**1.常用的关系型数据库**

* **Oracle：**功能强大，主要缺点就是贵
* **MySQL：**互联网行业中最流行的数据库，这不仅仅是因为MySQL的免费。可以说关系数据库场景中你需要的功能，MySQL都能很好的满足。
* **MariaDB：**是MySQL的分支，由开源社区维护，MariaDB虽然被看作MySQL的替代品，但它在扩展功能、存储引擎上都有非常好的改进。
* **PostgreSQL：**也叫PGSQL，PGSQL类似于Oracle的多进程框架，可以支持高并发的应用场景，PG几乎支持所有的SQL标准，支持类型相当丰富。PG更加适合严格的企业应用场景，而MySQL更适合业务逻辑相对简单、数据可靠性要求较低的互联网场景。

**2.NoSQL数据库（非关系型数据库）**

* **Redis：**提供了持久化能力，支持多种数据类型。Redis适用于数据变化快且数据大小可预测的场景。
* **MongoDB：**一个基于分布式文件存储的数据库，将数据存储为一个文档，数据结构由键值对组成。MongoDB比较适合表结构不明确，且数据结构可能不断变化的场景，不适合有事务和复杂查询的场景。
* **HBase：**建立在HDFS，也就是Hadoop文件系统之上的分布式面向列（宽列即有列族概念，但不是列式存储）的数据库。类似于谷歌的大表设计，HBase可以提供快速随机访问海量结构化数据。在表中它由行排序，一个表有多个列族以及每一个列族可以有任意数量的列。 HBase依赖HDFS可以实现海量数据的可靠存储，适用于数据量大，写多读少，不需要复杂查询的场景。
* **Cassandra：**一个高可靠的大规模分布式存储系统。支持分布式的结构化Key-value存储，以高可用性为主要目标。适合写多的场景，适合做一些简单查询，不适合用来做数据分析统计。
* **Pika：**一个可持久化的大容量类Redis存储服务， 兼容五种主要数据结构的大部分命令。Pika使用磁盘存储，主要解决Redis大容量存储的成本问题。

**3.NewSQL数据库（新一代关系型数据库）**

* **TiDB：**开源的分布式关系数据库，几乎完全兼容MySQL，能够支持水平弹性扩展、ACID事务、标准SQL、MySQL语法和MySQL协议，具有数据强一致的高可用特性。既适合在线事务处理，也适合在线分析处理。
* **OceanBase：**OceanBase是蚂蚁金服的数据库，OB是可以满足金融级的可靠性和数据一致性要求的数据库系统。当你需要使用事务，并且数据量比较大，就比较适合使用OB。不过目前OB已经商业化，不再开源。
* Clickhouse：列式数据库，相对于传统的MySQL（行式数据库），它会以列存储一个单元的完整数据，比传统数据库的优势在于数据分析时不需要读取整行的完整数据，只需要读取部分列数据即可，更加适用于OLAP大数据业务场景。

**二、事物特性及事物类型**

后面的详解知识点会展开介绍

**三、数据库的范式**

前关系数据库有六种范式：第一范式、第二范式、第三范式、巴斯-科德范式（BCNF）、第四范式和第五范式。**范式级别越高对数据表的要求越严格。**

* 第一范式要求最低，只要求表中字段不可用在拆分。
* 第二范式在第一范式的基础上要求每条记录由主键唯一区分，记录中所有属性都依赖于主键。
* 第三范式在第二范式的基础上，要求所有属性必须直接依赖主键，不允许间接依赖。
* 一般说来，数据库只需满足第三范式就可以了。

详解知识点一：数据库事务

**知识点**

**▌1.数据库事务特性**

数据库的特性是面试时考察频率非常高的题目，共4个特性：

* **原子性：**是指事务由原子的操作序列组成，所有操作要么全部成功，要么全部失败回滚。
* **一致性：**是指事务的执行不能破坏数据库数据的完整性和一致性，一个事务在执行之前和执行之后，数据库都必须处以一致性状态。比如在做多表操作时，多个表要么都是事务后新的值，要么都是事务前的旧值。
* **隔离性：**是指多个用户并发访问数据库时，数据库为每个用户执行的事务，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。事务的隔离级别我们稍后介绍。
* **持久性：**是指一个事务一旦提交并执行成功，那么对数据库中数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

