1. 引论

1.1数据是指表示信息的符号。

1.2数据管理发展阶段

人工管理阶段特点：数据不保存；没有专用的软件管理数据；应用程序完全依赖于数据；数据不能共享。

文件系统阶段特点：数据可以长时间保存；有专门的文件系统软件管理数据；数据是面向应用的；数据冗余度大；缺乏对数据统一的控制机制。

数据库系统阶段：数据结构化；数据独立性高（物理、逻辑独立性）；减少数据冗余；数据共享；统一的数据保护功能。

1.3数据库（DB）：存放在介质上的有组织的数据集合

数据库系统（DBS）：对数据提供存储管理和应用的计算机系统。

数据库管理系统（DBMS），具有数据定义、数据操纵、运行管理和维护等功能，是数据库系统的核心。

数据库管理员（DBA）

* 1. 数据模型是数据特征的抽象，包含三个方面（三要素）：数据结构（最重要）、数据操作、数据的完整性约束。

数据模型有两个层次：逻辑数据模型（表示逻辑结构）、物理数据模型（表示数据存储结构和存储方法）。

1.5数据库系统的三级模式结构：外模式、模式和内模式

模式（又称概念模式或数据模式），是全体数据的逻辑结构和特征描述，一个数据库对应一个模式，提供模式描述语言DDL。

外模式（又称子模式），是数据库用户的数据视图，通常为模式的子集，面向应用程序，一个外模式对应一个或多个应用，一个模式可以对应多个外模式。

内模式，数据的物理结构和存储方式的描述，一个数据库对应一个内模式。

三级模式对数据的三个抽象级别（视图级、概念级、物理级）

1.6三级模式结构的二级映像

目的：实现三个模式的联系和转换

外模式/模式映像：一个模式对应多个外模式，当模式结构改变，则只要修改该映像关系，不必修改外模式的局部逻辑结构，实现了数据的逻辑独立性。

模式/内模式映像：一个模式对应一个内模式，当数据库物理存储结构改变时，仅需修改该映像关系，提供了数据的物理独立性。

1.7数据库管理系统的功能

定义功能：定义模式、外模式、内模式

操纵功能：增删改查

保护功能：安全性控制、完整性控制、并发控制、恢复

维护功能：转储、装入、统计分析、重组

1.8数据库不同视图

数据库管理员：决定数据库逻辑结构和物理结构，定义数据安全性要求和约束条件，监控维护数据库的运行，数据库的改进、重组和重构

系统分析员：负责应用系统的需求分析和规范说明，与用户及DBA协商确定系统的软硬件配置，参与数据库系统的概要设计

数据库设计员：参与用户需求调查和系统分析，确定数据库中的数据，设计数据库各级模式

应用程序员：设计编写应用系统的程序模块，调试与安装

用户

其中参与数据库设计过程始终的是系统分析员、数据库设计员

其中主要参与需求分析阶段和运行维护的是DBA和用户

其中在系统实施阶段参与的是应用开发人员

1. 数据模型

2.1 数据模型包括概念模型、逻辑模型、物理模型

概念模型：按用户观点对数据和信息建模

逻辑模型：按计算机系统的观点对数据建模

物理模型：描述数据在系统内部的表示方式和存取方法

三要素为：数据结构、数据操作、数据的完整性约束

2.2 E-R概念模型（设计题第一步）

实体（用矩形表示）：客观存在并可相互区别的事物

属性（用椭圆形表示）：实体所具有的某一特性称为属性

联系（用菱形框表示，联系本身也可以有属性，例如选课联系的成绩）：在现实世界中，事物内部以及事物之间有联系的，具有基数比约束、参与约束、实体参与度等语义

两个实体集之间的三类联系：1:1联系、1:n联系、m:n联系

多个实体集之间存在联系：m:n:p联系（例：某供应商提供某零件给某工程），而三个实体间两两多对多联系仅能表示每两个实体集之间的联系（例：无法表示某工程所用某零件由哪个供应商供应）

同一实体集内不同实体也存在联系

弱实体（用双框矩形表示，与菱形框用双线联系）：依赖于强实体存在，如家长对学生是弱实体

子类实体

2.3层次模型和网状模型

层次数据模型：按层次组织数据，使用树形数据结构

网状模型：节点间的联系是任意的

数据间的联系用指针表示

2.4关系数据模型

基本数据结构是二维表

关系：一张二维表，用来描述一个实体集

属性：表示实体集某一特征，为关系的某一列

域：某一列的取值范围

元组：表示实体的实例，为关系的某一行

键：一个或多个属性构成，能唯一标识一个元组，一个关系中可能有多个键，选择其中一个作为主键，其余为候选键。若键由多个属性组成，称为联合键；若键由关系所有属性组成，称为全键。

