**复试数据库吐血整理**By赖佑

19年及之前的题库和问题整理，

主要看加粗的字，以理解为主，不要死记硬背

1. **DBMS支持那几种数据模型**

常用的是**层次模型，网状模型和关系模型**（最重要）。新兴的是面向对象数据模型和对象关系数据模型。

层次模型: 层次模型是指用**树型结构**表示实体及其之间的联系，树中每一个节点代表一个记录类型，树状结构表示实体型之间的联系。

在一个层次模型中的限制条件是：有且仅有一个节点，无父节点，此节点为树的根；其他节点有且仅有一个父节点。

网状模型的数据结构主要有以下两个特征:

(1)允许一个以上的节点无双亲。

(2)一个节点可以有多于一个的双亲。

网状模型：用**有向图结构**表示实体类型及实体间联系的数据结构模型称为网状模型

关系实际上就是关系模式在某一时刻的状态或内容。也就是说，关系模式是型，关系是它的值。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的，因为关系操作在不断地更新着数据库中的数据。但在实际当中，常常把关系模式和关系统称为关系

现实世界的实体以及实体间的各种联系均用关系来表示，从用户角度看，关系模型中数据的逻辑结构是一张**二维表**。

1. 数据模式

数据库存放数据的模式，正因为有数据模式，才能构造复杂的数据结构来建立数据之间的内在联系与复杂关系，从而构成数据的全局结构模式。

数据模式是基于选定的数据模型对数据进行的“型”的方面的刻画，而相应的“实例”则是对数据“值”的方面的描述。先有数据模型，才能据其讨论相应数据模式，有了数据模式，就能依据该模式得到相应的实例。

1. 数据库的**三级模式**

这张图表示数据库管理系统（DBMS）从三个层次来管理数据：外部层次（External Level）、概念层次（Conceptual Level）和内部层次（Internal Level）。其中**外部层次的数据是用户所看到的数据，所以又叫『用户』层次；概念层次的数据是DBMS中全局管理（可以简单地理解为所有的）数据，及数据之间的约束，所以又叫『逻辑』层次；内部层次的数据是存储在介质上的数据，包括存储路径、存储方式、索引方式等，所以又叫『物理』层次。**

而『模式』，就是『对于数据的结构性描述』，所以对于不同层次的数据，就对应着不同层次的模式，分别是：外模式（用户模式）、模式（概念模式/逻辑模式）与内模式（物理模式）。

在一个数据库系统中，**『模式』与『内模式』都只能有一个，但『外模式』可以有很多个**。原因是**『模式』是对DBMS中所有数据的全局性的结构描述**，所以只能有一种形式。举个例子：一个学校的数据库中要存放所有学生、课程与选课记录的信息。我们可以用三张表：学生信息表、课程信息表与选课记录表来存放所有的数据。但是存放学生的数据的时候，我们只能够选取某一种特定的结构对学生的信息进行描述，而不能多种不同的结构共存（比如两张不同结构的学生表，存放的都是所有学生的信息）。所以『模式』只能有一个。同样，**我们在介质上存放数据的时候，也只能选取某一种特定的存储路径（数据文件在哪个文件夹下）、存储方式（二进制/文本）、索引方式（聚簇索引，普通索引）等等**，可以理解为一种数据结构，和具体的计算机语言没什么关系，也不能够像程序一样执行。

**至于『外模式』，是以『视图』的形式展现给用户的**，所以我猜你对于外模式理解错误的原因，就是理解错误了『视图』的概念。这里的『视图』不是我们平时说的 UI ，而是指数据库中的视图，又叫『虚表』，是相对于『基本表』来说的。这里的『基本表』指的就是我们在建立数据库的时候需要创建的那些数据表（用 CREATE TABLE 创建出来的），而视图，是对这些基本表中的数据的某种展现形式，是这些数据的一个子集，但不限于某张表中的数据的子集，也可以是多张表联合起来进行显示的一组数据。比如在一个学校的教务系统中，我们希望每个学生只能够看到自己的成绩，普通老师可以看到他/她教的学生的成绩，系主任可以看到这个系中所有学生的成绩，校长可以看到学校中每个学生的成绩。那么我们可以基于所有学生的成绩数据，针对于不同权限的用户选取不同的子集，这个子集就是所谓的『视图』。但需要注意的是，『视图』本身并不『实际』存放数据，而只是对于如何选取数据的一种描述，这也是所谓的『虚』。当全局数据改变的时候，视图中的数据也会随之改变。通过外模式，可以让不同的用户（包括应用程序）看到他们希望看到的（比如可以同时看到学生本人的信息及他/她的所有成绩与对应的课程的信息，而这些数据实际上是分别存在于不同的数据表中的），或是我们希望他们看到的数据（通过不同的子集来实现）。

总而言之，三种层次的模式都是对于数据的描述，只不过描述的层次不同。

1、**模式**又称**概念模式**或逻辑模式，对应于概念级。它是由数据库设计者综合所有用户的数据，按照统一的观点构造的全局逻辑结构，是对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图(全局视图)。它是由数据库管理系统提供的数据模式描述语言(Data Description Language，DDL)来描述、定义的，体现、反映了数据库系统的整体观。

2、**外模式**又称子模式或**用户模式**，对应于用户级。它是某个或某几个用户所看到的数据库的数据视图，是与某一应用有关的数据的逻辑表示。外模式是从模式导出的一个子集，包含模式中允许特定用户使用的那部分数据。用户可以通过外模式描述语言来描述、定义对应于用户的数据记录(外模式)，也可以利用数据操纵语言(Data Manipulation Language，DML)对这些数据记录进行。外模式反映了数据库的用户观。

