MySQL数据库面试题

简述什么是关系型数据库

所谓关系型数据库，是指采用了关系模型来组织数据的数据库。关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。 关系型数据库是由许多数据表（Table）所组 成，表又是由许多记录（Row 或Record）所组成，而纪录又是由许多的字段（Column 或Filed）所组成。

什么是主键?

设置一个表中的某个字段为主键，这个字段能够唯一的确定该表中某条记录，这样的字段我们称为主键

什么是外键

外部键约束用于强制参照完整性，提供单个字段或者多个字段的参照完整性

主外键关联的作用?

保证数据完整性。

消除冗余数据。

SQL语句中文含义是?

Structured Query Language，结构化查询语言

什么是数据库

数据库是长期存储在计算机内的、有组织的、可共享的数据集合

什么数据库管理系统?

DBMS就是实现把用户意义下的抽象的逻辑数据转换成计算机中的具体的物理数据的处理软件。

什么是数据库?

数据库是按照数据结构来组织、存储和管理的仓库

关系型数据库的特点是什么?

1）数据以表格的形式出现

2)每行为各种记录名称

3)每列为记录名称所对应的数据域

4)许多的行和列组成一张表单

5)若干的表单组成数据库

常见的数据库有哪些?

SQLserver Access mysql oracle DB2 人大金仓

请介绍一下数据库常用的术语有哪些？并进行解释?

数据库:数据库是一些关联表的集合

数据表:表是数据的矩阵，在一个数据库中表看起来就像一个电子表格一样 列:一列包含了相同类型的数据

行:一行是一组相关的数据

冗余:存储两倍数据，冗余降低了性能，但是提高了数据的安全性

主键:主键是唯一的，一个数据表中只能包含一个主键，你可以使用主键来查询数据

外键:外键用于关联两个表

复合键:将多个列作为一个索引键，一般用于复合索引

索引:使用索引可以快速的访问数据表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结 构，类似于数的目录参照完整性:要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件

表头:每一列的名称

列:具有相同数据类型的数据的集合

行:每一行用来描述某条记录的具体信息

值:行的具体信息，每个值必须与该列的数据类型相同

键:键的值在当前列中具有唯一性

请简单的介绍一下MySQL数据库?

mysql是开源的，免费，支持大型的数据库，可以处理拥有上千万条记录的大型数据库，使用标准的SQL数据 语言形式，可以支持多种操作系统使用，支持多种编程语言的使用

SQL语句可以划分成几类?

1） DDL 数据定义语句主要针对是数据库中表的操作--创建修改删除

2) DML 数据操纵语句主要是对表中的数据进行插入、更新、删除

3) DQL 数据查询语句主要是对表中的数据进行查询

4) TCL 事务控制语句主要管理数据库的事务提交事务回滚事务

查看当前所有存在的数据库语句是?

show databases;

创建数据库语句是?

格式一:

create database 数据库名称;

格式二:

create database 数据库名称 character set utf-8;

查看创建好的数据库?

show create database 数据库名称;

选择/使用数据库

use 数据库名称;

删除数据库

drop database 数据库名称;

查看系统支持的引擎?

show engines;

查看默认存储引擎?

show variables like 'storage_engine';

常见的数据库类型有哪些?

字符串类型、整数类型、小数类型、日期/时间类型

创表语句如何编写?

create table 表名(

列名1 数据类型 primary key,

列名2 数据类型,

....

);

添加新字段

Alter table 表名 add 列名数据类型;

修改字段名称

alter table 表名 change 旧列名新列名新数据类型;

修改字段类型

alter table 表名 modify 列名数据类型;

删除一个字段

alter table 表名 drop 列名;

修改表的名字

alter table 旧表名 rename 新表名;

修改表的存储引擎

alter table 表名 engine=新存储引擎名称;

查看表结构

desc 表名;

向全部列插入数据

insert into 表名 values(列值1，列值2....);

查看表中全部数据

select *from 表名;

向指定列插入数据

insert into 表名(列名1，列名2....) values(列值1,列值2...);

表中的列名要与列值一一对应

同时插入多条数据

insert into 表名 values(列值1,列值2...),(列值1,列值2...),(列值1,列值2...);

insert into 表名(列名1,列名2...) values(列值1,列值2....),(列值1,列值2....),(列值1,列值2....);

更新语句的格式如何编写?

update 表名 set 列名1=该列新值，列名2=该列新值 where 条件;