1. 关系数据库

3.1 关系的规范化性质：

关系定义：n个域的笛卡尔积的子集，n称为关系的目或者度。

①每一列中的值是同类型的数据，来自同一个域。

②不同的列可以有相同的域，每一列称为属性。

③列的次序是无关紧要的。

④元组的每个分量是不可分的数据项。

⑤元组的次序是无关紧要的。

⑥每个元组是不同的，关系中不可出现重复的元组。

3.2关系数据模型

关系模式的形式：关系名（属性1，属性2,…,属性名n）

关系模式的集合称为关系数据库模式

数据完整性约束：实体完整性、参照完整性、用户自定义完整性。（实体、参照是必须的，且由系统自动支持）

数据操作：查询、插入、删除、修改。

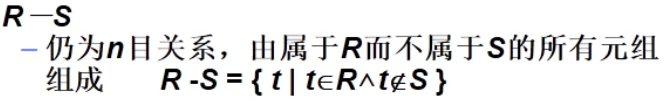
3.3关系代数运算

传统的集合运算：并、差、交、笛卡尔积

专门的关系运算：选择、投影、连接、除

五种基本运算：并、差、笛卡尔积、投影、选择

交

差

交

笛卡尔积

选择：一元运算，从关系中选择满足一定条件的元组子集

投影：从R中选择出若干属性列组成新关系

条件连接：

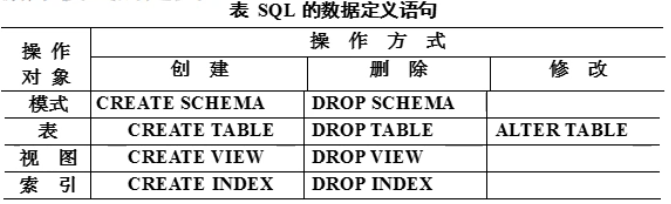
自然连接：在两个关系上共同属性的等值连接

1. 关系数据库标准语言SQL

4.1 SQL语言集数据定义语言DDL、数据操纵语言DML、数据控制语言DCL的功能于一体

4.2数据定义：create、drop、alter

SQL数据定义功能包括表、视图、索引和数据库模式的定义



4.3索引

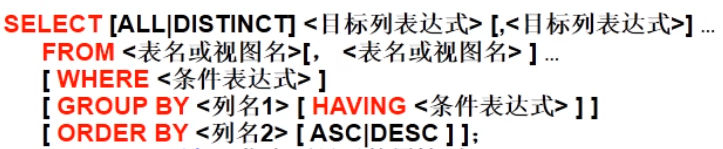
索引类型：聚集索引（索引顺序与数据库物理存储顺序相同，每个表只能有一个）、非聚集索引

聚集索引适用范围：经常查找范围值的列，很少对基表进行增删操作、很少对索引列进行修改操作



4.4数据查询与更新

语句格式：



Select子句：指定要显示的属性列

From子句：指定查询对象

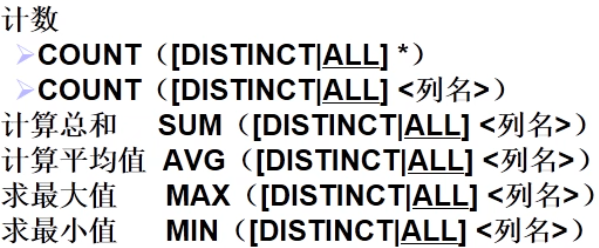
Where子句：指定查询条件

Group by子句：对查询结果按指定列分组，该属性列值相等的元组为一个组。

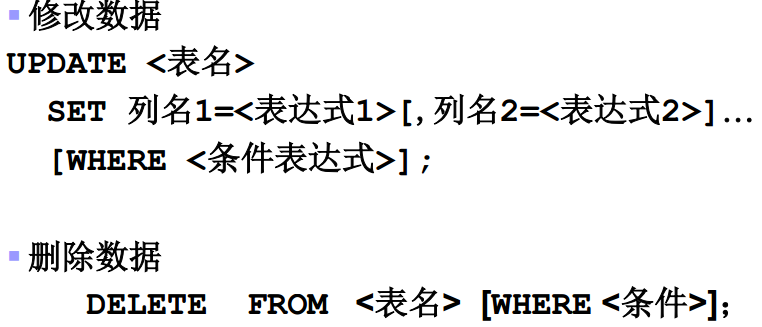
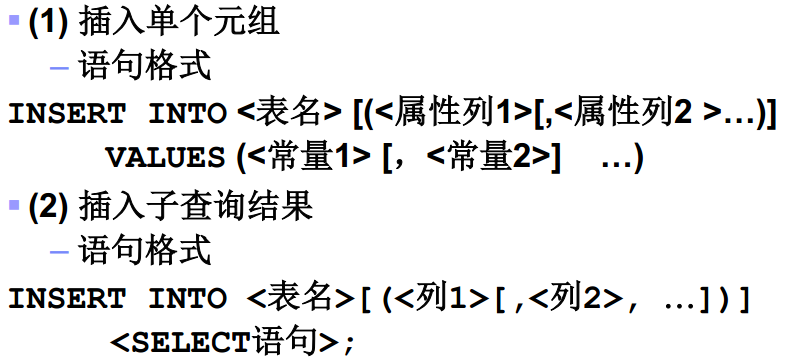
Having短语：筛选出值有满足指定条件的组

Order by子句：对查询结果按指定列值的升序或降序排序

集函数：主要集函数包括

distinct表示取消重复列，all不取消

使用Group by子句后，Select子句只能出现分组属性和集函数，使用having筛选出符合条件的组。



4.5视图

视图是建立在一个或多个基本表上的虚表。可以通过视图的更新转换成对基本表的更新。部分视图是无法更新的，对这些视图的更新不能唯一有效的转换为对基本表的更新

实现视图查询的方法：实体化视图、视图消解法（主要）。

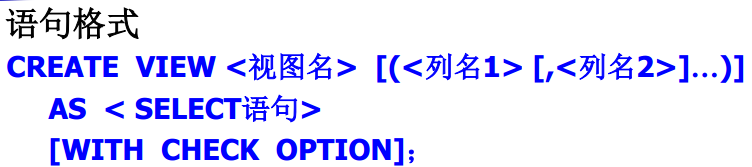
实体化视图：检查视图是否存在，将视图实体化为临时表并查询，删除临时表

视图消解法：检查视图是否存在，将视图定义中的子查询与用户查询结合，转化为等价的对基本表的查询，并执行转化后的查询

With check option 检查用户通过视图对不属于视图范围内的基本表数据更新。

仅在一个表上取其行列值且包含候选键的视图都是可更新的，称为行列子集视图。

视图优点：提供了逻辑数据独立性、简化用户视图、提供安全保护功能

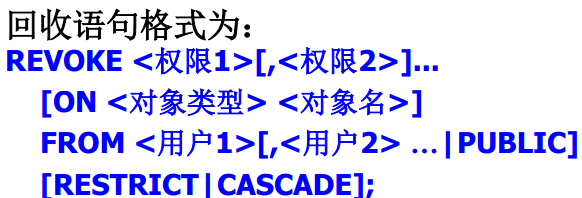
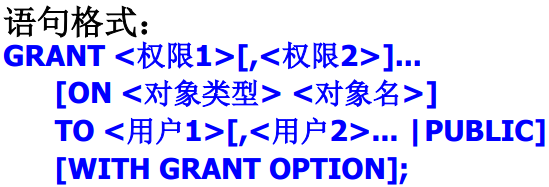


