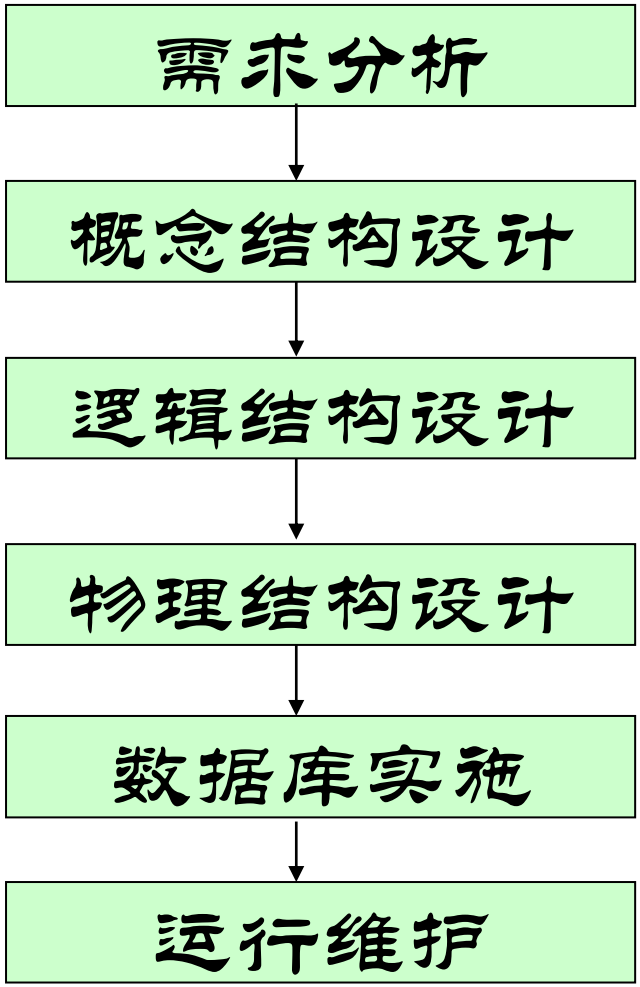
数据库原理

The Theory of Database System

第五章 数据库设计

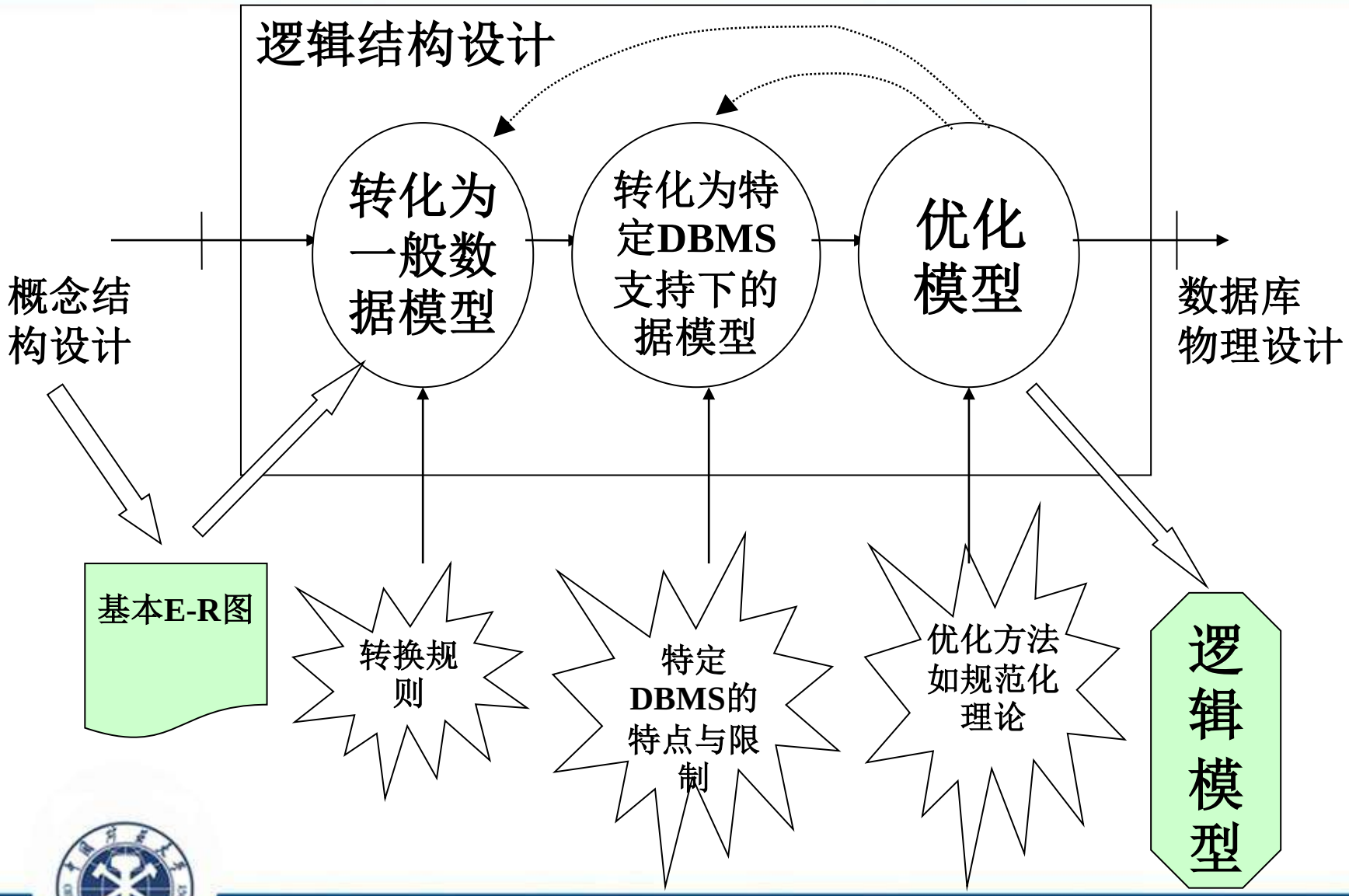
中国矿业大学计算机学院

数据库设计的基本步骤



5.4 逻辑结构设计