如果没有where条件将修改表中的全部数据

删除表中的数据

delete from 表名 where 条件;

删除表

drop table(if exists)表名;

查询表中指定列的数据

select 列名1,列名2.... from 表名;

给列起别名

select 列名1 as 别名1,列名2 as 别名2..... from 表名;

select 列名1 别名1,列名2 别名2 .... from 表名;

去掉重复的列值

select distinct 列名 from 表名;

限制查询

select /列名 from 表名 limit 初始位置,行数;

初始位置是从那行开始显示，一个可选值，默认为0，行数是要查询的行数

排序格式

select /列名 from 表名 order by 列名1 asc/desc,列名2 asc/desc;

asc 升序默认 desc 降序 如果对多列进行排序的时候，首先排序第一列，第一列中必须有相同的列值才会进行第二列排序

条件查询

select /列名 from 表名 where 条件;

条件包括关系运算符和逻辑运算符、特殊情况

关系运算符有哪些?

> < <= >=

逻辑运算符有哪些?

and or not

特殊情况有哪些?

该列列值为空 is null

该列列值不为空 is not null 在之间 between ... and select 列名 from 表名

where 列名 between 初值 and 终止;

用来比较一个列中的几个列值

select 列名 from 表名

where 列名 in(列值1,列值2....)

模糊查询

select 列名 from 表名 where 列名 like 条件;

%表示0个或者多个任意字符

_ 表示任意1个字符

统计该表中的数据总条数

count(列名/)

统计该列中所有列值的累加之和

sum(列名)

统计该列中,所有列值的平均值

avg(列名)

统计该列中最小值

min(列名)

统计该列中最大值

max(列名)

什么是分组查询?

根据某一列，把数据分成几组，可以对每一组数据使用聚合

语法格式:

select 列名/分组函数 from 表名

where 条件

group by 列名

order by 列名/分组函数 asc/desc;

请介绍下having语句?

对分组以后的数据再次进行过滤，经常和分组函数一起使用

having与where的区别是什么?

where 针对整张表数据进行过滤

having 针对分组后的数据进行过滤

where 可以单独使用

having 必须和group by 一起使用

having语法格式如何书写?

select 列名/分组函数 from 表名

where 条件 group by 列名 having 条件

order by 列名/分组函数 asc/desc;

常用的字符处理函数有哪些?

length(列名/字符串)统计出列值/字符串字符的个数

trim(列名/字符串)去掉列值或者字符串两端的空格

substring(参数1,参数2,参数3)用于字符串的截取参数1 要出处理的字符串参数2 从哪里开始截取 如果是正数表示从正数第几个开始截取(从1开始)，如果是负数，表示从倒数第几个开始截取参数3 表示截取字符的个数

reverse(列名/字符串)字符串逆序函数

concat(字符串/列名,字符串/列名)合并字符串

常用的数值处理函数有哪些?

round(数值，位数)四舍五入函数，如果位数大于0，表示小数点后保留几位，如果位数等于0，表示不保留小 数，如果位数小于0，表示小数点之前第几位进行四舍五入 truncate(数值,位数)截取函数，如果位数>0，表示保留小数点后几位，如果位数等于0，表示舍弃小数点后 所有的数字，如果位数小于0，表示舍弃小数点前第几位
rand(n)随机数函数随机生成0~1之间的数值，如果n值固定，随机数固定不变
sqrt(n)平方根函数 mod(x,y)余数函数 ifnull(n,m)处理null函数

常用的日期/时间处理函数有哪些?

curdate()获取当前系统日期 select curdate();
curtime()获取当前系统时间 select curtime();
sysdate()/now()获取当前系统时期时间 select sysdate(),now();
year(date)获取年份
month(date)获取月份
date_format(参数1，参数2)将数据库的日期转换为对应的字符串格式，参数1合法的日期格式，参数2:规 定日期/时间的输出格式
str_to_date 将时间格式的字符串，按照所提供的显示格式转换为日期类型的值，经常用于插入语句

查询昨天、今天、明天的日期

select curdate()-1,curdate(),curdate()+1;

什么是连接查询?