4.6SQL数据控制

权限包括：create、select、insert、delete、update、alter、index、drop

授权语句grant

回收语句revoke



DBA拥有所有权限，且可以授权和回收

用户对自己建立的基本表和视图有全部操作权限。

被授权用户若有许可则可传递授权

4.7 嵌入式SQL

嵌入式SQL的编译：所有嵌入式SQL语句必须加前缀EXEC SQL，以区分SQL语句与主语言语句。C语言一般表示为EXEC SQL <SQL语句>

数据库和主语言程序的通信：SQL中设有通信区SQLCA，会返回SQL语句执行情况，0表示成功，格式为EXEC SQL INCLUDE SQLCA；

数据库和主语言程序的数据交换：SQL语句中使用的主语言程序变量称为主变量，在SQL语句中要加”:”作为标志，在使用前均需declare语句说明

面向集合和面向记录的协调：SQL语言面向集合，主语言面向记录，引入游标机制协调。游标是数据缓冲器，存放SQL语句结果，包含结果集和指向结果集某一行的指针。使用步骤为定义、打开、推动、关闭。

引入游标机制，面向集合和记录。

1. 查询处理和查询优化

5.1查询处理过程

查询分析、查询检查（、建立查询内部表示）、查询优化、查询执行

查询分析：对查询语句语法检查和语法分析

查询检查：根据数据字典对合法的查询进行语义检查，包括用户权限和完整性约束

建立查询内部表示：转换成等价的关系代数表达式、查询树

查询优化：选择高效执行的查询处理策略，分为代数优化和物理优化，依据为基于规则、

代价、语义

查询执行：依据优化器得到的执行策略生成查询计划，代码生成器生成执行计划的代码

5.2基本算法

选择操作的实现：

全表扫描法，顺序扫描基本表，适合小表，不适合大表

索引（或散列）扫描法，适合选择条件中属性上有索引的选择，通过二分查找等形式找到满足条件的元组指针，再通过指针找到元组。

连接操作的实现：例如连接S、SC表

嵌套循环法，对外层循环的每个元组，检测内层循坏每个元组，在连接属性上是否相等，相等则串联后作为结果输出。因选择占用块数较小的表作为外表

排序合并法，适合连接的表已经对连接属性排好序的情况，两个表都只需要扫描一次

索引连接法，建立对连接属性的索引，与排序合并法类似

散列连接法，将连接属性作为hash码，用同一个hash函数把两个表中的元组散列到一个hash文件中，分为划分阶段和试探阶段

5.3查询优化

查询优化总目标：选择有效策略使得查询代价最小

代价主要包括：磁盘存取代价（I/O）、处理代价（CPU代价）、内存开销（、通信代价），其中I/O代价是最主要的

代数优化：按照一定规则改变代数表达式中操作的次序和组合，使查询更高效

基于存取路径的优化：

基于代价估算的优化：对多个可选的策略的代价进行估算，并选择代价较小的策略。

5.4代数优化策略

在关系代数表达式中尽可能早执行选择操作（最重要）

投影运算和选择运算可同时进行

将投影运算与其前或后的双目运算结合

把某些选择和笛卡尔积结合成为连接操作

找出公共子表达式

5.5 代数优化算法

把形如σF1∧F2∧…∧Fn(E)变换为σF1(σF2(…(σFn(E))…))。

对每个选择操作尽可能移到叶端

对投影操作尽可能移到叶端

把选择和投影的串接合并成单个选择、单个投影或一个选择后跟一个投影。

将每一双目运算和其直接祖先（选择投影）为一组

6. 数据库的安全性

6.1数据库安全性是指在数据库应用系统的不同层次提供安全防范措施，目标主要是保证数据的完整性、可用性、保密性和可审计性。

6.2 数据库的安全性主要包含的层次：数据库系统层、操作系统层、网络层。

6.3数据库系统所采用的安全技术主要包括以下几类：访问控制技术、存取控制技术、视图机制、数据加密技术、数据库审计。

6.4存取控制机制包括两部分内容，一是定义用户权限，而是合法权限的检查。控制机制分为两种，一是自主存取控制，二是强制存取控制

自主存取控制（DAC）：直接指定用户对于不同的数据库对象的存取权限，由三个要素组成（主体、客体、控制策略），用存取矩阵表示。用户的权限由数据对象和操作类型组成。

强制存取控制（MAC）：将数据库对象（客体）和用户（主体）被分为多个安全级别，对于任意一个对象，只有具有合法分类级别的用户才可以存取。规则为：仅当主体的许可证级别大于或等于客体的级别时，该主体才能读取相应的客体；仅当主体的许可证级别小于等于客体的级别时，该主体才能写相应的客体。

强制存取控制是对数据本身赋予密级标记，当数据复制时，数据与标记一起复制，从而提高了更高级别的安全性。

6.5视图机制：进行存取权限控制时可以为不同的用户定义不同的视图，把数据对象限制在一定范围内。通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的保护。

6.6数据加密：防止数据库数据在存储和传输过程中失密的有效手段。

基本思想：根据一定的算法将原始数据（属于为明文）变换为不可直接识别的格式（术语为密文）从而使得不知道解密算法的人无法获知数据的内容。

6.7数据库审计：把用户对数据库的所有操作自动记录下来放入审计日志中，DBA可以通过阅读审计文件，发现非法访问行为。一般分为用户审计（任何用户都可以设置）和系统审计（只能由DBA设置）。

