MySQL

MySQL优化的衡量指标

- TPS

  Transaction Per Second（每秒传输的事务处理个数），指服务器每秒处理的事务数。只适用于InnoDB存储引擎。

  TPS=(COM_COMMIT+COM_ROLLBACK)/UPTIME（服务器启动时间）

- QPS

  Queries Per Second（每秒查询处理量）同时适用于MyiSAM和InnoDB引擎。

  QPS=QUESTIONS/UPTIME

- 响应时间

MySQLSlap（了解）

MySQL压力测试工具

- 创建schema、table、test data；
- 运行负载测试，可以模拟多个客户端并发连接；
- 测试环境清理。

MySQL一些细节

- MySQL是多用户数据库，缓存数据（默认关闭）和优化过的SQL语句（默认开启）。
- 缓存

  //数据缓存是否开启

  SHOW VARIABLES LIKE '%query_cache_type%'

  //数据缓存大小

  SHOW VARIABLES LIKE 'query_cache_size'

  //设置数据缓存大小

  SET GLOBAL query_cache_size= 1048576

  生产环境不开启。

存储引擎

- **MyISAM**
  - 特性：并发性与锁级别-表级锁 支持全文检索　支持数据压缩（压缩删除.OLD文件后，只能查询， check table XXX　repair table XXX 后可修改、插入）
  - 适用场景：

    非事务性应用（数据仓库、报表、日志数据）、只读类应用、空间类应用（空间函数、坐标）
  - 数据存储文件

    tablename.frm：存储表结构，任何存储引擎都具备的

    tablename.myd：数据库文件

tablename.myi：索引文件

非聚集索引

- **InnoDB** （MySQL5.5级以后版本默认引擎）
  - 特性：支持事务、支持行级锁（并发程度更高）、支持外键
- 适用场景：InnoDB适合于大多数OLTP（ 联机事务处理 ）应用
- 数据存储文件

  tablename.frm：存储表结构，任何存储引擎都具备的

  tablename.ibd：独立表空间存储的数据+索引

  ibdataX ： 系统表空间存储的数据+索引
- 独立表空间和系统表空间

  innodb_file_per_table

  ON:独立的表空间 tablename.ibd：存储的数据+索引

  OFF:系统表空间 ibdataX

  5.6之前默认为系统表空间

  独立表空间优点：
  - 系统表空间删除数据后，无法简单的收缩文件大小（删除数据后，未释放空间）
  - 独立表空间删除数据后，可以通过optimize table收缩系统文件
  - 系统表空间会产生IO瓶颈（所有表公用一个系统文件）
  - 独立表空间可以同时向多个文件数显数据

  建议：InnoDB使用独立表空间

对比项	MyISAM	InnoDB
主外键	不支持	支持
事务	不支持	支持
行表锁	表锁：即使操作一条记录，也会 锁住整个表 不适合高并发操作	行锁：操作时只锁某一行， 不对其他行有影响 适合高并发操作
缓存	只缓存索引，不缓存真实数据	不仅缓存索引，还缓存数据， 对内存要求较高
表空间	小	大
关注点	性能	事务
默认安装	Y	Y

MySQL锁

锁的简介

- 为什么需要锁

  使用锁对有限资源进行保护，解决隔离和并发的矛盾。
- 锁的概念
  - 锁是计算机协调多个进程或线程并发访问某一资源的机制。

    - 在数据库中，数据也是一种供许多用户共享的资源。如何保证数据并发访问的一致性、有效性是所有数据库必须解决的一个问题。锁冲突也是影响数据库并发访问性能的一个重要因素。
- MySQL中的锁
    - 表级锁：开销小，加锁快；不会出现死锁；锁定颗粒度大，发生锁冲突的概率最高，并发度最低。

      适合于以查询为主，只有少量按索引条件更新数据库的应用。
    - 行级锁：开销大，加锁慢；会出现死锁；锁定颗粒度小，发生锁冲突的概率最低，并发度最高。

      适用于有大量按索引条件并发更新少量不同数据，同时又有并发数据查询的应用。
    - 页面锁：开销和加锁时间介于表锁和行锁之间；会出现死锁；锁定颗粒度介于表锁和行锁之间，并发度一般。