由于要查询的数据，分布在不同的表中，为了一次获取不同表中的数据，就需要使用连接查询

内连接查询

只查询满足条件的记录
格式:
select 别名1./列名1,别名2./列名2 ..... from 表1 别名1，表2 别名2 ....
where 关联条件;

外连接查询

1)左外连接查询以左边的表为主，查询左表中所有的数据以及与之关联的右表中的数据
格式:select 别名1./列名，别名2./列名 from 左表别名1 left join 右表别名2 on 关联条件;
2)右外连接查询以右边的表为主，查询右表中所有的数据，以及与之关联左表的数据
格式:select 别名1./列名,别名2./列名 from 左表别名1 right join 右表别名2 on 关联条件;

请介绍一下子查询

在查询内部又包含有个查询语句
1)any 关键字子查询
any表示满足其中任意一条件，就返回查询结果
2)all关键字子查询
all表示同时满足所有条件，就返回查询结果
3)exists关键字子查询
exists判断是否有结果，如果有，将继续查询
4)in 关键字子查询 比较多个结果时，使用in

请简单的描述一下什么是约束

对创建的表设置一些规则，只有满足这些规则，才可以插入数据，我们把这些规则叫做约束

常见的约束有哪些

主键约束 primary key
唯一约束 unique 默认值约束 default 非空约束 not null

请描述一下主键约束

主键约束就是用来标识表中的数据，避免表中出现重复的数据，被主键约束所修饰的列，该列的列值必须是唯
一且非空的，一张表中只能由一个主键约束，在创建表的时候，添加主键约束

请介绍一下什么是联合主键

使用主键约束来修饰2个列或者多个列的组合值，叫做联合主键也称为复合主键

在修改表时，如何添加主键约束，格式应该如何编写

alter table 表名 add primary key(列名1，列名2.....);

删除主键约束的格式如何编写?

alter table 表名 drop primary key;

请介绍一下唯一约束

用来指定一个列或者几个列的组合值，使其具有唯一性，防止用户在该列上输入重复的列值。

修改表时,如何添加唯一约束?

alter table 表名 add unique(列名1,列名2....);

删除唯一约束的格式是?

alter table 表名 drop index key_ame;

如何查看表中的key_mane

show keys from 表名;

请介绍下默认值约束

用来表示，在某列指定默认值，当执行插入操作的时候，如果该列没有插入列值，系统会自动把默认值作为该列的列值。每个列只能指定一个默认值

修改表时,如何添加默认值约束?

alter table 表名 modify 列名数据类型 default 默认值;

删除默认值约束

alter table 表名 modify 列名数据类型 default null;

请介绍一下非空约束

当执行插入操作的时候，被非空约束所修饰的列，该列的列值不能为null

修改表的时候,如何添加非空约束

alter table 表名 modify 列名数据类型 not null;

删除非空约束的格式是?

alter table 表名 modify 列名数据类型;

请介绍外键约束

用来维护两张表之间的关联关系，被外键约束所修饰的列，该列的列值必须参照与之关联另外一张表中主键所在的列的列值 父表/主表没有外键的表。子表，存在外键的表，被外键所修饰的列的列值，必须参照于主表的主键所在的列的 列值

修改表时,添加外键约束

alter table 子表 add foreign key(列名) references 主表(列名);

简单的说一下你对索引的了解?

索引是建立在表中列上的数据库对象，用于提高查询的速度，索引是一种提高查询效率的机制

创建索引的格式?

create index 索引名 on 表名(别名);

如何查看表中的索引

show index from 表名;

在修改表时如何添加索引?

alter table 表名 add index 索引名称(列名(长度))

如何删除索引

drop index 索引名 on 表名;

请说一下索引的优缺点

索引的优点:使用索引可以最大化最高查询的速度

索引的缺点:创建索引会占用磁盘物理存储空间，而且建立索引虽然能提高数据查询的速度，但是会减慢数据修 改

请简单介绍一下视图?

试图就是一个虚拟表,通过试图可以查看一个或多个表中数据

创建视图

create view 视图名称 as 查询语句;

修改视图

alter view 视图名称 as 查询语句;

创建存储过程

delimier // create procedure 存储过程名称 begin

执行 SQL语句 end // delimiter;

call 存储过程名称

查看存储过程

show procedure status like 条件;

删除存储过程

drop procedure 存储过程名称;

创建普通用户

create user'用户名'@'localhost'indentified by '密码';

修改用户密码

update 表名 set password=password("密码") where User ="用户名" and host="localhost"

删除用户 ·

drop user '用户名'@'localhost';

