版本号：v1.0 阶段：初稿

# 数据库面试宝典

MRL Liu

（未经许可，不得传播）

2022年03月07日

本文以LeetCode网站的《数据库知识手册》、《MySQL必知必会》为参考，介绍数据库领域面试相关的知识，深入研究可参考其他资料。

在IT行业中，数据库是一个至关重要的技术，无论是在项目开发还是算法岗位，都需要运用数据库技术处理数据。

# 第1章 数据库基础篇

## 一、什么是数据库

### 1、数据库的概念

**数据库**（Database）是保存**有组织的数据**的**容器**（通常是一个文件或一组文件），是通过**数据库管理系统**（DataBase- Management System，DBMS）创建和操纵的容器。DBMS的主要目标是提供一种可以方便、高效地存取数据库信息的途径。

我们常说XX数据库，其实本质上是指XX数据库管理系统，使用的是一个软件。目前，较为流行的DBMS有MySQL、SQL Server、Oracle 等。

### 2、数据库的作用

数据库技术用来帮助我们管理海量数据，面对庞大的网络数据量，只有使用数据库技术才可以高效且条理分明地存储数据，它使技术人员能够更加迅速和方便地管理数据。

现在我们对比下常见的数据保存方式：

|  |  |  |  |
| --- | --- | --- | --- |
| **保存方式** | **优点** | **缺点** | **应用场景** |
| 保存在内存 | 存取速度快 | 数据无法永久保存，程序运行结束，数据消失。 | 应用程序执行过程中读写的临时数据 |
| 保存在文件 | 数据可永久保存 | 在查询海量数据时效率低；操作速度比内存操作慢，需要频繁的IO操作，所以存储速度慢 | 适用于存储少量、不需要频繁更新、安全要求较低的数据 |
| 数据保存在数据库 | 数据可**永久保存**；  数据**安全性高**；  可**高效管理海量数据**； | 数据库移植不方便；  不支持集群；  不擅长业务逻辑的处理 | 适用于存储海量、可能需要频繁更新、安全要求较高的数据，一般用于后端存储用户数据 |

### 3、数据库的类型

数据库有两种类型，分别是**关系型数据库**和**非关系型数据库**。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **定义** | **优点** | **缺点** | **常见** |
| 关系型数据库 | 建立在关系模型基础上，由多张能互相连接的表组成的数据库 | 1. 使用表结构，格式一致，易于维护； 2. 支持SQL 语句； 3. 安全性高，数据库 | 1. 读写性能比较差； 2. 建立在关系模型上，不可避免空间浪费； 3. 固定的表结构，灵活度较低 | MySQL，Microsoft SQL Server，Oracle，PostgreSQL 等 |
| 非关系型数据库又被称为 NoSQL（Not Only SQL ) | 以对象的形式存储数据的数据库 | 1）应用场景更广泛，存储数据的格式可以是key-value、文档、图片等形式，；   1. 可轻松进行海量数据的维护和处理； 2. 具有可扩展、高并发、高稳定性、成本低的优势；4）可以实现数据的分布式处理 | 1. 不支持SQL语句； 2. 安全性低，无法保证数据的完整性； | Neo4j，Redis，MongoDB 等 |

本文档着重介绍以MySQL为代表的关系数据库。

## 数据库的三大范式

数据库范式是设计数据库时，需要遵循的一些规范。各种范式是条件递增的联系，越高的范式数据库冗余越小。常用的数据库三大范式为：

第一范式（1NF）：每个列都不可以再拆分，强调的是列的原子性。第一范式要求数据库中的表都是二维表。

第二范式（2NF）：在第一范式的基础上，一个表必须有一个主键，非主键列完全依赖 于主键，而不能是依赖于主键的一部分。

第三范式（3NF）：在第二范式的基础上，非主键列只依赖（直接依赖）于主键，不依赖于其他非主键。

## 触发器

触发器（trigger）是与表相关的数据库对象，是用户定义在关系表上的一类由事件驱动的特殊的存储过程，在满足定义条件时触发，并执行触发器中定义的 语句集合。触发器的这种特性可以协助应用在数据库端确保 数据库的完整性。

