# 数据库原理 复习资料（考试用）

## 数据库与数据库系统

1. 数据管理技术的发展主要经历了哪几个阶段？

文件管理，数据库管理。

1. 文件系统和数据库系统（ppt）

(1)DB数据量大。DB可以通过DBMS直接进行很多操作，而文件的话只提供了简单的操作。

(2)用文件的话，要针对不同的文件结构编写不同的代码。

(3)由于实际应用中并发同时访问导致数据访问不一致（文件没有解决办法，需要在应用软件层解决）

(4)故障处理（文件系统不具备这个能力，但DBMS会自动恢复）

(5)安全和访问控制

1. 比较用文件管理和用数据库管理数据的主要区别

DBS建立在文件系统之上。DBS与FS相比实际上是在应用程序和存储数据的数据库之间增加了一个系统软件，即数据库管理系统，使得以前在应用程序中由开发人员实现的很多繁琐的操作和功能，都可以由这个系统软件完成，这样应用程序不再需要关心数据的存储方式，而且数据的存储方式的变化也不再影响应用程序。而在文件系统中，应用程序和数据的存储是紧密相关的，数据的存储方式的任何变化都会影响到应用程序，因此不利于应用程序的维护。

FS面向某一应用程序，共享性差，冗余度大，数据独立性差，记录内有结构，整体无结构，由应用程序自己控制。

DBS面向现实世界，共享性高，冗余度小，具有较高的物理独立性和一定的逻辑独立性，整体结构化，用数据模型描述，由DBMS提供数据的安全性、完整性、并发控制和恢复能力。

FS与DBS的联系：都是计算机系统中管理数据的软件。

1. 数据、数据库、数据库管理系统和数据库系统的概念以及它们之间的关系

数据库系统面向数据密集型应用，以统一管理和共享数据为特征，具有存储、管理、维护数据的功能。

(1) 数据（Data）：描述事物的符号记录称为数据。数据的种类有数字、文字、图形、图像、声音、正文等。数据与其语义是不可分的。

(2) 数据库（Database，DB）：数据库是长期储存在计算机内的、有组织的、可共享的数据集合，以一定的组织形式存于存储介质上。数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度、较高的数据独立性和易扩展性，并可为各种用户共享。

(3) 数据库系统（Database System，DBS)：数据库系统是指在计算机系统中引入数据库后的系统构成，一般由数据库、数据库管理系统(及其开发工具)、应用程序、数据库管理员构成。

(4) 数据库管理系统（Database Management System，DBMS）：数据库管理系统是位于用户与操作系统之间的一层数据管理软件，用于科学地组织和存储数据、高效地获取和维护数据。DBMS的主要功能包括数据库的建立和维护功能、数据定义功能、数据组织存储和管理功能、数据操作功能、事务的管理和运行功能。

**联系**：数据库系统包括数据库、数据库管理系统、应用系统、数据库管理员，所以数据库系统是个大的概念。数据库是长期存储在计算机内的有组织、可共享的大量的数据集合，数据库管理系统是由管理员操作管理数据库的查询、更新、删除等操作的，数据库应用系统是用来操作数据库的。

1. 数据库管理系统包含的功能。

数据定义和操纵功能；数据库运行控制功能；数据库的组织、存储和管理；建立和维护数据库；数据通信接口

## 三级模式与数据独立性

1. 现代数据库怎么管理数据模式的？

数据库系统划分为三个层次，称为三级模式，分别为概念模式、外模式、内模式，都存于数据目录中，是数据目录的最基本内容，DBMS通过数据目录，管理和访问数据模式。

1. 数据库系统包含哪三级模式？试分别说明每一级模式的作用。

**外模式（views）：**对现实系统中用户感兴趣的整体数据结构的局部描述，用于满足不同数据库用户需求的数据视图，是数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，是对数据库整体数据结构的子集或局部重构。

**模式（conceptual schema）：**数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。

**内模式（physical schema）：**对整个数据库的底层表示，它描述了数据的存储结构。

1. 数据库三级模式划分的优点是什么？它能带来哪些数据独立性？

将用户、逻辑数据库与物理数据库进行了划分，使彼此之间的相互干扰减到最少。

带来物理独立性和逻辑独立性，使得底层的修改和变化尽量不影响到上层。

1. 数据库系统的两级映像的功能是什么？它带来了哪些功能？

**模式／内模式映像：**定义了概念视图和存储的数据库的对应关系，它说明了概念层的记录和字段在内部层次怎样表示。如果数据库的存储结构改变了，那么，必须对模式／内模式映像进行必要的调整，使模式能够保持不变。

**外模式／模式映像：**定义了特定的外部视图和概念视图之间的对应关系，当概念模式的结构可发生改变时，也可以通过调整外模式／模式映像间的映像关系，使外模式可以保持不变。

外模式中的视图都是概念模式相关信息的可计算映射，物理模式与逻辑模式之间存在存储映射。

1. 数据独立性

**物理数据独立性：**用户的应用程序与存储在磁盘上的数据库中的数据相互独立。当数据库的存储结构改变，将模式/内模式映象作相应改变，使模式保持不变，从而应用程序不必改变。

**逻辑数据独立性：**用户的应用程序与数据库的逻辑结构相互独立。当模式改变时，将外模式/模式的映象作相应改变，使外模式保持不变，从而应用程序不用修改。

**数据独立性的重要性：**应用程序和数据结构之间相互独立，互不影响。当数据的存储结构或数据的逻辑发生变化时，不会影响到应用程序。（数据库数据的存储结构与存取方法独立）

**数据独立性的好处是，**数据的物理存储设备更新了，物理表示及存取方法改变了，但数据的逻辑模式可以不改变。数据的逻辑模式改变了，但用户的模式可以不改变，因此应用程序也可以不变。这将使程序维护容易，另外，对同一数据库的逻辑模式，可以建立不同的用户模式，从而提高数据共享性，使数据库系统有较好的可扩充性，给DBA维护、改变数据库的物理存储提供了方便。

## 数据模式与模型

1. 举例说明什么是数据模型？什么是数据模式？两者的关系和区别？

Data model is a collection of concepts and definitions for describing data.