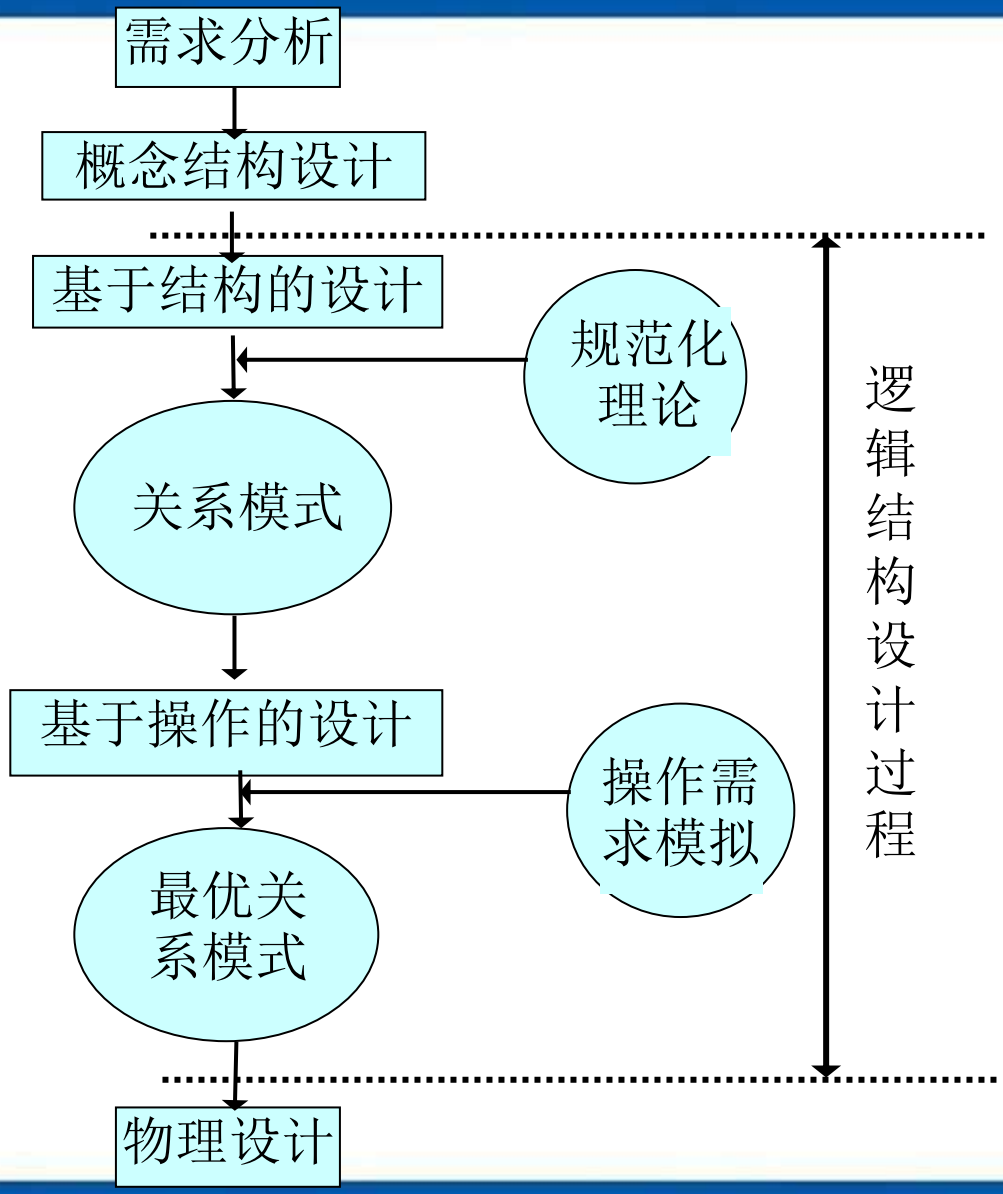
将全局概念结构转化为某个具体的DBMS所支持的数据模型，并根据逻辑结构设计准则、数据的语义约束、规范化理论等对数据模型的结构进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式，这就是逻辑结构设计所要完成的任务。