使用场景：

可以通过数据库中的相关表实现 级联更改；

实时监控某张表中的某个字段的更改，并需要做出相应的处理。

# 第2章 MySQL基础篇

## 一、关系数据库的术语

前文已知，MySQL是一种关系数据库，这里介绍下关系数据库中的专用术语。

### 1、表、列、行的区别

表（table）是存储某种特殊数据类型的结构化文件，数据库中的每个表的表名都是唯一的，描述表如何存储结构化数据的信息就是表的模式（schema）。

表由列（colomn）组成，列表示表的某部分信息，表中的每个列都有对应的数据类型（datatype），数据类型可以帮助正确地分类数据，同时更好地利用磁盘空间。

表的数据是按行（row）组成的，行就是数据库中的一条记录（record）。表中的每一行都有唯一标识自己的一列（或一组列），叫做主键（primary key）。作为主键的列需要满足以下条件：

任意两行的主键都不可重复，不可为空，不允许修改、更新或重用（删除一条记录后，其主键值不能被重新分配）。

### 主键、外键、索引的区别

|  |  |  |
| --- | --- | --- |
|  | 定义 | 作用 |
| 主键 | 唯一记录一条数据的字段，不允许重复，不允许为空 | 用来唯一标识数据，维持保存数据的完整性 |
| 外键 | 一条数据中存储另一张表的主键的字段，允许重复，可以为空 | 用于建立和其他表的联系 |
| 索引 | 一种特殊的数据结构，不允许重复，但可以为空 | 用于协助快速查询、更新数据库表中的数据 |

### SQL与MySQL的区别

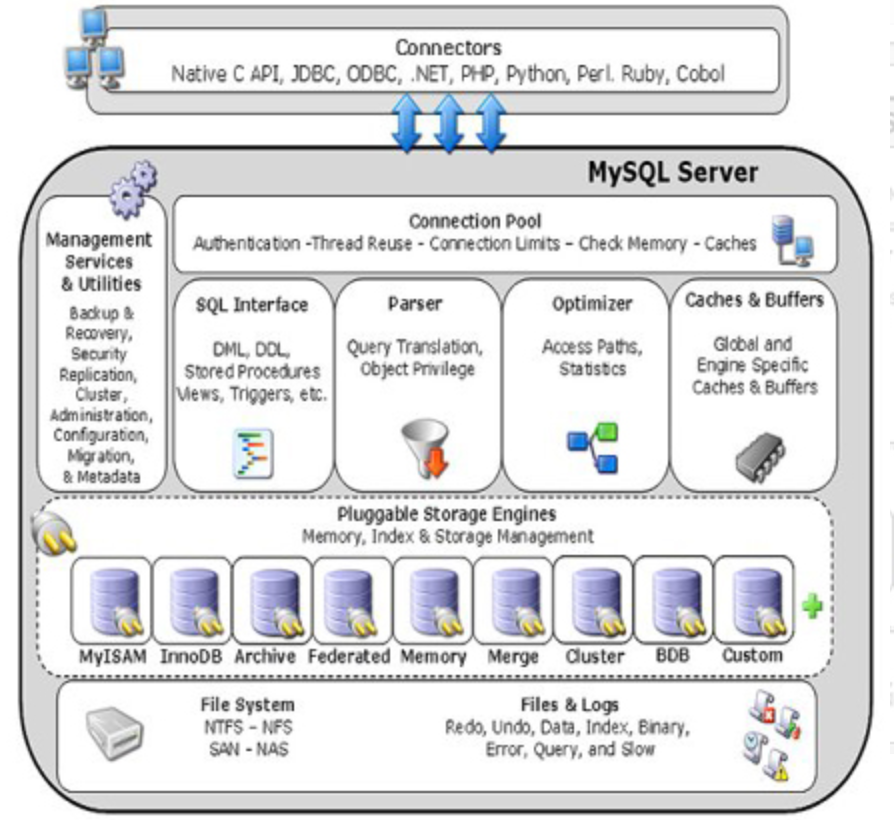
SQL（Structured Query Language）是结构化查询语言，请注意，SQL不是某个数据库供应商独有的语言，几乎所有主流的数据库软件都使用相同的SQL。