**▌2.事物并发问题与隔离级别**

**a.事务并发问题**

* **脏读：**脏读是指在一个事务处理过程里读取了另一个未提交的事务中的数据，例如，账户A转帐给B500元，B余额增加后但事务还没有提交完成，此时如果另外的请求中获取的是B增加后的余额，这就发生了脏读，因为事务如果失败回滚时，B的余额就不应该增加。
* **不可重复读：**不可重复读是指对于数据库中某个数据，一个事务范围内多次查询返回了不同的数据值，这是由于在多次查询之间，有其他事务修改了数据并进行了提交。
* **幻读：**是指一个事务中执行两次完全相同的查询时，第二次查询所返回的结果集跟第一个查询不相同。与不可重复读的区别在于，不可重复读是对同一条记录，两次读取的值不同。而幻读是记录的增加或删除，导致两次相同条件获取的结果记录数不同。

**b：事务的四种隔离级别**

可以用于解决这几种并发问题。如图右面，由上到下的4种隔离级别由低到高。

* **级别1读未提交：**也就是可以读取到其他事务未提交的内容，这是最低的隔离级别，这个隔离级别下，前面提到的三种并发问题都有可能发生。
* **级别2读已提交：**就是只能读取到其他事务已经提交的数据。这个隔离级别可以解决脏读问题。
* **级别三可重复读：**可以保证整个事务过程中，对同数据的多次读取结果是相同的。这个级别可以解决脏读和不可重复读的问题。MySQL默认的隔离级别就是可重复读。
* **级别四串行化：**这是最高的隔离级别，所有事务操作都依次顺序执行。这个级别会导致并发度下降，性能最差。不过这个级别可以解决前面提到的所有并发问题。

**▌3.事务分类**

共分5大类：

* **扁平化事务**：在扁平事务中，所有的操作都在同一层次，这也是我们平时使用最多的一种事务。它的主要限制是不能提交或者回滚事务的某一部分，要么都成功，要么都回滚。
* **带保存点的扁平事务：**为了解决第一种事务的弊端，就有了第二种带保存点的扁平事务。它允许事务在执行过程中回滚到较早的状态，而不是全部回滚。通过在事务中插入保存点，当操作失败后，可以选择回滚到最近的保存点处。
* **链事务：**可以看做是第二种事务的变种。它在事务提交时，会将必要的上下文隐式传递给下一个事务，当事务失败时就可以回滚到最近的事务。不过，链事务只能回滚到最近的保存点，而带保存点的扁平化事务是可以回滚到任意的保存点。
* **嵌套事务:**由顶层事务和子事务构成，类似于树的结构。一般顶层事务负责逻辑管理，子事务负责具体的工作，子事务可以提交，但真正提交要等到父事务提交，如果上层事务回滚，那么所有的子事务都会回滚。
* **分布式事务：**是指分布式环境中的扁平化事务。

**其中，常用的分布式事务解决方案共4种**

**a.XA协议：**是保证强一致性的刚性事务。实现方式有两段式提交和三段式提交。两段式提交需要有一个事务协调者来保证所有的事务参与者都完成了第一阶段的准备工作。如果协调者收到所有参与者都准备好的消息，就会通知所有的事务执行第二阶段提交。一般场景下两段式提交已经能够很好得解决分布式事务了，然而两阶段在即使只有一个进程发生故障时，也会导致整个系统存在较长时间的阻塞。三段式提交通过增加Pre-commit阶段来减少前面提到的系统阻塞的时间。三段式提交很少在实际中使用，简单了解就可以了。