MyISAM表锁

- **表共享读锁（Table Read Lock)**

  创建读锁的语法：LOCK TABLE material_input_myisam read

  1.同一个事务：
    - 在同一个session中查询该表 ok
    - 在同一个session中增、删、改 报错

      INSERT into material_input_myisam (id) VALUE (100)
      DELETE FROM material_input_myisam WHERE id=2
      UPDATE material_input_myisam set id=99 WHERE id=3
    - 在同一个session中操作其他表 报错

      INSERT into material_input_myisam_copy (id) VALUE (100)

  2.不同事务：
    - 在另外一个session中查询该表 ok

      INSERT into material_input_myisam (id) VALUE (100)
    - 在另外一个session中新增该表 等待

      INSERT into material_input_myisam_copy (id) VALUE (100)

  3.别名锁
    - lock table 表名 as 别名 read
      lock table material_input_myisam as myisam read
      SELECT myisam.* FROM material_input_myisam myisam

  4.解锁

  UNLOCK TABLE

  5.查看被锁过几次

  show status LIKE 'table_locks_readed'

- **表共享写锁（Table Write Lock)**

  创建写锁的语法：LOCK TABLE material_input_myisam write

  1.同一个事务：
    - 同一个session中增删改查该表 ok
    - 对不同的表操作 报错

2.不同事务：

- 不同session中，增删改查 等待

- **总结**
  - 对MyISAM表的读操作，不会阻塞其用户对同一表的读请求，但会阻塞对同一表的写请求。
  - 对MyISAM表的读操作，不会阻塞当前session对其他表的读，但当对其他表修改时会报错。
  - 一个session使用LOCK TABLE命令给表加了读锁，这个session可以查询锁定表中的记录，但更新或访问其他表会报错。
  - 对MyISAM表的读操作，另一个session可以查询表的记录，但更新就会出现等待。
  - 对MyISAM表的写操作，会阻塞其他用户对同一表的读和写操作。
  - 对MyISAM表的写操作，当前session可以对本表做增删改查，但对其他表进行操作会报错。

InnoDB行锁

- **锁类型**
  - 共享锁：读锁。当一个事务对某几行上读锁时，允许其他事务对这几行进行读操作，但不允许其他行进行写操作，也不允许其他事务给这几行上排它锁，但允许上读锁。

    lock in share mode

    select * from table_name where 条件 lock in share mode

  - 排它锁：写锁。当一个事务对某几行上写锁是，不允许其他事务写，但允许读。更不允许其他事务给这几行上任何锁。

    for update

    select * from table_name where 条件 for update

- **注意**
  - 两个事务不能锁同一个索引。
  - insert，delete，update在事务中默认加排它锁。
  - 行锁必须有索引才能实现，否则自动锁全表。
  - 开启一个事务的时候会解表锁。

物理结构修改

数据量、并发量很大的一个表修改表结构

1. 创建与原表结构相同的新表
2. 修改新表结构
3. 原表创建触发器
4. 原表数据copy到新表，copy过程中，触发器将原表更新数据，全部更新到新表
5. copy完后，用新表代替原表

MySQL事务

1.事务的简介

- 什么是事务

  数据库事务（transaction）：是指作为单个逻辑工作单元执行的一系列数据库操作，要么全部执行，要么全部不执行。

- 为什么需要事务

  许多软件都是多用户、多线程的，对同一个表可能有很多用户在使用，为了保持数据的一致性，所以提出事务的概念。

2.事务的特性

- **事务具有4个特性：原子性、一致性、隔离性、持久性，成为ACID特性。**

1. 原子性（atomicity）：

   原子性是指事务包含的全部数据库操作，要么全部成功，要么全部失败回滚。不存在中间状态。

2. 一致性（consistency）：

   事务必须是使数据库从一个一致性状态转变到另一个一致性状态。

3. 持久性（durability）：

   持久性是指一个事务一旦提交了，那么对数据库的操作是永久的，即便数据库遇到故障，提交的事务也不会丢失。

4. **隔离性（isolation）：**

   隔离性是指多个用户并发访问数据库时，数据库为每一个请求开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

