目录

[面试知识学习总结](#_Toc416_WPSOffice_Level1) [1](#_Toc416_WPSOffice_Level1)

[一、 数据库](#_Toc8822_WPSOffice_Level1) [2](#_Toc8822_WPSOffice_Level1)

[1. 数据库事务的四个特性](#_Toc8822_WPSOffice_Level2) [2](#_Toc8822_WPSOffice_Level2)

[(1) ACID](#_Toc8822_WPSOffice_Level3) [2](#_Toc8822_WPSOffice_Level3)

[2. 数据库视图](#_Toc10292_WPSOffice_Level2) [2](#_Toc10292_WPSOffice_Level2)

[(1) 定义：根据所查询的结果所生成的虚拟表，里面数据只在使用时动态生成](#_Toc10292_WPSOffice_Level3) [2](#_Toc10292_WPSOffice_Level3)

[(2) 常用操作：](#_Toc10937_WPSOffice_Level3) [2](#_Toc10937_WPSOffice_Level3)

[(3) 何时使用：](#_Toc17887_WPSOffice_Level3) [2](#_Toc17887_WPSOffice_Level3)

[(4) 优点（和上面使用相对应）：](#_Toc12930_WPSOffice_Level3) [2](#_Toc12930_WPSOffice_Level3)

[(5) 和表的关系](#_Toc32715_WPSOffice_Level3) [3](#_Toc32715_WPSOffice_Level3)

[(6) 内部算法（了解）](#_Toc13832_WPSOffice_Level3) [3](#_Toc13832_WPSOffice_Level3)

[3. drop,delete与truncate](#_Toc10937_WPSOffice_Level2) [3](#_Toc10937_WPSOffice_Level2)

[(1) 定义：](#_Toc29172_WPSOffice_Level3) [3](#_Toc29172_WPSOffice_Level3)

[(2) 区别：](#_Toc14199_WPSOffice_Level3) [3](#_Toc14199_WPSOffice_Level3)

[4. .数据库表连接](#_Toc17887_WPSOffice_Level2) [3](#_Toc17887_WPSOffice_Level2)

[(1) 外连接](#_Toc31630_WPSOffice_Level3) [3](#_Toc31630_WPSOffice_Level3)

[(2) 内连接](#_Toc12313_WPSOffice_Level3) [3](#_Toc12313_WPSOffice_Level3)

[(3) 交叉连接](#_Toc5111_WPSOffice_Level3) [4](#_Toc5111_WPSOffice_Level3)

[5. 三范式](#_Toc12930_WPSOffice_Level2) [4](#_Toc12930_WPSOffice_Level2)

[① 字段不可拆分](#_Toc6157_WPSOffice_Level3) [4](#_Toc6157_WPSOffice_Level3)

[6. 函数、存储过程、触发器](#_Toc32715_WPSOffice_Level2) [4](#_Toc32715_WPSOffice_Level2)

[(1) 函数](#_Toc8229_WPSOffice_Level3) [4](#_Toc8229_WPSOffice_Level3)

[(2) 存储过程](#_Toc8321_WPSOffice_Level3) [5](#_Toc8321_WPSOffice_Level3)

[(3) 触发器](#_Toc5235_WPSOffice_Level3) [6](#_Toc5235_WPSOffice_Level3)

[(4) 三者区别](#_Toc23738_WPSOffice_Level3) [7](#_Toc23738_WPSOffice_Level3)

[7. 分表和分区（了解）](#_Toc13832_WPSOffice_Level2) [7](#_Toc13832_WPSOffice_Level2)

[(1) 分表是将一张表分为多个子表；而分区还是一张表，但存储数据的区块变多](#_Toc19972_WPSOffice_Level3) [7](#_Toc19972_WPSOffice_Level3)

[8. 索引（MySQL）](#_Toc29172_WPSOffice_Level2) [7](#_Toc29172_WPSOffice_Level2)

[(1) 创建：](#_Toc14457_WPSOffice_Level3) [7](#_Toc14457_WPSOffice_Level3)

[(2) 唯一索引：](#_Toc10260_WPSOffice_Level3) [7](#_Toc10260_WPSOffice_Level3)

[(3) 索引的作用：](#_Toc20097_WPSOffice_Level3) [7](#_Toc20097_WPSOffice_Level3)

[9. In和exists](#_Toc14199_WPSOffice_Level2) [7](#_Toc14199_WPSOffice_Level2)

