7.4.2 数据模型的优化

- 数据库逻辑设计的结果不是唯一的。
- 得到初步数据模型后，还应该适当地修改、调整数据模型的结构，以进一步提高数据库应用系统的性能，这就是数据模型的优化。
- 关系数据模型的优化通常以规范化理论为指导。

优化数据模型的方法

1. 确定数据依赖

   按需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖

2. 消除冗余的联系

   对于各个关系模式中的数据依赖进行极小化处理，消除冗余的联系。

3. 确定所属范式

   - **按照数据依赖的理论对关系模式逐一进行分析**
   - **考查是否存在部分函数依赖、传递函数依赖、多值依赖等**
   - **确定各关系模式分别属于第几范式**

4. 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

注意：并不是规范化程度越高的关系就越优

例：在关系模式

学生成绩单(学号,英语,数学,语文,平均成绩) 中存在下列函数依赖：

学号→英语， 学号→数学， 学号→语文

显然有： 学号→(英语,数学,语文)

而 (英语, 数学, 语文)→平均成绩

因此该关系模式中存在传递函数依赖：学号→平均成绩，是2NF关系

虽然平均成绩可以由其他属性推算出来，但如果应用中需要经常查询学生的平均成绩，为提高效率，仍然可保留该冗余数据，对关系模式不再做进一步分解。

优化数据模型的方法

5. 按照需求分析阶段得到的各种应用对数据处理的要求，对关系模式进行必要的分解或合并，以提高数据操作的效率和存储空间的利用率

- 常用分解方法
  - 水平分解
  - 垂直分解

7.4.2 数据模型的优化

- 水平分解
  - ➢什么是水平分解
    - –把(基本)关系的元组分为若干子集合，定义每个子集合为一个子关系，以提高系统的效率
  - ➢水平分解的适用范围
    - –满足“80/20原则”的应用（一个大关系中，经常被使用的数据只是关系的一部分约20%，可以将这些数据分解出来，形成一个子关系）
    - –并发事务经常存取不相交的数据

例：有关系模式**客户（帐号，姓名，地区……）**，如果绝大多数查询一次只涉及一个地区，就可以把整个关系按地区进行水平分解。

- **湖北客户（帐号，姓名，地区…… ）**
- **湖南客户（帐号，姓名，地区…… ）**
- ……

7.4.2 数据模型的优化

- 垂直分解
  - ➢什么是垂直分解
    - – 把关系模式*R*的属性分解为若干子集合，形成若干子关系模式
  - ➢垂直分解的适用范围
    - – 取决于分解后*R*上的所有事务的总效率是否得到了提高

教师关系（教师号，姓名，性别，年龄，职称，工资，住址，电话，简历）

- 如果经常查询前六项，后三项很少使用，则可以对教师关系进行垂直分解：

教师关系1（教师号，姓名，性别，年龄，职称，工资）

教师关系2（教师号，住址，电话，简历）

7.4.3 设计用户子模式

- 定义数据库模式主要是从系统的时间效率、空间效率、易维护等角度出发。
- 定义用户外模式时应该更注重考虑用户的习惯与方便。包括三个方面：

(1) 使用更符合用户习惯的别名

- 合并各分E-R图曾做了消除命名冲突的工作，以使数据库系统中同一关系和属性具有唯一的名字。
- 但对于某些局部应用，由于改用了不符合用户习惯的属性名，可能会使他们感到不方便
- 因此在设计用户的子模式时可以重新定义某些属性名，使其与用户习惯一致。
- 例：负责学籍管理的用户习惯于称教师模式的职工号为教师编号。因此可以定义视图，在视图中职工号重定义为教师编号

7.4.3设计用户子模式

(2) 针对不同级别的用户定义不同的外模式，以满足系统对安全性的要求。

例：教师关系模式中包括职工号、姓名、性别、出生日期、婚姻状况、学历、学位、政治面貌、职称、职务、工资、工龄、教学效果等属性。

学籍管理应用：只能查询教师的职工号、姓名、性别、职称数据；

课程管理应用：只能查询教师的职工号、姓名、性别、学历、学位、职称、教学效果数据；

教师管理应用：则可以查询教师的全部数据。

7.4.3 设计用户子模式

定义两个外模式：

教师_学籍管理(职工号，姓名，性别，职称)

教师_课程管理(工号，姓名，性别，学历，位，职称，教学效果)