   事务隔离性的四个级别：

   - 未提交读（READ UNCOMMITED）
   - 已提交读（READ COMMITED）
   - 可重复读（REPEATABLE READ）
   - 可串行化（SERIALIZABLE）

- **事务并发问题**
  - 脏读（读取未提交数据）：事务A读取了事务B提交的数据，然后B回滚操作，那么A读到的数据就是脏数据。
  - 不可重复读（前后多次读取，内容不一致）：事务A多次多次读取同一数据，事务B在A多次读取的过程中，对数据做了更新并提交，导致事务A多次读到的数据不一致。
  - 幻读（前后多次读取，数据总量不一致）：事务A在执行读取操作，需要两次统计数据的总量，第一次查询数据总量后，事务B执行了新增/删除数据操作并提交，导致事务A第二次读取到的数据总量和第一次统计的不一样，称为幻读。

  不可重复读和幻读容易混淆：

  - 不可重复读侧重的修改数据（update），解决此问题需要行锁，锁住满足条件的行；
  - 幻读侧重于删除或增加（delete或insert），解决幻读需要锁表。
- **总结**

事务隔离级别	脏读	不可重复度	幻读
未提交读	是	是	是
已提交读	否	是	是
可重复度	否	否	是
可串行化	否	否	否

- 事务隔离级别为可重复度时，如果有索引的时候，以索引列为条件更新数据，会存在间隙、行锁、页锁的问题，从而锁住一些行；如果没有索引，更新数据会锁住整张表。
- 事务隔离级别为可串行化时，读写数据都会锁住整张表。
- 事务隔离级别越高，越能保证数据的完整性和一致性，但对并发性能影响越大，多数应用程序，事务隔离级别设置为读已提交，能避免脏读，具有较好的并发性。

3.事务的语法

- 开启事务

  begin

  start transaction（推荐）

  begin work
- 事务回滚

  rollback
- 事务提交

  commite
- 还原点

  savepoint

  start transaction

  insert into testdemo values(5,5,5);

  savepoint s1;

  insert into testdemo values(6,6,6);

  savepoint s2;

  insert into testdemo values(7,7,7);

  savepoint s3;

  rollback to savepoint s2

业务设计

1.逻辑设计

- **范式设计**
  - 第一大范式

    数据库表中所有字段都只具有单一属性

    单一属性的列是由基本数据类型组成的

    设计出来的表都是简单额二维表
  - 第二大范式

    表中只具有一个主键
  - 第三大范式

    指每一个非主键属性既不部分依赖于也不传递依赖于业务主键
- **范式化设计优缺点**
  - 优点：
    - 尽量减少数据冗余
    - 范式化的更新操作比反范式化要快
    - 范式化设计的表通常比反范式化设计的表小
  - 缺点
    - 对于查询需要多个表进行关联
    - 更难进行索引优化
- **反范式设计**
  - 反范式化设计是相对于范式化设计而言的。

    - 反范式化设计是为了性能和读取效率考虑，而适当的对范式化设计规则进行违反。
    - 允许少量冗余，空间换时间。
- **反范式化设计优缺点**
    - 优点：减少表的关联，更好的进行索引优化。
    - 缺点：存在数据冗余，增加数据修改的成本。

2.物理设计

- **定义数据库、表、字段的命名规范**：可读性原则、示意性原则、长名原则（不适应缩写）
- **选择合适的存储引擎**

对比项	MyISAM	InnoDB
主外键	不支持	支持
事务	不支持	支持
行表锁	表锁：即使操作一条记录，也会 锁住整个表 不适合高并发操作	行锁：操作时只锁某一行， 不对其他行有影响 适合高并发操作
缓存	只缓存索引，不缓存真实数据	不仅缓存索引，还缓存数据， 对内存要求较高
表空间	小	大
关注点	性能	事务
默认安装	Y	Y