[(1) in：](#_Toc13548_WPSOffice_Level3) [7](#_Toc13548_WPSOffice_Level3)

[(2) Exists：](#_Toc2118_WPSOffice_Level3) [8](#_Toc2118_WPSOffice_Level3)

[10. 优化数据库](#_Toc31630_WPSOffice_Level2) [8](#_Toc31630_WPSOffice_Level2)

[(1) 代码：](#_Toc32596_WPSOffice_Level3) [8](#_Toc32596_WPSOffice_Level3)

[(2) 数据库：](#_Toc30731_WPSOffice_Level3) [8](#_Toc30731_WPSOffice_Level3)

[(3) 服务器：](#_Toc8373_WPSOffice_Level3) [8](#_Toc8373_WPSOffice_Level3)

**面试知识学习总结**

# 数据库

1. 数据库事务的四个特性
   1. ACID
      1. 原子性：要么都完成要么都不完成
      2. 一致性：事务提交前和提交后，数据库的完整性约束不变
      3. 隔离性：两个事务对同个数据在相同时间只可以串行
      4. 持久性：事务提交后，所提交的数据就持久化存储在数据库中
2. 数据库视图
   1. 定义：根据所查询的结果所生成的虚拟表，里面数据只在使用时动态生成
   2. 常用操作：
      1. 创建：create view 视图名 as select 语句
      2. 修改：alter view 视图名 as select 语句
      3. 删除：drop view 视图名
   3. 何时使用：
      1. 经常需要用到的比较复杂的sql语句，可以使用视图来简化sql语句
         1. 比如有一个报表查询，语句十分复杂，各种join group等，我们就可以创建一个视图，这样我们查询该报表，只需使用 select \* from 视图名
      2. 需要进行权限控制的时候
         1. 比如有一张商品表，里面有进价等我们不想让别人看见的信息，我们就可以将查询商品表的权限关闭，同时定义一个视图来展现可以让别人看见的信息。 create view goods\_view as select goods\_id, goods\_name from goods
      3. 大数据分表时
         1. 当有张表数据news有几百万行后，我们可以将其分为四张表news1，news2...并且用视图，把四张表形成一张视图来查询create view news as select \* from news1 union select \* from news2 union ...
   4. 优点（和上面使用相对应）：
      1. 简化sql
      2. 控制权限，保护数据
      3. 大数据分表
   5. 和表的关系
      1. 表的数据变化，视图数据也会更新
      2. 视图数据增删改，表的数据也会变化
      3. 一般不对视图进行增删改，其中大部分视图是不可以更新的
         1. 简单关系的可以更新
         2. 多个子表或者包含分组 聚合函数的 不可更新（大部分视图都是复杂的）
   6. 内部算法（了解）
      1. 本质：数据库会有一些算法将对视图的查询语句和视图中的查询语句相结合，形成可以查询的sql语句
3. drop,delete与truncate
   1. 定义：
      1. Drop：删除表
      2. Delete：删除表中的一行数据
      3. Truncate：删除表中的所有数据（语法：TRUNCATE TABLE 表名）
   2. 区别：
      1. Delete，truncate删除的是数据，drop删除的是table
      2. Delete可以对table和view操作，truncate只能对table操作
      3. Delete from user 和 truncate table user 都是删除所有数据，但是truncate速度快，因为它使用的系统和事务日志资源少
         1. Delete每删除一行，都会记录日志
      4. Delete不会减少表和索引所占空间，truncate会恢复到初始，drop会全部释放
4. .数据库表连接
   1. 外连接
      1. Left join或left outer join 以左边表为基准,右边可能有空的
      2. Right join或right outer join 以右边表为基准,左边可能有空的
      3. Full join 或 full outer join 完全返回两边表,两边可能有空的
   2. 内连接
      1. Inner join 或 inner outer join 只返回符合两边连接条件的数据，不会有空的
   3. 交叉连接
      1. cross join没有连接条件
5. 三范式
   1. 1NF
      1. 字段不可拆分
   2. 2NF
      1. 有主键，非主键字段依赖主键
   3. 3NF
      1. 非主键字段不能相互依赖
   4. 目前有12345范式，和BC范式。每一个范式都是在上一个范式的基础上实现的。一般使用3范式就够了，性能好，方便管理数据
6. 函数、存储过程、触发器
   1. 函数
      1. 定义：如select Count(\*) ...中的count()就是函数
      2. 具体使用:
         1. 创建：