（数据模型是用于描述数据的概念和定义的集合，比如字段）

Data schema is a description of a particular collection of data, using a given data model.

（数据模式是使用给定数据模型对特定数据集合的描述，如数据表）

**数据模型：**是用来描述数据的一组概念和定义。以文件系统为例，它所用的数据模型包含文件，记录和字段。

**数据模式：**以一定的数据模型对一个单位数据的类型、结构、相互的关系所进行的描述（对给定的数据模型的具体描述，数据模型的实例化）。例如学生信息记录可以定义为姓名、学号、性别等属性和关系的形式。

**区别：**数据模型是描述数据的手段，而数据模式是用给定的数据模型对具体的数据的描述。

**关系：**数据模式反映一个单位的各种事物的结构、属性、联系和约束，实质上是用数据模型对一个单位的模拟。

**数据语言(数据库语言)：**包括数据定义、操作、查询等基本组成部分(可以以关系模型和SQL为例进一步展开)。

1. 数据模型怎么影响系统性能？

数据模型包含数据结构、数据操作、数据完整性约束，

影响性能的因素主要是数据结构的复杂度和数据操作的可优化程度。

1. 什么是结构化数据，半结构化数据，非结构化数据？

**结构化数据：**数据整体结构化，通过数据模型描述。

**半结构化数据：**单条记录内部的数据有结构，数据文件间无联系，整体无结构。

**非结构化数据：**数据间，数据文件间都没有结构。

1. 解释数据模型的概念，为什么要将数据模型分成两个层次？

数据模型是对现实世界数据特征的抽象。一般要满足：(1)能够比较真实地模拟现实世界；(2)容易被人们理解；(3)能够很方便地在计算机上实现。

由于用一种模型同时很好地满足这三方面的要求在目前是比较困难的，因此在数据库系统中就可以针对不同的使用对象和应用目的，采用不同的数据模型。根据模型应用的不同目的，将这些模型分为概念层数据模型和组织层数据模型两大类，以方便对信息的描述。

1. 概念层数据模型和组织层数据模型分别是针对什么进行的抽象？

**概念层数据模型：**对现实世界的抽象，形成信息世界模型；

**组织层数据模型：**对信息世界进行抽象和转换，形成具体的DBMS支持的数据组织模型。

## 几种数据模型

1. 关系型数据库的特点（课件）
2. 解释关系模型中的主码、外码、主属性、非主属性的概念，并说明主码、外码的作用。

主码：关系表中用于唯一确定一个元组的一个属性或最小属性组，其作用是使表中的每一行数据彼此不重复且有意义。

外码：引用另一个表的主键（也可以是候选键）的表中的一个列，其作用是表示两个或多个实体之间的关联关系。

主属性：关系模型中使用主码作为记录的唯一标识，主码所包含的属性称为关系的主属性。

非主属性：关系模型中使用主码作为记录的唯一标识，主码所包含的属性称为关系的主属性，其他的非主码属性称为非主属性。

1. 关系数据模型上的关系运算是什么？它们的区别和相似点是什么？（关系演算vs关系代数）。

**Differences:** Relational Algebra needs to specify the order of operations; while relational calculus only needs to indicate the logic condition the result must be fulfilled.

**Similiarities:** They are equivalent in terms of expression. SQL language can express any query that is expressible in relational algebra /calculus.

1. 在网状数据模型和关系数据模型中，如何表达两个记录型之间的m:n关系？一般而言，三个实体型之间的多对多联系和这三个实体型两两之间的三个多对多联系等价吗？分别举例说明。

网状数据模型中，通过Link记录来实现M:N关系，而关系模型中通过关系(表)来表示M:N关系，与网状数据模型相比，实现的是一种“软链接”。

不等价。以供应商、零件、工程三者的关系为例，他们之间的一个三元多对多关系和三个两两之间的二元多对多关系是不同的，因为如果供应商A能够提供零件B、A又是工程C的供应商、而工程C又需要零件B，并不意味着工程C中必须要使用供应商A提供的零件B，也就是说三个两两之间的二元关系并不能决定他们之间的一个三元关系。

1. 关系模型的优点和存在的问题是什么？

**优点：**关系模型中将现实世界中的实体以及实体之间的联系统一用关系（表）来表示，与层次和网状数据模型中用物理指针表示实体间的联系相比，关系模型实现的是一种“软连接”，由于被查询的对像是关系，查询结果还是关系，因此可构成一个代数系统，即关系代数。这使得我们可以用数学的方法来研究数据库中的问题，加之关系模型概念非常简单，又解决了查询效率的问题，因此自70年代出现后很快就取代层次和网状数据模型而成为主流。

**缺点：**以记录为基础，不能很好面向用户和应用；不能以自然的方式表达实体间联系；语义贫乏；数据类型太少，不能满足应用需要。

1. 传统数据模型评价 //P30

（1）都适用于OLTP【联机事物处理】应用

（2）以记录为基础，基于结构化数据，不能很好的面向用户和应用

（3）不能用很自然的方法表达现实世界的模型，表达能力有限

（4）模型本身缺乏语义信息

（5）支持的数据类型少，难以满足需求应用需求

## 关系演算

Q1：

DRC：

TRC： or t.age?