- **为表字段选择合适的数据类型**

Mysql 的存储引擎,myisam和innodb的区别。

1.MyISAM 是非事务的存储引擎，适合用于频繁查询的应用。表锁，不会出现死锁，适合小数据，小并发。

2.innodb是支持事务的存储引擎，合于插入和更新操作比较多的应用，设计合理的话是行锁（最大区别就在锁的级别上），适合大数据，大并发。

数据表类型有哪些

答：MyISAM、InnoDB、HEAP、BOB,ARCHIVE,CSV等。
MyISAM：成熟、稳定、易于管理，快速读取。一些功能不支持（事务等），表级锁。
InnoDB：支持事务、外键等特性、数据行锁定。空间占用大，不支持全文索引等。

MySQL数据库作发布系统的存储，一天五万条以上的增量，预计运维三年,怎么优化?

a. 设计良好的数据库结构，允许部分数据冗余，尽量避免join查询，提高效率。
b. 选择合适的表字段数据类型和存储引擎，适当的添加索引。
c. mysql库主从读写分离。
d. 找规律分表，减少单表中的数据量提高查询速度。
e。添加缓存机制，比如memcached，apc等。

f. 不经常改动的页面，生成静态页面。
g. 书写高效率的SQL。比如 SELECT * FROM TABEL 改为 SELECT field_1, field_2, field_3 FROM TABLE.

对于大流量的网站,您采用什么样的方法来解决各页面访问量统计问题?

答：a. 确认服务器是否能支撑当前访问量。
b. 优化数据库访问。
c. 禁止外部访问链接（盗链）, 比如图片盗链。
d. 控制文件下载。
e. 使用不同主机分流。
f. 使用浏览统计软件，了解访问量，有针对性的进行优化。

如何进行SQL优化?

答：
（1）选择正确的存储引擎
以 MySQL为例，包括有两个存储引擎 MyISAM 和 InnoDB，每个引擎都有利有弊。
MyISAM 适合于一些需要大量查询的应用，但其对于有大量写操作并不是很好。甚至你只是需要update一个字段，整个表都会被锁起来，而别的进程，就算是读进程都无法操作直到读操作完成。另外，MyISAM 对于 SELECT COUNT(*) 这类的计算是超快无比的。

InnoDB 的趋势会是一个非常复杂的存储引擎，对于一些小的应用，它会比 MyISAM 还慢。但是它支持“行锁”，于是在写操作比较多的时候，会更优秀。并且，他还支持更多的高级应用，比如：事务。

（2）优化字段的数据类型

记住一个原则，越小的列会越快。如果一个表只会有几列罢了（比如说字典表，配置表），那么，我们就没有理由使用 INT 来做主键，使用 MEDIUMINT, SMALLINT 或是更小的 TINYINT 会更经济一些。如果你不需要记录时间，使用 DATE 要比 DATETIME 好得多。当然，你也需要留够足够的扩展空间。

（3）为搜索字段添加索引

索引并不一定就是给主键或是唯一的字段。如果在你的表中，有某个字段你总要会经常用来做搜索，那么最好是为其建立索引，除非你要搜索的字段是大的文本字段，那应该建立全文索引。

(4)避免使用Select *从数据库里读出越多的数据，那么查询就会变得越慢。并且，如果你的数据库服务器和WEB服务器是两台独立的服务器的话，这还会增加网络传输的负载。即使你要查询数据表的所有字段，也尽量不要用*通配符，善用内置提供的字段排除定义也许能给带来更多的便利。

(5)使用 ENUM 而不是 VARCHAR

ENUM 类型是非常快和紧凑的。在实际上，其保存的是 TINYINT，但其外表上显示为字符串。这样一来，用这个字段来做一些选项列表变得相当的完美。例如，性别、民族、部门和状态之类的这些字段的取值是有限而且固定的，那么，你应该使用 ENUM 而不是 VARCHAR。

(6)尽可能的使用 NOT NULL

除非你有一个很特别的原因去使用 NULL 值，你应该总是让你的字段保持 NOT NULL。 NULL其实需要额外的空间，并且，在你进行比较的时候，你的程序会更复杂。 当然，这里并不是说你就不能使用NULL了，现实情况是很复杂的，依然会有些情况下，你需要使用NULL值。