逻辑结构设计
转化为一般数据模型
转化为特定DBMS支持下的据模型
优化模型
概念结构设计
数据库物理设计
基本E-R图
转换规则
特定DBMS的特点与限制
优化方法如规范化理论
逻辑模型

逻辑结构设计的方法

两类设计方法：

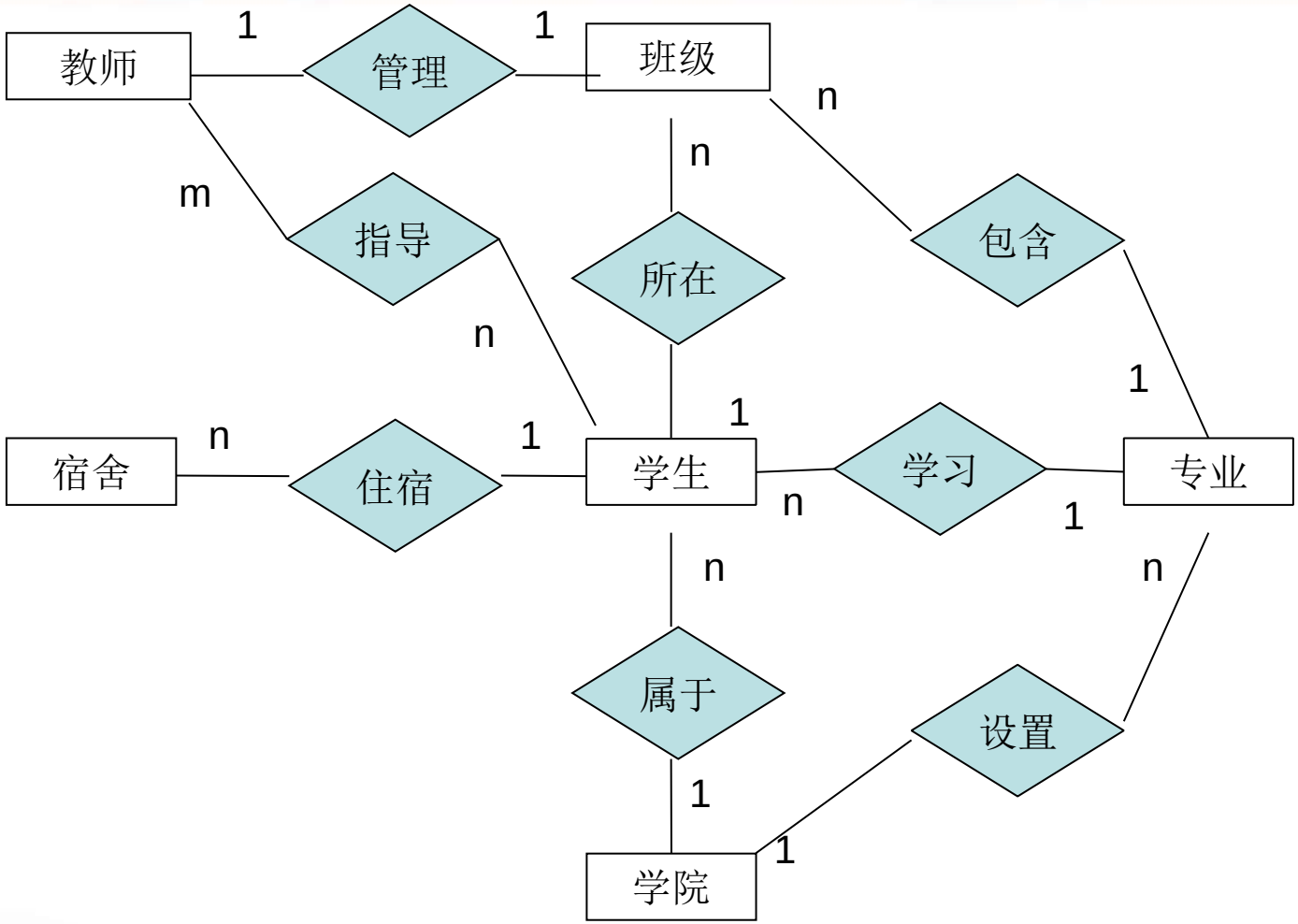
- 基于结构的设计方法
- 基于操作的设计方法



5.4.1 逻辑结构设计的任务和步骤

- 将概念模型转化为等价的关系模式
- 按需要对关系模式进行规范化
- 对规范化后的模式进行评价
- 根据局部应用的需要，设计用户外模式

学籍管理子系统总E-R图



5.4.2 E-R图向关系模型转换的原则

- **实体：一个实体转换为一个关系模式。**

实体的属性就是关系的属性，实体的码就是关系的码。

学生

↓

学生（学号，姓名，出生日期，学院编号，班级编号，宿舍编号）

班级　　学院　　专业　　宿舍　　教师

5.4.2 E-R图向关系模型转换的原则

- 类型为1:1 联系的转换规则
- 类型为1:n联系的转换规则
- 类型为n:m联系的转换规则

5.4.2 E-R图向关系模型转换的原则

➢ 联系类型为1:1

- 可以将联系转换为一个新的关系。

与该联系相连的各实体的码以及联系本身的属性构成新关系的属性，每个实体的码均是该关系的候选码。

管理（教师号，班级号）

管理（教师号，班级号）

管理（教师号，班级号）

5.4.2 E-R图向关系模型转换的原则

➢ 联系类型为1:1

•可以与任意一端对应的关系模式合并。在该关系模式中加入另一关系的码和联系的属性，该关系的码不变。

教师：｛职工号，教师姓名，性别，职称 ，班级号｝

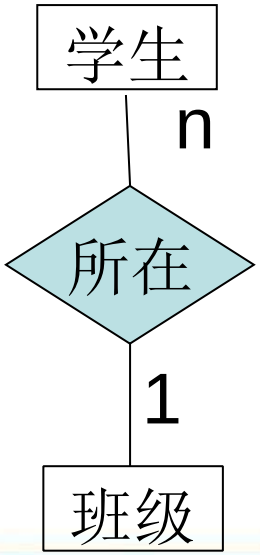
班级：｛班级编号，班级信息备注 ，职工号｝

5.4.2 E-R图向关系模型转换的原则

➢ 联系类型为1:n

- 可以将联系转换为一个新的关系。

将联系转换为一个新的关系模式：与该联系相连的各实体的码以及联系本身的属性构成新关系的属性，该关系的码是n端关系模式的码。