授权：学籍管理应用只能访问教师_学籍管理视图

授权：课程管理应用只能访问教师_课程管理视图

授权：教师管理应用能访问教师表

这样就可以防止用户非法访问本来不允许他们查询的数据，保证了系统的安全性。

7.4.3 设计用户子模式

[例] 关系模式产品（产品号，产品名，规格，单价，生产车间，生产负责人，产品成本，产品合格率，质量等级），可以在产品关系上建立两个视图：

- 为一般顾客建立视图：

  产品1（产品号，产品名，规格，单价）

- 为产品销售部门建立视图：

  产品2（产品号，产品名，规格，单价，车间，生产负责人）

  - 顾客视图中只包含允许顾客查询的属性
  - 销售部门视图中只包含允许销售部门查询的属性
  - 生产领导部门则可以查询全部产品数据
  - 可以防止用户非法访问不允许他们查询的数据，保证系统的安全性

7.4.3 设计用户子模式

(3) 简化用户对系统的使用

– 如果某些局部应用中经常要使用某些很复杂的查询，为了方便用户，可以将这些复杂查询定义为视图。

如**student（sno,sname,sage….）**

course(cno,cname….)

sc(sno,cno,grade)

若经常查某位同学学了某门课成绩是多少

则 **create view aaa**

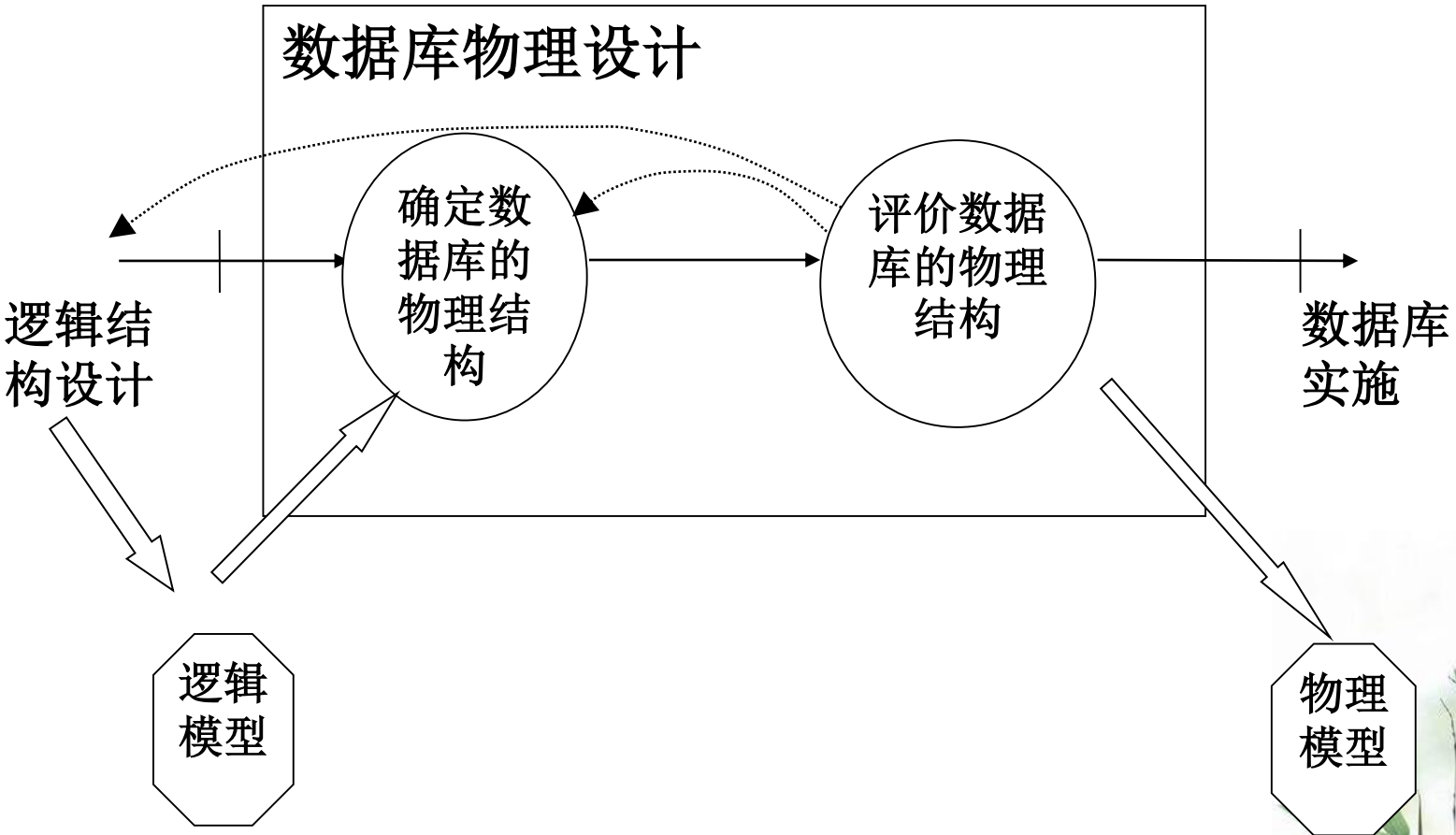
as

……..