数据备份

mysqldump -u user -p 数据库名称> 备份名称带后缀文件.sql

数据恢复

mysql -u user -p 数据库名称< 备份名称带后缀文件.sql

什么是主键、外键、超键、候选键

超键：在关系中能唯一标识元组的属性集称为关系模式的超键。一个属性可以为作为一个超键，多个属性组合在一起也可以作为一个超键。超键包含候选键和主键。

候选键：是最小超键，即没有冗余元素的超键。

主键：数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。一个数据列只能有一个主键，且主键的取值不能缺失，即不能为空值（Null）。

外键：在一个表中存在的另一个表的主键称此表的外键

为什么用自增列作为主键

如果我们定义了主键(PRIMARY KEY)，那么InnoDB会选择主键作为聚集索引、 如果没有显式定义主键，则InnoDB会选择第一个不包含有NULL值的唯一索引作为主键索引、 如果也没有这样的唯一索引，则InnoDB会选择内置6字节长的ROWID作为隐含的聚集索引(ROWID随着行记录 的写入而主键递增，这个ROWID不像ORACLE的ROWID那样可引用，是隐含的)。 数据记录本身被存于主索引（一颗B+Tree）的叶子节点上。这就要求同一个叶子节点内（大小为一个内存页或 磁盘页）的各条数据记录按主键顺序存放，因此每当有一条新的记录插入时，MySQL会根据其主键将其插入适 当的节点和位置，如果页面达到装载因子（InnoDB默认为15/16），则开辟一个新的页（节点） 如果表使用自增主键，那么每次插入新的记录，记录就会顺序添加到当前索引节点的后续位置，当一页写满， 就会自动开辟一个新的页 如果使用非自增主键（如果身份证号或学号等），由于每次插入主键的值近似于随机，因此每次新纪录都要被 插到现有索引页得中间某个位置，此时MySQL不得不为了将新记录插到合适位置而移动数据，甚至目标页面可 能已经被回写到磁盘上而从缓存中清掉，此时又要从磁盘上读回来，这增加了很多开销，同时频繁的移动、分 页操作造成了大量的碎片，得到了不够紧凑的索引结构，后续不得不通过OPTIMIZE TABLE来重建表并优化填 充页面。

触发器的作用

触发器是一种特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和
一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上 包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

什么是存储过程？用什么来调用

存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以
调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL语句执行要快。 调用：

1）可以用一个命令对象来调用存储过程。

2）可以供外部程序调用，比如：java程序。

什么叫视图？游标是什么？

视图：

是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多 个表的行或列的子集。对视图的修改会影响基本表。它使得我们获取数据更容易，相比多表查询。

游标： 是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行
检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十 分重要。

视图的优缺点

优点：

1对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。

2)用户通过简单的查询可以从复杂查询中得到结果。

3)维护数据的独立性，试图可从多个表检索数据。

4)对于相同的数据可产生不同的视图。

缺点：

性能：查询视图时，必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，那么就无法更改数据

视图和表的区别和联系

视图是已经编译好的sql语句，而表不是;

视图没有实际的物理记录，而表有;

表是内容，视图是窗口;

视图是在基本表之上建立的表，它的结构和内容都来自基本表，它依据基本表存在而存在。一个视图可以对应一个基本表，也可以对应多个基本表。视图是基本表的抽象和在逻辑意义上建立的新关系

drop、truncate、 delete区别

最基本：

drop直接删掉表。 truncate删除表中数据，再插入时自增长id又从1开始。 delete删除表中数据，可以加where字句。

（1） DELETE语句执行删除的过程是每次从表中删除一行，并且同时将该行的删除操作作为事务记录在日志 中保存以便进行进行回滚操作。TRUNCATE TABLE则一次性地从表中删除所有的数据并不把单独的删除操作 记录记入日志保存，删除行是不能恢复的。并且在删除的过程中不会激活与表有关的删除触发器。执行速度 快。

（2）表和索引所占空间。当表被TRUNCATE 后，这个表和索引所占用的空间会恢复到初始大小，而DELETE操 作不会减少表或索引所占用的空间。drop语句将表所占用的空间全释放掉。

（3）一般而言，drop > truncate > delete

（4）应用范围。TRUNCATE 只能对TABLE；DELETE可以是table和view

（5） TRUNCATE 和DELETE只删除数据，而DROP则删除整个表（结构和数据）。

（6） truncate与不带where的delete： 只删除数据，而不删除表的结构（定义）drop语句将删除表的结 构被依赖的约束（constrain),触发器（trigger)索引（index);依赖于该表的存储过程/函数将被保留， 但其状态会变为：invalid。

