数据库技术-数据库设计

赵亚伟

zhaoyw@ucas.ac.cn

中国科学院大学 大数据分析技术实验室

2017.12.10

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

数据库应用系统设计的两个方面

- 从系统开发的角度来看，数据库应用系统的设计应包括两个方面：

  （1）结构特性的设计

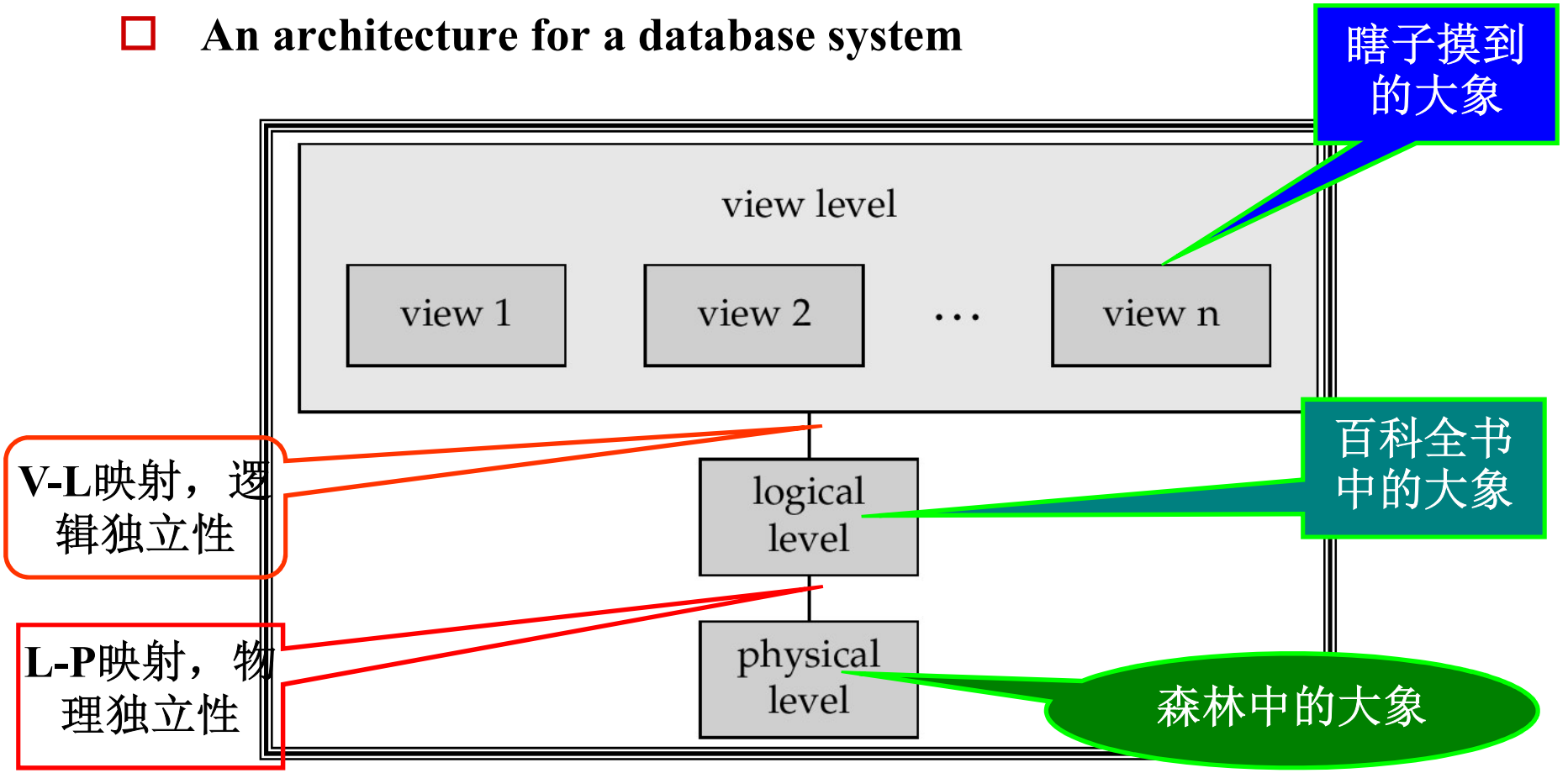
  结构特性的设计是指数据结构的设计，其结果是得到一个合理的数据模型，以反映现实世界中事物及事物间的联系，它包括各级数据库模式的设计。

