**数据库三大范式是什么**

第一范式：每个列都不可以再拆分。

第二范式：在第一范式的基础上，非主键列完全依赖于主键，而不是依赖于主键的一部分。(每个表中都要有一个主键，其他字段都依赖这个主键)

第三范式：在第二范式的基础上，非主键列只依赖于主键，不依赖于其他非主键。(其他字段不能相互依赖，只能依赖主键)

(在设计数据库结构的时候，要尽量遵守三范式，如果不遵守，必须有足够的理由。比如性能。事实上我们经常会为了性能而妥协数据库的设计。)

遵守的好处：

1. 减少数据冗余：避免重复存储相同数据；

2. 提高可维护性：数据结构清晰，修改数据时只需更新一处。

3. 增强扩展性：新增字段或表时不影响现有结构

**SQL语句优化**

Insert用批量插入提升性能

表连接不宜太多，索引不宜太多，一般5个以内

联合索引，要遵守最左前缀法则

避免使用 or或者!=，可能会使索引失效，从而全表扫描

避免进行null值的判断，索引可能失效

避免在字段开头模糊查询，会导致全表扫描

使用数值代替字符串类型，字符会降低查询和连接的性能

使用varchar代替char，对于查询来说，在一个相对较小的字段内搜索，效率更高

使用union all代替union

只查询一条数据的时候，使用limit 1，SQL中加了limit 1，如果第一条就命中目标return， 没有limit的话，还会继续执行扫描表。

inner join 、left join、right join，三种连接如果结果相同，优先使用inner join

order by 排序时,使用有索引的字段进行排序

尽量根据主键/索引字段进行数据更新

性能：count(\*) ≈ count(1) > count(主键) > count(字段)

1.慢查询日志记录慢SQL

2.explain分析SQL的执行计划

第一个看的字段是possible\_keys, 理论上用到的索引

第二个看的字段是key，实际上用到的索引

第三个看的的字段是type，采取什么类型的扫描数据

3.profile 分析执行耗时

select\_type标记了单表查询、关联查询、子查询等信息

1、type

system表只有一行记录(等于系统表)