(7)固定长度的表会更快

如果表中的所有字段都是“固定长度”的，整个表会被认为是 “static” 或 “fixed-length”。 例如，表中没有如下类型的字段： VARCHAR，TEXT，BLOB。只要你包括了其中一个这些字段，那么这个表就不是“固定长度静态表”了，这样，MySQL 引擎会用另一种方法来处理。

固定长度的表会提高性能，因为MySQL搜寻得会更快一些，因为这些固定的长度是很容易计算下一个数据的偏移量的，所以读取的自然也会很快。而如果字段不是定长的，那么，每一次要找下一条的话，需要程序找到主键。

并且，固定长度的表也更容易被缓存和重建。不过，唯一的副作用是，固定长度的字段会浪费一些空间，因为定长的字段无论你用不用，他都是要分配那么多的空间。

如何设计一个高并发的系统

① 数据库的优化，包括合理的事务隔离级别、SQL语句优化、索引的优化

② 使用缓存，尽量减少数据库 IO

③ 分布式数据库、分布式缓存

④ 服务器的负载均衡

锁的优化策略

① 读写分离

② 分段加锁

③ 减少锁持有的时间

④ 多个线程尽量以相同的顺序去获取资源

等等，这些都不是绝对原则，都要根据情况，比如不能将锁的粒度过于细化，不然可能会出现线程的加锁和释放次数过多，反而效率不如一次加一把大锁。这部分跟面试官谈了很久

索引的底层实现原理和优化

B+树，经过优化的B+树

主要是在所有的叶子结点中增加了指向下一个叶子节点的指针，因此InnoDB建议为大部分表使用默认自增的主键作为主索引。

什么情况下设置了索引但无法使用

① 以“%”开头的LIKE语句，模糊匹配

② OR语句前后没有同时使用索引

③ 数据类型出现隐式转化（如varchar不加单引号的话可能会自动转换为int型）

SQL语句的优化

order by要怎么处理

alter尽量将多次合并为一次

insert和delete也需要合并

等等

实践中如何优化MySQL

我当时是按以下四条依次回答的，他们四条从效果上第一条影响最大，后面越来越小。

① SQL语句及索引的优化

② 数据库表结构的优化

③ 系统配置的优化

④ 硬件的优化

sql注入的主要特点

变种极多，攻击简单，危害极大

sql注入的主要危害

未经授权操作数据库的数据

恶意篡改网页

私自添加系统账号或者是数据库使用者账号

网页挂木马

优化数据库的方法

1. 选取最适用的字段属性，尽可能减少定义字段宽度，尽量把字段设置NOTNULL，例如'省份'、'性别'最好适用ENUM
2. 使用连接(JOIN)来代替子查询
3. 适用联合(UNION)来代替手动创建的临时表
4. 事务处理
5. 锁定表、优化事务处理
6. 适用外键，优化锁定表
7. 建立索引
8. 优化查询语句

简单描述mysql中，索引，主键，唯一索引，联合索引的区别，对数据库的性能有什么影响（从读写两方面）

索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。

普通索引(由关键字KEY或INDEX定义的索引)的唯一任务是加快对数据的访问速度。

普通索引允许被索引的数据列包含重复的值。如果能确定某个数据列将只包含彼此各不相同的值，在为这个数据列创建索引的时候就应该用关键字UNIQUE把它定义为一个唯一索引。也就是说，唯一索引可以保证数据记录的唯一性。

主键，是一种特殊的唯一索引，在一张表中只能定义一个主键索引，主键用于唯一标识一条记录，使用关键字 PRIMARY KEY 来创建。

索引可以覆盖多个数据列，如像INDEX(columnA, columnB)索引，这就是联合索引。

索引可以极大的提高数据的查询速度，但是会降低插入、删除、更新表的速度，因为在执行这些写操作时，还要操作索引文件。

数据库中的事务是什么?

事务（transaction）是作为一个单元的一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作完成，事务则提交，其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚，该事务所有操作的影响都将取消。ACID 四大特性,原子性、隔离性、一致性、持久性。

了解XSS攻击吗？如何防止?

