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》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。

# 数据库基础篇

## 一、什么是数据库

**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，其实实质上是XX数据库管理系统。目前，较为流行的数据库管理系统有MySQL、SQL Server、Oracle 等。

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。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. 使用SQL 语句，可用于复杂查询； 3. 数据存储在磁盘中，安全性高 | 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 | 非关系型数据库又被称为 NoSQL（Not Only SQL )，通常指数据以 对象 的形式存储在数据库中，而对象之间的关系通过每个对象自身的属性来决定。 | 存储数据的格式可以是 key-value、文档、图片等形式，应用场景更广泛；   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）不提供 SQL 支持；2）无事务处理，无法保证数据的完整性和安全性；3）功能没有关系型数据库完善 | Neo4j，Redis，MongoDB 等 |

注：SQL是结构化查询语言（Structured Query Language）的缩写，是一种数据库查询语言，用于存取数据、查询、更新和管理关系数据库系统。与其他语言（如英语以及 Java 等程序设计语言）不一样，SQL由少量的描述性很强的词构成，简单易学。

## 为什么要使用数据库

常见的数据保存的方式：

|  |  |  |
| --- | --- | --- |
| 数据保存方式 | 优点 | 缺点 |
| 数据保存在内存 | 存取速度快 | 数据无法永久保存 |
| 数据保存在文件 | 数据可永久保存 | 查询数据不方便；操作速度比内存操作慢，频繁的 IO 操作 |
| 数据保存在数据库 | 数据可永久保存且数据安全性高；使用 SQL 语句，查询方便效率高；便于数据管理、智能化数据分析 | 数据库移植不方便；不支持集群；不擅长业务逻辑的处理 |

总而言之，面对庞大的网络数据量，使用数据库可以高效且条理分明地存储数据，它使人们能够更加迅速和方便地管理数据。

## SQL与MySQL有什么区别

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 | 更新 |
| SQL | 结构化查询语言 | 本质是一种语言，用于在数据库上执行各种操作，例如访问，更新和操作数据库中的数据，用户使用时需要学习该语言，然后编写查询。 | SQL的命令一般是稳定的 |
| 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 | 定期获得各种更新 |

## 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。

## 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。

## 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。

# 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

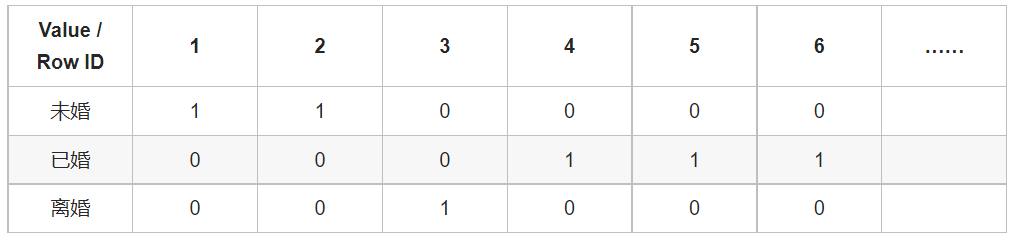
哈希索引采用一定的哈希算法（常见哈希算法有 直接定址法、平方取中法、折叠法、除数取余法、随机数法），将数据库字段数据转换成定长的Hash值，与这条数据的行指针一并存入Hash表的对应位置，如果发生Hash碰撞（两个不同关键字的Hash值相同），则在对应Hash键下以链表形式存储。

**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：



# 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 |  |  |