const 使用常量进行索引查询

eq\_ref 唯一索引扫描，通常使用主键约束

ref 非唯一性索引扫描

range 索引范围扫描

index 全索引扫描

ALL 全表扫描

性能排名：system > const > eq\_ref > ref > range > index > all。

实际sql优化中，最后达到ref或range级别。

key：实际使用的索引。

possible\_keys：显示可能应用在这张表中的索引

key\_len：使用的索引的长度。

**SQL调优**

1. 定位问题：通过监控或日志找到慢查询

2. 分析执行计划：通过EXPLAIN分析执行计划，找出性能瓶颈，确认是否存在全表扫描、低效连接等问题。

3. 针对性优化：

优化索引：添加索引，在查询频繁的列上创建索引；避免过多索引，索引虽能加速查询，但会降低写操作的性能；尽量使用联合索引

优化SQL语句，目标就是减少计算量、避免全表扫描、优化数据访问路径；

避免SELECT \*；用JOIN代替子查询（子查询可能生成临时表）；用EXISTS代替IN；避免在WHERE子句中对列使用函数或表达式，这会导致索引失效。

优化数据库设计与配置

表结构合理范式化：减少冗余，但可能增加JOIN复杂度；

分库分表；

参数调优：控制连接池的最大连接数，避免资源竞争，关闭非必要日志

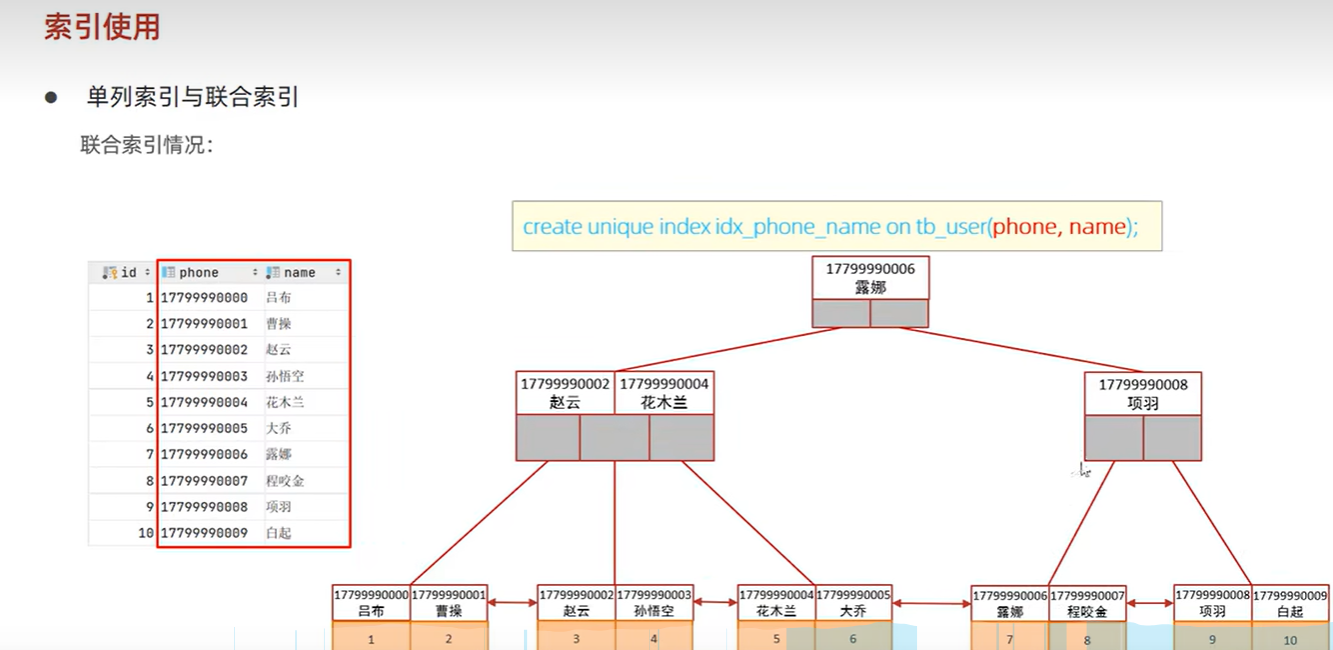
合理分配数据库内存参数，如缓冲池大小

4. 验证效果：对比优化前后执行时间和资源消耗。

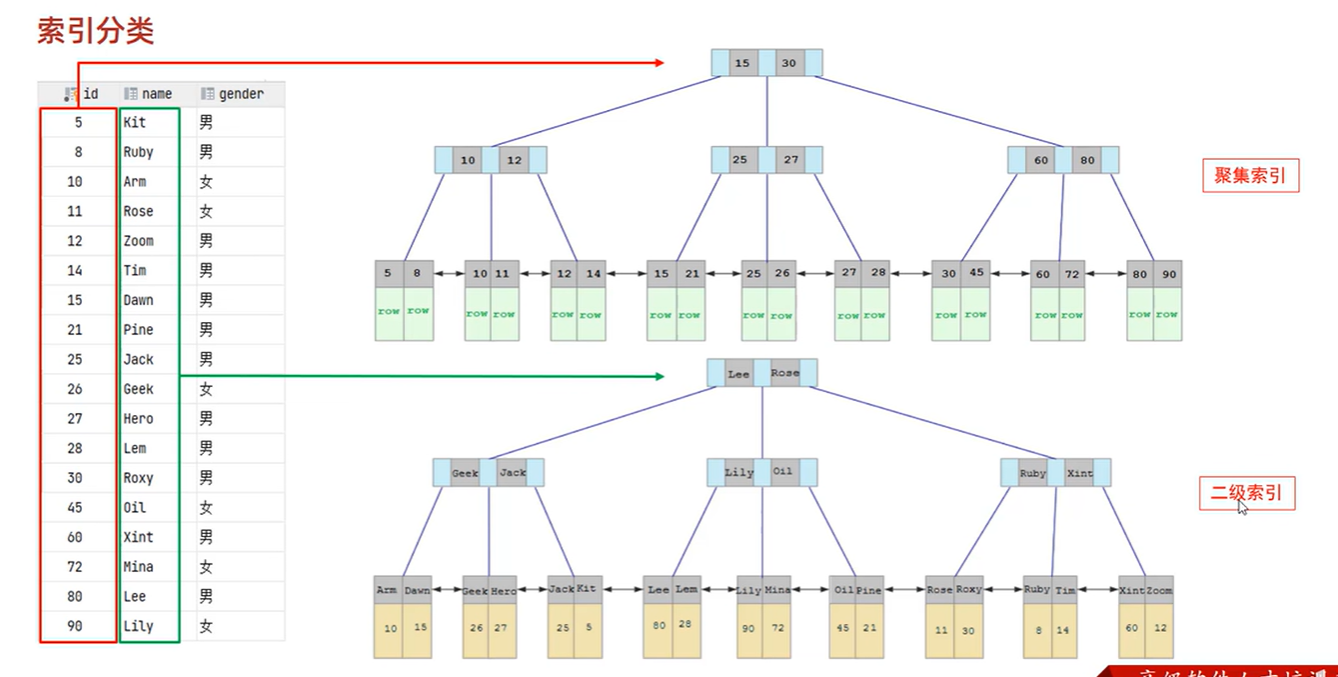
5. 持续监控：防止业务增长后性能再次下降。

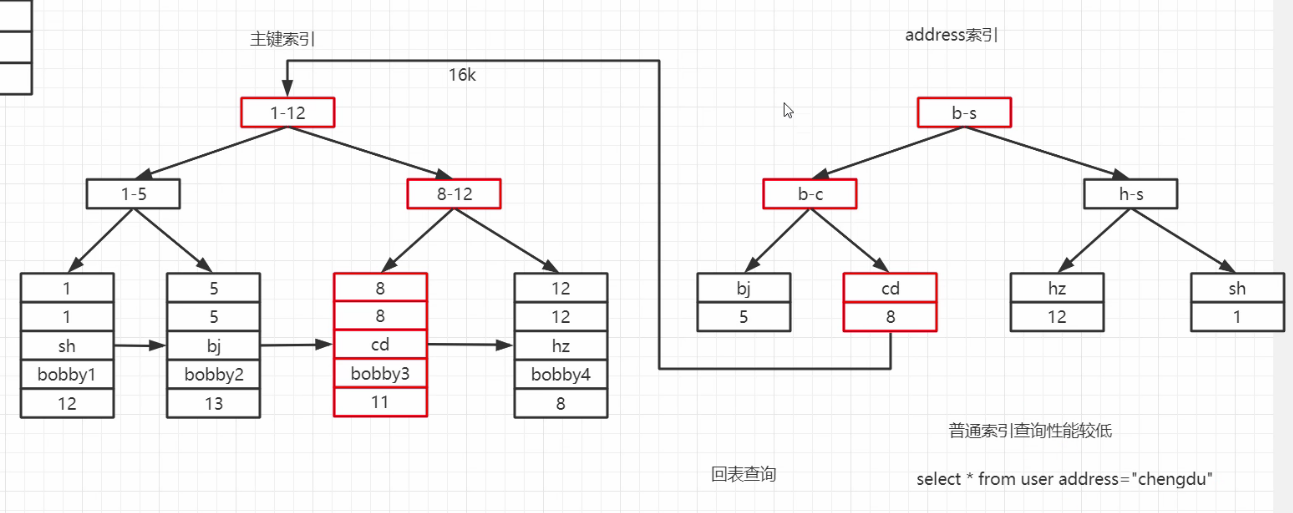
**什么是索引？**

索引是帮助MySQL高效获取数据的数据结构，相当于目录



联合索引，先按照phone排序，phone相同按照name排序name局部有序





**索引有哪些优缺点？**

优点：可以加快数据的检索速度。

缺点：创建索引需要耗费时间，存储索引需要占用空间。降低了更新表的速度，需要维护索引的结构

**索引有哪几种类型？**

主键索引: 唯一，非空，一个表只能有一个主键

唯一索引: 唯一，允许有空值，一个表允许多个列创建唯一索引

普通索引: 没有任何限制，可以不唯一，允许有空值

联合索引：多列值组成一个索引

全文索引

**创建索引的原则（重中之重）**

索引虽好，但也不是无限制的使用，最好符合一下几个原则

1） 最左前缀匹配原则