  （2）行为特性的设计

  行为特性的设计是指应用程序的设计。在分析系统功能需求的基础上，完成各个功能模块的设计。

关于三级模式两级映射

- 数据抽象：划分层次— 三级模式两级映像
- **An architecture for a database system**



抽象层次

- **Physical level describes how a record (e.g., customer) is stored.**物理层描述如何保存(磁介质等)：*森林中的大象*
- **Logical level: describes data stored in database, and the relationships among the data.**

```
type customer = record
                    name : string;
                    street : string;
                    city : integer;
                end;   //PASCAL语言描述的一个记录
```

逻辑层描述数据之间的联系：*百科全书中的大象*

- **View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.** 视图层：应用程序隐藏细节，可欺上瞒下， *瞎子摸到的大象*

抽象层次

- 物理层、逻辑层、视图层是三个不同的抽象层次，物理层是最低层次，视图是最高层次。
- 三个层次还有别名，即内模式（又称存储模式）、模式（又称逻辑模式）、外模式（又称用户模式），所以又称——数据库的三级模式
  - 外模式：用户模式，用户能够看见和使用的局部数据的逻辑结果和特征描述
  - 模式：逻辑模式，数据库中全体数据逻辑结构和特征的描述
  - 内模式：存储模式，数据物理结构和存贮方式的描述，是数据在数据库内部的表示方式
- 一个数据库系统只有一个内模式和模式，但可能会有多个外模式
- 数据库系统的所有方法和技术基本上都围绕这三个层次展开

模式间映像

- 数据库的三级模式是对数据的三个抽象级别，为了能够实现这三个抽象层次的联系和转换，在这三级模式之间提供了二级映像。
  - 外模式/模式映像
  - 模式/内模式映像
- 对于一个数据库系统，外模式/模式映像可能是多个，而模式/内模式映像是唯一的
- 二级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。
- 两层独立性的目的：下面变化了，上层可以不变或少变