Q2：

DRC：

TRC：

Q3：

DRC：

TRC：

除法：

文本

描述已自动生成

**R/S的元组关系演算**

文本

描述已自动生成



IMG_256

文本

描述已自动生成

## SQL

1. 用户接口与SQL

关系代数是一种过程语言，以此设计的数据库语言，用户要说明需要什么数据，和获得这些数据的过程。

对用户来说，最好只说明需要什么数据，而如何获得这些数据则不必用户说明，而由系统来实现。即非过程语言。

非过程关系数据库语言里最成功的应用最广的：SQL。

1. SQL和关系代数的联系和区别

关系代数（以关系代数为基础的数据库语言是过程性的）、SQL、关系演算(元组关系演算和域关系演算)，它们的非过程化程度依次递增，主要应用领域也不同。SQL是关系数据库的标准语言，关系代数和关系演算是它的理论基础。

**联系：**关系代数是SQL的理论基础。

**区别：**SQL是结构化查询语言，是数据库具体的技术标准和规范。关系代数是数学理论。

1. 什么是基本表？什么是视图？两者的区别和联系是什么？

**基本表：**实际存储在数据库中的二维表，它是本身独立存在的表，在SQL中一个关系就对应一个表。

**视图：**关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制。

**区别：**视图是从一个或几个基本表（或视图）中导出的表，是一个虚表，数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。

1. 使用视图的好处。

1、简化数据查询语句

2、使用户能从多角度看待同一数据

3、提高了数据的安全性

4、提供了一定程度的逻辑独立性

1. 数据类型问题

1、char(10)和nchar(10)的区别是什么？它们各能存放多少个字符？占用多少空间？

答：char(10)是定长字符串类型，是普通字符编码，nchar(10)是定长字符串类型，统一字符编码。char(10)存放10个字符，占用10个字节空间；nchar(10)存放10个字符，占用20个字节空间。

2、char(n)和varchar(n)的区别是什么？其中n的含义是什么？各占用多少空间？

答：char(n)是定长字符串类型，varchar(n)是可变长字符串类型，它按数据的实际长度来分配空间。其中n的含义都是能够存储的字符的个数，对于char(n)类型其所占的空间固定为n个字节；对于varchar(n)其所占空间最多不超过n或2\*n个字节。

1. SQL问题

查询只有一名仓库管理员预定的服装的编号；

SELECT服装编号 FROM采购订单 GROUP BY服装编号 HAVING COUNT(DISTINCT 库管员编号) = 1

## DBMS

1. DBMS的核心组件

用户写的应用程序经过接口转为基础的数据库语言（SQL等）

SQL经过词法及语法分析产生语法树

之后检查用户是否有权访问语法树涉及的数据对象，若通过则继续执行，否则拒绝执行

授权检查通过后，进行语义分析与处理，对四种语言分别做不同处理（通常统称为查询处理），其中QL最复杂也最基本，存在多种存取路径选择问题（查询优化）后就形成SQL语句执行

DLL：存放SQL的动态链接库

并发控制：SQL语句执行过程都有并发控制以防止多用户并发访问数据引起数据不一致

恢复机制：发生故障时，能使数据库恢复到最近的一致状态或先前的某个一致状态

DBMS是操作系统之上的软件系统，是操作系统的用户,它对系统资源的调用需要请求操作系统为其服务，通过系统调用来实现

1. DBMS的进程结构

一个应用程序经过DBMS编译后作为一个单独的可执行文件，以一个单独进程进行执行。DBMS core作为这个可执行文件的一个函数。

一个应用程序进程对应一个DBMS core进程，每个客户对应一个线程。

## 数据库存储结构

1. 文件的基本类型

**堆文件：**记录按插入顺序存放，就像堆货物一样。物理地址不一定连续，逻辑地址连续。插入容易，查找难，只能顺序搜索，删除麻烦，一般不在物理上删除，只是打删除标志，以免引起大量记录转移。

**直接文件：**将记录某一属性（一般为主键）用散列函数映射成记录的地址。按散列键访问快，但不同散列键可能映射同一地址，且只对散列键到记录的访问有效，键映射空间固定，不变处理变长记录，用得少。

**索引文件：**索引+堆文件/簇集。在记录的某一属性（组）上建立索引，索引项由索引键的值及其对应的记录的地址货地址集组成。提高查询效率，增加索引维护开销，付出储存空间代价。

1. B树和B+树

在B+树中，键是存储在内部节点中的索引，而记录存储在叶节点。在B树中，key不能重复存储，这意味着没有重复的键或记录。在B+树中，叶子节点相互链接以提供顺序访问。在B树中，叶子节点之间没有链接。B+树是平衡二叉搜索树。B+树确保所有叶节点保持在一样的高度。在B+树中，叶节点使用链表链接。因此，B+树可以支持随机访问和顺序访问。

What are the differences between B Tree and B+ tree? Why do we use B+ tree in DBMS?