**b.TCC：**是满足最终一致性的柔性事务方案。TCC采用补偿机制，核心思想是对每个操作，都要注册对应的确认和补偿操作。它分为三个阶段：Try阶段主要对业务系统进行检测及资源预留；Confirm阶段对业务系统做确认提交。Cancel阶段是在业务执行错误，执行回滚，释放预留的资源。

**c.消息事务：**第三种方案是消息一致性方案。基本思路是将本地操作和发送消息放在一个事务中，保证本地操作和消息发送要么都成功要么都失败。下游应用订阅消息，收到消息后执行对应操作。

**d.GTS/Fescar:**阿里云中的全局事务服务GTS，对应的开源版本是Fescar。Fescar基于两段式提交进行改良，剥离了分布式事务方案对数据库在协议支持上的要求。使用Fescar的前提是分支事务中涉及的资源，必须是支持ACID事务的关系型数据库。分支的提交和回滚机制，都依赖于本地事务来保障。 Fescar的实现目前还存在一些局限，比如事务隔离级别最高支持到读已提交级别。

**详解知识点二：MySQL数据库**

**▌1.常用SQL语句**

需要能手写常用SQL语句，这里没有什么特殊的技巧，根据如图列出的语句类型多做一些练习

**▌2.数据类型**

要知道MySQL都提供哪些基本都数据类型，不同数据类型占用的空间大小。

**▌3.MySQL中主要的存储引擎**

MyISAM是MySQL官方提供的存储引擎，其特点是支持全文索引，查询效率比较高，缺点是不支持事务、使用表级锁。InnoDB在5.5版本后成为了Mysql的默认存储引擎，特点是支持ACID事务、支持外键、支持行级锁提高了并发效率。TokuDB是第三方开发的开源存储引擎，有非常快的写速度，支持数据的压缩存储、可以在线添加索引而不影响读写操作。但是因为压缩的原因，TokuDB非常适合访问频率不高的数据或历史数据归档，不适合大量读取的场景。

**▌4.MySQL中的锁**

MyIASAM使用表级锁，InnoDB使用行级锁。表锁开销小，加锁快，不会出现死锁；但是锁的粒度大，发生锁冲突的概率高，并发访问效率比较低。行级锁开销大，加锁慢，有可能会出现死锁，不过因为锁定粒度最小，发生锁冲突的概率低，并发访问效率比较高。

**注：**

* 共享锁也就是读锁，其他事务可以读，但不能写。MySQL可以通过Lock In Share Mode语句显示使用共享锁。
* 排他锁就是写锁，其他事务不能读取，也不能写。对于Update、Delete和INSERT语句，InnoDB会自动给涉及的数据集加排他锁，或者使用select for update显示使用排他锁。

**▌5.索引**

后文重点讲解

**▌6.MySQL的存储过程与函数**

存储过程和函数都可以避免开发人员重复编写相同的SQL语句，并且存储过程和函数都是在MySQL服务器中执行的，可以减少客户端和服务器端的数据传输。

存储过程能够实现更复杂的功能，而函数一般用来实现针对性比较强的功能，例如特殊策略求和等。存储过程可以执行包括修改表等一系列数据库操作，而用户定义函数不能用于执行修改全局数据库状态的操作。

存储过程一般是作为一个独立的部分来执行，而函数可以作为查询语句的一个部分来调用。SQL语句中不能使用存储过程，但可以使用函数。

不过存储过程一般与数据库实现绑定，使用存储过程会降低程序的可移植性，应谨慎使用。

**▌7.新特性**

可以了解MySQL8.0的一些新特性，例如默认字符集格式改为了UTF8；增加了隐藏索引的功能，隐藏后的索引不会被查询优化器使用，可以使用这个特性用于性能调试；支持了通用表表达式，使复杂查询中的嵌入表语句更加清晰；新增了窗口函数的概念，它可以用来实现新的查询方式。窗口函数与 SUM、COUNT等集合函数类似，但不会将多行查询结果合并，而是将结果放在多行中。即窗口函数不需要GROUP BY。