SQL和MySQL是DBMS中最令人困惑的两个术语，二者之间存在本质上的区别。

|  |  |  |  |
| --- | --- | --- | --- |
|  | **本质** | **用途** | **稳定性和费用** |
| SQL | 结构化查询语言（Structured Query Language） | 用于操作数据库（取数据、查询、更新和管理关系数据库系统），和数据库进行通信，与其他编程语言不一样，SQL由少量的描述性很强的词构成，简单易学。 | SQL的命令一般是稳定的，不存在专门的供应商，几乎所有主流的数据库软件底层都使用相同的SQL。 |
| MySQL | 关系数据库管理系统（RDBMS） | 本质是一个软件，会为用户提供一个界面，只需单击一些按钮即可用于执行各种数据库操作。该软件底层使用SQL执行所有数据库操作 | 由数据库供应商提供版本更新，可能需要商业付费。 |

## 二、MySQL数据库的架构

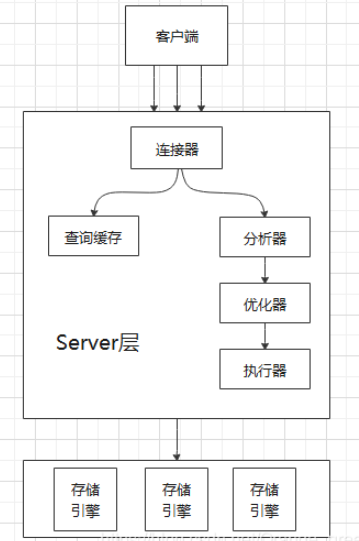
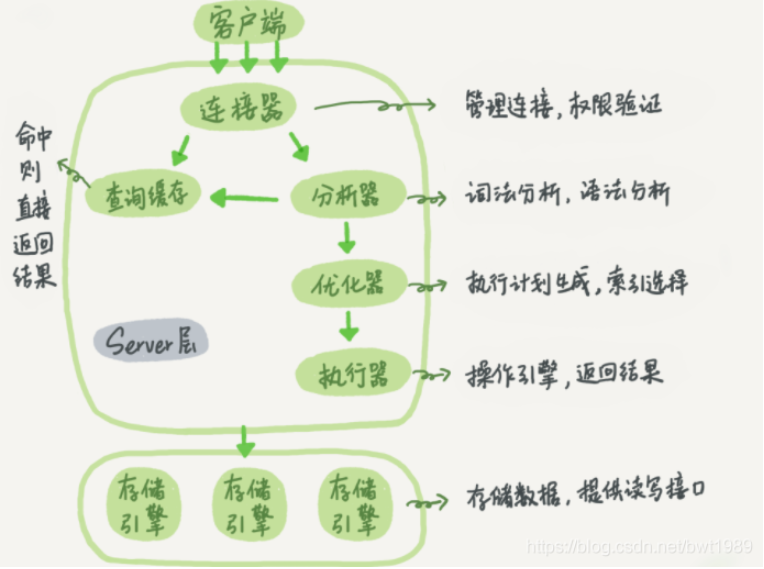
MySQL采用了典型的C/S架构，即使用MySQL Server作为服务端提供数据库服务，使用MySQL Client作为客户端。Client负责连接到Server，Server负责完成Client的请求和操作。