（7） delete语句为DML（data maintain Language),这个操作会被放到 rollback segment中,事 务提交后才生效。如果有相应的 tigger,执行的时候将被触发。

（8） truncate、drop是DLL（data define language),操作立即生效，原数据不放到 rollback segment中，不能回滚。

（9）在没有备份情况下，谨慎使用 drop 与 truncate。要删除部分数据行采用delete且注意结合where 来约束影响范围。回滚段要足够大。要删除表用drop;若想保留表而将表中数据删除，如果于事务无关，用 truncate即可实现。如果和事务有关，或老师想触发trigger,还是用delete。

（10） Truncate table 表名速度快,而且效率高,因为:?truncate table 在功能上与不带 WHERE 子 句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的 系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

（11） TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的 计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。

（12）对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的
DELETE 语句。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器。

什么是临时表，临时表什么时候删除?

临时表可以手动删除：

DROP TEMPORARY TABLE IF EXISTS temp_tb;

临时表只在当前连接可见，当关闭连接时，MySQL会自动删除表并释放所有空间。因此在不同的连接中可以创 建同名的临时表，并且操作属于本连接的临时表。 创建临时表的语法与创建表语法类似，不同之处是增加关键字TEMPORARY，

如：

CREATE TEMPORARY TABLE tmp_table (NAME VARCHAR (10) NOT NULL,

time date NOT NULL

);

select * from tmp_table;

非关系型数据库和关系型数据库区别，优势比较

非关系型数据库的优势：

性能：NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以 性能非常高。

可扩展性：同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。 关系型数据库的优势：

复杂查询：可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。 事务支持：使得对于安全性能很高的数据访问要求得以实现。

其他：

1.对于这两类数据库，对方的优势就是自己的弱势，反之亦然。

2.NOSQL数据库慢慢开始具备SQL数据库的一些复杂查询功能，比如MongoDB。

3.对于事务的支持也可以用一些系统级的原子操作来实现例如乐观锁之类的方法来曲线救国，比如Redis set nx

数据库范式，根据某个场景设计数据表

第一范式:(确保每列保持原子性)所有字段值都是不可分解的原子值。

第一范式是最基本的范式。如果数据库表中的所有字段值都是不可分解的原子值，就说明该数据库表满足了第 一范式。 第一范式的合理遵循需要根据系统的实际需求来定。比如某些数据库系统中需要用到“地址”这个属性，本来直 接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么 就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操 作的时候将非常方便。这样设计才算满足了数据库的第一范式，如下表所示。 上表所示的用户信息遵循了第一范式的要求，这样在对用户使用城市进行分类的时候就非常方便，也提高了数 据库的性能。 第二范式:(确保表中的每列都和主键相关)在一个数据库表中，一个表中只能保存一种数据，不可以把多种数 据保存在同一张数据库表中。 第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与 主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中，一个表中只能保存一种数据， 不可以把多种数据保存在同一张数据库表中。 比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联 合主键。 第三范式:(确保每列都和主键列直接相关,而不是间接相关)数据表中的每一列数据都和主键直接相关，而不能 间接相关。

第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关。 比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的关系。而不可以在订单 表中添加关于客户其它信息（比如姓名、所属公司等）的字段。 BCNF:符合3NF，并且，主属性不依赖于主属性。 若关系模式属于第二范式，且每个属性都不传递依赖于键码，则R属于BC范式。 通常BC范式的条件有多种等价的表述：每个非平凡依赖的左边必须包含键码；每个决定因素必须包含键码。 BC范式既检查非主属性，又检查主属性。当只检查非主属性时，就成了第三范式。满足BC范式的关系都必然 满足第三范式。 还可以这么说：若一个关系达到了第三范式，并且它只有一个候选码，或者它的每个候选码都是单属性，则该 关系自然达到BC范式。

一般，一个数据库设计符合3NF或BCNF就可以了。 第四范式:要求把同一表内的多对多关系删除。 第五范式:从最终结构重新建立原始结构。

什么是内连接、外连接、交叉连接、笛卡尔积等

内连接: 只连接匹配的行

左外连接: 包含左边表的全部行（不管右边的表中是否存在与它们匹配的行），以及右边表中全部匹配的行右外连接: 包含右边表的全部行（不管左边的表中是否存在与它们匹配的行），以及左边表中全部匹配的行 例如1：