3、**内模式**又称**存储模式**，对应于物理级，它是数据库中全体数据的内部表示或底层描述，是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式和物理结构，对应着实际存储在外存储介质上的数据库。内模式由内模式描述语言来描述、定义，它是数据库的存储观。

在一个数据库系统中，只有唯一的数据库，因而作为定义、描述数据库存储结构的内模式和定义、描述数据库逻辑结构的模式，也是唯一的，但建立在数据库系统之上的应用则是非常广泛、多样的，所以对应的外模式不是唯一的，也不可能是唯一的。

其中**内模式处于最低层**，它反映了**数据在计算机物理结构中的实际存储形式**，**概念模式处于中层**，它反映了设计者的**数据全局逻辑要求**，而**外模式处于最外层**，它反映了**用户对数据的要求**。

1. **SQL的四个组成部分**？

1、数据**定义**语言（DDL）；DROP 、CREATE 、ALTER

2、数据**查询**语言（QL）；SELECT

3、数据**操纵**语言（DML）；INSERT(插入) UPDATE(修改) DELETE(删除)

4、数据**控制**语言（DCL）。GRANT  、REVOKE、 COMMIT 、ROLLBACK

数据库操纵语言举例

数据操纵语言DML（Data Manipulation Language），用户通过它可以实现对数据库的基本操作。例如，对表中数据的查询、插入、删除和修改

什么是数据库控制语言,试举例说明

1. 介绍下有哪些**应用数据库**？

桌面型：Access、Foxpro、DB2

企业型：SQL Server系列、MySQL、Oracle、Sybase

1. 什么是**数据独立性**。

数据独立性表示应用程序与数据库中存储的数据不存在依赖关系，包括逻辑数据独立性和物理数据独立性。

**数据与程序的的逻辑独立性**：

**当模式改变时**（例如增加新的关系，新的属性，改变属性的数据类型等），由数据库管理员对各个**外模式/模式的映像**做相应的改变，可以使外模式保持不变。应用程序是依据数据的外模式编写的，从而**应用程序不必修改**，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。

**数据与程序的物理独立性**：

当**数据库的存储结构改变时**，由数据库管理员对**模式/内模式的映像**做相应的改变，可以使模式保持不变，从而应用程序也不必改变，保证了数据与程序的物理独立性，简称数据的物理独立性。

数据库管理系统在三级模式之间提供的两层映像保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性

1. **两种独立性**

物理独立性指的是应用程序与数据存放在相互独立的磁盘地址，**内模式与模式映象**保证了其物理独立特性

逻辑独立性指的数据与程序逻辑结构上的独立特性，数据或应用程序的逻辑结构发生变化性都不影响对方，**外模式与模式映象**保证了其逻辑独立性。

1. 范式是啥

第一范式 是数据库最基本的要求，即**属性不可分**

第二范式 在1NF基础上**消除**数据库非主属性对码的**部分函数依赖**

第三范式 在2NF基础上**消除**非主属性对码的**传递函数依赖**

主属性：在候选码中出现过

非主属性：没在候选码中出现过

BCNF:

1. **关系数据库都有哪些操作**，特点是什么？（基本的关系操作有哪些？）

◇**查询**:选择、投影、连接、除、并、交、差

◇**数据更新**:插入、删除、修改

关系操作的特点:集合操作方式，即操作的对象和结果都是**集合**。相应地，非关系数据模型的数据操作方式则为一次一记录的方式。

1. 数据库系统中的**常见故障**有哪些?（数据库故障的种类？数据库有哪几种故障及各自的恢复策略。数据库恢复技术有哪些）

1、事务内部的故障2、系统故障3。介质故障4。计算机病毒

一、事务故障  
什么是**事务故障**：**某个事务在运行过程中由于种种原因未运行至正常终止点**事务故障的常见原因：**输入数据有误、运算溢出、违反了某些完整性限制、某些应用程序出错、并行事务发生死锁**

事务故障的恢复：**事务撤消**（UND）  
恢复程序要在不影响其它事务运行的情况下，强行回滚（RBACK）该事务，即清除该事务对数据库的所有修改，使得这个事务象根本没有启动过一样

### 系统恢复的步骤：

#### （1）反向扫描日志文件（即从最后向前扫描日志文件），查找该事物的更新操作。

#### （2）对该事务的更新操作执行逆操作。即将日志记录中"更新前的值"写入数据库。这样如果记录中是插入操作，则相当于做删除操作；若记录中是删除操作，则做插入操作；若是修改，则相当于修改前值代替修改后值。

#### （3）继续反向扫描日志文件，查找该事务的其他更新操作，并作同样处理。

#### （4）如此处理下去，直至读到此事务的开始标记，事务故障恢复就完成了。

二、系统故障  
什么是**系统故障**：由于**某种原因造成整个系统的正常运行突然停止，致使所有正在运行的事务都以非正常方式终止**。发生系统故障时，内**存中数据库缓冲区的信息全部丢失，但存储在外部存储设备上的数据未受影响**系统故障的常见原因：**操作系统或DBMS 代码错误、操作员操作失误、特定类型的硬件错误（如CPU 故障）、突然停电**  
系统故障的恢复  
1. 清除尚未完成的事务对数据库的所有修改  
如果DBMS 无法确定哪些事务已更新过数据库，则系统重新启动后，恢复程序要**强行撤消所有未完成事务**，使这些事务象没有运行过一样。  
2. 将已完成事务提交的结果写入数据库  
如果DBMS 无法确定哪些事务的提交结果尚未写入物理数据库，则系统重新启动后，恢复程序需要**重做所有已提交的事务**。