2） 较频繁作为查询条件的字段才去创建索引

3） 更新频繁的字段不适合创建索引

4）不能有效区分数据的字段不适合创建索引(如性别，男女未知，最多也就三种，区分度实在太低)

5） 尽量的扩展索引，不要新建索引。比如表中已经有a的索引，现在要加(a,b) 的索引，那么只需要修改原来的索引即可。

6） 有外键的数据列要建立索引。

7） 对于那些查询中很少涉及的列，重复值比较多的列不要建立索引。

8）对于定义为text、image和bit的数据类型的列不要建立索引。

针对数据量较大，且查询比较频繁的表建立索引

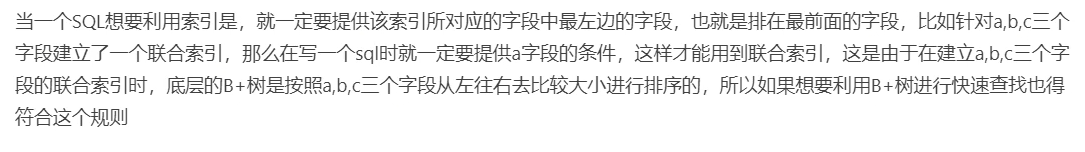
较频繁作为查询条件的字段才去创建索引

尽量选择区分度高的列作为索引

如果是字符串字段，长度较长，建立前缀索引

尽量使用联合索引，减少单列索引

**最左前缀原则**

****

当一个SQL想要利用联合索引时，就一定要提供该索引所对应的字段中最左边的字段。在创建联合索引时，where子句中使用最频繁的一列放在最左边。当创建(a,b,c)联合索引时，想要索引生效的话，只能使用 a和ab、ac和abc等组合最左前缀法则指的是查询从索引的最左列开始，并且不跳过索引中的列。如果跳跃某一列，索引将部分失效(后面的字段索引失效)。与where条件的顺序无关。