XSS是跨站脚本攻击，首先是利用跨站脚本漏洞以一个特权模式去执行攻击者构造的脚本，然后利用不安全的Activex控件执行恶意的行为。
使用htmlspecialchars()函数对提交的内容进行过滤，使字符串里面的特殊符号实体化。

SQL注入漏洞产生的原因？如何防止?

SQL注入产生的原因：程序开发过程中不注意规范书写sql语句和对特殊字符进行过滤，导致客户端可以通过全局变量POST和GET提交一些sql语句正常执行。

防止SQL注入的方式：
开启配置文件中的magic_quotes_gpc 和 magic_quotes_runtime设置

执行sql语句时使用addslashes进行sql语句转换

Sql语句书写尽量不要省略双引号和单引号。

过滤掉sql语句中的一些关键词：update、insert、delete、select、 * 。

提高数据库表和字段的命名技巧，对一些重要的字段根据程序的特点命名，取不易被猜到的。

Php配置文件中设置register_globals为off,关闭全局变量注册

控制错误信息，不要在浏览器上输出错误信息，将错误信息写到日志文件中。

为表中得字段选择合适得数据类型（物理设计）

字段类型优先级: 整形>date,time>enum,char>varchar>blob,text
优先考虑数字类型，其次是日期或者二进制类型，最后是字符串类型，同级别得数据类型，应该优先选择占用空间小的数据类型

存储时期

Datatime:以 YYYY-MM-DD HH:MM:SS 格式存储时期时间，精确到秒，占用8个字节得存储空间，datatime类型与时区无关
Timestamp:以时间戳格式存储，占用4个字节，范围小1970-1-1到2038-1-19，显示依赖于所指定得时区，默认在第一个列行的数据修改时可以自动得修改timestamp列得值
Date:（生日）占用得字节数比使用字符串.datatime.int储存要少，使用date只需要3个字节，存储日期月份，还可以利用日期时间函数进行日期间得计算
Time:存储时间部分得数据
注意:不要使用字符串类型来存储日期时间数据（通常比字符串占用得储存空间小，在进行查找过滤可以利用日期得函数）
使用int存储日期时间不如使用timestamp类型

对于关系型数据库而言，索引是相当重要的概念，请回答有关索引的几个问题：

a)、索引的目的是什么?
快速访问数据表中的特定信息，提高检索速度

创建唯一性索引，保证数据库表中每一行数据的唯一性。

加速表和表之间的连接

使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间

b)、索引对数据库系统的负面影响是什么?
负面影响：
创建索引和维护索引需要耗费时间，这个时间随着数据量的增加而增加；索引需要占用物理空间，不光是表需要占用数据空间，每个索引也需要占用物理空间；当对表进行增、删、改、的时候索引也要动态维护，这样就降低了数据的维护速度。

c)、为数据表建立索引的原则有哪些?
在最频繁使用的、用以缩小查询范围的字段上建立索引。

在频繁使用的、需要排序的字段上建立索引

d)、 什么情况下不宜建立索引?
对于查询中很少涉及的列或者重复值比较多的列，不宜建立索引。

对于一些特殊的数据类型，不宜建立索引，比如文本字段（text）等

简述在MySQL数据库中MyISAM和InnoDB的区别

区别于其他数据库的最重要的特点就是其插件式的表存储引擎。切记：存储引擎是基于表的，而不是数据库。

InnoDB与MyISAM的区别：

InnoDB存储引擎: 主要面向OLTP(Online Transaction Processing，在线事务处理)方面的应用，是第一个完整支持ACID事务的存储引擎(BDB第一个支持事务的存储引擎，已经停止开发)。

特点：

· 行锁设计、支持外键,支持事务，支持并发，锁粒度是支持mvcc得行级锁；

MyISAM存储引擎: 是MySQL官方提供的存储引擎，主要面向OLAP(Online Analytical Processing,在线分析处理)方面的应用。
特点：

不支持事务，锁粒度是支持并发插入得表级锁，支持表所和全文索引。操作速度快，不能读写操作太频繁；

解释MySQL外连接、内连接与自连接的区别

先说什么是交叉连接: 交叉连接又叫笛卡尔积，它是指不使用任何条件，直接将一个表的所有记录和另一个表中的所有记录一一匹配。

内连接 则是只有条件的交叉连接，根据某个条件筛选出符合条件的记录，不符合条件的记录不会出现在结果集中，即内连接只连接匹配的行。
外连接 其结果集中不仅包含符合连接条件的行，而且还会包括左表、右表或两个表中的所有数据行，这三种情况依次称之为左外连接，右外连接，和全外连接。