### 系统的恢复步骤是：

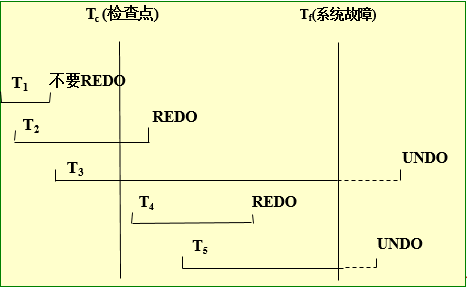
#### （1）正向扫描日志文件（即从头扫描日志文件），找出在故障发生前已经提交的事务（这些事务既有BEGIN TRANSACTION记录，也有COMMIT记录），将其事务标识记入重做（REDO）队列。同时找出故障发生时尚未完成的事务（这些事务只有BEGIN TRANSACTION记录，无相应的COMMIT记录）,将其事务标识记入撤销队列。

#### （2）对撤销队列中的各个事务进行撤销（UNDO）处理。

#### 进行UNDO处理的方法是，反向扫描日志文件，对每一个UNDO事务的更新操作执行逆操作，将将日志记录中"更新前的值"写入数据库（该方法和事务故障的解决方法一致）。

#### （3）对重做队列中的各个事务进行重做（REDO）处理。

#### 进行REDO处理的方法是：正向扫描日志文件，对每一个REDO事务从新执行日志文件登记的操作。即将日志记录中"更新后的值"写入数据库。



## 检查点的理解

## ###首先查看有几个检查点，找到里故障最近的一次检查点，之后看看哪些事务执行检查的时间是在检查点之前完成的 （1）如果事务在点之前完成的则不需要REDO（如:T1） （2）如果事务在检查点之前执行，在检查点之后、故障之前提交则需要REDO(如：T2,T4) （3）如果事务在故障后，则需要执行UNDO(如：T3,T5)

三、介质故障  
什么是**介质故障**：**硬件故障使存储在外存中的数据部分丢失或全部丢失**  
介质故障比前两类故障的可能性小得多，但破坏性最大。

介质故障的常见原因：**硬件故障、磁盘损坏、磁头碰撞**操作系统的某种潜在错误  
瞬时强磁场干扰  
介质故障的**恢复**  
**装入数据库发生介质故障前某个时刻的数据副本**  
**重做自此时始的所有成功事务 ，将这些事务已提交的结果重新记入数据库**故障的种类小结  
数据库系统中各类故障对数据库的影响  
数据库本身被破坏 （介质故障）  
数据库处于不一致状态  
数据库中包含了未完成事务对数据库的修改（事务故障、系统故障）  
数据库中丢失了已提交事务对数据库的修改（系统故障）  
  
不同类型的故障应采用不同的恢复操作  
故障的种类小结（续）  
恢复操作的基本原理：简单  
原理：利用 存储在系统其它地方的冗余数据 来重建 数据库中已经被破坏或已经不正确的那部分数据

1. **数据库恢复策略**有哪几种？

1、事务故障：

发生事务故障时，被迫中断的事务可能已对数据库进行丁修改，为了消除该事务对数据库的影响，要利用日志文件中所记载的信息，强行回滚该事务，将数据库恢复到修改前的初始状态。

为此，要检查日志文件中由这些事务所引起的发生变化的记录，取消这些没有完成的事务所做的一切改变，这类恢复操作称为事务撤销。

2、系统故障：

系统故障的恢复要完成两方面的工作，既要撤销所有末完成的事务，还要重做所有已提交的事务，这样才能将数据库真正恢复到一致的状态。

3、介质故障：

这需要装入发生介质故障前最新的后备数据库副本，然后利用日志文件重做该副本后所运行的所有事务。

1. 数据转存（数据冗余）2.日志文件
2. **DB并发操作通常带来哪三类问题**？

**丢失修改**：**当一个事务修改了数据，并且这种修改还没有还没有提交到数据库中时，另外一个事务又对同样的数据进行了修改，并且把这种修改提交到了数据库中**。这样，数据库中没有出现第一个事务修改数据的结果，好像这种数据修改丢失了一样。

**脏读**：当一个事务正在访问数据，并对数据进行了修改，**而这种修改还没有提交到数据库中，这时，另一个事务也访问这个数据，然后使用了这个数据**。因为这个数据是还没有提交的数据，那么另一个事务读到的这个数据是脏数据，依据脏数据所做的操作可能是不正确的。