数据库设计的特点：

三分技术；

七分管理；

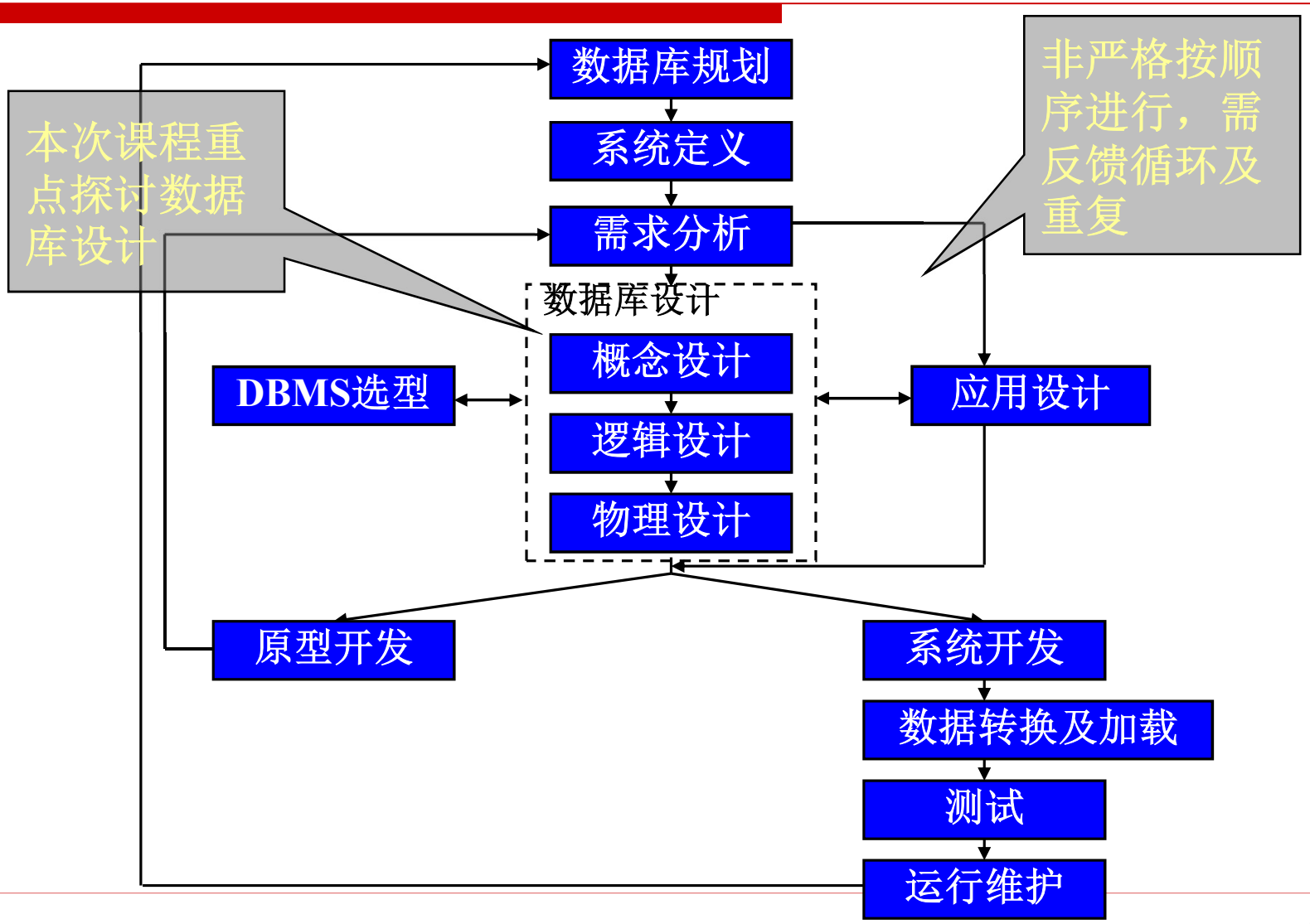
十二分基础数据。

注意：不能轻视技术，技术是创新的基础

数据库应用系统生命周期

- 数据库规划
- 系统定义
- 需求收集及分析
- 数据库设计
- **DBMS选型(可选)**
- 应用系统设计
- 原型开发(可选)
- 开发实现
- 数据转换及加载
- 测试
- 运行维护

数据库应用系统生命周期



目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

数据库设计的三阶段

- 数据库设计分为三个阶段
  - 概念设计
  - 逻辑设计
  - 物理设计
- 数据模型的主要用途
  - 帮助理解数据的含义
  - 便于信息沟通，对需求分析很有利
- 两个常用的数据模型：**ER**模型及关系模型都具有简洁、明了的特征。因此，**ER**模型很适合与用户进行交流沟通，关系模型适合设计人员之间的交流沟通。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

