结果2/语句2;  ELSE 结果N/语句N;  END CASE;  IF语法：  IF 条件1 THEN 语句1；  ELSEIF 条件2 THEN 语句2；  ELSE 语句N；  END IF； | CREATE FUNCTION test\_if(score FLOAT) RETURNS CHAR  BEGIN  DECLARE ch CHAR DEFAULT 'A';  IF score>90 THEN SET ch='A';  ELSEIF score>80 THEN SET ch='B';  ELSEIF score>60 THEN SET ch='C';  ELSE SET ch='D';  END IF;  RETURN ch;  END $  CREATE FUNCTION test\_case(score FLOAT) RETURNS CHAR  BEGIN  DECLARE ch CHAR DEFAULT 'A';  CASE  WHEN score>90 THEN SET ch='A';  WHEN score>80 THEN SET ch='B';  WHEN score>60 THEN SET ch='C';  ELSE SET ch='D';  END CASE;  RETURN ch;  END $ |
| 循环 | WHILE语法：  [标签: ]WHILE循环条件 DO  循环体;[LEAVE 标签];  END WHILE [标签];  LOOP语法：  [标签:] LOOP  循环体；  END LOOP [标签];  REPEAT语法：  [标签: ]REPEAT  循环体;  UNTIL循环条件  END REPEAT [标签]; | CREATE PROCEDURE test\_while(IN count INT)  BEGIN  DECLARE i INT DEFAULT 1;  a:WHILE i<=count DO  INSERT INTO admin(username,`password`) VALUES(i,'0000');  IF i>=20 THEN LEAVE a;END IF;  SET i=i+1;  END WHILE a;  END $ |

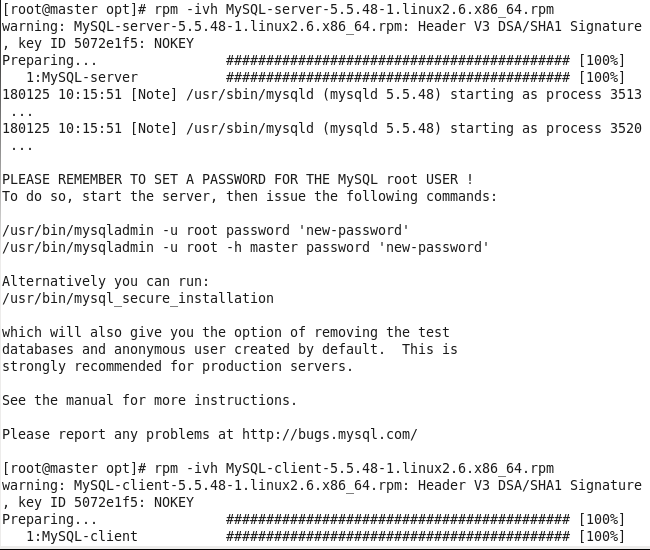
# 综合练习：批数据脚本

|  |  |  |
| --- | --- | --- |
| 步骤 |  |  |
| 建库建表 | CREATE DATABASE zhp;  CREATE TABLE dept(  id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  dname VARCHAR(20) NOT NULL DEFAULT ""  ); | CREATE TABLE emp(  id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  ename VARCHAR(20) NOT NULL,  manager INT UNSIGNED NOT NULL,  sal DOUBLE NOT NULL,  dept INT  ); |
| 开启 | show variables like 'log\_bin\_trust\_function%';  set global log\_bin\_trust\_function\_creators=1; |  |
| 函数 | DELIMITER $  CREATE FUNCTION rand\_string(n INT) RETURNS VARCHAR(255)  BEGIN  DECLARE chars VARCHAR(60) DEFAULT 'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';  DECLARE str VARCHAR(60) DEFAULT '';  DECLARE i INT DEFAULT 0;  WHILE i<n DO  SET str=CONCAT(str,SUBSTRING(chars,FLOOR(1+RAND()\*52),1));  SET i=i+1;  END WHILE;  RETURN str;  END $ | DELIMITER $  CREATE FUNCTION rand\_num(n INT) RETURNS INT(5)  BEGIN  DECLARE i INT DEFAULT 0;  SET i=FLOOR(RAND()\*n+1);  RETURN i;  END $ |
| 存储过程 | CREATE PROCEDURE insert\_emp(IN num INT)  BEGIN  DECLARE i INT DEFAULT 0;  SET autocommit=0;  REPEAT  SET i=i+1;  INSERT INTO emp(ename,manager,sal,dept)  VALUES(rand\_string(8),rand\_num(5),5000,rand\_num(10));  UNTIL i=num  END REPEAT;  COMMIT;  END $ | CREATE PROCEDURE insert\_dept(IN num INT)  BEGIN  DECLARE i INT DEFAULT 0;  SET autocommit=0;  REPEAT  SET i=i+1;  INSERT INTO dept(dname) VALUES(rand\_string(10));  UNTIL i=num  END REPEAT;  COMMIT;  END $ |
| 执行 | CALL insert\_emp(10) | CALL insert\_dept(10) |