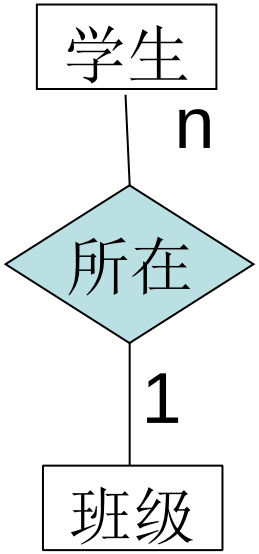


所在（学号，班级号）

5.4.2 E-R图向关系模型转换的原则

➢ 联系类型为1:n

•可以与n端对应的关系模式合并。在n端关系模式中加入1端关系模式的码和联系的属性，关系的码仍为n端关系的码。



学生（学号，姓名，性别，
出生日期，班级号）

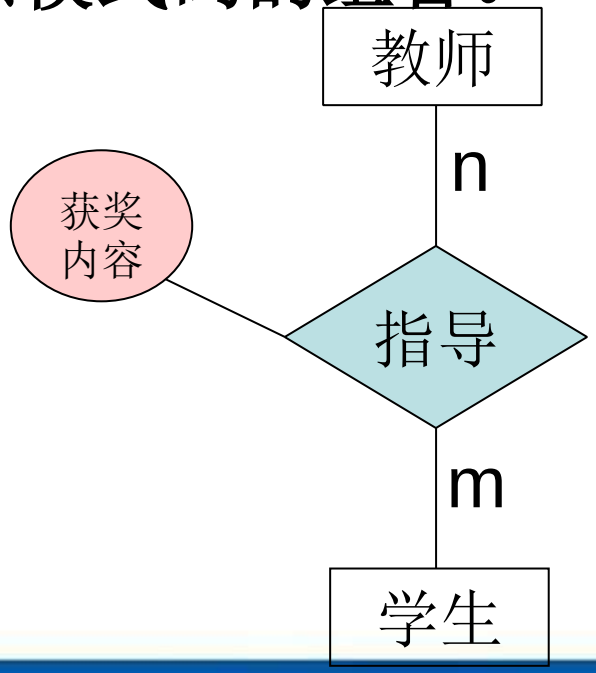
5.4.2 E-R图向关系模型转换的原则

➢ 联系类型为n:m

- 只能将联系转换为一个新的关系。

与该联系相连的各实体的码以及联系本身的属性构成新关系的属性，该关系的码是两端关系模式码的组合。

指导（学号，教师号，获奖内容）



学籍管理子系统的关系模式

- 学生（学号，姓名，性别，出生日期，学院编号，班级编号，宿舍编号）
- 班级（班级编号，班级信息备注）
- 宿舍（宿舍编号，地址，电话）
- 学院（学院编号，学院名称，院长，办公电话）
- 专业（专业编号，专业名称）
- 教师（职工号，教师姓名，性别，职称）

1:1 和1：n关系采用生成一个新关系的策略

- 管理（教师编号，班级编号）
- 所在（学号，班级编号）
- 住宿（学号，宿舍编号）
- 属于（学号，学院编号）
- 学习（学号，专业编号）
- 包含（班级编号，专业编号）
- 设置（专业编号，学院编号）
- 指导（教师编号，学号，获奖内容）

1:1 和1：n关系采用生成一个新关系的策略

- 学生（学号，姓名，性别，出生日期，学院编号，班级编号，宿舍编号}

- 住宿（学号，宿舍编号）
- 包含（班级编号，专业编号）
- 所在（学号，班级编号）

消除冗余

- 学生（学号，姓名，性别，出生日期，学院编号，班级编号，宿舍编号）
- 班级（班级编号，班级信息备注）
- 宿舍（宿舍编号，地址，电话）
- 学院（学院编号，学院名称，院长，办公电话）
- 专业（专业编号，专业名称）
- 教师（教师编号，教师姓名，性别，职称）
- 学习（学号，专业编号）
- 管理（教师编号，班级编号）
- 包含（班级编号，专业编号）
- 设置（专业编号，学院编号）
- 指导（教师编号，学号，获奖内容）