Create function 函数名 返回值

Begin

方法体

End

例：创建一个比较两数大小的函数

create function my\_fun1(a1 int, a2 int) returns int

begin

if a1>a2 then

return a1;

else

return a2;

end if;

end

* + - 1. 查询：
         1. Show function status （查询所有）
         2. Show create function 函数名 （查询某个函数的创建于巨）
      2. 调用：
         1. Select my\_fun1(1, 2)
      3. 删除：
         1. Drop function 函数名
  1. 存储过程
     1. 定义：执行特定功能的sql代码块
     2. 具体使用：
        1. 创建：

Create procedure 过程名

Begin

方法体

End

例：从1加到99

CREATE PROCEDURE my\_pro2()

BEGIN

DECLARE i int default 1; --DECLARE i 局部变量

SET @sum = 0; --@sum是会话变量 WHILE i<100 DO

SET @sum= @sum+@i;

SET i = i+1;

end WHILE;

SELECT @sum;

END

* + - 1. 查询：
         1. Show procedure status （查询所有）
         2. Show create procedure 过程名 （查看该存储过程创建语句）
      2. 调用：
         1. Call 存储过程名
      3. 删除：
         1. Drop procedure 过程名
      4. 参数（详情见项目 testMysql）：
         1. In 只读进来
         2. Out 不读进来，只输出到外面
         3. Inout 既读又输出
  1. 触发器
     1. 定义：对表数据进行操作后会触发某功能的sql代码块
     2. 具体使用：
        1. 创建：
           1. create trigger 触发器名 时间（before after）对表操作(insert update delete) on 表名 for each row

例如(新增一个订单后，对应商品数量减少)：

Create trigger mytri1 after insert on lj\_order for each row

Begin

update lj\_product set productNum = productNum - new.productNum where id = new.productId;

End

* + - 1. 查询：
         1. Show triggers （查询所有）
         2. Show create trigger 触发器名
      2. 调用：
         1. 满足操作条件后，自动触发
      3. 删除：
         1. Drop trigger 触发器名
  1. 三者区别
     1. 三者都是sql语句块
     2. 函数有返回值，并且方法体中不能用select语句查询，除非是select...into...来赋值某个变量。
     3. 触发器是满足条件，自动调用。其他两个手动调用

1. 分表和分区（了解）
   1. 分表是将一张表分为多个子表；而分区还是一张表，但存储数据的区块变多
2. 索引（MySQL）
   1. 创建：
      1. CREATE INDEX 索引名ON 表名(列名(length)); 直接创建
      2. ALTER table 表名 ADD INDEX 索引名(列名) 修改创建
      3. CREATE TABLE mytable( 建表时指定

ID INT NOT NULL,

username VARCHAR(16) NOT NULL,

INDEX [indexName] (username(length))

);

* 1. 唯一索引：
     1. CREATE unique INDEX 索引名ON 表名(列名(length)); 直接创建
     2. ALTER table 表名 ADD unique 索引名(列名) 修改创建
     3. 和上面普通的类似，将index换位unique
  2. 索引的作用：
     1. 好比是书的目录，提高检索速度，当数据多时，索引的作用很明显
     2. 索引并不是越多越好，它会占用资源，对于经常查询使用的建立索引，无用的不要检索引

1. In和exists
   1. in：
      1. 它是先执行 in里面的内容，然后再查询外面的，将二者笛卡尔积，取得符合条件的

如：

SELECT \* FROM mb\_user WHERE mb\_user.id IN (SELECT user\_id FROM mb\_order

WHERE mb\_order.user\_id = mb\_user.id)

* 1. Exists：
     1. 直接跟在where后面，如下。她是先执行外面的SELECT \* FROM mb\_user，然后每行都执行exists里面的sql，如果符合，就返回true,最终该行查询就保留下来。返回false，就不保留

SELECT \* FROM mb\_user WHERE Exists (SELECT user\_id FROM mb\_order

WHERE mb\_order.user\_id = mb\_user.id)

1. 优化数据库
   1. 代码：
      1. Sql语句方面的优化，如in 与exist，having和where等
      2. 使用缓存等机制，如hibernate中的一级缓存，二级缓存等
   2. 数据库：
      1. 建立索引
      2. 建立视图
      3. 拆分表
      4. 使用范式建表，减少冗余
      5. 使用非范式建表，适量冗余
   3. 服务器：
      1. 砸钱