概念设计阶段

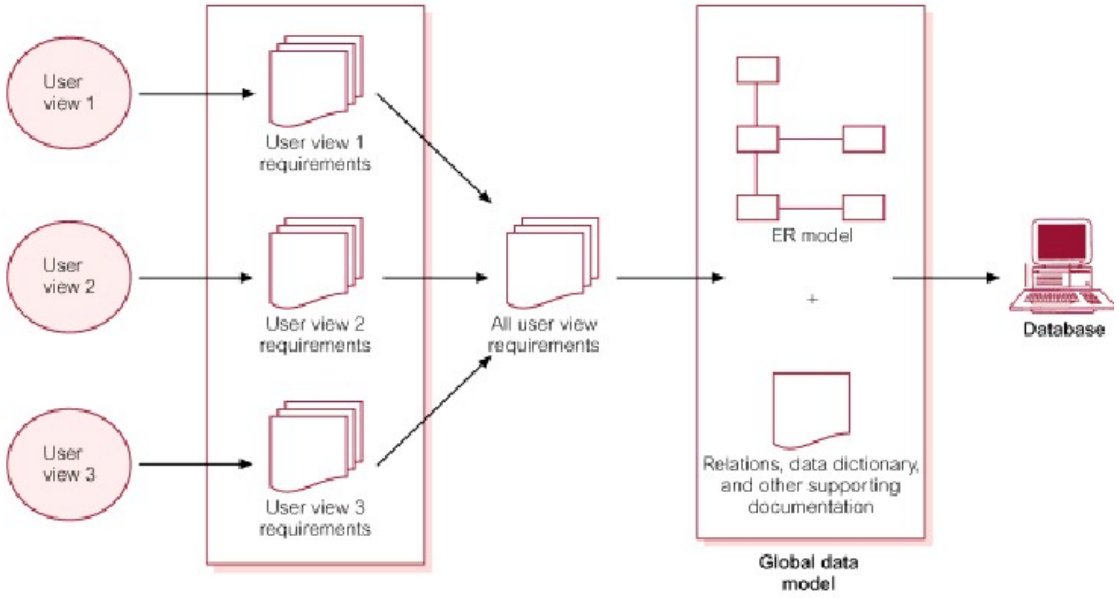
- 概念设计阶段构建的数据模型独立于计算机的物理实现－便于沟通
- 根据需求说明中的信息构建信息模型－有据可依
- 为逻辑设计阶段提供信息－承前启后

概念模型设计

- 概念模型设计的目标是产生一个用户易于理解的、反映信息需求的整体数据库概念模型。
- 概念模型是系统中各个用户共同关心的信息结构。
- 概念模型设计的常用方法是实体联系方法（**ER**模型）。用实体联系方法对具体数据进行抽象加工，将实体集合抽象成实体类型，用实体间联系反映现实世界事物间的内在联系。