左外连接，也称左连接，左表为主表，左表中的所有记录都会出现在结果集中，对于那些在右表中并没有匹配的记录，仍然要显示，右边对应的那些字段值以NULL来填充。右外连接，也称右连接，右表为主表，右表中的所有记录都会出现在结果集中。左连接和右连接可以互换，MySQL目前还不支持全外连接。

写出三种以上MySQL数据库存储引擎的名称（提示：不区分大小写）

MyISAM、InnoDB、BDB（BerkeleyDB）、Merge、Memory（Heap）、Example、Federated、Archive、CSV、Blackhole、MaxDB 等等十几个引擎

Myql中的事务回滚机制概述

事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位，事务回滚是指将该事务已经完成的对数据库的更新操作撤销。

要同时修改数据库中两个不同表时，如果它们不是一个事务的话，当第一个表修改完，可能第二个表修改过程中出现了异常而没能修改，此时就只有第二个表依旧是未修改之前的状态，而第一个表已经被修改完毕。而当你把它们设定为一个事务的时候，当第一个表修改完，第二表修改出现异常而没能修改，第一个表和第二个表都要回到未修改的状态，这就是所谓的事务回滚

SQL语言包括哪几部分？每部分都有哪些操作关键字？

答：SQL语言包括数据定义(DDL)、数据操纵(DML),数据控制(DCL)和数据查询（DQL）四个部分。

数据定义：Create Table,Alter Table,Drop Table, Craete/Drop Index等

数据操纵：Select ,insert,update,delete,

数据控制：grant,revoke

数据查询：select

完整性约束包括哪些?

答：数据完整性(Data Integrity)是指数据的精确(Accuracy)和可靠性(Reliability)。

分为以下四类：

1) 实体完整性：规定表的每一行在表中是惟一的实体。

2) 域完整性：是指表中的列必须满足某种特定的数据类型约束，其中约束又包括取值范围、精度等规定。

3) 参照完整性：是指两个表的主关键字和外关键字的数据应一致，保证了表之间的数据的一致性，防止了数据丢失或无意义的数据在数据库中扩散。

4) 用户定义的完整性：不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性即是针对某个特定关系数据库的约束条件，它反映某一具体应用必须满足的语义要求。

与表有关的约束：包括列约束(NOT NULL（非空约束）)和表约束(PRIMARY KEY、foreign key、check、UNIQUE) 。

什么是事务？及其特性?

答：事务：是一系列的数据库操作，是数据库应用的基本逻辑单位。

事务特性：

（1）原子性：即不可分割性，事务要么全部被执行，要么就全部不被执行。

（2）一致性或可串性。事务的执行使得数据库从一种正确状态转换成另一种正确状态

（3）隔离性。在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事务，

（4） 持久性。事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

或者这样理解：

事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过ACID测试，即原子性，一致性，隔离性和持久性。

什么是锁?

答：数据库是一个多用户使用的共享资源。当多个用户并发地存取数据时，在数据库中就会产生多个事务同时存取同一数据的情况。若对并发操作不加控制就可能会读取和存储不正确的数据，破坏数据库的一致性。

加锁是实现数据库并发控制的一个非常重要的技术。当事务在对某个数据对象进行操作前，先向系统发出请求，对其加锁。加锁后事务就对该数据对象有了一定的控制，在该事务释放锁之前，其他的事务不能对此数据对象进行更新操作。

基本锁类型：锁包括行级锁和表级锁

什么叫视图？游标是什么？

答：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，视图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。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。

什么是存储过程？用什么来调用？

答：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。可以用一个命令对象来调用存储过程。

索引的作用？和它的优点缺点是什么？

答：索引就一种特殊的查询表，数据库的搜索引擎可以利用它加速对数据的检索。它很类似与现实生活中书的目录，不需要查询整本书内容就可以找到想要的数据。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。