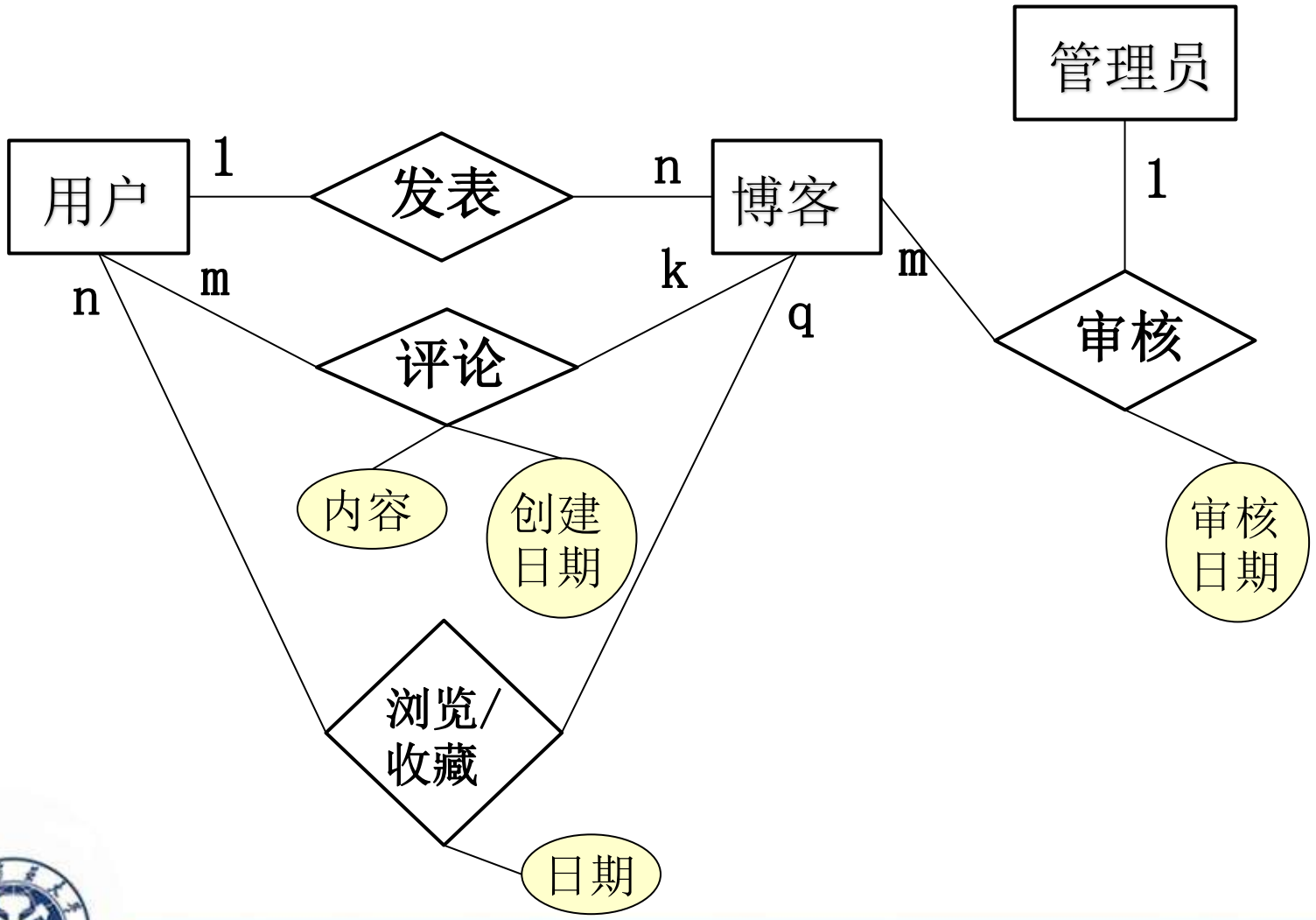
1:1 和1：n关系采用合并的策略

- 学生（学号，姓名，性别，出生日期，*学院编号，班级编号，宿舍编号，专业编号*）
- 班级（班级编号，班级信息备注，*教师编号，专业编号*）
- 宿舍（宿舍编号，地址，电话）
- 学院（学院编号，学院名称，院长，办公电话）
- 专业（专业编号，专业名称，*学院编号*）
- 教师（教师编号，教师姓名，性别，职称）
- 指导（*教师编号，学号*，获奖内容）

个人博客系统

- 用户发表博客
- 管理员审核博客
- 用户浏览、收藏喜欢的博客
- 用户可以评论博客或者

个人博客系统



个人博客系统（逻辑结构设计）

用户(用户ID, 用户名, 密码, 邮箱, 注册日期, 头像, 角色)

博客(博客ID, 标题, 内容, 创建日期, 修改日期
, 用户ID, 审核, 审核日期, 审核人)

评论(评论ID, 内容, 创建日期, 用户ID, 博客ID , 父评论ID ）

浏览收藏(用户ID, 博客ID, 日期 , 是否收藏 ）

标签(标签ID, 标签名称)

博客标签(博客ID, 标签ID)