出现范围查询(>，<)，范围查询右侧的列索引失效，出现范围查询的那一列的索引不失效

**索引失效的情况**

在索引列上进行运算操作，索引失效

字符串不加引号，索引失效

头部模糊匹配，索引失效

联合索引，不满足最左前缀匹配原则‌

使用了select \*

**B树和B+树的区别**

在B树中，叶子节点和非叶子节点都存放键和值；但在B+树中，非叶子节点只存放键，不存放值，叶子节点存放键和值。

B+树的叶子节点有指针相连，而B树没有。

对于内部结点中的key，左子树中的key都小于它，右子树中的key都大于等于它。叶子结点也按照key的大小排列

**使用B+树的优点**

高效的查询性能：B+ 树是多叉树，树的高度较低，减少了磁盘 I/O 次数；数据都存储在叶子节点，查询时间复杂度稳定为 O(log n)。

减少内存占用：只有叶子节点存储数据，内部节点仅存储键，减少了内存占用。

适合范围查询：叶子节点通过指针连接，便于顺序访问，适合范围查询（如 BETWEEN、>、< 等操作）。

**聚集索引和二级索引**

聚集索引，索引结构的叶子节点保存了行数据，必须有，而且只有一个。默认主键索引就是聚集索引

二级索引，索引结构的叶子节点保存了主键，可以存在多个

**非聚簇索引一定会回表查询吗？**

不一定，查询语句所要求的字段全部命中了索引，就不必再进行回表查询。

select age from employee where age < 20

**什么是数据库事务？**

事务是一系列操作组成的工作单元，该工作单元内的操作是不可分割的，要么所有操作都做，要么所有操作都不做。

**事物的四大特性(ACID)介绍一下?**

原子性：要么所有操作都做，要么所有操作都不做

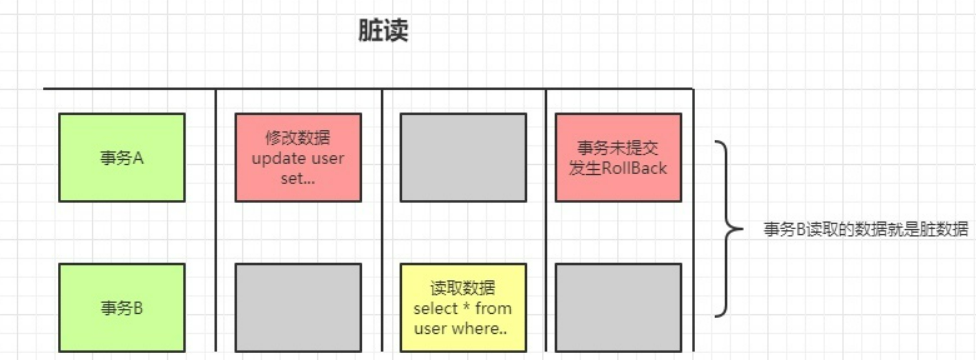
一致性：执行事务前后，数据的完整性必须保持一致。(拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性)