7.5 数据库的物理设计

- 什么是数据库的物理设计
  - 数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于选定的数据库管理系统。
  - 为一个给定的逻辑数据模型选取一个最适合应用环境的物理结构的过程，就是数据库的物理设计。

物理设计的步骤



7.5 数据库的物理设计

7.5.1 物理结构设计的任务

- 对于给定的逻辑模型，设计一个最适合应用要求的物理结构。
- **约束条件：**
  - 应用需求（响应时间、吞吐率、存储空间利用率、安全）
  - 应用特征（数据量、事务频率、涉及的关系和属性）
  - 具体DBMS产品的特点（存取方法、存储结构、参数）
  - 存储设备特性
- **设计内容：**
  - 设计存取方法
  - 设计存储方案
  - 确定系统配置参数

7.5.2 关系模式存取方法选择

- 数据库系统是多用户共享的系统，对同一个关系要建立多条存取路径才能满足多用户的多种应用要求。
- 物理设计的第一个任务就是要确定选择哪些存取方法，即建立哪些存取路径。

❖ **DBMS常用存取方法**

- **索引方法，目前主要是B+树索引方法**
- **聚簇（Cluster）方法**
- **HASH方法**

一、索引存取方法的选择

- 索引是与基表的属性（组）相关的数据对象，能够提供对基表数据的快速访问。
- 同一基表上可建多个索引，执行查询语句时，**DBMS自动选择**合适的索引访问数据。
- 当表中的数据更新时，**DBMS自动更新索引**。
- 数据越多，索引的优势越明显。
- 索引可以提高查询效率，但是增加了空间开销以及维护的代价。