In the B+ tree, keys are the indexes stored in the internal nodes and records are stored in the leaf nodes. In B tree, keys cannot be repeatedly stored, which means that there is no duplication of keys or records.

In B+ tree, the leaf nodes are linked to each other to provide the sequential access. In the B tree, leaf nodes are not linked to each other.

The B+ tree is a balanced binary search tree. B+ tree ensures that all leaf nodes remain at the same height. In the B+ tree, the leaf nodes are linked using a link list. Therefore, a B+ tree can support random access as well as sequential access.

1. B-树为代表的树形索引成为当前数据库系统主流索引具有必然性

索引文件是一种适应面比较广的文件结构，因此在数据库系统中得到了广泛的应用。对于经常变动的文件，静态索引的性能会随时间变化而变坏，所以目前在数据库系统中应用更多的是动态索引。而B-树就是为了磁盘或其它存储设备而设计的一种平衡多分树，能很好地进行动态索引。B-树提供了三种存取路径：

1.通过索引集进行树形搜索；

2.通过顺序集进行顺序搜索；

3.先通过索引找到入口，再沿顺序集顺序搜索。

B-树不仅提供了灵活的存取路径，而且能够自动保持平衡，不须定期重组，因此B-树为代表的树索引系列在数据库系统中应用甚广，成为数据库主流具有必然性。

1. 什么是数据字典？作用是什么？

数据字典是系统中各类数据描述的集合。

作用：供DBMS在处理数据存取时快速查找有关对象的信息，供DBA查询掌握系统的运行情况，支持数据库设计和系统分析。

## 查询优化

1. 代数优化与运算优化

**代数优化原则**

1. 先一元操作，后二元操作(Push down the unary operations as low as possible)

2. 优先连接小关系(Look for and combine the common sub-expression)

**操作优化**

因为连接开销很大，其中二元连接最基本用的最多，所以历来是查询优化研究重点

**优化策略**

俩关系已按属性排序，优先考虑排序归并法。一个以排序，另一个很小，则也可用排序归并法。

若两关系有一个关系在连接属性上有索引或散列，则可以另一关系为外关系，顺序扫描，并利用内关系上索引或散列寻找匹配元组。

若上述都不满足，且关系比较小，用嵌套循环法；否则，可用散列连接法

1. 相较层次和网状数据库系统，查询优化对关系数据库系统更为重要。

对。层次和网状数据库系统的语言是过程性的，用户不仅要说明需要什么数据，还要告诉数据库系统获得这些数据的过程。层次和网状数据库的数据模型使用指针表示属性之间的关系，这样的结构也固定了这两种数据库的查询路径，进行优化的空间有限。

而关系数据库系统的语言是非过程性的，用户只用说明需要什么数据，不说明查询的过程，由DBMS确定合理有效的执行策略。对于同一个查询语句，对应的关系代数等价的不同表达式的查询效率有着很大的差异；集合操作不同的执行规则和策略也对查询效率有着很大的影响；同时关系数据在物理存储形式和存取方式和路径上都没有限制。因此对于关系数据库系统来说，查询优化就更为重要，它对系统的性能有着很大的影响。

\*关系模型用表来表示实体间的联系，是一种软连接，查询时涉及大量连接操作，因此必须优化以解决效率问题。

RDBMS和其他DBMS的**本质不同**：采用非过程化的查询语言。

1. 简述索引对关系型数据库系统查询优化的意义

关系型数据库查询优化的途径之一是依赖于存取路径的优化，而在关系型数据库中索引是用得最多的一种存取路径，建立合适的索引是实现查询优化的首要前提。索引提供了对数据的快速访问，根据操作建立合适的索引能够很大程度上优化存取路径，从而提高查询效率。

**意义：**1.大大加快数据的检索速度；

2.创建唯一性索引，保证数据库表中每一行数据的唯一性；

3.加速表和表之间的连接；

4.在使用分组和排序子句进行数据检索时，可以显著减少询中分组和排序的时间。

1. 应该在什么时候使用索引？是不是正在任何情况下使用索引都能得到益处？举例说明。

一般在数据量较大的时候适合使用索引，在需要频繁进行查询、排序或分组操作的属性性建立合适的索引。

索引并不是越多越好，索引固然可以提高相应的select的效率，但同时也降低了insert及update的效率，因为有可能会重建索引。所以怎样建索引需要慎重考虑，根据实际操作需求考虑。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。

例：应用程序非常频繁地更新数据或磁盘空间有限时，则可能需要限制索引的数量。（索引占用磁盘空间，并且降低添加、删除和更新行的速度）。

例：对于小文件上的顺序查找使用堆文件的形式的开销不大，使用索引反而可能会增加开销（存储+维护）。对于增删改操作频繁的属性上建立索引可能会起到反作用。

1. 稠密索引是否一定能够提高针对索引属性查询的效率？

不一定。

1.如果是查询小文件中的全部或相当多的记录时，使用索引并不能提高查询效率，反而会因为索引查询增加开销；

2.如果稠密索引为次索引，且不是簇集索引，则再最低索引中，每个键值对应的不是一个地址而是一个地址集。很可能一个键值对应的多条记录分散在不同的物理块中；当一个键值对应的记录较多时，取这些记录时访问物理块的I/O开销反而会降低查询的效率。

1. 使用视图是否可以加快数据的查询速度？

不对。因为数据库管理系统在对视图进行查询时，首先检查要查询的视图是否存在，如果存在，则从数据字典中提取视图的定义，把视图的定义语句对视图语句结合起来，转换成等价的对基本表的查询，然后再执行转换后的查询，所以使用视图不可以加快数据的查询速度。

1. 从查询优化角度分析，为什么SQL查询where子句应尽量避免使用OR？

为了加快查询速度，优化查询效率，主要原则就是应尽量避免全表扫描，尽量在where及orderby涉及的列上建立索引。然而在where子句中使用or来当连接条件时，会导致引擎放弃使用索引而进行全表扫描，改用union后，性能会大大提高。**//P124**

SQL Query Optimization – OR vs. UNION

 - OR requires Stage 2 processing.