**▌8.MySQL调优**

后文重点讲解。

***➔重点讲解一：MySQL索引***

索引可以大幅增加数据库的查询的性能，在实际业务场景中，或多或少都会使用到。

**但是索引是有如下2个代价的:**

a.需要额外的磁盘空间来保存索引

b.对于插入、更新、删除等操作由于更新索引会增加额外的开销

因此索引比较适合用在读多写少的场景。

**▌1.MySQL索引类型**

如左面的模块，共分为5类：

* **唯一索引：**就是索引列中的值必须是唯一的，但是允许出现空值。这种索引一般用来保证数据的唯一性，比如保存账户信息的表，每个账户的id必须保证唯一，如果重复插入相同的账户id时会MySQL返回异常。
* **主键索引：**是一种特殊的唯一索引，但是它不允许出现空值。
* **普通索引：**与唯一索引不同，它允许索引列中存在相同的值。例如学生的成绩表，各个学科的分数是允许重复的，就可以使用普通索引。
* **联合索引：**就是由多个列共同组成的索引。一个表中含有多个单列的索引并不是联合索引，联合索引是对多个列字段按顺序共同组成一个索引。应用联合索引时需要注意最左原则，就是Where查询条件中的字段必须与索引字段从左到右进行匹配。比如，一个用户信息表，用姓名和年龄组成了联合索引，如果查询条件是姓名等于张三，那么满足最左原则；如果查询条件是年龄大于20，由于索引中最左的字段是姓名不是年龄，所以不能使用这个索引。
* **全文索引：**前面提到了，MyISAM引擎中实现了这个索引，在5.6版本后InnoDB引擎也支持了全文索引，并且在5.7.6版本后支持了中文索引。全文索引只能在CHAR,VARCHAR,TEXT类型字段上使用，底层使用倒排索引实现。要注意对于大数据量的表，生成全文索引会非常消耗时间也非常消耗磁盘空间。

**▌2.索引实现**

如右面的模块，索引实现共分4种形式：

* **B+树实现**：b+树比较适合用作'>'或'<'这样的范围查询，是MySQL中最常使用的一种索引实现。
* **R-tree：**是一种用于处理多维数据的数据结构，可以对地理数据进行空间索引。不过实际业务场景中使用的比较少。
* **Hash：**是使用散列表来对数据进行索引，Hash方式不像Btree那样需要多次查询才能定位到记录，因此Hash索引的效率高于B-tree，但是不支持范围查找和排序等功能.实际使用的也比较少。
* **FullText：**就是我们前面提到的全文索引，是一种记录关键字与对应文档关系的倒排索引。

**➔*重点讲解二：MySQL调优***

MySQL的调优也是研发人员需要掌握的一项技能

**一般MySQL调优有图中的4个纬度：**

* 针对数据库设计、表结构设计以及索引设置纬度进行的优化；
* 对业务中使用的SQL语句进行优化，例如调整Where查询条件；
* 对mysql服务的配置进行优化，例如对链接数的管理，对索引缓存、查询缓存、排序缓存等各种缓存大小进行优化;
* 对硬件设备和操作系统设置进行优化，例如调整操作系统参数、禁用Swap、增加内存、升级固态硬盘等等。

这四个纬度从优化的成本角度来讲，从左到右优化成本逐渐升高；从优化效果角度来看，从右到左优化的效果更高。

对于研发人员来说，前两个纬度与业务息息相关，因此需要重点掌握，后两个纬度更适合DBA进行深入学习，简单了解就好。

**本文将重点关注前两个纬度**

**▌1.表结构和索引的优化**

如左面的模块，应该掌握如下6个原则：

**第1个原则**：要在设计表结构时，考虑数据库的水平与垂直扩展能力，提前规划好未来1年的数据量、读写量的增长，规划好分库分表方案。比如设计用户信息表，预计1年后用户数据10亿条，写QPS约5000，读QPS30000，可以设计按UID纬度进行散列，分为4个库每个库32张表，单表数据量控制在KW级别；