SELECT a.,b. FROM luntan LEFT JOIN usertable as b ON a.username=b.username

例如2：

SELECT a.,b. FROM city as a FULL OUTER JOIN user as b ON a.username=b.username

全外连接: 包含左、右两个表的全部行，不管另外一边的表中是否存在与它们匹配的行。

交叉连接: 生成笛卡尔积 - 它不使用任何匹配或者选取条件，而是直接将一个数据源中的每个行与另一个数据 源的每个行都一一匹配

例如：

SELECT type,pub_name FROM titles CROSS JOIN publishers ORDER BY type

varchar和char的使用场景?

1.char的长度是不可变的，而varchar的长度是可变的。

定义一个char[10]和varchar[10]。 如果存进去的是‘csdn’,那么char所占的长度依然为10，除了字符‘csdn’外，后面跟六个空格，varchar就 立马把长度变为4了，取数据的时候，char类型的要用trim()去掉多余的空格，而varchar是不需要的。

2.char的存取数度还是要比varchar要快得多，因为其长度固定，方便程序的存储与查找。 char也为此付出的是空间的代价，因为其长度固定，所以难免会有多余的空格占位符占据空间，可谓是以空 间换取时间效率。

varchar是以空间效率为首位。

3.char的存储方式是：对英文字符（ASCII）占用1个字节，对一个汉字占用两个字节。

varchar的存储方式是：对每个英文字符占用2个字节，汉字也占用2个字节。

4.两者的存储数据都非unicode的字符数据。

like %和-的区别

通配符的分类:

%百分号通配符:表示任何字符出现任意次数(可以是0次).

_下划线通配符:表示只能匹配单个字符,不能多也不能少,就是一个字符.

like操作符: LIKE作用是指示mysql后面的搜索模式是利用通配符而不是直接相等匹配进行比较.

注意: 如果在使用like操作符时,后面的没有使用通用匹配符效果是和=一致的,SELECT * FROM products WHERE products.prod_name like '1000';

只能匹配的结果为1000,而不能匹配像JetPack 1000这样的结果.

%通配符使用: 匹配以"yves"开头的记录:(包括记录"yves") SELECT FROM products WHERE products.prod_name like 'yves%';

匹配包含"yves"的记录(包括记录"yves") SELECT FROM products WHERE products.prod_name like '%yves%';

匹配以"yves"结尾的记录(包括记录"yves",不包括记录"yves ",也就是yves后面有空格的记录,这里需要注意) SELECT FROM products WHERE products.prod_name like '%yves';

通配符使用: SELECT FROM products WHERE products.prod_name like '_yves'; 匹配结果为:像"yyves"这样记录.

SELECT FROM products WHERE products.prodname like 'yves'; 匹配结果为: 像"yvesHe"这 样的记录.(一个下划线只能匹配一个字符,不能多也不能少)

注意事项: 注意大小写,在使用模糊匹配时,也就是匹配文本时,mysql是可能区分大小的,也可能是不区分大小写的,这个 结果是取决于用户对MySQL的配置方式.如果是区分大小写,那么像YvesHe这样记录是不能被"yves "这样的 匹配条件匹配的.

注意尾部空格,"%yves"是不能匹配"heyves "这样的记录的. 注意NULL,%通配符可以匹配任意字符,但是不能匹配NULL,也就是说SELECT FROM products WHERE products.prod_name like '%;是匹配不到products.prod_name为NULL的的记录.

技巧与建议:

正如所见， MySQL的通配符很有用。但这种功能是有代价的：通配符搜索的处理一般要比前面讨论的其他搜索 所花时间更长。这里给出一些使用通配符要记住的技巧。 不要过度使用通配符。如果其他操作符能达到相同的目的，应该使用其他操作符。 在确实需要使用通配符时，除非绝对有必要，否则不要把它们用在搜索模式的开始处。把通配符置于搜索模式 的开始处，搜索起来是最慢的。

仔细注意通配符的位置。如果放错地方，可能不会返回想要的数.

count()、count(1)、count(column)的区别

count()对行的数目进行计算,包含NULL count(column)对特定的列的值具有的行数进行计算,不包含NULL值。 count()还有一种使用方式,count(1)这个用法和count()的结果是一样的。

性能问题:

1.任何情况下SELECT COUNT() FROM tablename是最优选择;