**不可重复读**：在一个事务内，多次读同一数据。在这个事务还没有结束时，另一个事务也访问该同一数据，那么，**在第一个事务中的两次读数据之间，由于第二个事务的修改，第一个事务两次读到的数据可能是不一样的**。

1. 数据库：数据库**采用什么机制来保证数据库并发操作的正确性**。------

### ****并发控制的主要技术****

* **封锁**(Locking)
* **时间戳：**这时可以给该表添加一个timestamp的字段，当添加、修改数据的时候，timestamp类型会自动生成一个新值，此时我们把timestamp字段的值也作为update的where子条件之一就可以避免覆盖其他用户的修改行为了。
* **乐观控制法：**认为事务执行时很少发生冲突，因此不对事务进行特殊的管制，而是让它自由执行，事务提交前再进行正确性检查。如果检查后发现该事务执行中出现过冲突并影响了可串行性，则拒绝提交并回滚该事务。乐观锁机制在一定程度上解决了这个问题。乐观锁，大多是基于数据版本
* （ Version ）记录机制实现。何谓数据版本？即为数据增加一个版本标识，在基于
* 数据库表的版本解决方案中，一般是通过为数据库表增加一个 “version” 字段来
* 实现。
* 读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。此时，将提
* 交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果提交的数据
* 版本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。
* 对于上面修改用户帐户信息的例子而言，假设数据库中帐户信息表中有一个
* version 字段，当前值为 1 ；而当前帐户余额字段（ balance ）为 $100 。
* 操作员 A 此时将其读出（ version=1 ），并从其帐户余额中扣除 $50
* （ $100-$50 ）。
* 在操作员 A 操作的过程中，操作员 B 也读入此用户信息（ version=1 ），并
* 从其帐户余额中扣除 $20 （ $100-$20 ）。
* 操作员 A 完成了修改工作，将数据版本号加一（ version=2 ），连同帐户扣
* 除后余额（ balance=$50 ），提交至数据库更新，此时由于提交数据版本大
* 于数据库记录当前版本，数据被更新，数据库记录 version 更新为 2 。
* 操作员 B 完成了操作，也将版本号加一（ version=2 ）试图向数据库提交数
* 据（ balance=$80 ），但此时比对数据库记录版本时发现，操作员 B 提交的
* 数据版本号为 2 ，数据库记录当前版本(再查一遍该记录)也为 2 ，不满足 “ 提交版本必须大于记
* 录当前版本才能执行更新 “ 的乐观锁策略，因此，操作员 B 的提交被驳回。
* 这样，就避免了操作员 B 用基于 version=1 的旧数据修改的结果覆盖操作
* 员 A 的操作结果的可能。
* 从上面的例子可以看出，乐观锁机制避免了长事务中的数据库加锁开销（操作员 A
* 和操作员 B 操作过程中，都没有对数据库数据加锁），大大提升了大并发量下的系
* 统整体性能表现。

**多版本并发控制：**在数据库中通过维护数据对象的多个版本信息来实现高效并发控制的一种策略。版本是指数据库中数据对象的一个快照，记录了数据对象某个时刻的状态。随着计算机系统存储设备价格的不断降低，可以考虑为数据库系统的数据对象保留多个版本，以提高系统的并发操作程度。

1. **两段锁协议**

所谓两段锁协议是指所有事务必须**分两个阶段对数据项加锁和解锁**：

1. **在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁**，而且

2. **在任一解锁之后，事务不再申请和获得任何其他加锁**。

所谓“两段”锁的含义是，事务分为两个阶段，第一阶段是获得封锁，也称为扩展阶段。在这阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁。第二阶段是释放封锁，也称为收缩阶段。在这阶段，事务可以释放任何数据项上的任何类型的锁，但是不能再申请任何锁。

（可以证明，若并发执行的所有事务均遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的。）

另外要注意两段锁协议和**防止死锁的一次封锁法**的异同之处。一次封锁法要求**每个事务必须一次将所有要使用的数据全部加锁**，否则就不能继续执行，因此一次封锁法遵守两段锁协议；但是**两段锁协议并不要求事务必须一次将所有要使用的数据全部加锁，因此遵守两段锁协议的事务可能发生死锁。**

**加锁阶段**：在该阶段可以进行加锁操作。在对任何数据进行读操作之前要申请并获得S锁，在进行写操作之前要申请并获得X锁。加锁不成功，则事务进入等待状态，直到加锁成功才继续执行。

**解锁阶段**：当事务释放了一个封锁以后，事务进入解锁阶段，在该阶段只能进行解锁操作不能再进行加锁操作。

两段封锁法可以这样来实现：事务开始后就处于加锁阶段，一直到执行ROLLBACK和COMMIT之前都是加锁阶段。ROLLBACK和COMMIT使事务进入解锁阶段，即在ROLLBACK和COMMIT模块中DBMS释放所有封锁。

1. 在对任何数据进行读、写操作之前，要申请并获得对该数据的封锁。

2. **每个事务中，所有的封锁请求先于所有的解锁请求**。

1. 事务的四个特点
2. 原子性2。一致性3。隔离性4。持续性

事务的概念：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。

事务具有四个特征：原子性（ Atomicity ）、一致性（ Consistency ）、隔离性（ Isolation ）和持续性（ Durability ）。这四个特性简称为 ACID 特性。  
1 、**原子性**  
事务是数据库的逻辑工作单位，**事务中包含的各操作要么都做，要么都不做**  
2 、**一致性**  
**事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态**。**因此当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态**。如果数据库系统 运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是 不一致的状态。  
3 、**隔离性**  
**一个事务的执行不能其它事务干扰**。即一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰。  
4 、**持续性**  
也称永久性，**指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的**。接下来的其它操作或故障不应该对其执行结果有任何影响。

1. SQL主键约束和唯一约束有什么区别呢？（数据库：primary key与unique的不同）

1.**主键约束**  
1) 主键用于唯一地标识表中的每一条记录，可以**定义一列或多列为主键**。  
2) 是**不可能（或很难）更新**．  
3) 主键**不允许重复**，**不允许空**（NULL）．  
4) 主健**可作外健**  
2.**唯一性约束**  
1) 唯一性约束用来限制不受主键约束的列上的数据的唯一性，用于作为访问某行的可选手段，**一个表上可以放置多个唯一性约束**．  
2) **只要唯一就可以更新**．  
3) 即表中任意两行在指定列上都不允许重复，**允许空**（NULL）．

相同点：都不允许**重复**

不同点：唯一性约束允许在该列上存在**空值**，主键约束不允许有空值

1. 什么是**表**？什么是**视图**？两者的区别和联系是什么？

表是实实在在得**保存数据的实体**，写入的数据都保存在表中，而视图是不保存数据的，也没有数据。**视图就是sql语句**，实际上视图从表中去取数据。表可以建立各种触发器，可以建立索引，可以建立主健，约束等。但是视图不能建立这些对象(视图可以建立替代触发器)。**表和视图可以更新**，但是**视图的更新受到约束**。比如，group by和表连接生成的视图不能更新。