隔离性：多个事务并发执行时，一个事务的执行不应影响其他事务的执行

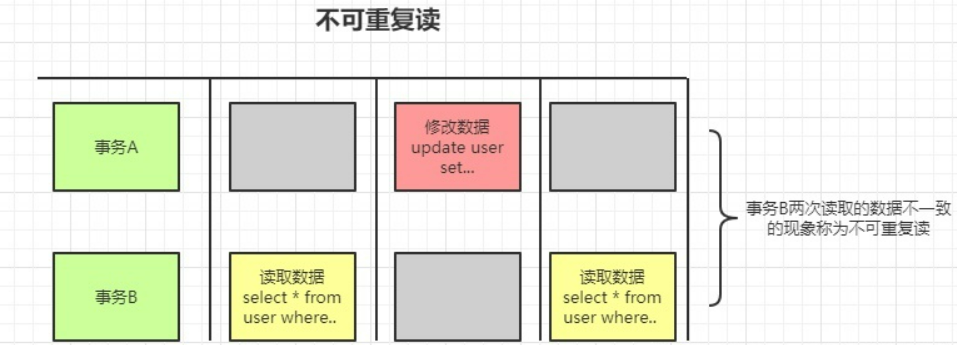
持久性：事务对数据的改变是持久的

**什么是脏读？幻读？不可重复读？**

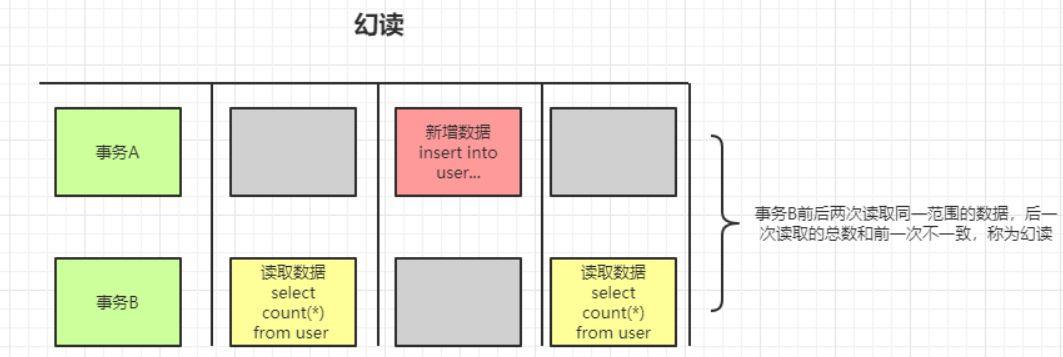
脏读：脏读是指一个事务读取到其他事务没有提交的数据。



不可重复读：指一个事务内多次根据同一查询条件查询出来的值不一致。



幻读：指一个事务内多次根据同一查询条件查询出来的记录行数不一致。



**什么是事务的隔离级别？MySQL的默认隔离级别是什么？**

为了达到事务的四大特性，数据库定义了4种不同的事务隔离级别，由低到高依次为Read uncommitted、Read committed、Repeatable read、

Serializable，这四个级别可以逐个解决脏读、不可重复读、幻读这几类问题。

mysql默认的事务隔离级别为repeatable-read



**SQL 标准定义了四个隔离级别：**

READ-UNCOMMITTED(读取未提交)：允许读取未提交的数据

READ-COMMITTED(读取已提交)：允许读取已提交的数据

REPEATABLE-READ(可重复读)： 对同一字段的多次读取结果都是一致的

SERIALIZABLE(可串行化)：所有的事务依次逐个执行(最高的隔离级别，完全服从ACID的隔离级别)

**对MySQL的锁了解吗**

当数据库有并发事务的时候，可能会产生数据的不一致，这时候需要一些机制来保证访问的次序，锁机制就是这样的一个机制。

**隔离级别与锁的关系**

在Read Uncommitted级别下，读取数据不需要加共享锁，这样就不会跟被修改的数据上的排他锁冲突

在Read Committed级别下，读操作需要加共享锁，但是在语句执行完以后释放共享锁；

在Repeatable Read级别下，读操作需要加共享锁，但是在事务提交之前并不释放共享锁，也就是必须等待事务执行完毕以后才释放共享锁。

SERIALIZABLE 是限制性最强的隔离级别，因为该级别锁定整个范围的键，并一直持有锁，直到事务完成。

**按照锁的粒度分数据库锁有哪些？**

全局锁，锁定数据库中的所有表，处于只读状态

行级锁是锁一行或者多行记录，锁粒度最小，锁冲突几率低，并发度高。行锁是基于索引加载的，所以行锁是要加在索引响应的行上，即命中索引

表级锁是锁一整张表，锁粒度大，锁冲突几率高，并发度低。表锁响应的是非索引字段，即全表扫描。

默认情况下，表锁和行锁都是自动获得的，不需要额外的命令。

(锁冲突是等待别人释放需要的锁资源)

**行锁->行级锁，行级锁分为共享锁和排他锁**

共享锁：又称为读锁，一个事务给某行数据加了读锁，其他事务也可以读，但是不能写

排他锁: 又称为写锁，一个事务给某行数据加了写锁，其他事务不能读，也不能写

**数据库的乐观锁和悲观锁是什么？怎么实现的？**

悲观锁：总认为会发生并发冲突，认为别人会修改数据，需要将数据锁定。实现方式：使用数据库中的锁机制

乐观锁：总认为不会发生并发冲突，只有提交数据更新时，才会对数据是否冲突进行检测。

**SQL语句主要分为哪几类**

数据定义语言DDL，CREATE，DROP，ALTER等 即对逻辑结构等有操作的，其中包括表结构，视图和索引。

数据查询语言DQL，SELECT，各种简单查询，连接查询等都属于DQL。

数据操纵语言DML，INSERT，UPDATE， DELETE

数据控制语言DCL，GRANT，REVOKE，COMMIT，ROLLBACK

**超键、候选键、主键、外键分别是什么？**

超键：能唯一标识元组的属性集称为超键。一个属性可以作为一个超键，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：不含有多余属性的超键称为候选键