6.8统计数据库安全性

统计数据库特点：以统计为主，统计数据是公开的，个别数据是保密的。

需要堵塞可能存在的隐蔽的信息通道，防止通过合法的查询推导出不合法信息。

解决方法：用随机数修改保密的属性值、采样法。

7.数据库完整性

7.1数据库的完整性是指数据的正确性、有效性、相容性，其主要目的是防止错误的数据进入数据库。

7.2完整性约束条件，是完整性控制机制的核心，包括实体完整性、参照完整性、用户自定义完整性。

约束条件的对象：关系-元组、元组集合间、关系间的约束；元组-元组的各列之间的约束；列级约束；

约束的状态：静态（某个确定状态时数据的约束）；动态（状态转移时新旧值之间的约束）

7.3实现数据完整性的方法

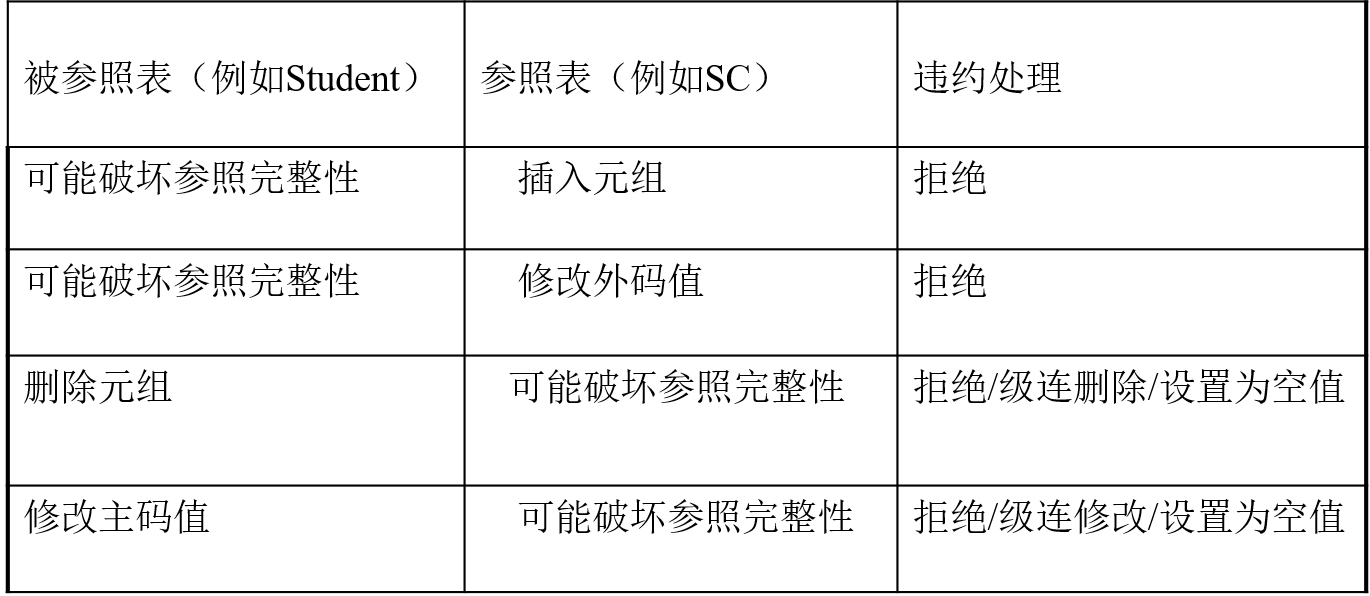
DBMS在完整性约束控制方面应当具有以下功能：提供定义完整性约束条件的机制；提供完整性检查的方法；违约处理。

7.4实体完整性：规定主键的值唯一且不能取空值，使用PRIMARY KEY定义

插入或对主码列进行更新操作时，RDBMS自动检查主码值是否唯一、主码各个属性是否为空，若不合规则就拒绝插入或修改

7.5参照完整性：其核心是不允许引用不存在的元组，使用 FOREIGN KEY（定义哪些列为外码）和REFERENCES（指明这些外码参照哪些表的主码）定义。

拒绝执行（no action）、级联操作（cascade）、设为空值（set-null）



7.6属性上约束条件定义：

列值非空（NOT NULL）、列值唯一（UNIQUE）、检查列值是否满足一个布尔表达式（CHECK）、默认值（DEFAULT）

7.7完整性约束命名子句

CONSTRAINT <完整性约束条件名> [PRIMARY KEY短语|FOREIGN KEY短语|CHECK短语]

违约处理：拒绝执行

7.8域中的完整性限制，使用CREATE DOMAIN语句建立一个域

使用ALTER DOMAIN <域名> ADD\DROP CONSTRAINT <限制名> …… 来添加或删除域上的条件限制

7.9触发器：用户定义在关系表上的一类由事件驱动的特殊过程

语法格式如下：

CREATE TRIGGER <触发器名>

{BEFORE | AFTER} <触发事件> ON <表名>

FOR EACH {ROW | STATEMENT}

[WHEN <触发条件>］

<触发动作体>

触发事件包括：INSERT、DELETE、UPDATE

触发器分为行级触发器和语句级触发器。

以下为一个BEFORE行级触发器的例子：

CREATE TRIGGER Insert\_Or\_Update\_Sal

BEFORE INSERT OR UPDATE ON Teacher

/\*触发事件是插入或更新操作\*/

FOR EACH ROW /\*行级触发器\*/

AS BEGIN /\*定义触发动作体，是PL/SQL过程块\*/

IF (new.Job='教授') AND (new.Sal < 4000) THEN

new.Sal :=4000;

END IF;

END;