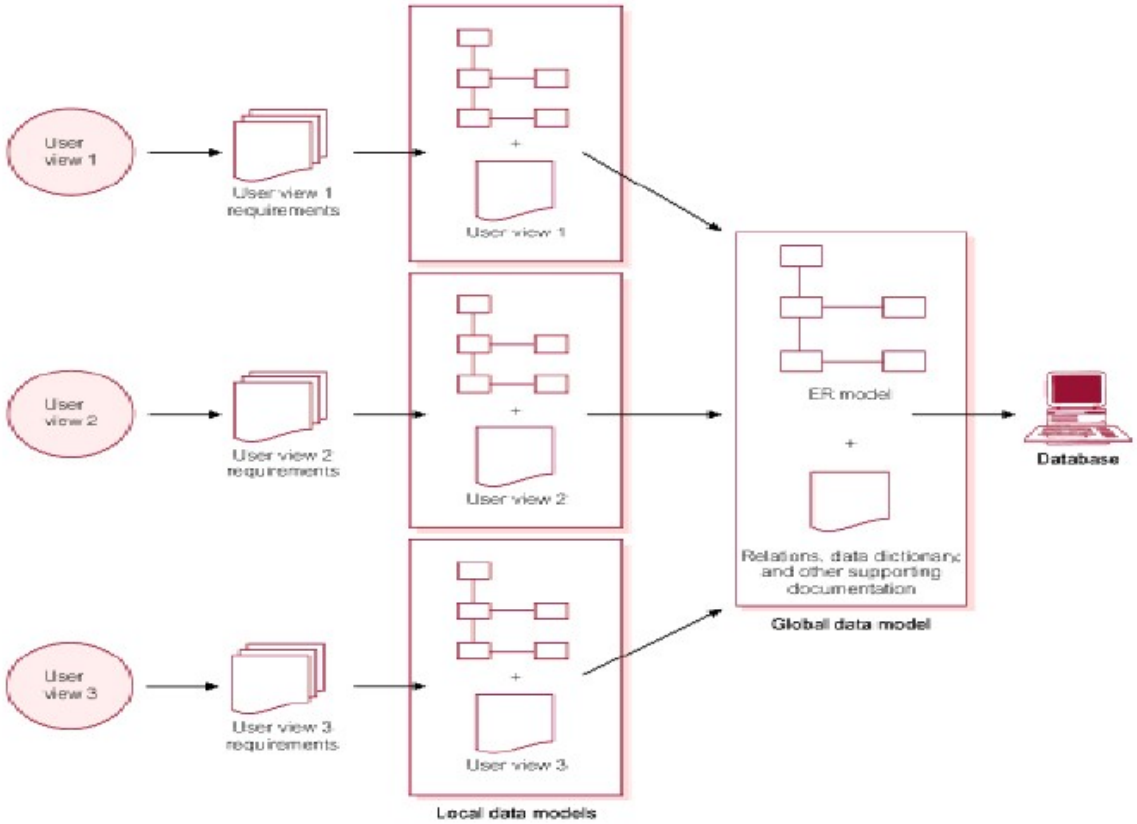
Global data model

- 将每个用户视图合并(Merge)成一个需求集合。
- 根据综合的需求集合形成一个单独的全局数据模型，该模型描述了所有用户的需求。难度大



Local data model

- 根据每个用户需求建立相互独立的Local data model（各用户需求图表和文档）
- 然后将局部数据模型合并成一个全局数据模型（描述了所有用户对数据库的需求）。各个击破，较常用



Local data model的集成

- 集成的主要任务是归并和重构局部数据模型，最后得到统一的整体视图。集成后的视图应满足以下要求：
  - 完整性和正确性，即整体数据模型应包含各局部视图所表达的所有语义；正确表达与所有局部视图应用相关的数据观点；
  - 最小化，系统中同一个对象原则上只在一个地方表示；
  - 可理解性，即整体数据模型对于设计者和用户都应是易于理解的。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

逻辑设计

- 逻辑模型设计的主要目标是产生一个**DBMS**可处理的数据模型和数据库模式。该模型必须满足数据库的存取、一致性及运行等各方面的用户需求。
- 逻辑结构设计的步骤
  - 将概念模型转换为一般的关系模型（参**ER**模型部分）
  - 对数据模型进行优化（参**NF**部分）
- 逻辑设计仍然独立于特定的**DBMS**和其他物理实现。
- 逻辑数据模型是概念设计模型的精确和映射。

关系规范化处理

- 关系规范化处理是一个优化过程，目标是提高数据库应用系统的性能。
- 具体方法
  - 确定函数依赖：写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间的数据依赖
  - 对各个关系模式之间的数据依赖进行极小化处理（注意：是模式级的），消除冗余联系
  - 按照数据依赖的理论对关系模式逐一进行分析，考察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
  - 按照需求分析阶段的处理要求，分析各模式是否适合应用环境，确定是否要对某些模式进行合并或分解

- 并不是分解程度越高就越优越。如当查询经常涉及到两个或多个关系模式的属性时，连接操作的代价是关系模式低效的主要因素。因此，这种情况下，**1NF**或**2NF**可能效率更高。
- 对关系模式进行必要的分解，提高数据操作的效率和存储空间的利用率。
- 常用的分解：
  - 水平分解：把元组分为若干子集，定义每个子集为一个子关系。划分原则为**80/20**原则，即经常被使用的数据为**20%**，可以分解出来。
  - 垂直分解：把属性分解为若干子集，把经常在一起使用的属性分解出来形成子模式

逻辑模型评价

- 逻辑模型是否符合用户的应用要求，要进一步审查。主要从功能和性能两个方面考虑。
- 功能评价主要依据需求分析后确定的系统功能，主要包括
  - 审查逻辑模型是否支持用户的所有应用要求
  - 检查每个处理所涉及的信息是否在逻辑结构中或可从中导出
  - 针对冗余的属性或关系，审查是真正冗余还是为系统扩充而设计
- 如果逻辑模型不能满足应用的要求，就要对模型进行修正，如果属于需求分析或概念设计阶段的问题，则应返回到前两个阶段重新设计。