联系：视图中的数据来源于基本表，都可以进行查询

区别：1.基本**表中存储实际的数据**，而**视图只存储定义，不存放实际数据**；

2.对基本表可以进行任意的**更新**操作，而对视图的更新是受限的。一般来说你可以用update，insert，delete等sql语句修改表中的数据，而对视图只能进行select操作。但是也存在可更新的视图，对于这类视图的update，insert和delete等操作最终会作用于与其相关的表中数据。

3.**视图的建立和删除只影响视图本身，不影响对应的基本表**

4.从**安全的角度**说，视图可以不给用户接触数据表，从而不知道表结构。

1. 数据库中**视图的优点**

**简单性**。视图不仅可以**简化用户对数据的理解，也可以简化他们的操作**。那些被经常使用的查询可以被定义为视图，从而使用户不必为以后的操作每次都指定全部的条件。当一个查询你需要频频的作为子查询使用时，视图可以简化代码，直接调用而不是每次都去重复写这个东西。

● **安全性**。**通过视图用户只能查询和修改他们所能见到的数据**。数据库中的其他数据则既看不见也取不到。数据库授权命令可以使每个用户对数据库的检索限制到特定的数据库对象上，但不能授权到数据库特定行和特定的列上。通过视图，用户可以被限制在数据的不同子集上。

1。可以**访问表中列的子集**。在表中有些列是比较敏感的数据不想用户看到的，如用户密码，员工工资等，用视图可以隐藏这些列。

2。可以访问表中**行的子集**。有时不想让用户看到与TA无关的数据时，可以在where条件中过滤，如企业中子公司的员工只能看到TA所在子公司的同事资料，而不想其它子公司员工资料也显示出来，用过滤后查询出来的数据集组成视图。

● **逻辑数据独立性**。视图可以**使应用程序和数据库表在一定程度上独立**。如果没有视图，应用一定是建立在表上的。有了视图之后，程序可以建立在视图之上，从而程序与数据库表被视图分割开来。

1. **哪些视图是可以更新、哪些是不可以更新的**，请举例说明

1：**简单视图**就是由一个表生成出来的视图，这种情况你更新她就和更新表一样

2：二次加工出来的简单视图仍然是一个表出来的视图，但是视图中存在通过函数或计算二次加工出来的其他字段。更新的时候只要不更新这些加工出来的字段也是可以更新的。

3：组合视图通过表之间关联联合等出来的复杂视图。这种视图更新的时候要注意你**所更新的列要来自同一个表**，也是可以更新的。

4：静态视图：这种视图等同于表可以直接更新，但是更新的数据仅在视图中反映出来，不反映到原表

5：其他视图通过表函数等其他生成的更为复杂的视图。一般不可更新

**教材：如果定义视图的查询满足下列条件，就可更新：（p72）**

1. **from子句只有一个数据库关系**
2. **Select**子句中只包含关系的属性名，**不包含任何表达式、聚集、distinct声明**
3. 任**何没有出现在select子句中的属性可以取空值**；即这些属性上**没有not null约束，也不构成主码的一部分**。
4. 查询中**不含有group by 或having子句**。
5. **1NF有哪些缺点**？

1，**数据冗余太大**（每一个系的系主任名字重复出现）

2，**更新异常**（某个系更换系主任之后，必须将该系学生有关的每一个元组都改变系主任名字）

3，**插入异常**（如果一个系刚成立，尚无学生，就无法将老师存入到数据库中）

4，**删除异常**（如果学生都毕业了，在删除学生时，把这个系的系主任的所有信息都删除了）

1. **数据字典**通常包含五个部分

**1。数据项2。数据结构3。数据流4。数据存储5。处理过程**

定义：数据字典是关于数据库中数据的描述，即元数据，而不是数据本身。  
它是进行详细的数据收集和数据分析所得的成果。数据字典不单单是对数据库中表的描述，所有数据库中的数据都可以通过数据字典找到相关定义

1. **数据模型的三要素**

数据结构、数据操作、完整性约束条件

1。 **数据结构**

数据结构是**所研究的对象类型的集合**。这些对象是数据库的组成成分，数据结构指对象和对象间联系的表达和实现，是对系统静态特征的描述，包括两个方面：

（1）**数据本身**：类型、内容、性质。例如关系模型中的域、属性、关系等。

（2）**数据之间的联系**：数据之间是如何相互关联的，例如关系模型中的主码、外码联系等。

2 。**数据操作**

对数据库中对象的实例允许**执行的操作集合**，主要指**检索和更新（插入、删除、修改）**两类操作。数据模型必须定义这些操作的确切含义、操作符号、操作规则（如优先级）以及实现操作的语言。数据操作是对系统动态特性的描述。

3 。**数据完整性约束**

数据完整性约束是一组**完整性规则的集合**，规定数据库状态及状态变化所应满足的条件，以保证数据的**正确性、有效性和相容性**。

1. **数据库索引**

目的：提供多种存储路径，**加快查找速度**。建立索引需要考虑的问题：1。没有查询、统计的需要则不建2。**数据增删改频繁，系统会花费许多时间来维护索引**

**什么场景不适合创建索引**

第一，对于那些**在查询中很少使用或者参考的列**不应该创建索引。这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。

第二，对于那些**只有很少数据值的列**也不应该增加索引。因为本来结果集合就是相当于全表查询了，所以没有必要。这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比 例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。

第三，**对于那些定义为text, image和bit数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少。**

第四，**当修改性能远远大于检索性能时，不应该创建索引**。这是因为，修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因 此，当修改性能远远大于检索性能时，不应该创建索引。