如何通俗地理解三个范式？

答：第一范式：1NF是对属性的原子性约束，要求属性具有原子性，不可再分解；

第二范式：2NF是对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；

第三范式：3NF是对字段冗余性的约束，即任何字段不能由其他字段派生出来，它要求字段没有冗余。。

范式化设计优缺点:

优点:

可以尽量得减少数据冗余，使得更新快，体积小

缺点:对于查询需要多个表进行关联，减少写得效率增加读得效率，更难进行索引优化

反范式化:

优点:可以减少表得关联，可以更好得进行索引优化

缺点:数据冗余以及数据异常，数据得修改需要更多的成本

什么是基本表？什么是视图？

答：基本表是本身独立存在的表，在 SQL 中一个关系就对应一个表。 视图是从一个或几个基本表导出的表。视图本身不独立存储在数据库中，是一个虚表

试述视图的优点?

答：(1) 视图能够简化用户的操作 (2) 视图使用户能以多种角度看待同一数据； (3) 视图为数据库提供了一定程度的逻辑独立性； (4) 视图能够对机密数据提供安全保护。

NULL是什么意思

答：NULL这个值表示UNKNOWN(未知):它不表示""(空字符串)。对NULL这个值的任何比较都会生产一个NULL值。您不能把任何值与一个 NULL值进行比较，并在逻辑上希望获得一个答案。

使用IS NULL来进行NULL判断

主键、外键和索引的区别?

主键、外键和索引的区别

定义：

主键–唯一标识一条记录，不能有重复的，不允许为空

外键–表的外键是另一表的主键, 外键可以有重复的, 可以是空值

索引–该字段没有重复值，但可以有一个空值

作用：

主键–用来保证数据完整性

外键–用来和其他表建立联系用的

索引–是提高查询排序的速度

个数：

主键–主键只能有一个

外键–一个表可以有多个外键

索引–一个表可以有多个唯一索引

你可以用什么来确保表格里的字段只接受特定范围里的值?

答：Check限制，它在数据库表格里被定义，用来限制输入该列的值。

触发器也可以被用来限制数据库表格里的字段能够接受的值，但是这种办法要求触发器在表格里被定义，这可能会在某些情况下影响到性能。

说说对SQL语句优化有哪些方法？（选择几条）

（1）Where子句中：where表之间的连接必须写在其他Where条件之前，那些可以过滤掉最大数量记录的条件必须写在Where子句的末尾.HAVING最后。

（2）用EXISTS替代IN、用NOT EXISTS替代NOT IN。

（3） 避免在索引列上使用计算

（4）避免在索引列上使用IS NULL和IS NOT NULL

（5）对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

（6）应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

（7）应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描

SQL语句中‘相关子查询’与‘非相关子查询’有什么区别?

答：子查询：嵌套在其他查询中的查询称之。

子查询又称内部，而包含子查询的语句称之外部查询（又称主查询）。

所有的子查询可以分为两类，即相关子查询和非相关子查询

（1）非相关子查询是独立于外部查询的子查询，子查询总共执行一次，执行完毕后将值传递给外部查询。

（2）相关子查询的执行依赖于外部查询的数据，外部查询执行一行，子查询就执行一次。

故非相关子查询比相关子查询效率高

char和varchar的区别?

答：是一种固定长度的类型，varchar则是一种可变长度的类型，它们的区别是：

char(M)类型的数据列里，每个值都占用M个字节，如果某个长度小于M，MySQL就会在它的右边用空格字符补足．（在检索操作中那些填补出来的空格字符将被去掉）在varchar(M)类型的数据列里，每个值只占用刚好够用的字节再加上一个用来记录其长度的字节（即总长度为L+1字节）．

varchar得适用场景:

字符串列得最大长度比平均长度大很多 2.字符串很少被更新，容易产生存储碎片 3.使用多字节字符集存储字符串

Char得场景:

存储具有近似得长度（md5值,身份证，手机号）,长度比较短小得字符串（因为varchar需要额外空间记录字符串长度），更适合经常更新得字符串，更新时不会出现页分裂得情况，避免出现存储碎片，获得更好的io性能