7.10激活触发器：同一个表上多个触发器激活过程如下

执行该表上的BEFORE触发器；

激活触发器的SQL语句；

执行该表上的AFTER触发器；

8．数据库恢复技术

8.1事务：用户定义的一个数据库操作序列，是恢复和并发控制的基本单位。

事务处理技术主要包括数据库恢复技术和并发控制技术。

定义事务（显示、隐式）

特性：原子性A、一致性C、隔离性I、持续性D，简称ACID特性。

8,2事务状态

数据库恢复到事务执行前的状态，称为事务回滚。

活动状态：事务执行时的状态。

部分提交状态：语句执行完毕，但数据还有部分驻留在内存中的状态。

失败状态：事务没有成功完成，必须回滚。

终止状态：事务回滚且数据库已经恢复到事务执行前的状态。

提交状态：数据更改完全写入磁盘，写入事务日志信息，事务成功完成。

8.3数据库故障

系统故障：造成系统停止运转的任何事件，使得系统要重新启动，运行事务全部非正常终止，内存中的数据缓存区信息全部丢失，外存不受影响。

事务内部故障：事务未运行至正常终止点就夭折，通常指非预期的情况，比如运算溢出、违反约束、应用程序出错、发生死锁。

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。

8.4恢复的实现技术

原理：数据冗余，利用存储在系统其它地方的冗余数据来重建数据库中已被破坏或不正确的那部分数据。

关键问题：如何建立冗余数据（数据转储、登记日记文件）；如何利用冗余数据实施数据库恢复。

8.5数据转储：DBA定期地将整个数据库复制到磁带或另一个磁盘上保存起来。

转储方法：静态转储和动态转储；海量转储和增量转储。

静态转储：在系统中无运行事务时进行转储，转储期间不允许对数据库的任何存取、修改。

动态转储：转储操作与用户事务并发进行，转储期间允许对数据库进行存取或修改。但无法保证副本数据正确有效，要配合日志恢复。

海量转储：每次转储全部数据库。

增量转储：只转储上次转储后更新过的数据。

8.6基于日志的数据库恢复

日志文件：记录有关事务的数据库操作信息，内容包括事务标识、操作类型、操作对象、更新前后的数据值。

登记原则：登记的次序严格按并行事务执行的时间次序；必须先写日志文件，后写数据库。

恢复技术：

Redo技术：如果发生故障，用Redo操作重做事务，恢复已完成的事务。

Undo技术：在事务执行过程中修改了数据库的事务尚未提交，如果此时发生故障，用Undo将被修改的数据项恢复到事务开始前的状态。（顺序与写入日志顺序相反）

8.7事务故障的恢复

恢复方法：利用日志文件撤销事务对数据的修改，系统回滚值事务执行前的状态。

事务故障的恢复由系统自动完成，不需要用户干预。

过程：反向扫描日志，查找该事务的更新操作，并对更新执行逆操作，直至读到此事务的开始标记。

8.8系统故障的恢复

系统故障的恢复由系统在重新启动时自动完成，不需要用户干预。

原因：一些未完成事务对数据库的更新已写入数据库、一些已提交事务对数据库的更新还留在缓冲区没来得及写入数据库。

恢复方法：Undo故障发生时未完成的事务，Redo已完成的事务

过程：正向扫描日志文件，将故障发生前已提交的事务加入Redo队列，发生时未完成的事务加入Undo队列；反向扫描日志文件，对Undo队列事务进行Undo处理；正向扫描日志文件，对每个Redo队列事务进行Redo处理。

8.9介质故障的恢复

介质故障的恢复需要DBA介入。

方法：重装数据库、重做已完成的事务。

过程：装入最新的后备数据库副本，使数据库恢复到最近一次转储时的状态（对于静态转储，装入后即处于一致性状态；对于动态转储，还需要装入转储时刻的日志文件副本，利用与恢复系统故障相同的方法）；重做已完成的事务。

8.10检查点恢复技术

检查点：记录在日志中表示数据库是否正常运行的一个标志，记录当前所有活动的事务。

内容：建立检查点时刻所有正在执行的事务清单、这些事务最近一个日志记录的地址；

重启动文件：记录各个检查点记录在日志文件中的地址。

写检查点的步骤：把日志缓冲区记录写入日志文件；在日志文件写入检查点记录；把数据缓冲区的内容写入数据库；把检查点记录在日志文件中地址写入重启动文件。

过程：从重启动文件中找到最后一个检查点记录，把该检查点的事务清单中的事务暂时放入Undo队列；从检查点开始正向扫描日志文件，若有新开始的事务暂时放入Undo队列，若有提交事务T，则将T从Undo队列移到Redo队列；对Undo队列事务进行Undo处理，对Redo队列事务进行Redo处理。

8.11数据库镜像恢复技术

数据库镜像：DBMS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。出现故障时，由镜像盘继续提供使用，同时DBMS自动利用镜像磁盘数据进行数据库恢复。

9.并发控制

9.1并发操作带来的数据不一致性（破坏了事务的隔离性）

丢失更新：事务1读->事务2读->事务1改->事务2改。事务1修改丢失。

不可重复读：事务1读->事务2改->事务1读。事务1前后读取不同。

读“脏数据”：事务1改->事务2读->事务1回滚。事务2读取的数据不一致。

9.2并发调度的可串行性：多个事务的并发执行是正确的，当且仅当并发执行结果与这些事务按某一串行顺序执行的结果相同，这种调度策略被称为可串行调度。

冲突操作：指不同的事务对同一个数据的读写操作和写写操作。

不同事务的冲突操作和同一事务的两个操作不能交换

冲突等价：如果调度S能通过一些列非冲突操作执行顺序的交换变成调度S1，则S和S1冲突等价。

可串行化调度的充分条件：与某串行调度冲突等价

9.3封锁技术：当事务存取某个数据项时，不允许其他事务修改这一数据项。

对某个数据对象操作之前，向系统发出请求，对其加锁；加锁后事务对该数据对象有了一定控制，其他事务无法更新次对象。

S锁（共享锁）：事务T对对象A加上S锁后，其他事务可读取对象、加S锁，不能修改对象、加X锁。

X锁（排它锁）：事务T对对象A加上X锁后，其他事务不可加任意类型的锁。



9.4封锁协议（何时申请封锁、持锁时间、何时释放封锁）

一级封锁协议：事务T在修改数据A之前必须先对其加X锁，直到事务结束才释放。无丢失更新，因为读数据时不加锁，不保证可重复读和不读脏数据。

二级封锁协议：在一级协议的基础上，事务T在读数据A前必须先对其加S锁，读完后即可释放S锁。防止读脏数据。

三级封锁协议：在二级协议的基础上，某一事物施加的S锁要保持到该事务结束才释放。

9.4封锁技术的问题：

活锁：某个事务请求封锁但始终没有获得锁而长时间处于等待状态。采用先来先服务的策略避免活锁。

死锁：同时处于等待状态的两个或多个事务相互封锁了对方请求的资源，而永远等待下去。如T1 Xlock R1->T2 Xlock R2->T1 Xlock R2->T2 Xlock R1->T1、T2相互等待对方解锁。

9.5解决死锁的方法

预防死锁的策略并不很适合数据库特点，普遍采用的是诊断并解除死锁的方法

预防死锁：一次封锁法（将事务要封锁的数据对象全部加锁，降低了系统的并发度）；顺序封锁法（预先对数据对象规定封锁顺序，所有事务按照顺序进行封锁，维护成本高，难以实现）；事务重试发（使用抢占机制和事务回滚）

死锁的检测：超时法（如果一个事务的等待时间超过了规定时限，就认为发生死锁，可能误判或无法及时发现）；事务等待图法（若事务V1等待V2，则在V1、V2间画一条有向边，若存在回路，则发生死锁）