 - Consider rewriting the query as the union of 2 SELECTs, making index access possible.

 - UNION ALL avoids the sort, but duplicates are included.

 - Monitor and EXPLAIN the query to decide which is best.

1. 设外关系R的物理块数为bR，内关系S的物理块数为bS，nB是可供连接的缓冲块数， 简述应怎样改进连接操作的嵌套循环算法，使其总的访问物理块数为：bR＋⌈bR/(nB-1)⌉×bS

DBMS以物理块为基本存取单元，nB－1块缓冲区给外循环关系，1块缓冲区给内循环关系，目的是尽量减少对内循环关系的扫描读取次数。只需将书上原算法对外循环关系的每个元组将内循环关系读取扫描一遍，改为对外循环关系一次读取nB－1个物理块到内存缓冲区，利用另一块缓冲区将内循环关系读取扫描一遍，在内存中仍利用两重循环进行两个关系元组间的两两比较。

1. 建立簇集索引的条件-P240；建立索引和不建索引的条件-P242

## 事务

1. 事务的概念

事务是用户定义的数据操作系列，这些操作可作为一个完整的工作单元，一个事务内的所有语句被作为一个整体，要么全部执行，要么全部不执行。

1. 事务的4个性质。每一个性质由DBMS的哪个子系统实现？对DBS有什么益处？

1）原子性：一个事务对DB的所以操作，是一个不可分割的工作单位。由DBMS的事务管理子系统实现的。保证了DBS的完整性。

2）一致性：数据不会因事务的执行而遭受破坏。由DBMS的完整性子系统实现。保证数据库的完整性。

3）隔离性：事务的并发执行与这些事务单独执行时结果一样。由DBMS的并发控制子系统实现的。使并发执行的事务不必关心其他事务，如同在单用户环境下执行一样。

4）持久性：事务对DB的更新应永久地反映在DB中。由DBMS的恢复管理子系统实现。保证DB具有可恢复性。

1. 如事务不遵守ACID准则，则对数据库产生何种后果？为什么一般不涉及数据库的程序中不提ACID准则？

数据库中会产生脏数据，数据不一致、难以恢复等情况。不涉及数据库的程序不提ACID是因为它们很少需要满足数据库这样的高并发性和一致性需求。

一个事务是由应用程序中对数据库的一组操作序列组成的。如果事务不遵守ACID准则，则数据库中数据的完整性和一致性等就可能会因为事务的执行而遭到破坏。而一般不涉及数据库的程序不存在多用户之间数据的共享问题，所以在一般不涉及数据库的程序中不提ACID准则。

1. 描述更新事务的提交规则和日志提前规则的必要性 //P142

提交规则是为了确保在事务提交前将AI写入非易失性存储器中，这样即使事务进入commit阶段后发生故障，记录的AI仍然可以用于重做和更新，以保证事务符合ACID原则。

日志提前规则是指如果在事务提交之前直接将AI写入数据库，则必须在事务提交之前将相应的BI写入日志，以便在发生故障时进行undo在交易进入之前commit阶段，事务的执行符合ACID原则。

1. 登记日志文件时为什么必须先写日志文件，后写数据库？

把数据的修改写到数据库和把对数据的修改操作写到日志文件是两个不同的操作，在两个操作之间可能会发生故障如果先写了数据库修改，而在日志文件中没有登记这个修改，在恢复的时候就无法恢复这个修改了；如果先写日志，但没有修改数据库，按日志恢复时只是多执行一次撤消操作，并不影响数据库的正确性，为了安全，一定要先写日志文件。

1. 试述提交规则和先记后写规则对更新事务的必要性。

提交规则是保证后像在事务提交前写入非挥发存储器，这样即使事务进入提交阶段以后发生故障，仍可利用记录下来的后像重做更新，从而确保事务满足ACID原则。

先记后写规则是指如果后像在事务提交前直接写入数据库，则必须在此之前将相应的前像写入运行记录(Log)，以备当事务进入提交阶段之前发生故障时做undo，使事务的执行满足ACID原则。

## 故障与恢复

1. 数据库故障分类

事务内部的故障，系统故障，其他故障

1. 数据库恢复的技术

数据库备份技术，事务日志备份技术

1. 数据库的破坏：P171
2. 假设运行记录与数据库的存储磁盘具有独立失效模式，介质失效恢复时，对运行记录中上一检查点以前的已提交事务应该redo否？为什么？

是。因为介质失效会丢失数据库中所有的数据，恢复时需要再加载最近的后备副本后，根据运行记录中的后像，重做最近后备副本以后提交的所有更新事务。因此最近一次检查点以前提交的事务也要做redo操作。在取后备副本后，以前运行的数据就失去价值，可以清空CTL表。

1. 怎样进行系统故障的恢复?