5.4.3 逻辑结构的优化

- 确定数据依赖
- 对于各关系模式间的数据依赖进行极小化处理，消除冗余的联系。
- 规范化
- 分解或合并关系模式

常用分解方法

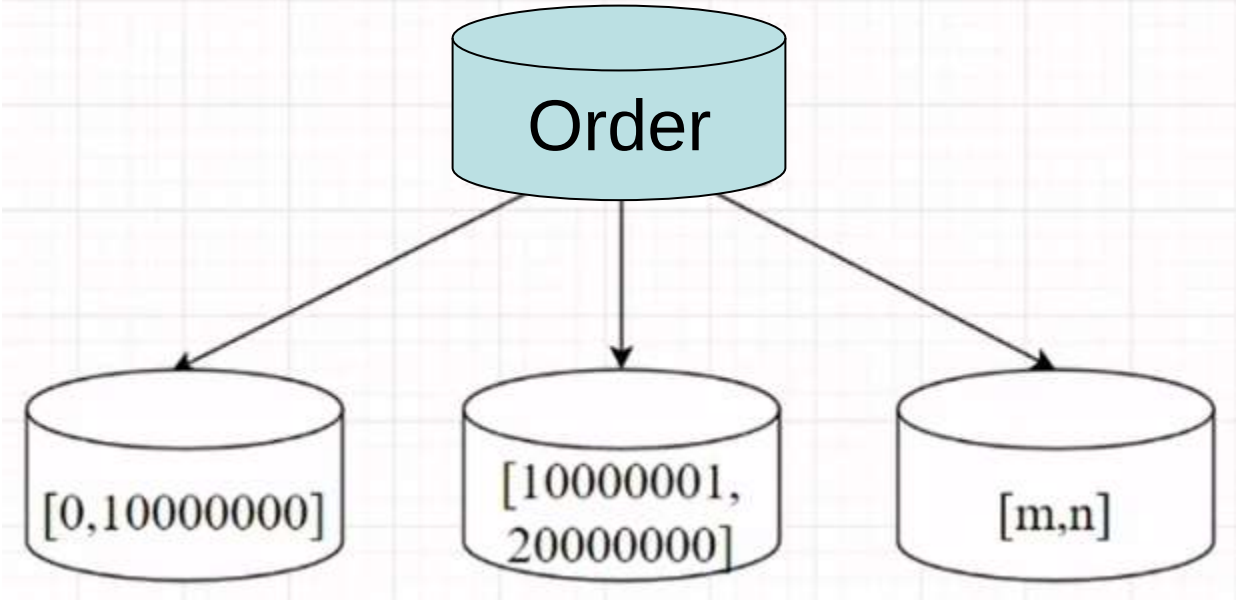
- 水平分解
- 垂直分解

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。
- 水平分解的方法
  - 平均分解
  - 根据业务逻辑分解
  - Hash分解