一、索引存取方法的选择

- 选择索引存取方法的主要内容
  - 对哪些属性列建立索引
  - 对哪些属性列建立组合索引
  - 对哪些索引要设计为唯一索引

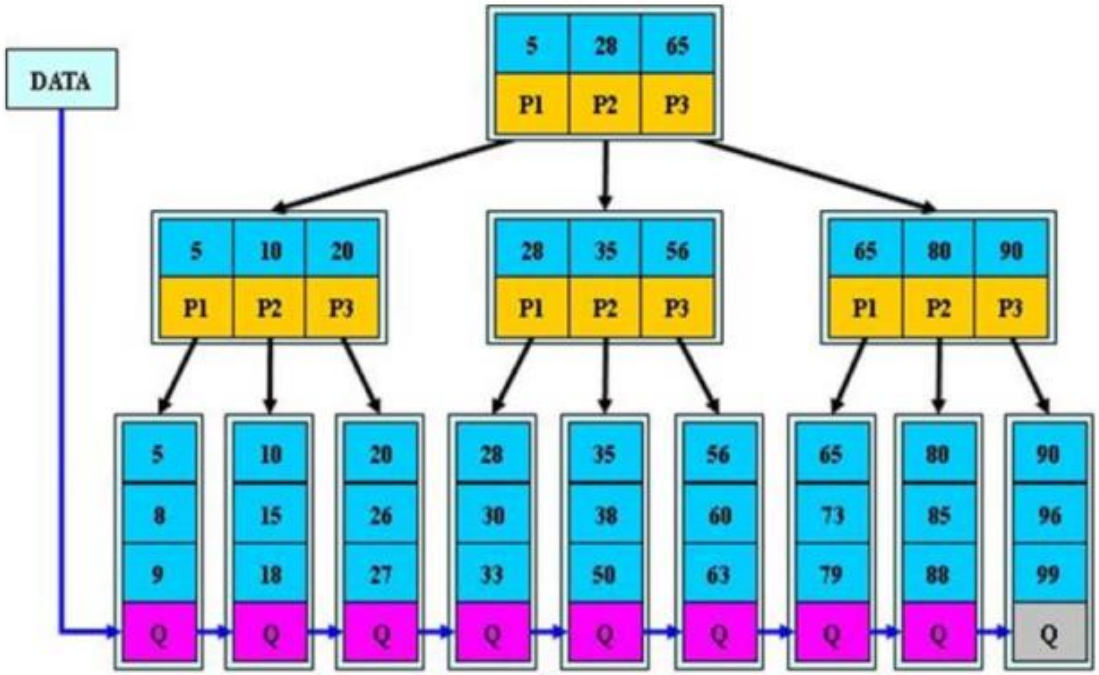
❖选择索引存取方法的一般规则

- 一个(或一组)属性经常在查询条件中出现。
- 一个属性经常作为最大值和最小值等聚集函数的参数
- 一个(或一组)属性经常在连接操作的连接条件中出现。

B+树索引

适用范围广：

- 等值查询
- 范围查询
- 聚集函数计算
- 排序
- 遍历



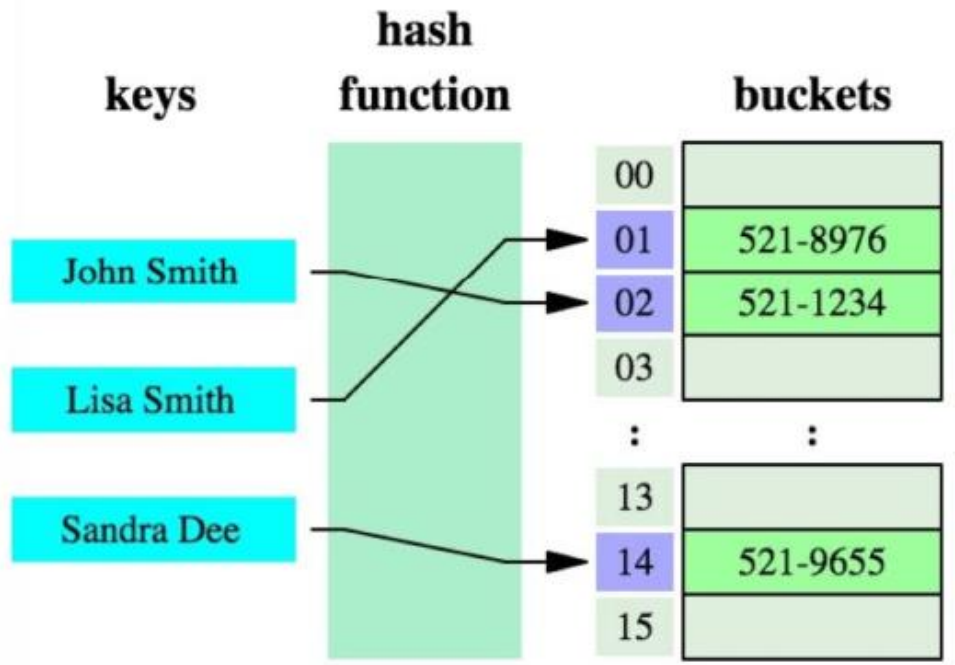
Hash索引

适用于：

- 等值查询
- 等值连接

不适用于：

- 范围查询
- 排序
- 模糊



二、聚簇存取方法的选择

- 什么是聚簇
  - 为了提高查询速度，把某个属性（组）上具有相同值的元组集中存放在连续的物理块中，该属性（组）称为聚簇码。

❖ 聚簇索引

- 建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中元组的物理顺序一致。

CREATE CLUSTER INDEX Stusname ON Student(Sname);

在**Student**表的**Sname**（姓名）列上建立一个聚簇索引，而且**Student**表中的记录将按照**Sname**值的升序存放

二、聚簇存取方法的选择

- **单表聚簇**：将单个表的数据按聚簇码分组有序的存储。
  - **例如**：查询学生关系中计算机系的学生信息。
  - 如果按照“系别”建聚簇，可以显著地减少访问磁盘的次数。
- **多表聚簇**：把多个表的数据按聚簇码集中存储在一起。

例：假设用户经常要按系别查询学生成绩单，这一查询涉及学生关系和选修关系的连接操作，即需要按学号连接这两个关系，为提高连接操作的效率，可以把具有相同学号值的学生元组和选修元组在物理上聚簇在一起。这就相当于把多个关系按“预连接”的形式存放，从而大大提高连接操作的效率。

二、聚簇存取方法的选择

聚簇的设计原则

- 一个表最多只能属于一个聚簇。
- 聚簇维护开销很大。
- 频繁更新的属性不适合作为聚簇码。
- 单表聚簇：最常用于等值比较的属性，且该属性上的重复取值较多。
- 多表聚簇：经常进行连接操作的表可以建立聚簇。