系统故障造成数据库不一致状态的原因：(1)未完成事务对数据库的更新可能已写入数据库，(2)已提交事务对数据库的更新可能还留在缓冲区没来得及写入数据库。因此恢复操作就是要撤消故障发生时未完成的事务，重做已完成的事务。

⑴正向扫描日志文件，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消(UNDO)队列。

⑵对撤消队列中的各个事务进行撤消(UNDO)处理。反向扫描日志文件，对每个UNDO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。

⑶对重做队列中的各个事务进行重做(REDO)处理。正向扫描日志文件，对每个REDO事务重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

1. 怎样进行介质故障的恢复?

发生介质故障后，磁盘上的物理数据和日志文件被破坏，恢复方法是重装数据库，然后重做已完成的事务。

⑴装入最新的数据库后备副本（离故障发生时刻最近的转储副本），使数据库恢复到最近一次转储时的一致性状态。对于动态转储的数据库副本，还须同时装入转储开始时刻的日志文件副本，利用恢复系统故障的方法（即REDO+UNDO），才能将数据库恢复到一致性状态。

⑵装入相应的日志文件副本（转储结束时刻的日志文件副本），重做已完成的事务。即：首先扫描日志文件，找出故障发生时已提交的事务的标识，将其记入重做队列。然后正向扫描日志文件，对重做队列中的所有事务进行重做处理。即将日志记录中“更新后的值”写入数据库。

1. 怎样进行事务故障的恢复

事务故障是指事务在运行至正常终止点前被中止，这时恢复子系统应利用日志文件撤消（UNDO）此事务已对数据库进行的修改。事务故障的恢复是由系统自动完成的，对用户是透明的。系统的恢复步骤是：

⑴反向扫描文件日志（即从最后向前扫描日志文件），查找该事务的更新操作。

⑵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（因此时“更新前的值”为空）。若记录中是删除操作，则做插入操作，若是修改操作，则相当于用修改前值代替修改后值

⑶继续反向扫描日志文件，查找该事务的其他更新操作，并做同样处理。

⑷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。

1. 什么是数据库镜像？它有什么用途？

根据DBA的要求，自动把整个数据库或其中的关键数据复制到另一个磁盘上。每当主数据库更新时，DBMS自动把更新后的数据恢复过去，即DBMS自动保证镜像数据与主数据的一致性。

用途：1）一旦出现介质故障，可由镜像磁盘继续提供使用，同时DBMS自动利用镜像磁盘数据进行数据库恢复，不需要关闭系统和重装数据库副本。2）在没有出现故障时，数据库镜像还可以并发操作。提高数据库的可用性。

## 并发控制

1. 有关数据库系统并发的问题

（1）什么叫并发？DBMS 可同时接纳多个事务，事务在时间上可以重叠执行。

（2）为什么要并发？改善系统资源利用率；改进响应时间。

（3）并发会带来什么问题？丢失更新，读脏数据，读值不可重复。

（4）什么样的并发执行才是正确的？可串行化。

（5）如何避免并发所引起的问题？采用封锁发、时间戳法等并发控制方法。

（6）既然目标可串行化调度比冲突可串行化调度多，我们为什么要强调冲突可串行化而非目标可串行化呢？目标可串行化的判断算法是NP完全问题，有没有保证目标可串行化的简单实用的规则。冲突可串行化覆盖了绝大部分可串行化的调度实例，测试算法简单、易实现。

1. 并发的问题

丢失更新：两个事务同时对一个数据项更新，并发写入会有一个事务的更新被覆盖丢失。与两事务串行执行结果不同。write-write conflict

读脏数据：一个事务对某个元组中两个属性进行更新，一个事务则读取这个元组，并发执行会让这个元组的两个属性不一致，比如一个已更新一个未更新。读取的是个不存在的元组。read-write conflict

读值不可复现：一个事务读取一个数据两次，另一个事务并发修改了这个事务导致读取的两个数据不同。两事务若串行执行不会有此现象。read-write conflict

1. 在数据库中为什么要有并发控制机制？

答：数据库系统一个明显的特点是多个用户共享数据库资源，尤其是多个用户可以同时存取相同数据。在这样的系统中，在同一时刻同时运行的事务可达数百个。若对多用户的并发操作不加控制，就会造成数据库存、取的错误，破坏数据的一致性和完整性。

1. 判断并发事务运行正确性标准是什么？封锁法的基本思想是什么？它是怎么样保证并发事务的正确执行的？采用封锁法以后必须解决的问题是什么？ //P148

（对于串行调度，各事务的操作没有交叉，也就没有互相干扰，当然不会产生并发引起的问题。可串行化调度和某一个串行化调度等价，所以它也不会产生并发所引起的问题。）

**正确性准则：**可串行化（若两个事务集合并发执行的结果与串行执行的结果相同）

**封锁法的基本思想：**并发事务对同一数据对象操作前，向系统发出请求对操作对象进行加锁。从而强迫有冲突的事务按照抢到锁的次序执行。

**如何保证：**事务对某个数据对象的加锁请求获准后，该事务便对该对象有了一定的控制，在这个事务释放它的锁之前，其他事务对该数据对象的锁的请求需要根据相容矩阵进行锁的申请，如果没有申请到锁（证明有冲突），则无法对其进行操作，从而避免了访问冲突，保证并发事务正确执行。

**需要解决的问题：**活锁（先申请，先服务），死锁（等待-死亡，击伤-等待**P158**），其中必须解决的问题是由于事物之间的循环等待导致的死锁问题。

（当事务申请锁未获准时不是一律等待，而是让一些事务卷回重执(retry),以避免循环等待。为区别事务开始执行的先后，给他们一个随时间增长的时间标记ts(time stamp)）