死锁的恢复：选择一个处理死锁代价最小的事务，将其撤销；释放此事务持有的所有锁，使其他事务能继续运行下去。

9.6两阶段封锁协议

所有事务必须分两阶段对数据项加锁和解锁：在对任何数据进行读写操作前，必须先获得对该数据的封锁；释放一个封锁后，事务不再获得其他任何封锁。

使用两阶段封锁协议产生的是可串行化调度，使用两阶段封锁协议是可串行化调度的充分不必要条件。

拓展阶段（第一阶段）：事务可申请获得任何数据项上任何类型的锁，但不可释放锁

收缩阶段（第二阶段）：事务可以释放任何数据项上任何类型的锁，但不可申请锁

遵守两阶段封锁协议的事务可能发生死锁。

9.7锁表

锁管理器为目前已加锁的数据项维护一个记录链表，每个锁请求为一个记录，按请求的到达顺序排序这个表称为锁表。

封锁请求处理：事务T请求A上的锁；若没有A的锁表项，则表明A无锁，创建A的锁表项；若存在锁表项，则根据锁表项决定封锁请求

9.8多粒度封锁：在一个系统中同时支持多种封锁粒度供不同的事务选择

需要处理多个关系的大量元组的用户事务：以数据库为封锁单位

需要处理大量元祖的用户事务：以关系为封锁单元

只处理少量元组的用户事务：以元组为封锁单元

封锁粒度约小，被封锁对象越多，并发度越高，系统开销越大

多粒度封锁协议：多粒度树中每个节点被独立地加锁（显式封锁），同时这个节点的所有后裔节点也被加以同样类型的锁（隐式封锁）。

9.9意向锁（目的是提高对某个数据对象加锁时系统的检查效率）

对任一节点加基本锁，必须先对它的上层结点加意向锁。

IS锁（意向共享锁）：对一个数据对象加IS锁表示对它后裔节点拟加S锁。

IX锁（意向排它锁）：对一个数据对象加IX锁表示对它后裔节点拟加X锁。

SIX锁（共享意向排他锁）：对一个数据对象加SIX锁表示对它加S锁，再加IX锁。

申请封锁时应该自上而下的次序进行

释放封锁时应该自下而上的次序进行



10关系数据库设计理论

10.1关系模型的存储异常：数据冗余、插入异常、删除异常、更新异常。

原因：数据间存在一定的依赖关系。

数据依赖，包括函数依赖、多值依赖、连接依赖。

10.2函数依赖（FD），表示了关系中属性间的一种制约关系。

FD X->Y ： X决定Y。

平凡函数依赖Y⊆X

完全函数依赖**对任意的**X**’** ⊂X**，** X**’**→Y**都不成立**

传递函数依赖：若FD X→Y**，** Y不决定X**，** Y→Z**，则必有**FD X→Z。

逻辑蕴涵：函数依赖集F和关系模式R(U)，X、Y⊆U，若R满足FD X→Y，则F逻辑蕴涵FD X→Y

闭包：F+所有被F逻辑蕴涵的函数

10.3函数依赖公理

Armstrong公理

自反律：若Y⊆X⊆U, 则 F |= X→Y；

增广律：若X→Y且Z⊆U，则 F |= XZ→YZ；

传递律：若X→Y, Y→Z，则 F |= X→Z.

合成规则：若X→Y, X→Z，则XZ→W

分解规则：若X→Y, Z⊆Y，则X→Z

伪传递规则：若X→Y, YZ→W，则XZ→W

10.4属性闭包

XF+={Ai | 所有用公理由F推出的X→Ai}

能够由Armstrong公理从F导出X→Y成立的充要条件是Y⊆ XF+

10.5最小函数依赖集

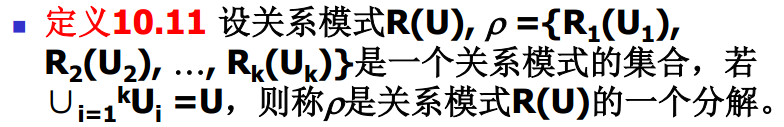
满足以下条件成为最小函数依赖集：

F中所有函数依赖其右部都是单属性；

F中任一函数依赖X→A，F-{X→A}与F不等价；

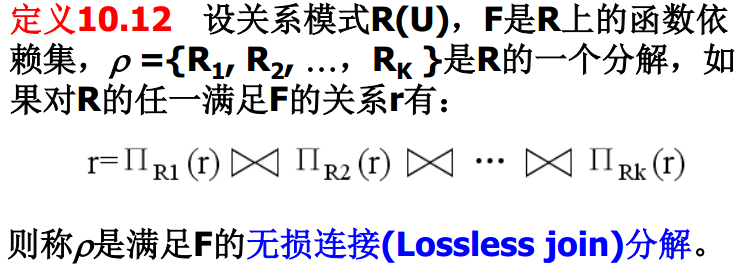
F中任一函数依赖X→A，F-{X→A}**∪**{Z→A }与F不等价，Z为X真子集

10.6模式分解：将大的关系模式用小的替代。



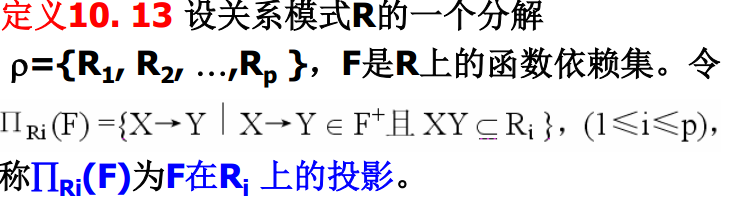
分解后的模式应当与原模式等价，等价包括：不损失任何信息，保持原来的函数依赖。

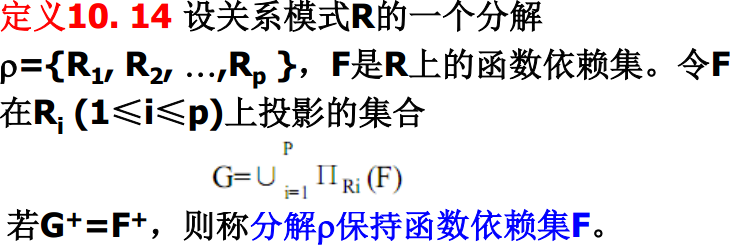
10.7无损连接分解



判断分解为无损链接分解的方法 算法10.2

10.8保持函数依赖





检验分解是否保持函数依赖 算法10.3

10.8关系模式的规范化

通过分解关系模式，解决插入异常、删除异常、更新异常、数据冗余等问题。

第一范式（1NF）：如果关系模式R的每个属性对应的域值都是不可再分的，则称模式R属于第一范式。

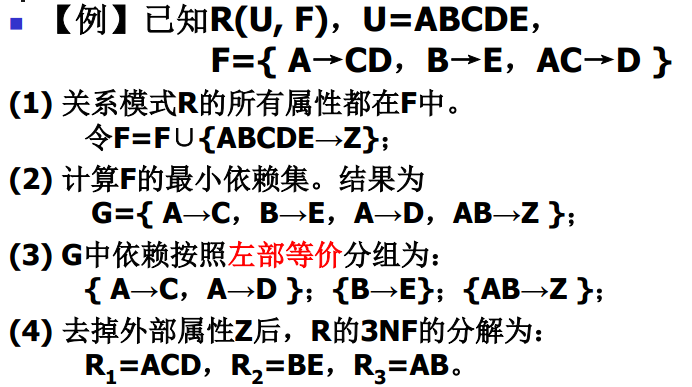
第二范式（2NF）：如果关系模式R∈1NF，且所有非主属性完全依赖与R的每一个候选键，则R∈2NF。