# Linux安装配置MySQL

rpm安装步骤：



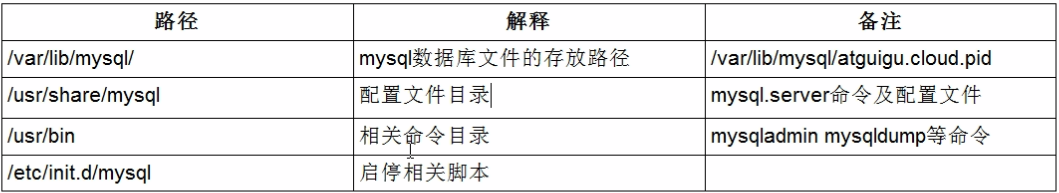


配置

设置密码：/usr/bin/mysqladmin -u root password 123456

设置开机启动：chkconfig mysql on

配置文件：



5.5版本：cp /usr/share/mysql/my-huge.cnf /etc/my.cnf

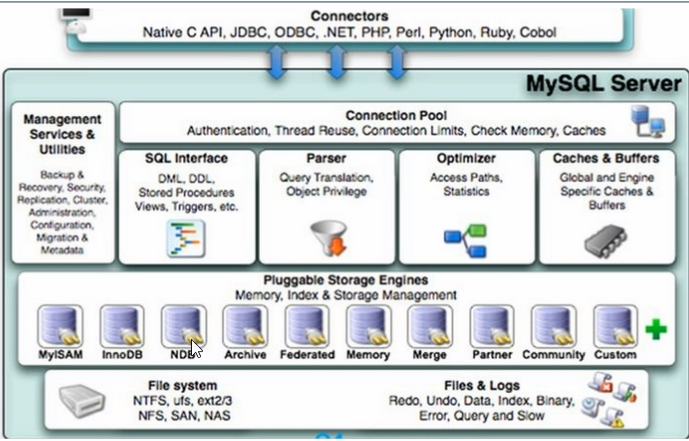
5.6版本：cp /usr/share/mysql/my-default.cnf /etc/my.cnf

查看字符集：show valiables like '%char%';

修改my.cnf配置文件并重启服务：

|  |
| --- |
| [client]  #password = your\_password  port = 3306  socket = /var/lib/mysql/mysql.sock  default-character-set=utf8  [mysqld]  port = 3306  character\_set\_server=utf8  character\_set\_client=utf8  collation-server=utf8\_general\_ci  [mysql]  no-auto-rehash  default-character-set=utf8 |