主键：指定一个候选键为主键，主键唯一，非空

外键：外键就是另一个表的主键

**SQL 约束有哪几种？**

NOT NULL: 非空约束

UNIQUE: 唯一约束

PRIMARY KEY: 主键约束

FOREIGN KEY: 外键约束

CHECK: 控制字段的值范围

**关联查询**

内连接分为隐式内连接和显式内连接，隐式内连接from后面跟上表1，表2，where后面跟上条件，显式内连接用inner join

外连接分为左外连接和右外连接

左外连接：LEFT JOIN，以左表为主，先查询出左表所有数据，按照ON后面的关联条件匹配右表，没有匹配到的用NULL填充

右外连接：RIGHT JOIN, 以右表为主，先查询出右表，按照ON后的关联条件匹配左表，没有匹配到的用NULL填充

自连接可以是内连接也可以是外连接，一张表自己连接自己

联合查询（UNION与UNION ALL）：

把多次查询的结果合并起来，联合查询的列要相等，UNION ALL，不会合并重复的记录行

**什么是子查询**

sql语句中嵌套select语句，称为子查询

**子查询的四种情况**

标量子查询，单个值，父查询使用：= <> > < 等运算符

列子查询，一列多行，父查询使用：in 、not in运算符

行子查询，一行多列，父查询使用：= 、<>、in 、not in运算符。用括号把多个列名包起来

表子查询，多行多列，父查询使用：in。用括号把多个列名包起来

**varchar与char的区别**

char表示定长字符串，如果插入数据的长度小于char的固定长度，则用空格填充

varchar表示可变长字符串，插入的数据是多长，就按照多长来存储

**int(20)中20的涵义**

是指显示字符的长度。20表示最大显示宽度为20，但仍占4字节存储，存储范围不变；

**为什么要使用数据库**

数据永久保存、管理数据方便

SQL语句查询效率高

**什么是SQL？**

SQL代表结构化查询语言，是访问关系型数据库的通用语言

**什么是MySQL?**

MySQL是一个关系型数据库管理系统

**mysql有哪些数据类型**

整型有TINYINT、SMALLINT、MEDIUMINT、INT、

BIGINT，分别表示1字节、2字节、3字节、4字节、8字节整数。

浮点型有FLOAT、DOUBLE、DECIMAL

字符串有VARCHAR、CHAR、TEXT

VARCHAR用于存储可变长字符串，它比定长类型更节省空间。

日期和时间类型有DATETIME、DATE、TIMESTAMP

**mysql大数据表查询优化**

1.建立索引：索引可以显著提高查询速度，特别是在大数据表中。应该将索引放置在经常用于检索的列上，以尽可能减少查询时间。

2.缓存查询结果：缓存查询结果可以极大地减少查询时间，特别是针对重复查询的情况。

3.分库分表：分表，将一张表拆成几张表，水平拆分，每个表的结构都一样。垂直拆分，一般需要关联id。分库，水平分库, 每个库的结构都一样，垂直分库，将不同的表拆分到不同的库中（应用程序的代码编写难度增大，使用mycat，不用调整代码即可实现分库分表）