第三范式（3NF）：如果关系模式R∈1NF，且在R中没有非主属性传递依赖于R的候选键，则关系模式R∈3NF。若R∈3NF，则R∈2NF

BCNF：如果关系模式R∈1NF，且在R中没有任何属性传递依赖于R中的任一关键字，则模式R∈BCNF；如果关系模式R∈1NF，F是R上的函数依赖集，对于F中每个函数依赖X→Y，必有X是R的一个候选键，则R∈BCNF。若R∈BCNF则R∈3NF。

10.9模式分解算法

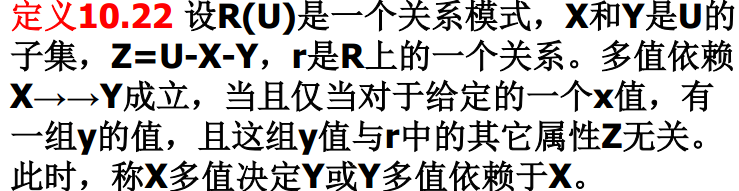
生成3NF，算法10.4，具有无损连接性和保持依赖性，例如下

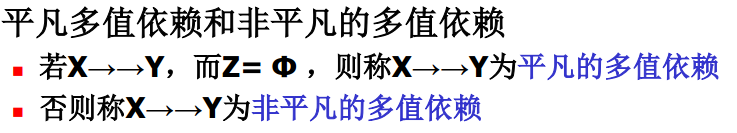


生成BCNF，算法10.5，具有无损连接性，不一定具有保持依赖性，例如下



10.10 多值依赖

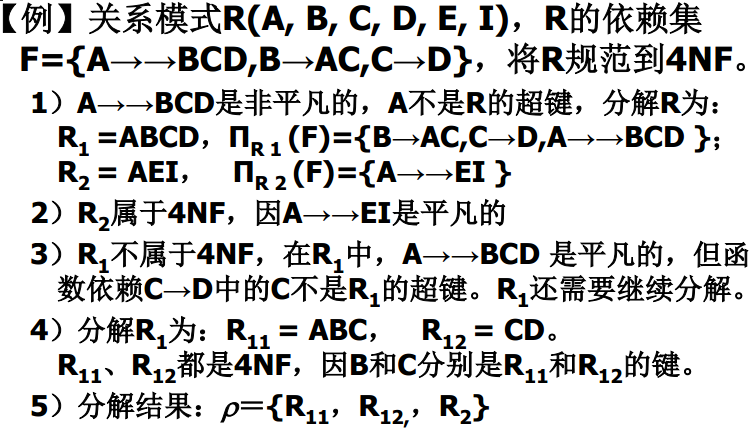




10.11第四范式

设关系模式R∈1NF，对于R上任何一个非平凡的多值依赖X→→Y，X是R的一个超键，则R∈4NF。4NF的属性之间不存在非平凡的多值依赖。

分解生成4NF的算法10.6，例如下：



11.数据库设计

11.1基本步骤：

需求分析：准确了解与分析需求，最困难耗时

概念结构设计：对需求综合、归纳、抽象，形成概念模型

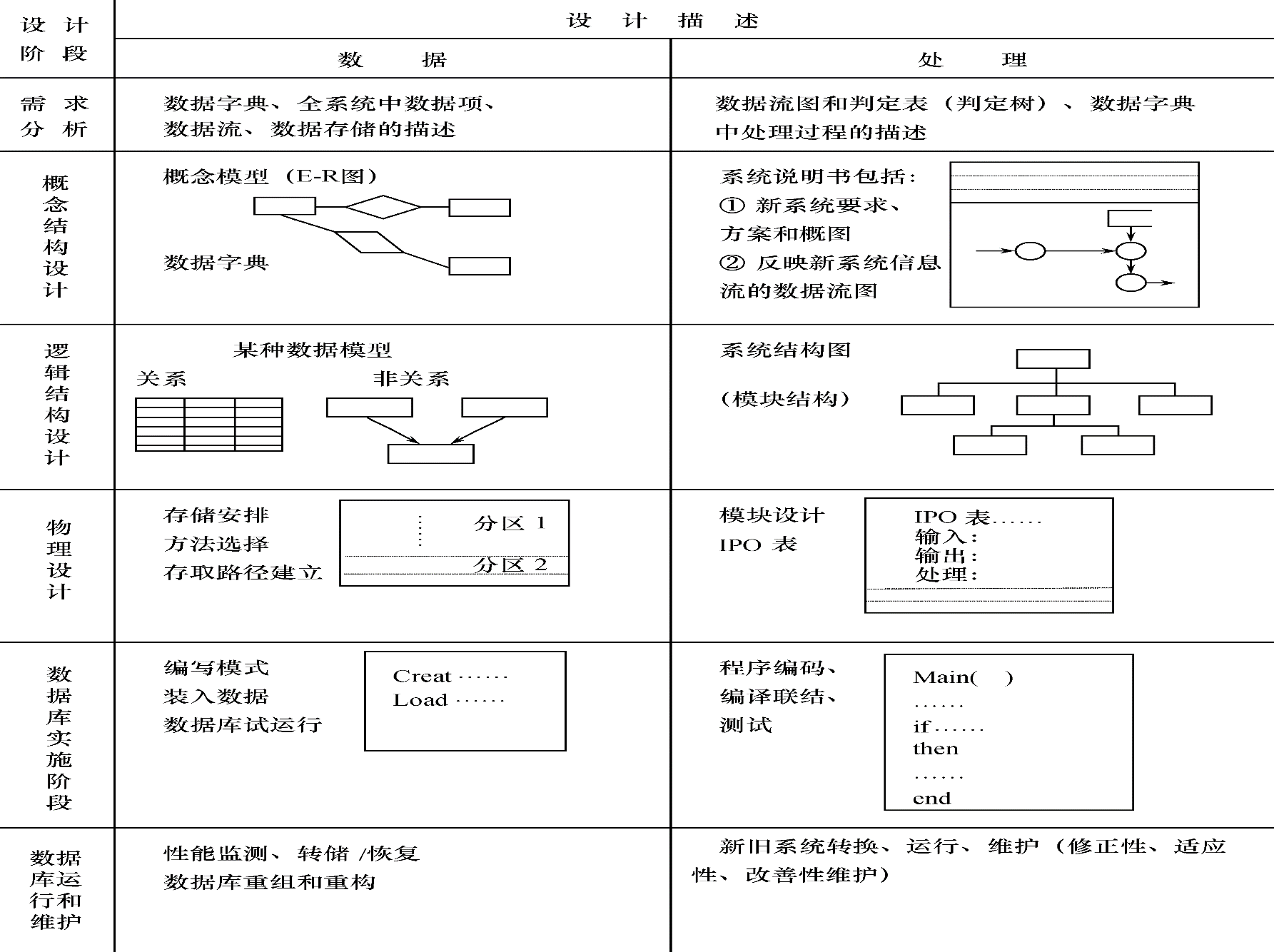
逻辑结构设计：转为DBMS支持的数据模型并优化

物理结构设计：选取适合应用环境的物理结构，包括存储结构和存取方法

数据库实施：运用DBMS语言和宿主语言设计

数据库运行和维护

完整的设计过程是以上六阶段的重复与补充



11.2需求分析

信息要求、处理要求、安全性与完整性要求

11.3数据字典

进行详细数据收集和数据分析所获得的的主要结果，在分析阶段建立，在设计过程中不断完善

数据项：不可再分的数据单位

数据结构：反映数据间组合关系，可有数据项和数据结构组成

数据流：数据结构在系统内传输的路径

数据存储：数据结构保存的地方，数据流的来源和去处之一

处理过程：一般用判定表或判定树描述

11.4概念结构设计

将用户需求抽象为信息结构

自底向上设计概念结构的步骤：抽象数据设计局部视图（分E-R图）；集成局部视图，得到全局概念结构（总E-R图）