2.尽量减少SELECT COUNT() FROM tablename WHERE COL = ‘value’这种查询;

3.杜绝SELECT COUNT(COL) FROM tablename WHERE COL2 = ‘value’的出现。 如果表没有主键,那么count(1)比count()快。 如果有主键,那么count(主键,联合主键)比count()快。 如果表只有一个字段,count()最快。

count(1)跟count(主键)一样,只扫描主键。count()跟count(非主键)一样,扫描整个表。明显前者更快一些

最左前缀原则

多列索引:

ALTER TABLE people ADD INDEX lname_fname_age (lame,fname,age); 为了提高搜索效率，我们需要考虑运用多列索引,由于索引文件以B－Tree格式保存，所以我们不用扫描任何记 录，即可得到最终结果。 注：在mysql中执行查询时，只能使用一个索引，如果我们在lname,fname,age上分别建索引,执行查询时， 只能使用一个索引，mysql会选择一个最严格(获得结果集记录数最少)的索引。 最左前缀原则：顾名思义，就是最左优先，上例中我们创建了lname_fname_age多列索引,相当于创建了 (lname)单列索引，(lname,fname)组合索引以及(lname,fname,age)组合索引。

什么是索引?

何为索引:

数据库索引，是数据库管理系统中一个排序的数据结构，索引的实现通常使用B树及其变种B+树。 在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数 据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

索引的作用？它的优点缺点是什么

索引作用:

协助快速查询、更新数据库表中数据。 为表设置索引要付出代价的： 一是增加了数据库的存储空间；二是在插入和修改数据时要花费较多的时间(因为索引也要随之变动)

索引的优缺点

创建索引可以大大提高系统的性能（优点）:

1.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。

2.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。

3.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。

4.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。

5.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。

增加索引也有许多不利的方面(缺点):

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

2.索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

哪些列适合建立索引、哪些不适合建索引?

索引是建立在数据库表中的某些列的上面。在创建索引的时候，应该考虑在哪些列上可以创建索引，在哪些列上不能创建索引。 一般来说，应该在这些列上创建索引：

（1）在经常需要搜索的列上，可以加快搜索的速度；

（2）在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；

（3）在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；

（4）在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；

（5）在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时 间；

（6）在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。

对于有些列不应该创建索引：

（1）对于那些在查询中很少使用或者参考的列不应该创建索引。 这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引， 反而降低了系统的维护速度和增大了空间需求。

（2）对于那些只有很少数据值的列也不应该增加索引。 这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行 的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。

（3）对于那些定义为text, image和bit数据类型的列不应该增加索引。 这是因为，这些列的数据量要么相当大，要么取值很少。 (4)当修改性能远远大于检索性能时，不应该创建索引。 这是因为，修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改性能远远大于检索性能时，不应该创建索引。

什么样的字段适合建索引

唯一、不为空、经常被查询的字段

MySQL常见的三种存储引擎（InnoDB、MyISAM、MEMORY）的区别?

两种存储引擎的大致区别表现在：

1.InnoDB支持事务，MyISAM不支持，这一点是非常之重要。事务是一种高级的处理方式，如在一些列增删改 中只要哪个出错还可以回滚还原，而MyISAM就不可以了。

2.MyISAM适合查询以及插入为主的应用。

3.InnoDB适合频繁修改以及涉及到安全性较高的应用。

4.InnoDB支持外键，MyISAM不支持。

5.从MySQL5.5.5以后，InnoDB是默认引擎。

6.InnoDB不支持FULLTEXT类型的索引。

7.InnoDB中不保存表的行数，如select count() from table时，InnoDB需要扫描一遍整个表来计算有多少行，但是MyISAM只要简单的读出保存好的行数即可。注意的是，当count()语句包含where条件时 MyISAM也需要扫描整个表。