4. 主从复制，实现读写分离，主库用来写数据，从库用来读数据，增删改的请求操作主库，查询的请求操作从库，降低主库的访问压力。

3. 分区：将大的数据表分为几个小的数据表，可以将查询操作分布在不同的服务器上，从而大大提高查询速度。

4. 使用适当的数据类型：使用适当的数据类型可以减少存储空间，从而提高查询速度。例如，使用INT代替VARCHAR，可以显著减少查询时间。

5. 冗余数据：通过将重要的数据复制到多个表中，可以避免查询过程中的连接操作，从而提高查询速度。

1. 避免使用子查询：子查询通常比其他查询慢得多，因此应尽量避免使用它们。

2. 避免使用LIKE查询：LIKE查询通常比其他查询慢得多，应尽可能使用其他查询方法。

3. 避免使用太多JOIN：JOIN操作需要花费大量的时间，应尽量减少它们的使用。

**Mysql引擎**

1. innoDB支持事务、外键。MyISAM不支持

2. innoDB支持到行级锁，MyISAM仅支持表级锁；

3. innodb不保存表具体行数；myisam保存表具体行数；

4. 执行大量的update和insert最好用innoDB，执行大量select，最好用MyISAM

5.myisam使用非聚集索引，索引和数据分开，只缓存索引；innodb使用聚集索引，索引和数据存在一个文件。

6. Innodb不支持fulltext全文索引，MyISAM支持

**SQL执行顺序**

FROM操作哪些表

ON 、JOIN 对表进行关联

WHERE过滤数据

GROUP BY 分组

AGG\_FUNC 计算聚合函数

HAVING对分组之后的数据进行过滤

SELECT选取列

DISTINCT去重

ORDER BY排序

LIMIT/OFFSET 分页

**MYSQL函数**

curdate(),获取当前日期和获取当前时间函数

ADDDATE(date,n)函数：计算日期参数date加上n天后的日期。

DATEDIFF(date1,date2) 返回日期参数date1和date2之间相隔的天数

month（date）获取月份函数

CONCAT、AVG()、SUM() 、MAX()、MIN()、COUNT()

**建表考虑的点**

1. 字段设计

选择合适的数据类型，避免浪费存储或溢出

是否允许 NULL，尽量定义非空，减少索引复杂度（NULL 会增加索引存储开销）

主键设计，优先使用自增整数作为主键，提高查询效率

2. 索引优化

添加索引，在查询频繁的列上创建索引；避免过多索引，索引虽能加速查询，但会降低写操作的性能；尽量使用联合索引

3. 字符集推荐使用 `utf8mb4` 字符集

4. 约束与数据完整性

非空约束、唯一约束、主键约束、外键约束、CHECK 约束

5. 命名规范与注释

表名、字段名使用小写字母+下划线（如 `user\_order`）

为表和字段添加 `COMMENT`注释

6. 表引擎选择

InnoDB、MyISAM

**UUID字段作为索引的局限**

由于UUID是随机的，新插入的值可能落在任意位置，导致节点频繁分裂，增加插入操作的开销

较大的UUID字段会增加索引的大小，可能导致更多的磁盘I/O操作，降低查询速度

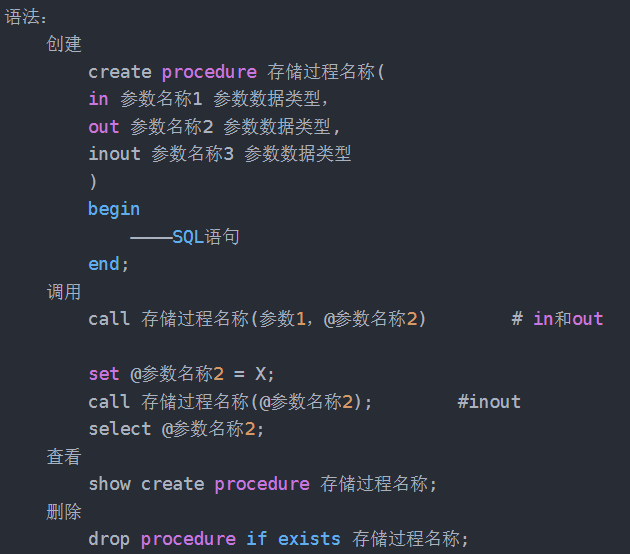
**联合索引和单列索引**

联合索引适合高频多列查询，单列索引适合单列查询频率高

**存储过程**

1. 提高性能，存储过程在首次执行时编译，后续调用直接使用编译后的代码

2. 减少网络开销，应用程序只需传递调用命令，而非大量SQL语句



**union all和union区别**

union去除重复，union all不会去除重复记录+

union all要比 union快的多

**事务回滚**

COMMIT; -- 提交事务

ROLLBACK; -- 回滚事务