数据抽象：分类（定义一组对象的类型），聚集（定义某一类型的组成部分、属性），概括（定义了子集联系）

设计局部E-R图：标定实体、实体的属性、标定实体的码，确定实体间的联系，属性必须是不可分的数据项，且不可与其他实体有联系

E-R图的集成：主要工作为消除各分E-R图的冲突，E-R图冲突分为属性冲突、命名冲突、结构冲突。

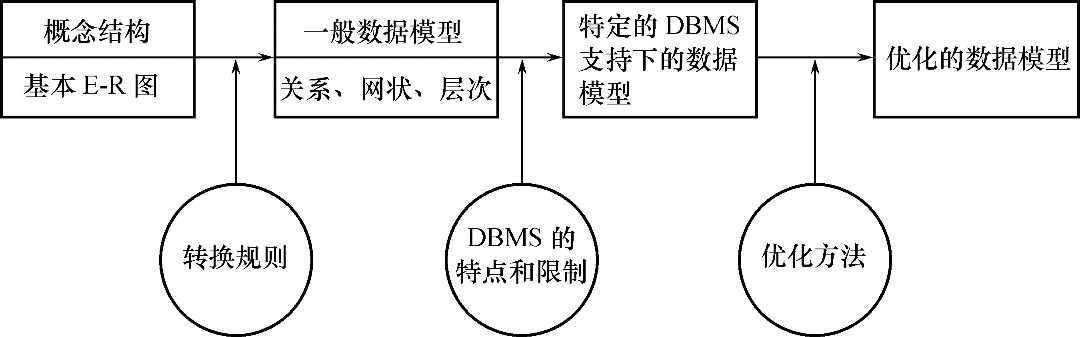
属性冲突：属性域冲突（不同分E-R图中同一类属性的域不同），取值单位冲突（比如元和万元）

命名冲突：同名异义、异名同义

结构冲突：同一对象在不同应用中有不同抽象、同一实体集在不同E-R图属性不同、实体集之间的联系在不同局部视图中呈现不同类型。

11.5逻辑结构设计

将概念结构转化为一般的关系模型，像DBMS支持下的数据模型转换，然后优化，最后设计用户子模式



E-R模型向关系模型转换：

1:1联系，可以转换成一个独立的关系模式，也可以与任意一端对应的关系模式合并

1:n联系，可以转换成一个独立的关系模式，也可以与n端对应的关系模式合并

m:n联系，转换成一个独立的关系模式

三个或三个以上实体间的多元联系，可以转换成一个关系模式

具有相同码的关系模式可以合并

优化方法：

确定数据依赖，按需求得到的语义写出每个关系模式的依赖

消除冗余联系

确定所属范式，一般而言第三范式就足够了

分析应用环境是否适合模式

对关系模型进行必要的分解，以提高数据操作的效率和存储空间利用率

11.6物理结构设计

步骤：确定物理结构，在关系数据库中主要指存取方法（建立存取路径）和存储结构，确定系统配置；对物理结构进行评价，主要是时间和空间效率

常用存取方法：索引方法、聚簇方法（将某属性具有相同值的元组存放在连续物理块上）、hash方法

建立索引的一般规则：某列经常在查询条件中出现、某属性经常作为聚集函数的参数、某属性经常在连接操作的连接条件中出现

聚簇适用在聚簇码访问或连接是该关系主要应用时

确定存储结构：存放位置和存储结构，决定因素为存取时间、存储空间利用率、维护代价

存放位置基本原则：易变部分和稳定部分分开存放；存取频率高部分和存取频率低的部分分开存放。

11.7数据库试运行

主要工作为功能测试和性能测试

功能测试若不满足需要对应用程序部分作出修改调整

性能测试若不满足需要调整物理结构，修改系统参数，甚至返回逻辑设计阶段