第五，不会出现在where条件中的字段不该建立索引。

1.3、**什么样的字段适合创建索引**

1、表的主键、外键必须有索引；**外键是唯一的，而且经常会用来查询**

2、**数据量超过300的表应该有索引**；

3、经常与其他表进行连接的表，在连接字段上应该建立索引；**经常连接查询，需要有索引**

4、经常出现在Where子句中的字段，加快判断速度，特别是大表的字段，应该建立索引，建立索引，一般用在select ……where f1 and f2 ，我们在f1或者f2上建立索引是没用的。只有两个使用**联合索引**才能有用

5、**经常用到排序的列上**，因为索引已经排序。

6、**经常用在范围内搜索的列上创建索引**，因为索引已经排序了，其指定的范围是连续的

**索引优缺点**

2.1、优点

索引由数据库中一列或多列组合而成，其作用是**提高对表中数据的查询速度**

索引的优点是可以提高检索数据的速度

2.2、缺点

索引的缺点是**创建和维护索引需要耗费时间**

索引可以提高查询速度，会**降低更新表的速度**

建立索引会占用磁盘空间的索引文件。

**索引分类：普通索引、唯一索引、全文索引、组合索引。**

**聚集–非聚集**：聚集索引：对正文内容按照一定规则排列的目录称为聚集索引

非聚集索引：目录自己按照一定规则排列，正文自己按照另一种规则排列，目录主要是保存对正文的一个映射关系，这种称为非聚集索引

1. **日志文件**的定义与作用。

日志文件是用来**记录事务对数据库的更新操作的文件**。作用：在**数据库恢复**中起着重要的作用。并协助后备副本进行介质故障恢复。

1. **数据库保护**(访问)的内容有哪些？

1.利用**权限**机制，2.利用**完整性约束**防止非法数据进入数据库，

3.提供**故障恢复**能力， 4.提供**并发访问控制**。

1. 什么是**完整性约束**（关系完整性包括哪三个方面）

数据完整性约束指的是为了防止不符合规范的数据进入数据库，在用户对数据进行插入、修改、删除等操作时，DBMS自动按照一定的约束条件对数据进行监测，使不符合规范的数据不能进入数据库，以确保数据库中存储的数据正确、有效、相容。

完整性约束可确保数据库中的数据正确性和相容性。主要包括：实体完整性，域完整性，参照完整性，用户自定义完整性。

**1) 实体完整性**：规定表的每一行在表中是惟一的实体。（每个数据表都必须有**主键**，而作为主键的所有字段，其属性必须是独一及非空值）

**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（**列的值域**的完整性。如数据类型、格式、值域范围、是否允许空值等）

**3) 参照完整性**：是指**两个表的**[**主关键字**](https://baike.baidu.com/item/%E4%B8%BB%E5%85%B3%E9%94%AE%E5%AD%97)**和**[**外关键字**](https://baike.baidu.com/item/%E5%A4%96%E5%85%B3%E9%94%AE%E5%AD%97/1239625)**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。（参照的关系中的属性值必须能够在被参照关系找到或者取空值）

**4) 用户定义的完整性**：不同的[关系数据库系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F)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。

1. 数据库管理系统的**完整性**要完成**的功能** （在DBMS中，在完整性方面需要完成哪些任务？）

-- 防止不合语意的数据，防止无效操作和错误结果

1. **数据库完整性与安全性**的区别？

完整性和安全性是两个不同的概念。**前者是为了防止数据库中存在不符合语义的数据，防止错误信息的输入和输出造成的无效操作和错误结果，**   **后者是防止数据库被恶意的破坏和非法的存取**。当然，完整性和安全性是密切相关的。特别是从系统实现的方法来看，某**一种机制常常既可以用于安全保护亦可用于完整性保证**。

## 数据库完整性

数据库中数据的完整性：指**数据的正确、有效、相容**。

**数据库管理系统采取的方法**：

* 提供**完整性约束条件（规则）的定义机制** 实体 参照 用户完整性约束
* 提供完整性检查的方法 数据修改时由DBMS自动完成
* 进行违约处理：**拒绝操作**
* **断言**创建以后，任何**对断言中所涉及关系的操作都会触发**关系数据库管理系统对断言的检查，任何使断言不为真值的操作都会被**拒绝执行**。

## 安全性

数据库的安全性保护数据库以防止不合法使用所造成的**数据泄漏、更改或破坏**

**数据库的不安全因素**：

* 非授权用户对数据库的恶意存取和破坏 ---**用户身份鉴定 存取控制 视图**
* 数据库中重要或敏感的数据被泄露 ---**强制存取控制、数据加密存储、加密传输。审计**
* 安全环境的脆弱 **与操作系统的安全性、网络协议安全保障有关**

1. 数据库有哪几种**备份策略**，大概的操作

1.**全量数据备份**   
   备份整个数据库，恢复时恢复所有。优点是简单，缺点是数据量太大，非常耗时

2.**增量数据备份**(Differential Backups)   
    所谓增量，就是以某个起始时间点的全量数据为基础，备份该时间点以后的数据。而起始时间点的全量数据，就是通过全量备份而为的。   
   如果有人告诉你“每周一进行全量备份，每天进行一次增量备份。”，这就意味着，星期一作一次全量配份，形成一个起始时间点的全量数据；星期二备份星期一以来的数据；星期三也备份星期一以来的数据；.......星期天也备份星期一以来的数据。到第二周的星期一时，又执行一次全量配份，再开始新的备份周期。   
   如果要恢复星期三的数据，则要先恢复星期一的全量数据，然后再恢复在星期一到星期三之间的增量数据。

3.**日志备份**   
   周一做一次全量数据备份，周二时备份 周一至周二 的日志，周三时配份 周二至周三 的日志......。   
   若要恢复周三的数据，则先恢复到周一的全量数据，再按 周一至周二的日志、 周二至周三的日志 进行数据库操作