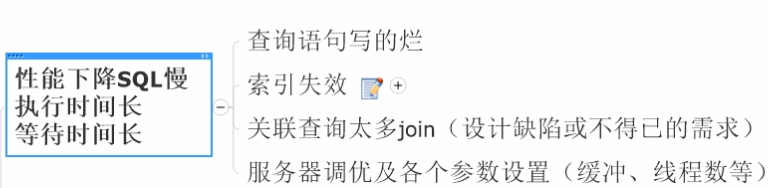
逻辑架构



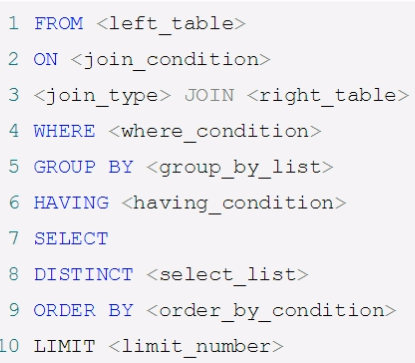
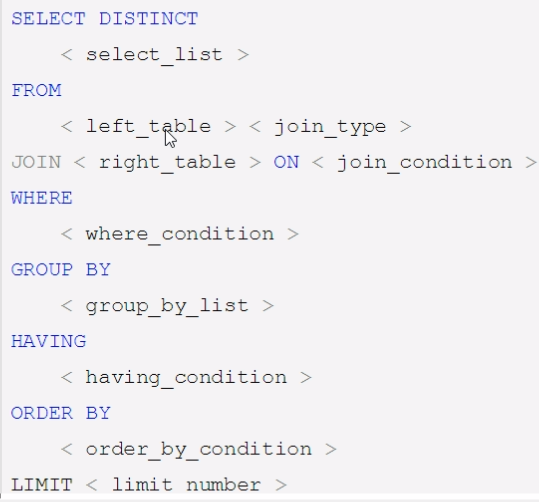
存储引擎

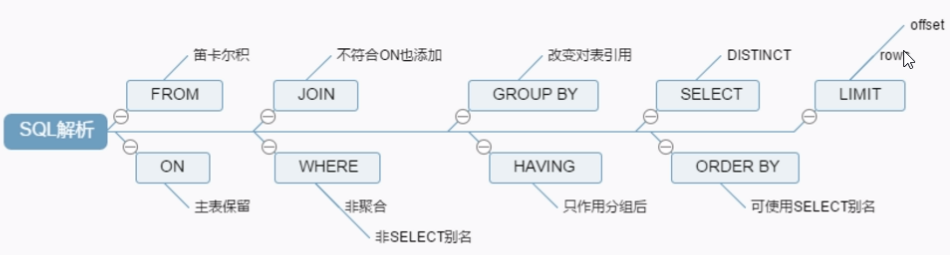


# 性能分析



执行循序：





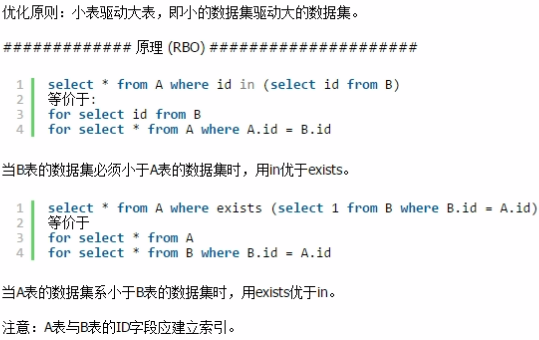
性能分析结果(explain)：

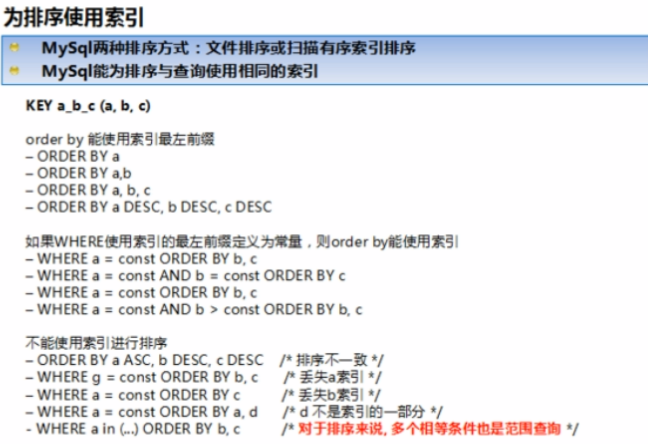
|  |  |
| --- | --- |
| explain表头 | 解释 |
| id | id相同，执行循序从上至下；  id不同，若是子查询，id会递增，id值越大，优先级越高； |
| select\_type | 包括：SIMPLE,PRIMARY,SUBQUERY,DERIVED,UNION,UNION RESULT；  由于区别查询类型； |
| type | 包括：ALL,INDEX,RANGE,REF,EQ\_REF,CONST/SYSTEM,NULL,FULLTEXT等；  访问类型；system>const>eq\_ref>ref>fulltext>range>index>all |
| possible\_keys  key | possible\_keys：可能用到的索引；key：实际使用到的索引；  可能possible\_keys为空，但实际key不为空，即覆盖索引； |
| key\_len | 表示索引中使用的字节数。保证精度下，越短越好；  是最大可能长度，并非实际使用长度； |
| ref | 显示索引的哪一列被使用 |
| rows | 每张表有多少行被优化器查询 |
| extra | 包含不适合在其他列显示，但十分重要的信息；  包括：Using filesort,Using temporary,Using index,Using where,Using join buffer,  impossible where,select tables optimized away,distinct; |