8.对于自增长的字段，InnoDB中必须包含只有该字段的索引，但是在MyISAM表中可以和其他字段一起建立联 合索引。

9.DELETE FROM table时，InnoDB不会重新建立表，而是一行一行的删除，效率非常慢。MyISAM则会重建 表。

10.InnoDB支持行锁（某些情况下还是锁整表，如 update table set a=1 where user like '%lee%'。

查询语句不同元素（where、jion、limit、group by、having等等）执行先后顺序

查询中用到的关键词主要包含六个，并且他们的顺序依次为 select--from--where--group by--having--order by 其中select和from是必须的，其他关键词是可选的，这六个关键词的执行顺序与sql语句的书写顺序并不是 一样的，而是按照下面的顺序来执行

from:需要从哪个数据表检索数据

where:过滤表中数据的条件

group by:如何将上面过滤出的数据分组 having:对上面已经分组的数据进行过滤的条件 select:查看结果集中的哪个列，或列的计算结果 order by :按照什么样的顺序来查看返回的数据

from后面的表关联，是自右向左解析而where条件的解析顺序是自下而上的 也就是说，在写SQL语句的时候，尽量把数据量小的表放在最右边来进行关联（用小表去匹配大表），而把能 筛选出小量数据的条件放在where语句的最左边（用小表去匹配大表）

使用explain优化sql和索引?

对于复杂、效率低的sql语句，我们通常是使用explain sql 来分析sql语句，这个语句可以打印出，语句的执行。这样方便我们分析，进行优化 table：显示这一行的数据是关于哪张表的 type：这是重要的列，显示连接使用了何种类型。从最好到最差的连接类型为const、eq_reg、ref、 range、index和ALL

all:full table scan ;MySQL将遍历全表以找到匹配的行；

index: index scan; index 和 all的区别在于index类型只遍历索引；

range：索引范围扫描，对索引的扫描开始于某一点，返回匹配值的行，常见与between ，等查询； ref：非唯一性索引扫描，返回匹配某个单独值的所有行，常见于使用非唯一索引即唯一索引的非唯一前缀进 行查找； eq_ref：唯一性索引扫描，对于每个索引键，表中只有一条记录与之匹配，常用于主键或者唯一索引扫描；

const，system：当MySQL对某查询某部分进行优化，并转为一个常量时，使用这些访问类型。如果将主键 置于where列表中，MySQL就能将该查询转化为一个常量。

possible_keys：显示可能应用在这张表中的索引。如果为空，没有可能的索引。可以为相关的域从WHERE 语句中选择一个合适的语句 key：实际使用的索引。如果为NULL，则没有使用索引。很少的情况下，MySQL会选择优化不足的索引。这 种情况下，可以在SELECT语句中使用USE INDEX（indexname）来强制使用一个索引或者用IGNORE INDEX

（indexname）来强制MySQL忽略索引 key_len：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 ref：显示索引的哪一列被使用了，如果可能的话，是一个常数 rows：MySQL认为必须检查的用来返回请求数据的行数 Extra：关于MySQL如何解析查询的额外信息。将在表4.3中讨论，但这里可以看到的坏的例子是Using temporary和Using filesort，意思MySQL根本不能使用索引，结果是检索会很慢。

MySQL慢查询怎么解决

slow_query_log 慢查询开启状态。

slow_query_log_file 慢查询日志存放的位置（这个目录需要MySQL的运行帐号的可写权限，一般设置为MySQL的数据存放目录）。

long_query_time 查询超过多少秒才记录。

mysql都有什么锁

MySQL有三种锁的级别：页级、表级、行级。

表级锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高,并发度最低。 行级锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低,并发度也最高。 页面锁：开销和加锁时间界于表锁和行锁之间；会出现死锁；锁定粒度界于表锁和行锁之间，并发度一般

什么是死锁?

死锁: 是指两个或两个以上的进程在执行过程中。因争夺资源而造成的一种互相等待的现象,若无外力作用,它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁,这些永远在互相等竺的进程称为死锁进程。 表级锁不会产生死锁.所以解决死锁主要还是针对于最常用的InnoDB。 死锁的关键在于：两个(或以上)的Session加锁的顺序不一致。 那么对应的解决死锁问题的关键就是：让不同的session加锁有次序

死锁的解决办法

1.查出的线程杀死 kill

SELECT trx_MySQL_thread_id FROM information_schema.INNODB_TRX;

2.设置锁的超时时间

Innodb 行锁的等待时间，单位秒。可在会话级别设置，RDS 实例该参数的默认值为50（秒）。 生产环境不推荐使用过大的 innodb_lock_wait_timeout参数值 该参数支持在会话级别修改，方便应用在会话级别单独设置某些特殊操作的行锁等待超时时间，如下： set innodb_lock_wait_timeout=1000; —设置当前会话 Innodb 行锁等待超时时间，单位秒。

3.指定获取锁的顺序

在一个查询语句中，使用哪个关键字可以去掉重复列

distinct