7.5.3 确定数据库的存储结构

- 确定数据库物理结构的内容
  - 确定数据的存放位置和存储结构
    - 关系
    - 索引
    - 聚簇
    - 日志
    - 备份
- 基本原则
  - 根据应用情况将
    - 易变部分与稳定部分
    - 存取频率较高部分与存取频率较低部分

    分开存放，以提高系统性能。

确定数据的存放位置

例

- 数据库数据备份、日志文件备份等由于只在故障恢复时才使用，而且数据量很大，可以考虑存放在磁带或光盘上。
- 如果计算机有多个磁盘，可以考虑将表和索引分别放在不同的磁盘上，在查询时，由于两个磁盘驱动器分别在工作，因而可以保证物理读写速度比较快。
- 可以将比较大的表分别放在两个磁盘上，以加快存取速度，这在多用户环境下特别有效。
- 可以将日志文件与数据库对象（表、索引等）放在不同的磁盘上，提高I/O并发速度，以改进系统的性能。
- 对空间效率要求较高的应用，考虑压缩存储
- 对安全需求较高的应用，考虑加密存储

确定系统配置

- DBMS产品一般都提供了一些存储分配参数
- 系统都为这些变量赋予了合理的缺省值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要根据应用环境确定这些参数值，以使系统性能最优。
- 在物理设计时对系统配置变量的调整只是初步的，在系统运行时还要根据系统实际运行情况做进一步的调整，以期切实改进系统性能。

确定系统配置

- 最大连接数：如果并发连接请求量较大，建议调高
- 内存配置参数（查询缓存、日志缓存、数据缓存.....）：受总的物理内存大小的约束，参数之间相互制约
- 存储配置参数（数据文件的大小、日志文件大小、物理块大小、物理块装填因子）
- .....

- 参数优化不可能一次性完成，需要不断的观察及调试。

7.5.3 确定数据库的存储结构

- 影响数据存放位置和存储结构的因素
  - 硬件环境
  - 应用需求
    - 存取时间
    - 存储空间利用率
    - 维护代价

这三个方面常常是相互矛盾的

例：消除一切冗余数据虽能够节约存储空间和减少维护代价，但往往会导致检索代价的增加

必须进行权衡，选择一个折中方案。

7.5.4 评价物理结构

- 评价内容
  - 对数据库物理设计过程中产生的多种方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。
- 评价方法
  - 定量估算各种方案
    - 存储空间
    - 存取时间
    - 维护代价
  - 对估算结果进行权衡、比较，选择出一个较优的合理的物理结构
  - 如果该结构不符合用户需求，则需要修改设计

7.6 数据库的实施和维护

完成数据库的物理设计之后

- 设计人员就要用**RDBMS**提供的数据定义语言和其他实用程序将数据库逻辑设计和物理设计结果严格描述出来，成为**DBMS**可以接受的源代码。经过调试产生目标模式。
- 组织数据入库

一、数据载入

- 人工方法：适用于小型系统
  - 步骤

1) 筛选数据。需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。

2) 转换数据格式。筛选出来的需要入库的数据，其格式往往不符合数据库要求，还需要进行转换。这种转换有时可能很复杂。

3) 输入数据。将转换好的数据输入计算机中。

4) 校验数据。检查输入的数据是否有误。

一、数据载入

- 计算机辅助数据入库：适用于中大型系统
  - 步骤
    1) 筛选数据
    2) 输入数据。由录入员将原始数据直接输入计算机中。数据输入子系统应提供输入界面。
    3) 校验数据。数据输入子系统采用多种检验技术检查输入数据的正确性。
    4) 转换数据。数据输入子系统根据数据库系统的要求，从录入的数据中抽取有用成分，对其进行分类，然后转换数据格式。抽取、分类和转换数据是数据输入子系统的主要工作，也是数据输入子系统的复杂性所在。
    5) 综合数据。数据输入子系统对转换好的数据根据系统的要求进一步综合成最终数据。

一、数据装载

- 如果数据库是在原数据库系统的基础上设计的，则数据输入子系统只需要完成转换数据、综合数据两项工作，直接将老系统中的数据转换成新系统中需要的数据格式。要充分利用数据转换工具。

- 为了保证数据能够及时入库，应在数据库物理设计的同时编制数据输入子系统。

二、编制与调试应用程序

- 数据库应用程序的设计应该与数据设计并行进行。
- 在数据库实施阶段，当数据库结构建立好后，就可以开始编制与调试数据库的应用程序。调试应用程序时由于数据入库尚未完成，可先使用模拟数据。