# 索引优化

|  |
| --- |
| 防止索引失效: |
| 最佳左前缀法则：多列索引，查询从索引最左列开始，并且不跳过索引中的列； |
| 不在索引列中做计算、函数、类型转换等操作；如：varchar类型不加引号会导致索引失效； |
| 索引列若使用了范围条件（范围条件会使用索引），则索引顺序之后的列不能使用索引； |
| 使用不等于或is null、is not null的时候，无法使用索引； |
| like ‘%abc…’索引失效；解决方法：使用覆盖索引； |
| 使用or连接会导致索引失效； |

# 查询优化





# 慢查询日志

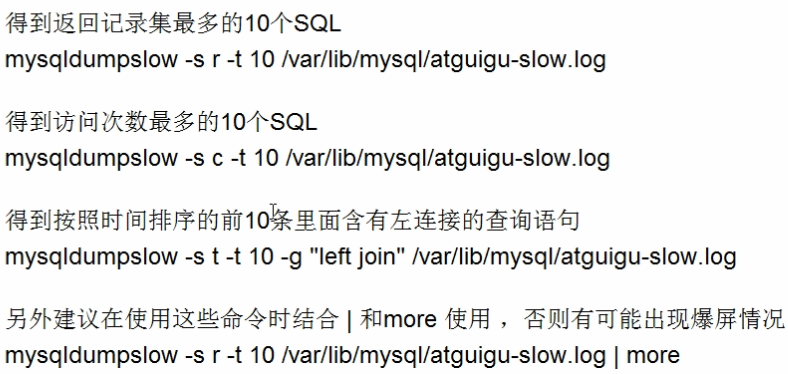
查看：show variables like '%slow\_query\_log%'; 默认关闭，一般不打开。（注意加单引号）

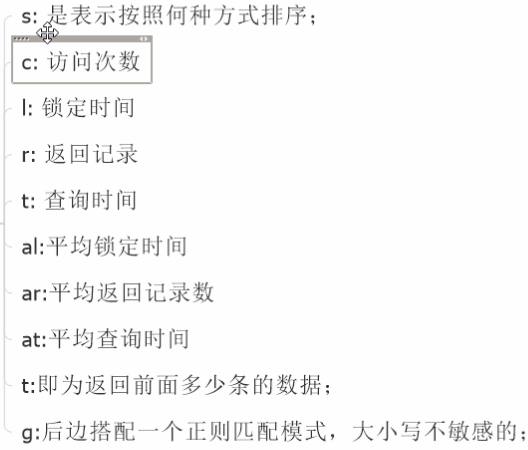
开启：set global slow\_query\_log=1;重启服务失效；

永久开启：修改配置文件etc/my.cnf添加两行；

慢查询时间：show variables like '%long\_query\_time%';