一个事务日志备份只备份事务日志中的信息。事务日志备份必须与至少一次全数据库备份联用，这是因为如果恢复数据，必须要有一个开始点。事务日志备份还允许你实施一种特殊类型的恢复，即时间点恢复。

1. **数据库系统安全和计算机系统安全**的关系

安全性问题不是数据库系统独有的，所有计算机系统都有这个问题。只是在数据库系统中**大量数据集中存放**，而且**为许多最终用户直接共享**，从而使**安全性问题更为突出**。

**系统安全保护措施是否有效是数据库系统的主要指标之一**。

数据库的安全性和计算机系统的安全性，包括操作系统，网络系统的安全性是**紧密联系，相互支持的**。

1. 数据库的**笛卡尔积**：对于A中的每一个元素，都有对于在B中的所有元素做连接运算 。可以见得对于两个元组分别为m，n的表。笛卡尔积 后得到的元组个数为m x n个元组。
2. **DBA的职责**是什么？

DBA则是这个机构的一个(组)人员，负责全面管理和控制数据库系统.职责有:

(1)决定数据库中的**信息内容和结构**:数据库中要存放哪些信息

(2)决定数据库的**存储结构和存取策略**:获得较高的存取效率和存储空间利用率

(3)定义数据的**安全性要求和完整性约束条件**:负责确定各个用户对数据库的存取权限、数据的保密级别和完整性约束条件

(4)**监控数据库的使用和运行**:转储数据、维护日志文件、故障恢复

(5)数据库的**改进和重组重构**:对运行情况进行记录、统计分析，以此来不断改进数据库设计

数据库管理员的主要职责有以下几个方面：

1.设计：**数据库设计**，包括字段、表和关键字段；资源在辅助存储设备上是怎样使用的，怎样增加和删除文件及记录，以及怎样发现和补救损失。

2.监视：**监控数据库的警告日志，定期做备份删除**。灾难出现时对数据库信息进行恢复。监控数据库的日常会话情况。碎片、剩余表空间监控，及时了解表空间的扩展情况、以及剩余空间分布情况。

3.**备份**：对数据库的备份监控和管理数据库的备份至关重要。

4.修改密码：规范数据库用户的管理，定**期对管理员等重要用户密码进行修改。**

5.SQL语句：对SQL语句的**书写规范的要求一个SQL语句**，如果写得不理想，对数据库的影响是很大的。所以，每一个程序员或相应的工作人员在写相应的SQL语句时，应该严格按照《SQL书写规范》一文,最后要有DBA检查才可以正式运行。

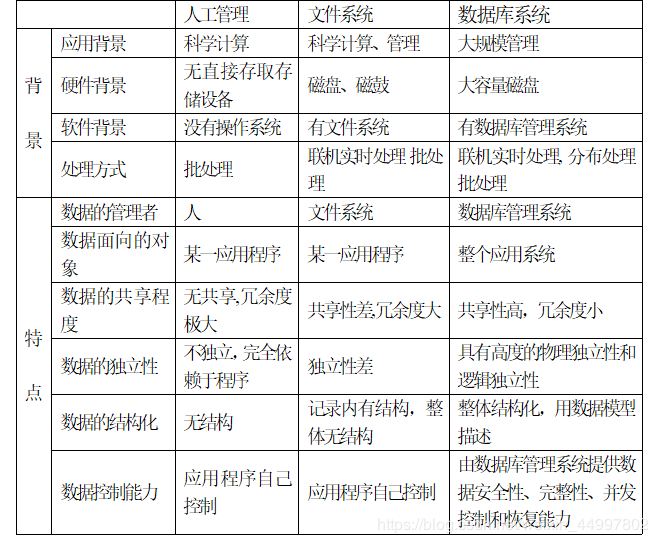
6.最终用户服务和协调：数据库管理员**规定用户访问权限和为不同用户组分配资源**。

7.数据库安全：数据库管理员能够为不同的数据库管理系统用户规定不同的访问权限，**以保护数据库不被未经授权的访问和破坏**。例如，允许一类用户只能检索数据，而另一类用户可能拥有更新数据和删除记录的权限。

1. **数据库系统和文件系统相比**有什么优点？

1、数据库设计时面向数据模型对象,**数据库设计的基础是数据模型**。2、数据库系统的数据**冗余度小**、数据**共享度高** 3、数据库系统的**数据和程序之间具有较高的独立性（物理独立性，逻辑独立性**）4、数据库系统**通过DBMS进行数据安全性和完整性的控制**5、数据库中数据的最小存取单位是数据项

|  |  |
| --- | --- |
| 文件系统 | 数据库管理系统 |
| 某一应用 | 现实世界 |
| 共享性差，冗余度大 | **共享性高，冗余度小** |
| 记录内有结构，整体无结构 | **整体结构化，用数据模型描述** |
| 应用程序自己控制 | **由数据库管理系统提供数据安全性，完整性，并发控制和恢复能力** |
| 独立性差 | **具有高度的物理独立性和一定的逻辑独立性** |



**数据库系统DBS的四个特点**：

1，**整体数据结构化**

数据库中的任何数据都不属于任何应用，数据是公共的，结构是全面的。在数据库中，数据文件的个数是有限的，固定的，但数据库系统的应用却是无限制的。

好处：整体数据的结构化可减少乃至消除不必要的数据冗余，因此节约了整体数据的存储空间，避免了数据的不一致性和不相容性（数据不符合规定的约束条件）。

2，**数据的共享度高**

数据与数据的逻辑结构同时存储在数据库中，显示数据时，可同时显示数据的逻辑结构；整个组织的整体数据被综合考虑，整体数据结构化。因而，DBS的数据共享度较高。

好处：合法用户都可以方便的访问使用数据库中的数据，且不用担心出现数据的不一致性和不相容性。数据库中的数据可适应各种合法用户的合理要求以及各种应用的要求，可以方便的扩充新的应用。