7.6.2 数据库试运行

- 应用程序调试完成，并且已有一小部分数据入库后，就可以开始数据库的试运行。

- 数据库试运行也称为联合调试，其主要工作包括：

  1）**功能测试**：实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能。

  2）**性能测试**：测量系统的性能指标，分析是否符合设计目标。

一、数据库性能指标的测量

- 数据库物理设计阶段在评价数据库结构估算时间、空间指标时，作了许多简化和假设，忽略了许多次要因素，因此结果必然很粗糙。

- 数据库试运行则是要实际测量系统的各种性能指标（不仅是时间、空间指标），如果结果不符合设计目标，则需要返回物理设计阶段，调整物理结构，修改参数；有时甚至需要返回逻辑设计阶段，调整逻辑结构。

二、数据的分批入库

- 重新设计物理结构甚至逻辑结构，会导致数据重新入库。
- 由于数据入库工作量实在太大，所以可以采用分批输入数据的方法
  - 先输入小批量数据供先期联合调试使用
  - 待试运行基本合格后再输入大批量数据
  - 逐步增加数据量，逐步完成运行评价

三、数据库的转储和恢复

- 在数据库试运行阶段，系统还不稳定，硬、软件故障随时都可能发生
- 系统的操作人员对新系统还不熟悉，误操作也不可避免
- 因此必须做好数据库的转储和恢复工作，尽量减少对数据库的破坏。

7.6.3 数据库运行和维护

- 数据库试运行结果符合设计目标后，数据库就可以真正投入运行了。
- 对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。
  - 应用环境在不断变化
  - 数据库运行过程中物理存储会不断变化
- 在数据库运行阶段，对数据库经常性的维护工作主要是由DBA完成的，包括：

1. 数据库的转储和恢复

- 转储和恢复是系统正式运行后最重要的维护工作之一。
- DBA要针对不同的应用要求制定不同的转储计划，定期对数据库和日志文件进行备份。
- 一旦发生介质故障，即利用数据库备份及日志文件备份，尽快将数据库恢复到某种一致性状态。

2.数据库的安全性、完整性控制

- DBA必须根据用户的实际需要授予不同的操作权限
- 在数据库运行过程中，由于应用环境的变化，对安全性的要求也会发生变化，DBA需要根据实际情况修改原有的安全性控制。
- 由于应用环境的变化，数据库的完整性约束条件也会变化，也需要DBA不断修正，以满足用户要求。

3.数据库性能的监督、分析和改进

- 在数据库运行过程中， DBA必须监督系统运行，对监测数据进行分析，找出改进系统性能的方法。
  - 利用监测工具获取系统运行过程中一系列性能参数的值;
  - 通过仔细分析这些数据，判断当前系统是否处于最佳运行状态，应当做哪些改进，例如调整系统物理或对数据库进行重组织或重构造。

4. 数据库的重组织和重构造

1）数据库的重组织

- 为什么要重组织数据库
  - 数据库运行一段时间后，由于记录的不断增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。
- 重组织的形式
  - 全部重组织
  - 部分重组织
    - 只对频繁增、删的表进行重组织
- 重组织的目标
  - 提高系统性能

4.数据库的重组织和重构造

- 重组织的工作
  - 按原设计要求
    - 重新安排存储位置
    - 回收垃圾
    - 减少指针链
  - 数据库的重组织不会改变原设计的数据逻辑结构和物理结构
- DBMS一般都提供了供重组织数据库使用的实用程序，帮助DBA重新组织数据库。

4.数据库的重组织和重构造

2）数据库的重构造

– 为什么要进行数据库的重构造

- 数据库应用环境发生变化，会导致实体及实体间的联系也发生相应的变化，使原有的数据库设计不能很好地满足新的需求
  - –增加新的应用或新的实体
  - –取消某些已有应用
  - –改变某些已有应用

4.数据库的重组织和重构造

- 数据库重构造的主要工作
  - 根据新环境调整数据库的模式和内模式
    - 增加新的数据项
    - 改变数据项的类型
    - 改变数据库的容量
    - 增加或删除索引
    - 修改完整性约束条件

4.数据库的重组织和重构造

- 重构造数据库的程度是有限的
  - 若应用变化太大，已无法通过重构数据库来满足新的需求，或重构数据库的代价太大，则表明现有数据库应用系统的生命周期已经结束，应该重新设计新的数据库系统，开始新数据库应用系统的生命周期。