数据库原理常见概念题

数据库设计中，数据模式遵循的范式是越高越好么?

数据库设计中，数据模式遵循的范式不是越高越好，应取决于应用。数据库的范式主 要目的是防止数据冗余，更新异常、插入异常和删除异常，而范式高会存在处理速度缓慢和处理逻辑复杂的问题，从而降低数据库性能，因此需要权衡考虑。

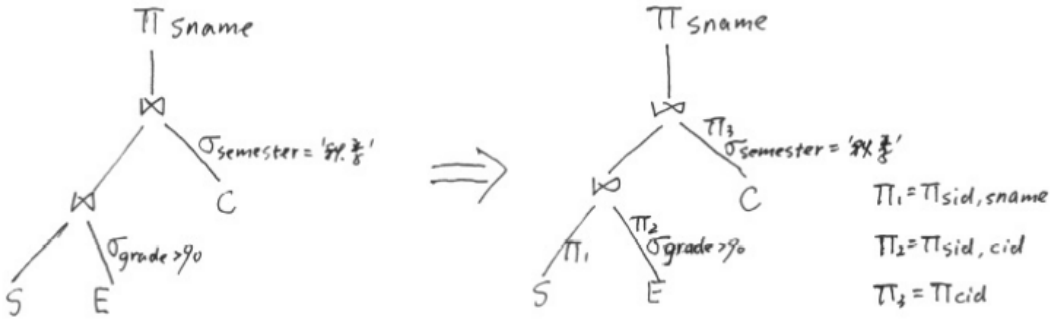
现代数据库系统如何实现数据的独立性?

- 两级映射和三级结构

操作优化在关系模型的重要作用

4. 为什么查询优化对关系数据库管理系统(RDBMS)来说特别重要？与网状、层次模型的数据库管理系统相比，RDBMS 的查询处理有什么本质的不同？试写出上题查询(1)的代数优化过程(10%)。

4．要点：因为关系模型仍然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；本质不同：关系数据库系统采用非过程化的查询语言；代数优化过程：



B+树和B树有什么不同？为什么要这样改进

- B+树非叶子节点不存在数据只存索引，B树非叶子节点存储数据。可以使得节点大小相同时，其内部节点相对B树高度更小查询产生的I/O更少。
- B+树使用双向链表串连所有叶子节点，可以使得区间查询效率更高，扫描数据库只需从叶子结点扫描一遍就行了。

稠密索引是否一定能够提高针对索引属性查询的效率?为什么?

稠密索引不一定能提高针对索引属性的查询效率。

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，反而会因为索引增加开销；

② 如果稠密索引为次索引，但不是簇集索引，也就是说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 理课的 I/O 开反而会降低查询的效率。

什么是数据库系统？它有哪些重要组成部分？数据模型和数据语言有什么关系？

> 书本 P5

（1） 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。

（2） 数据库系统由应用程序、数据库管理系统、数据库、数据库管理员等组成。

（3） 数据模型是用来描述现实世界数据的一组概念和定义；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）。

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。

（1） 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“软连接”；

（2） 不等价；

（3） 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。

试回答下列有关数据库系统并发的问题

（1） 什么叫并发？

（2） 为什么要并发？

（3） 并发会带来什么问题？

（4） 什么样的并发执行才是正确的？

（5） 如何避免并发所引起的问题？

（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？

（1） 是指 DBMS 可同时接纳多个事务，事务在时间上可以重叠执行；

（2） 目的：①改善系统资源利用率；②改进响应时间；

（3） 问题：①丢失更新；②读脏数据；③读值不可重复；

（4） 可串行化；

（5） 可采用封锁法、时间戳法等并发控制方法；

（6） 目标可串行化的判断算法是 NP 完全问题，也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。

数据库系统中，若事务并发执行的调度是可串行化的，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。

试述提交规则和先记后写规则对更新事务的必要性

> 书本P142

4、要点：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。

关于死锁避免方法，为什么说把操作 系统中可用的Requesting all locks at initial time of transaction方法用 于数据库系统，理论上可行实际却做 不到?

- 数据库中的数据对象会有动态查删改变化，还有动态SQL，运行前都不确定要访问哪些数据对象，要在程序运行前一次性锁住所有要访问的对象，当然做不到。

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。

U锁的好处？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?

S 和 U 锁相容，可以提高并发度；U 和 U 锁相容，造成死锁。

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。

假设运行记录与数据库的存储磁盘有独立失效模式，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?

介质失效指磁盘发生故障，数据库受损，如磁盘、刺头破损。

介质失效后应该在新介质中加载最近后备副本，并用档案存储器内运行记录中的后像，redo 后备副本以后提交的所有更新事物。如果检查点比后备副本要新，则对后备副本以后，检查点以前的事物也应该 redo。

Check Point

（1）Checkpoint 用于数据库发生系统失效时，避免大量的无效 redo。

（2）可以减少，具体与取 CP 的周期有关。

（3）取后备副本后可以清空。

试分析空值产生的原因。为了处理空值，DBMS要做哪些主要工作?

（1） 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；

（2） 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。

关系模型的优缺点

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；

分布式数据库的前景

7. 试分析分布式数据库系统出现的技术背景和应用背景。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(10%)?

（1）背景：分布式处理技术的发展，当时网络带宽还不足，因此自然出现了由联网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。
（2）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。

分布式数据库死锁举例

- 节点A依次请求 R1 R2，A已经持有R1，等待R2，节点B依次请求 R3 R2，B已经占有R2，等待R1。
- 由于分布式数据系统事务只能同时commit或者abort，导致全局的循环等待。

联邦式数据库系统与分布式数据库系统的异同

同：物理分布相同，数据模式分布于各个网络节点

异：无全局模式；节点高度自治，其原本数据模式保留；各节点通过协商共享数据满足服务；

“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”？

> 书本P107

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-数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。

组织 SQL 查询语句时，一般不建议在 Where 条件中使用“or”谓词?

> 书本P124

使用 Or 谓词的析取选择条件，并没有好的优化条件，只能按其中各个条件分别选出一个元 组集，再求这些元祖集的并。并是开销大的操作，而且在 OR 连接的多个条件中，只要有一 个条件没有合适的存取路径，就不得不采用顺序扫描来处理这种查询，导致效率大大降低。