1. 若事务并发执行的调度可串行化，即认为该并发结果是正确的，为什么？

事务并发执行的调度是可串行化的，也就是说对于该事务并发执行的调度与该事务的串行调度等价；对于串行调度，各事务的操作没有交叉，没有互相干扰，因此不会产生并发执行时的冲突问题，因此与之等价的事务并发调度也不会产生冲突，即并发结果是正确的。

1. 两段锁协议的概念

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；

2）在释放一个封锁之后，事务不再申请和获得对该数据的封锁。

两段锁含义：事务分为两个阶段，第一阶段是获得封锁，也称为扩展阶段。在这个阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不释放任何锁。第二阶段是释放封锁，也称为收缩阶段。在这个阶段，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。

1. 已有的(S, X）、(S, U, X)锁能解决事务并发中的死锁问题么？为什么？

不能解决并发事务中的死锁问题。当一个事物A占用数据对象a的X锁，事务B占用数据对象b的X锁，事务A和事务B又分别申请数据对象b和数据对象a的锁，在（S,X）和（S,U,X）锁中，均无法获准，需要等待对方事务释放锁，而进入等待状态则无法释放自己所占用的锁，从而陷入循环等待，即死锁。

1. （S,U,X）锁的相容矩阵为什么已经加了U锁，不允许其它事务申请加U锁？如果允许会出现什么情况？

U锁表示事务对数据对象进行更新的操作，在最后写入阶段事务再将其升级为X锁，导致最终写操作时若在U锁阶段允许其他事务申请U锁，则在事务A想将U锁升级为X锁进行数据写操作时，由于存在其他事物对数据对象的U锁，而无法升级为X锁，从而导致死锁。

## 约束和范式

1. 关系模型的数据完整性包含哪些内容？分别说明每一种完整性的作用。

数据的完整性是为了防止数据库中存在不符合应用语义的数据，为了维护数据的完整性，数据库管理系统提供了一种机制来检查数据库中的数据，看其是否满足语义规定的条件。这些加在数据库数据之上的语义约束条件就是数据完整性约束条件。

实体完整性：保证关系数据库中所有的表都必须有主键，且主键不允许为空。

参照完整性：用于描述实体之间的关联关系。

用户定义完整性：约束关系中属性的取值范围，即保证数据库中的数据符合现实语义。

**一些约束：**

在进行插入操作时检查DEFAULT约束。在进行插入和更新操作时检查CHECK约束。

UNIQUE约束的作用是保证数据的取值不重复。

1. 最基本的三个完整性约束

域完整性约束：属性值得在域中

实体完整性约束：一个关系内的约束，每个关系都要有一个主键，每个元组（实体）的主键值应唯一且不为 NULL

引用完整型约束：外键要么空缺，要么引用实际存在的主键值

1. 完整性约束的说明

用过程说明：把约束的说明和检验交给应用程序

用断言说明：断言指数据库状态必须满足的逻辑条件

在基表定义中加入CHECK 子句约束

用触发子表示约束

1. 分析空值产生的原因。为了处理空值，DBMS要做哪些主要工作？

原因：①某些数据不确定；②某些数据当前仍不知道；③某些数据不存在。

DBMS需要在查询处理中支持包括True、False、Null的三值逻辑，需要支持针对空值的实体完整性约束、引用完整性约束等。

1. 依赖

**数据依赖：**关系数据库内数据之间存在一定的数据依赖关系。

**函数依赖：**一个属性的值可以唯一的决定其它属性的值

**多值依赖：**一个属性的值决定其它属性的一组值，实际生活存在但很少。

**连接依赖：**关系属性之间能够无损连接（连接后的元组数一个不多，一个不少），实际生活存在但很少。

**函数依赖是行，多值依赖是列。**

1. 范式

**1NF：**关系中的每一个属性必须是原子的（不可再分），不支持表中套表。

**2NF：**1NF+不存在属性间的部分函数依赖。

·例子：若属性由（学号、姓名、班级、课程号、成绩）五个属性组成，其中（学号，课程号）共同组成主键。此时不满足二范式。其中的姓名和班级，只需要依赖主键中的学号就可以得到。