修改：set global long\_query\_time=3;重启会话生效；重启服务失效；

使用命令工具mysqldumpslow：



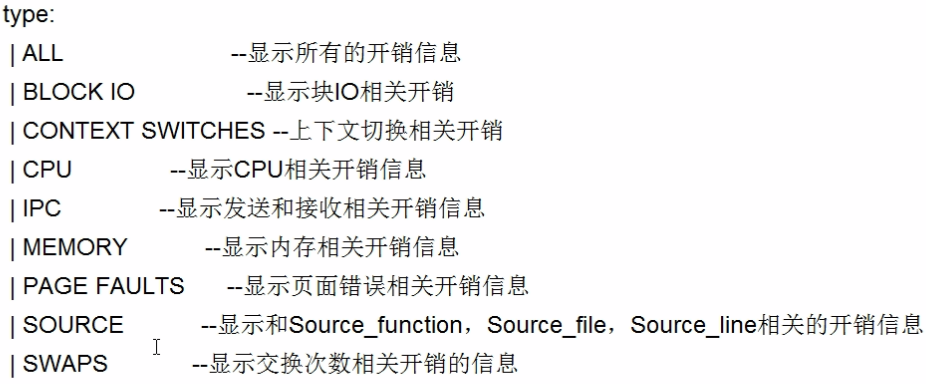
# show profile

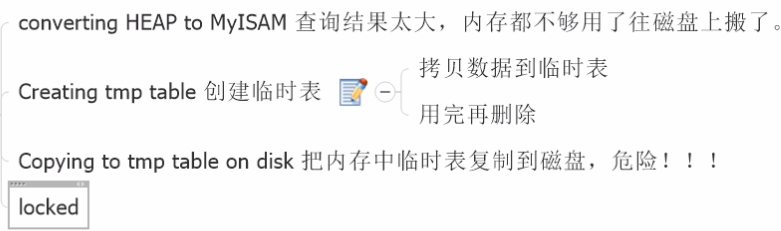
查看：show session variables like '%profiling%'; 默认关闭

开启：set profiling=on;

查看语句执行情况：show profiles;

查看某语句具体情况：show profile cpu,block io for query [id];





# 全局查询日志

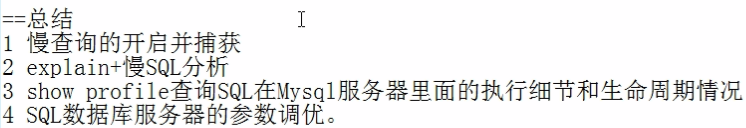
开启：set global general\_log=1;set global log\_output=’TABLE’;select \* from mysq.general\_log;

配置开启：

general\_log=1

general\_log\_file=/path/logfile

log\_output=FILE



# 锁

查看表锁：show open tables；

加表锁：lock table user read,mylock write;

解表锁：unlock tables;

查看表锁状态：show status like 'table%';

查看行锁状态：show status like 'innodb\_row\_lock%';

锁定一行：SQL+for update；

|  |  |  |
| --- | --- | --- |
| 类型 | 当前表 | 其他表 |
| 读锁 | 可以读锁表；  不可以写锁表；  不可以读其他表；  不可以写其他表； | 可以读锁表；  不可以写锁表（被阻塞）；  可以读其他表；  可以写其他表； |
| 写锁 | 可以读写锁表；  不可以读写其他表； | 不可以读写锁表（被阻塞）；  可以读写其他表； |
| 总结 | 读锁会阻塞写，但不会阻塞读；写锁会阻塞读写；  MyISAM默认读操作会加读锁，写操作会加写锁；默认是写优先，不适合做写为主的表引擎； | |

索引失效会导致行锁变表锁；

# 主从复制

|  |  |  |
| --- | --- | --- |
| 步骤 | MySQL5.5（Windows）主机 | MySQL5.5（Linux）从机 |
| 修改配置文件 | my.ini的[mysqld]中添加：  server-id=1  log-bin=D:/MySQL/master/mysqlbin  log-err=D:/MySQL/master/mysqlerr  basedir="D:/MySQL/"  tmpdir="D:/MySQL/"  binlog-ignore-db=mysql  #binlog-do-db=ssh | my.cnf的[mysqld]中添加：  log-bin=mysql-bin  server-id=2  (需先注释掉默认的server-id=1) |
| 重启服务并关闭防火墙 | 重启服务并关闭防火墙 | service mysql restart  service iptables stop |
| 主机账户授权 | grant replication slave on \*.\* to 'zhp'@'192.168.168.13' identified by '123456';  flush privileges;  show master status; | change master to  master\_host='192.168.168.100',  master\_user='zhp',  master\_password='123456',  master\_log\_file='mysqlbin.000001',  master\_log\_pos=107;  start slave;  show slave status\G  stop slave; |