3，数据的**独立性高**

数据的独立性是指数据与应用程序之间的关联性。数据与数据的结构是存储在数据库中的（在外存上），由DBMS管理的。应用程序既不存储数据，也不存储数据的逻辑结构。

物理独立性：数据库中数据的世纪存储方式改变时，DBMS可以适当改变转换数据的方式，使用户面对数据的逻辑结构保持不变，从而处理数据的应用程序也保持不变。

逻辑独立性：数据库中数据的逻辑结构发生变化时，DBMS可以适当改变数据的转换方式，用户面对数据的逻辑结构保持不变。

好处：数据与程序相互独立，可以方便的编制各种应用程序，大大减轻应用程序的维护工作。

4，高度的**数据控制能力**

a,较高的**安全性**

b,较好的**数据完整性**

c,较强的**并发控制能力**

d,较强的**数据恢复**能力

## 数据库管理系统[DBMS主要功能](https://www.cnblogs.com/wjdtyp/p/4003011.html)

1.数据定义：DBMS提供数据定义语言DDL（Data Definition Language），供用户定义数据库的三级模式结构、两级映像以及完整性约束和保密限制等约束。DDL主要用于建立、修改数据库的库结构。DDL所描述的库结构仅仅给出了数据库的框架，数据库的框架信息被存放在数据字典（Data Dictionary）中。

2.数据操作：DBMS提供数据操作语言DML（Data Manipulation Language），供用户实现对数据的追加、删除、更新、查询等操作。

3.数据库的运行管理：数据库的运行管理功能是DBMS的运行控制、管理功能，包括多用户环境下的并发控制、安全性检查和存取限制控制、完整性检查和执行、运行日志的组织管理、事务的管理和自动恢复，即保证事务的原子性。这些功能保证了数据库系统的正常运行。

4.数据组织、存储与管理：DBMS要分类组织、存储和管理各种数据，包括数据字典、用户数据、存取路径等，需确定以何种文件结构和存取方式在存储级上组织这些数据，如何实现数据之间的联系。数据组织和存储的基本目标是提高存储空间利用率，选择合适的存取方法提高存取效率。

5.数据库的保护：数据库中的数据是信息社会的战略资源，所以数据的保护至关重要。DBMS对数据库的保护通过4个方面来实现：数据库的恢复、数据库的并发控制、数据库的完整性控制、数据库安全性控制。DBMS的其他保护功能还有系统缓冲区的管理以及数据存储的某些自适应调节机制等。

6.数据库的维护：这一部分包括数据库的数据载入、转换、转储、数据库的重组合重构以及性能监控等功能，这些功能分别由各个使用程序来完成。

7.通信：DBMS具有与操作系统的联机处理、分时系统及远程作业输入的相关接口，负责处理数据的传送。对网络环境下的数据库系统，还应该包括DBMS与网络中其他软件系统的通信功能以及数据库之间的互操作功能。

1. DBMS控制管理功能
2. 数据库：DBMS的控制操作有哪些
3. 附加：函数模板和类模板的区别

函数模板：

在C++中，数据的类型也可以通过参数来传递，在函数定义时可以不指明具体的数据类型，当发生函数调用时，编译器可以根据传入的实参自动推断数据类型。这就是类型的参数化。  
值（Value）和类型（Type）是数据的两个主要特征，它们在C++中都可以被参数化。  
**所谓函数模板，实际上是建立一个通用函数，它所用到的数据的类型（包括返回值类型、形参类型、局部变量类型）可以不具体指定，而是用一个虚拟的类型来代替（实际上是用一个标识符来占位），等发生函数调用时再根据传入的实参来逆推出真正的类型。这个通用函数就称为函数模板**（Function Template）。  
在函数模板中，数据的值和类型都被参数化了，发生函数调用时编译器会根据传入的实参来推演形参的值和类型。换个角度说，函数模板除了支持值的参数化，还支持类型的参数化。  
一但定义了函数模板，就可以将类型参数用于函数定义和函数声明了。说得直白一点，原来使用 int、float、char 等内置类型的地方，都可以用类型参数来代替。

类模板：

**是对一批仅仅成员数据类型不同的类的抽象**，程序员只要为这一批类所组成的整个类家族创建一个类模板，给出一套程序代码，就可以用来生成多种具体的类，（这类可以看作是类模板的实例），从而大大提高编程的效率。

类模板是一个类家族的抽象，它只是对类的描述，[编译程序](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E7%A8%8B%E5%BA%8F)不为类模板(包括成员函数定义)创建程序代码，但是**通过对类模板的实例化可以生成一个具体的类以及该具体类的对象**。

与[函数模板](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E6%A8%A1%E6%9D%BF)不同的是：**函数模板的实例化**是由编译程序在处理[**函数调用**](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E8%B0%83%E7%94%A8)**时自动完成的**，而**类模板的实例化必须由程序员在程序中显式地指定**，

# -- 实例化不同。 [共享锁（读锁）和排他锁（写锁）](https://www.cnblogs.com/nickup/p/9804020.html)

共享锁（S锁）：共享 (S) 用于不更改或不更新数据的操作（只读操作），如 SELECT 语句。

如果事务T对数据A加上共享锁后，则其他事务只能对A再加共享锁，不能加排他锁。获准共享锁的事务只能读数据，不能修改数据。

排他锁（X锁）：用于数据修改操作，例如 INSERT、UPDATE 或 DELETE。确保不会同时同一资源进行多重更新。

如果事务T对数据A加上排他锁后，则其他事务不能再对A加任任何类型的封锁。获准排他锁的事务既能读数据，又能修改数据。

1. 主键关键字什么的？（题都没有看完）