我们来简单分析下这张图，其整体分为Client的Connectors和MySQL Server。接下来我们先讲解一条SQL语句在MySQL中的执行过程

### 1、SQL语句在MySQL中的执行

我们可以将上图中的MySQL Server的各个组件简化为如下关系：



接下来我们详细分析下：

|  |  |  |
| --- | --- | --- |
| 应用场景 | 组件 | 作用 |
|  | 连接池（Connection Pool） | 主要负责Client与Server的服务的连接工作，本质上是两个进程间的通信。  1、根据Client提供的账号和密码校验身份，通过获取对应权限；  2、监控Client的动作，长时间没有动作则断开连接（默认8小时） |
|  | 查询缓存 | 收到一个查询请求时，MySQL会先到查询缓存查看是否执行过这条语句，可能会以key-value的形式，key是查询的语句，value是查询的结果。如果可以找到key，则直接返回value给Client，这个过程叫做命中。 |
|  | 分析器 | 如果没有命中缓存，SQL语句就会进入分析器，开始真正的执行。分析器主要对SQL语句进行 “词法分析”和“语法分析”。  词法分析：分析SQL语句里面的字符串分别是什么，代表什么。如：需要识别字符串“T”为为“表名T”，字符串“ID”为“列ID"。  语法分析：根据词法分析的结果，语法分析器根据语法规则，判断输入的SQL是否满足MySQL语法。  如果你的sql语句有语法错误，就会返回一个异常：“You have an error in your SQL syntax” |
|  | 优化器 | 经过了分析器，MySQL知道了你要做什么。在开始执行前，还要经过优化器的处理。  优化器主要负责SQL语句执行方案的选择。如表里面有多个索引的时候，决定使用哪个索引。或者在一个语句有多表关联的时候，决定各个表的连接顺序。  优化器主要是通过你的sql语句，选择一种最优的方式，告诉MySQL该如何执行该语句。优化器会自动判断出效率最高的一种执行方式，通知MySQL去执行。 |
|  | 执行器 | 执行器负责语句的具体执行  首先，执行器会先判断一下你对这个表有没有执行查询的权限。如果没有就会返回没有权限的错误。  如果有权限，就会进入下面的执行步骤： |
|  | （插件式）存储引擎 |  |
|  | 文件系统和日志系统 |  |

执行器通过操作引擎，将sql语句执行的结果，返回给客户端。查询缓存顾名思义是用来缓存结果数据的。为什么放到最后来说，是因为在MySQL8.0版本已经没有这个功能了。这个功能弊大于利，当查询结果命中查询缓存时，会直接返回结果。但大多数时候，我们的使用场景更新的频率会非常频繁，当某一个表中有一条更新数据时，会将该表的查询缓存结果全部清空，效率会非常低。可以使用在一些系统配置表，等更新不频繁的表中。当然我们可手动选择是否开启，参数是：query\_cache\_type.

### 2、MySQL的连接操作

#### （1）Client和MySQL Server的连接

#### （2）长连接和短连接的区别

长连接：指建立连接后，如果客户端有请求，则一直使用同一个连接。

短连接：每次执行完很少的几次查询就断开连接，下次查询重新建立新的连接。

建立连接过程比较复杂，频繁建立连接会消耗大量网络资源，所以推荐使用长连接。

但是长连接会有一个问题，Mysql在执行过程中，临时使用的内存是管理在连接对象里面的。这些资源直到连接断开的时候才会释放。长期累积下来可能会导致内存占用过大，被系统强行杀掉。

由两种方式可以解决：

定期断开长连接。使用一段时间后，或者执行过一个占用内存的大查询后，断开连接，需要查询的时候重连。

如果用的是5.7以后的版本。可以在执行过大查询后，执行mysql\_reset\_connection来重新初始化连接资源。

#### （3）数据库连接泄露

数据库连接泄露指的是如果在某次使用或者某段程序中没有正确地关闭 Connection、Statement 和 ResultSet 资源，那么每次执行都会留下一些没有关闭的连接，这些连接失去了引用而不能得到重新使用，因此就造成了数据库连接的泄漏。数据库连接的资源是宝贵而且是有限的，如果在某段使用频率很高的代码中出现这种泄漏，那么数据库连接资源将被耗尽，影响系统的正常运转。

黑名单机制、单位时间内请求次数限制，超过某个极值，做服务降级，建议去了解一下服务熔断服务降级。

### 3、MySQL的存储引擎

MySQL和其他关系型数据库不一样的地方在于它的弹性以及可以通过插件形式提供不同种类的存储引擎，MySQL请求处理过程会根据不同的存储引擎发生变化，比如事务性的InnoDB和非事务性的MyISAM，数据的存储和SQL的执行会产生很大的差异。

MySQL 数据库区别于其他数据库的最重要的一个特点就是其插件式的表存储引擎。需要特别注意的是，存储引擎是基于表的，而不是数据库。

# 第3章 索引篇

在面试环节中，谈及数据库问题时，不可避免地会涉及**数据库索引**知识，关于索引的概念、原理和意义等等问题，我们都该有所了解并掌握。本章主要介绍有关索引知识点的面试重点。

## 一、索引的优缺点

索引本质上是一种**数据结构**。数据库索引是DBMS中一个**排序的数据结构**，以协助快速查询、更新数据库表中数据。索引的实现通常使用B树（平衡树）以及变种B+树。