**第2个原则**：要为字段选择合适的数据类型，在保留扩展能力的前提下，优先选用较小的数据结构。例如保存年龄的字段，要使用TINYINT而不要使用INT；

**第3个原则**：可以将字段多的表分解成多个表，必要时增加中间表进行关联。假如一张表有4、50个字段显然不是一个好的设计；

**第4个原则**：是设计关系数据库时需要满足第三范式，但为了满足第三范式，我们可能会拆分出多张表。而在进行查询时需要对多张表进行关联查询，有时为了提高查询效率，会降低范式的要求，在表中保存一定的冗余信息，也叫做反范式。但要注意反范式一定要适度；

**第5个原则**：要擅用索引，比如为经常作为查询条件的字段创建索引、创建联合索引时要根据最左原则考虑索引的复用能力，不要重复创建索引；要为保证数据不能重复的字段创建唯一索引等等。不过要注意索引对插入、更新等写操作是有代价的，不要滥用索引。比如像性别这样唯一很差的字段就不适合建立索引；

**第6个原则**：列字段尽量设置为Not Null，MySQL难以对使用Null的列进行查询优化，允许Null会使索引、索引统计和值更加复杂。允许Null值的列需要更多的存储空间，还需要MySQL内部进行特殊处理。

**▌2.SQL语句进行优化的原则**

如右面的模块，共分5个原则：

**第1个原则：**要找的最需要优化的SQL语句。要么是使用最频繁的语句，要么是优化后提高最明显的语句，可以通过查询MySQL的慢查询日志来发现需要进行优化的SQL语句；

**第2个原则**：要学会利用MySQL提供的分析工具。例如使用Explain来分析语句的执行计划，看看是否使用了索引，使用了哪个索引，扫描了多少记录，是否使用文件排序等等。或者利用Profile命令来分析某个语句执行过程中各个分步的耗时；

**第3个原则**：要注意使用查询语句是要避免使用Select \*，而是应该指定具体需要获取的字段。原因一是可以避免查询出不需要使用的字段，二是可以避免查询列字段的元信息；

**第4个原则**：是尽量使用Prepared Statements，一个是性能更好，另一个是可以防止SQL注入；

**第5个原则**：是尽量使用索引扫描来进行排序，也就是尽量在有索引的字段上进行排序操作。

以上为数据库操作须掌握的内容，可以进行差缺补漏，希望对研发人员有一定的帮助

***1.***必须了解数据库的基本原理、使用场景以及常用队列、数据库的特点。MySQL提供了多种引擎可以支持事务型与非事务型的关系对象库服务等等。

***2.***要深刻理解数据库事务的ACID特性，了解并发事务可能导致的并发问题和不同的数据库隔离级别如何解决这些并发问题。

***3.***要掌握常用的MySQL语句，比如WHERE条件查询语句、JOIN关联语句、ORDER BY排序语句等等。还要熟悉常用的自带函数，例如SUM、COUNT等等。

***4.***要了解MySQL数据库不同引擎的特点及不同类型的索引实现。比如最长使用的InnoDB非常擅长事务处理，MyISAM比较适合非事务的简单查询场景。比如知道MySQL的唯一索引、联合索引、全文索引等不同索引类型，以及最长使用等B+树索引实现等等。

面试加分项

**加分项**

***1.***要了解新特性，例如MySQL8.0中提供了窗口函数来支持新的查询方式；支持通用表表达式，使复杂查询中的嵌入表语句更加清晰等等。

***2.***要知道数据库表设计原则，如果有过线上业务数据库的设计经验就更好了，你能够知道如何对容量进行评估，也知道适当分库分表来保证未来服务的可扩展性，这会对面试起到积极的影响。

***3.***最好有过数据库调优经验，例如明明建立了索引的语句，但是查询效率还是很慢，通过Explain分析发现表中有多个索引，MySQL的优化器选用了错误的索引，导致查询效率偏低，然后通过在SQL语句中使用Use Index来指定索引解决。