·不满足2NF问题：

①插入异常：不能插入一个还未选课的学生的信息

②删除异常：如果一个学生申请休学，把选过的课退了，那么他的信息也会被删除。

③更新异常：更新中难以保持数据的一致性，上例的设计有大量的数据冗余

·（学号、姓名、班级）（学号、课程号、成绩）

**3NF：**2NF+不存在属性对主键的传递依赖。

·例子：若属性由（职工编号、工资级别、工资）三个属性组成，其中（职工编号）为主键。工资取决于工资级别，工资级别取决于职工编号。

·不满足3NF问题：

①插入异常：当一个人的工资级别还没定的时候，他对应的工资也没有。

②删除异常：若只是删除一个员工的工资信息时，会把对应的工资级别信息也删除了。

③更新异常：有大量的数据冗余。

·（职工编号、工资级别）、（工资级别、工资）

**2NF、3NF解决方法：**一视一地：一张表只管一件事情。

**4NF：**3NF+消除多值依赖。

**5NF：**4NF+消除连接依赖。

**BCNF与3NF基本等价，而BCNF的决定子都是主键。**

1. 数据模式遵循的范式越高越好吗？

范式主要目的是防止数据冗余、更新异常、插入异常和删除异常，因此达到该目的即可。但范式越高可能带来处理速度缓慢和处理逻辑复杂的问题，因此需要权衡考虑。

并不是应用的范式越高越好，要看实际情况而定。应用的范式等级越高，则表越多。表多会带来很多问题：查询时要连接多个表，增加了查询的复杂度，降低了数据库查询性能。

1. 关系规范化中的操作异常有哪些？它是由什么引起的？解决的方法是什么？

操作异常：数据冗余问题、数据更新问题、数据插入问题和数据删除问题。

它是由关系模式中某些属性之间存在的“不良”的函数依赖关系一起的。

解决的方法是进行模式分解，即把一个关系模式分解成两个或多个关系模式，在分解的过程中消除那些“不良”的函数依赖，从而获得良好的关系模式。

## 数据库设计

1. 数据库设计流程

需求分析，与用户交流确定需求

概念设计，分析数据之间的关系、实体及实体间的逻辑

逻辑设计，看采用的数据库系统，生成基表

物理设计，考虑数据在内存上到底如何存储

1. 概念结构应该具有哪些特点？

有丰富的语义表达能力。易于交流和理解。易于更改。易于像各种数据模型转换。

1. 什么是数据库的逻辑结构设计？简述其设计步骤。

逻辑结构设计的任务是把在概念结构设计中设计的基本E-R模型转换为具体的数据库管理系统支持的组织层数据模型，也就是导出特定的DBMS可以处理的数据库逻辑结构。

步骤：1、将概念结构转换为某种组织层数据模型。

2、对组织层数据模型进行优化。

1. 数据模型的优化包含哪些方法？

1、确定各属性间的函数依赖关系。根据需要分析阶段得出的语义，分别写出每个关系模式的各属性之间的函数依赖以及不同关系模式中各属性之间的数据依赖关系。

2、对各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

3、判断每个关系模式的范式，根据实际需要确定最合适的范式。

4、根据需要分析阶段得到的处理要求，分析这些模式对于这样的应用环境是否合适，确定是否要对某些模式进行分解或合并。

1. 对于课程/学生和选课表的数据库，（1）关系模式的设计考虑了哪些问题？（2）表达每门课的先修情况，如何调整数据库的设计？设计方案和理由。

(1)考虑了完整性约束和规范化的问题。

①数据库中各个关系表都有主键，主键的值是唯一的且不为空，满足了实体完整性约束。同时enroll表中定义了对course和student表的外键，满足了引用完整性约束。

②关系中的属性都是原子的，因此满足第一范式的要求；同时各关系表中不存在部分函数依赖和传递函数依赖，因此该数据库的设计也满足第二和第三范式的要求。

(2)先修课程包括课程号、课程名、开课院系等属性。如果将先修课程情况直接作为属性附加到course关系表中，会导致关系中出现部分函数依赖，不符合二范式的要求。而且先修课程可能不止一门，上述的做法还会产生大量冗余。因此需要新增一个关系表表达先修课程情况。关系表设计如下：

|  |  |  |  |
| --- | --- | --- | --- |
| cno | dname | cno1 | dname1 |

其中cno和dname表示当前课程的课程号和开课院系，cno1和dname1表示先修课程的课程号和开课院系。cno和dname、cno1和dname1都是对course表的外键。

## 分布式数据库

1. 试分析分布式数据库系统出现的技术背景和应用背景。它与后来出现的联邦式数据库系统的类似之处和本质区别是什么？

**背景：**分布式处理技术的发展，当时网络宽还不足，因此出现了由网络连接的多台计算机共同协作解决大量数据的存储、管理、查询的需求，通过将数据就近存放提高访问效率。

**本质区别：**前者在物理上分布的，但逻辑上却是集中的。这种分布式数据库只适宜用途比较单一的、不大的单位或部门。而联邦式分布数据库系统在物理上和逻辑上都是分布的。由于组成联邦的各个子数据库系统是相对“自治”的，这种系统可以容纳多种不同用途的、差异较大的数据库，比较适宜于大范围内数据库的集成。

**相似之处：**分布式数据库系统的不同类别。是在集中式数据库系统的基础上发展来的。是数据库技术与网络技术结合的产物。包含分布式数据库管理系统(DDBMS)和分布式数据库(DDB)。

1. 为什么虽然主流的数据库产品基本上都支持分布式数据库功能，但实际应用中分布式数据库的成功案例并不多？

应用并不理想的原因：①系统实现和使用都比较复杂，从数据库设计角度看，物理上分布、逻辑上集中的数据库设计对开发人员要求较高：②难于管理，单个DBA很难维护大量物理上分布的数据库，多个DBA又很难协调，系统安全较难控制；③网络带宽和服务器性能的飞速提升，使当初的一些瓶颈不再成为障碍，基于集中式数据库服务器的三层（多层）信息系统架构成为流行。

1. 分布式数据库的全局死锁。

由于系统提供的资源数比多个进程所需的资源数少，并且系统的资源分配策略和进程并发执行的速度不当。死锁是占有资源并申请资源的事务之间循环等待造成的。（举一个例子，T1握有T2需要的资源的同时等待T2的握有的资源），客户端A向服务器提交事务Ta，Ta需要R1,R2,R3,R4,R5五个资源，并已申请到R1和R2；而客户端B几乎同时提交事务Tb，需要R5,R4,R3,R2,R1五个资源，并已申请到R5和R3.至此Ta和Tb的循环等待开始，造成全局死锁。