更通俗地说，索引就相当于目录，其存在是为了方便数据内容查找，本身也占用物理空间。

|  |  |
| --- | --- |
| **索引的优点** | **索引的缺点** |
| 可以加快数据的检索速度（主要原因）；  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；  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；  通过使用索引，可以在查询的过程中，使用 **优化隐藏器**，提高系统性能。 | 时间上，创建和维护索引都要耗费时间，这种时间随着数据量的增加而增加，具体地，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；  空间上，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。 |

## 二、索引的数据结构

数据库索引根据结构分类，主要有****B树索引****、****Hash索引****和****位图索引****三种。

### B树索引

B树索引，又称平衡树索引，是MySQL数据库中使用最频繁的索引类型，MySQL、Oracle 和SQL Server数据库默认的都是B树索引（实际是用B+树实现的，因为在查看表索引时，MySQL一律打印BTREE，所以简称为B树索引）。

B树索引以树结构组织，它有一个或者多个分支结点，分支结点又指向单级的叶结点。其中，分支结点用于遍历树，叶结点则保存真正的值和位置信息。

B+树是在B树基础上的一种优化，使其更适合实现外存储索引结构。

一棵m阶B-Tree的特性如下：

|  |  |
| --- | --- |
| **数据库以B-Tree的数据结构存储数据** | **数据库以B+Tree的数据结构存储数据** |
|  |  |
| 每个结点最多m个子结点；  除了根结点和叶子结点外，每个结点最少有m/2（向上取整）个子结点；  所有的叶子结点都位于同一层；  每个结点都包含k个元素（关键字），这里m/2≤k<m，这里m/2向下取整；  每个节点中的元素（关键字）从小到大排列；  每个元素子左结点的值，都小于或等于该元素，右结点的值都大于或等于该元素。 | 所有的非叶子结点只存储 关键字信息；  所有具体数据都存在叶子结点中；  所有的叶子结点中包含了全部元素的信息；  所有叶子节点之间都有一个链指针。 |

### Hash索引

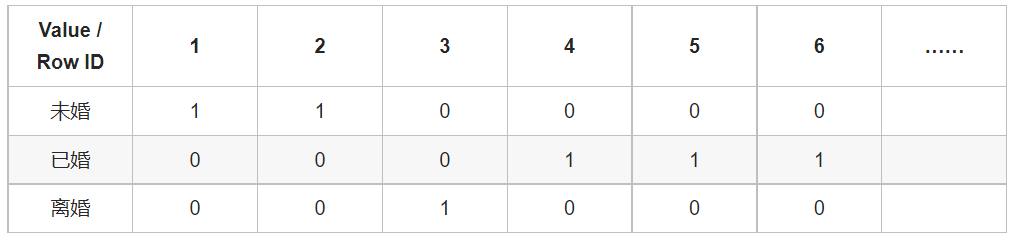
哈希索引采用一定的哈希算法（常见哈希算法有 直接定址法、平方取中法、折叠法、除数取余法、随机数法），将数据库字段数据转换成定长的Hash值，与这条数据的行指针一并存入Hash表的对应位置，如果发生Hash碰撞（两个不同关键字的Hash值相同），则在对应Hash键下以链表形式存储。

**检索时不需要类似B+树那样从根节点到叶子节点逐级查找，只需一次哈希算法即可立刻定位到相应的位置，速度非常快，平均检索时间为O(1)**。

### 位图索引

B树索引擅长于处理包含许多不同值的列，但是在处理基数较小的列时会变得很难使用。如果用户查询的列的基数非常的小， 即只有几个固定值，如性别、婚姻状况、行政区等等，要么不使用索引，查询时一行行扫描所有记录，要么考虑建立位图索引。

位图索引为存储在某列中的每个值生成一个位图。例如针对表中婚姻状况这一列，生成的位图索引大致如下所示：



位图索引适合静态数据，而不适合索引频繁更新的列。

# KV存储系统项目

使用跳表的数据结构

# 迭代日志表

|  |  |  |
| --- | --- | --- |
| **迭代版本** | **迭代工作** | **迭代日期** |
| V1.0 | 建立初稿文档，完成第1章和第2章的整理。 | 2022-02.02-02.05 |
| V1.1 |  |  |
| V1.2 |  |  |