- 性能评价主要考虑数据的存取效率和空间利用率。**LRA方法(Logical Record Access Method)**是度量数据库逻辑结构性能的一种方法，其主要指标有：
  - **LRA(Logic Record Access)**：单位时间内所有处理需存取记录的总数；
  - **TrVol(Transport Volume)**：单位时间内所有处理需存取的数据量；
  - **StroS(Storage Space)**：数据占用空间，由数据空间**DStor**和指针空间**PtrStor**组成。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

物理设计

- 物理设计是数据库逻辑设计在数据库工具上的实现过程。
- 物理设计的目标是从逻辑模型出发，设计一个在限定的软、硬件条件和应用环境下可实现的，并具有尽可能高的运行效率的物理数据库结构。
- 关系数据库的物理模型设计相对于其它模型而言是较为简单的，这是由于关系数据模型提供了较高的逻辑数据和物理数据的独立性，而且大多数物理设计因素都由**RDBMS**处理，留给设计人员控制的因素已经很少。

物理设计内容

- 一般来说，物理设计阶段，设计人员需要考虑以下内容：
  - 选择数据库文件的存储结构。如流水文件、顺序文件、聚簇文件、哈希(Hash)文件、B^+树文件等
  - 选择索引。选择索引的目的在于实施关键字约束和提高访问效率。
  - 分配存储空间。许多DBMS的存储空间是以区段和页块来组织的。设计人员可按DBMS所提供的计算公式计算系统空间占用量，计算时要考虑当前数据量、增长率和索引情况，并为未来的扩展留有余地。

DBMS选型（可选）

- DBMS选型是物理设计阶段的重要工作。
- 选择DBMS的主要步骤
  - 定义分析条目
  - 列出几个备选产品
  - 进行评估
  - 给出产品选型建议报告
- DBMS评估指标反映了其性能情况，不同的DBMS产品具有不同的优势，只有根据需求进行选择

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

应用设计

- 应用设计是指界面设计和事务处理设计，应用设计实质是如何保证有效地使用和处理数据库
- 应用设计与数据库设计是并行的活动，可以同时进行，两个环节交互完成设计。
- 应用设计包括
  - 界面设计
  - 事务处理设计

应用设计－事务处理设计

- 由一个用户或应用程序执行的一个活动或一系列活动，或者完整执行或者不执行，这样的活动称为事务。
- 事务是访问和改变数据库的基本活动。
- 事务处理设计依据数据处理的需求分析，这些信息在需求分析阶段可以获得。
- 事务处理设计本质上属于物理设计阶段。

事务特征

- 从用户的角度看，事务的重要特征
  - 事务对数据进行操作
  - 具有特定功能
  - 有输出
  - 对用户来说是重要的
  - 有一定的使用率
- 有三种类型的事务：恢复、更新及二者混合。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

开发实现

- 数据库的开发实现主要采用**SQL**语句进行系统的实现，包括
  - 使用**DDL**创建数据库模式及空的数据库文件
  - 使用**DDL**创建用户视图
  - 创建应用程序，包括使用**DML**或嵌入式**SQL**实现事务处理

  - 前两个任务本质上属于物理设计阶段。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

测试

- 运行应用程序发现存在的问题。
- 使用详细的测试计划和数据。
- 测试不能发现潜在的错误，只能够发现显式的软件的错误。
- 根据需求测试数据库及应用，判断能否发布。

目录

- 数据库生命周期
- 数据库设计
  - 概念设计
  - 逻辑设计
  - 物理设计
- 应用系统设计
- 开发实现
- 测试
- 运行维护

运行维护

- 系统上线后的监测和维护活动。
- 监测系统性能
  - 如果性能下降，则需要进行调优或重新组织数据库
- 如果需要，则需要升级数据库应用。
- 使新的需求并入数据库应用中。

- **DA及DBA**的职责（设计、监控、维护），国内一般合二为一。

数据库设计参考

- 数据库设计模板
  - 数据库设计模板很多，很多企业根据自身情况制定企业标准
  - 国家标准（**GB 8567-88**，数据库设计说明书）

小结

- 设计一个数据库应用系统的重点应放在需求分析、概念设计、逻辑设计三个主要阶段，设计过程中往往还会有许多反复
- 贯穿整个设计过程之中的是数据模型
- 设计要尽量做到简洁明了