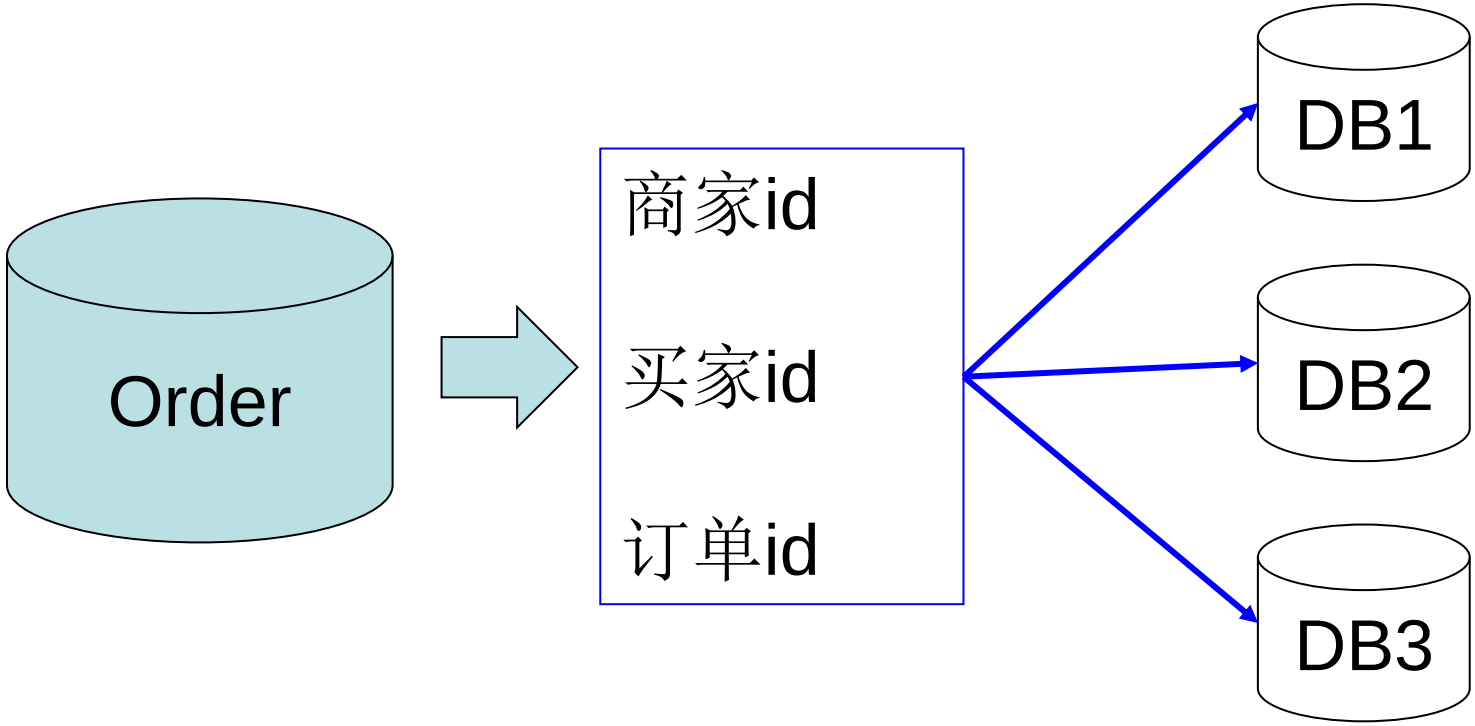
分库分表

（1）平均分解



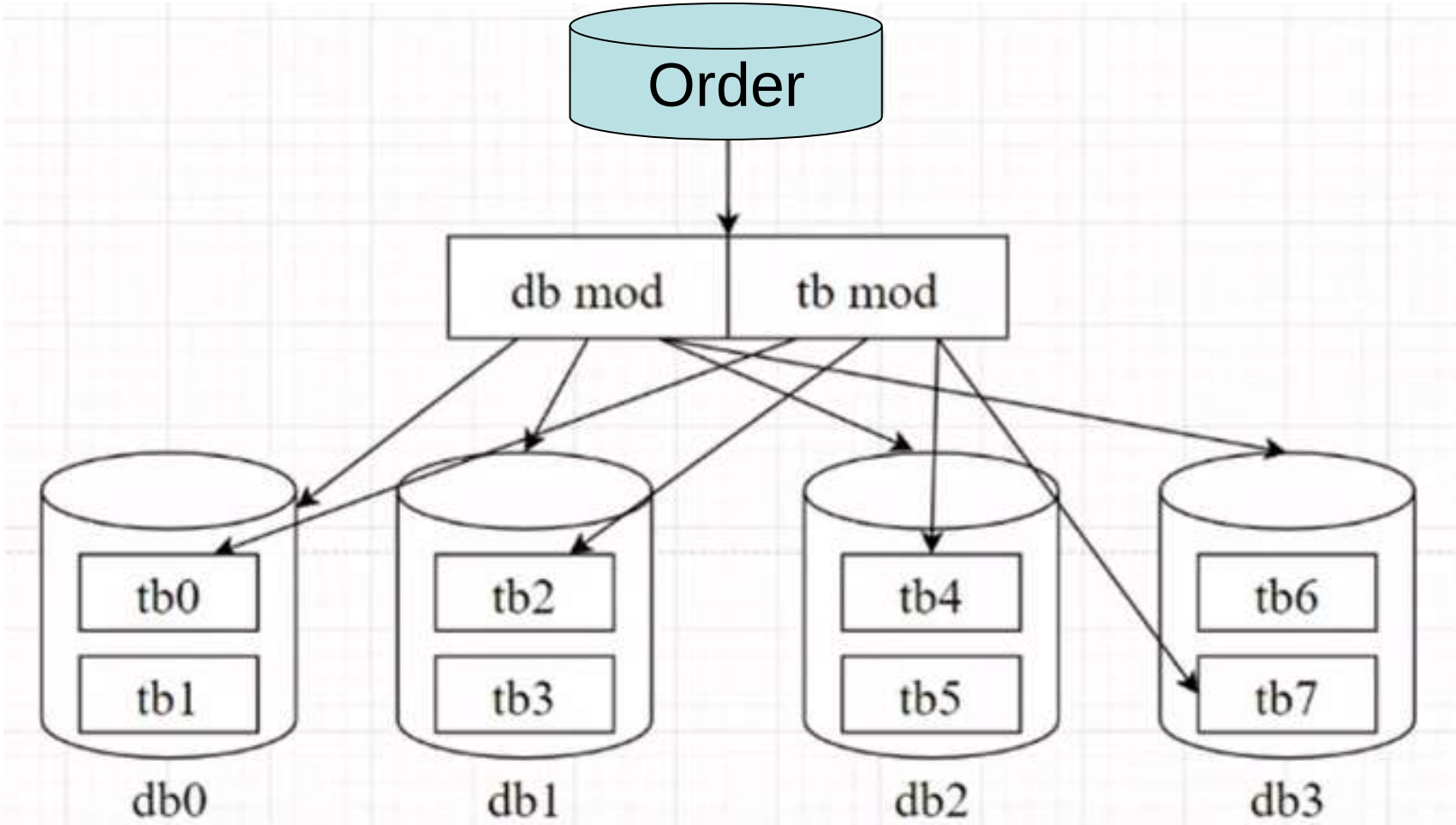
分库分表

(2) 根据业务逻辑分解



分库分表

(3) Hash分解



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。
- 垂直分解的原则
  - 经常在一起使用的属性从*R*中分解出来形成一个子关系模式。

垂直分解

- 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。
- 进行垂直分解的方法
  - 简单情况：直观分解
  - 复杂情况：用第4章中的模式分解算法
  - 垂直分解必须不损失关系模式的语义(保持无损连接性和保持函数依赖)。

5.4.4 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：

设计用户子模式（续）

(1) 使用更符合用户习惯的别名

- 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

设计用户子模式（续）

(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。

例如教务系统中：

学生用户只能查看到学号，姓名，班级，平时成绩，期末成绩，总成绩。

教师用户可以查看学号，姓名，班级，能够更新平时成绩，期中成绩，实验成绩，期末成绩；并且成绩提交后更新权限消失。

管理员则可以查询并更新全部数据。

设计用户子模式（续）

(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。

5.8 数据库设计实例

校园公共自行车管理系统

校园公共自行车管理系统

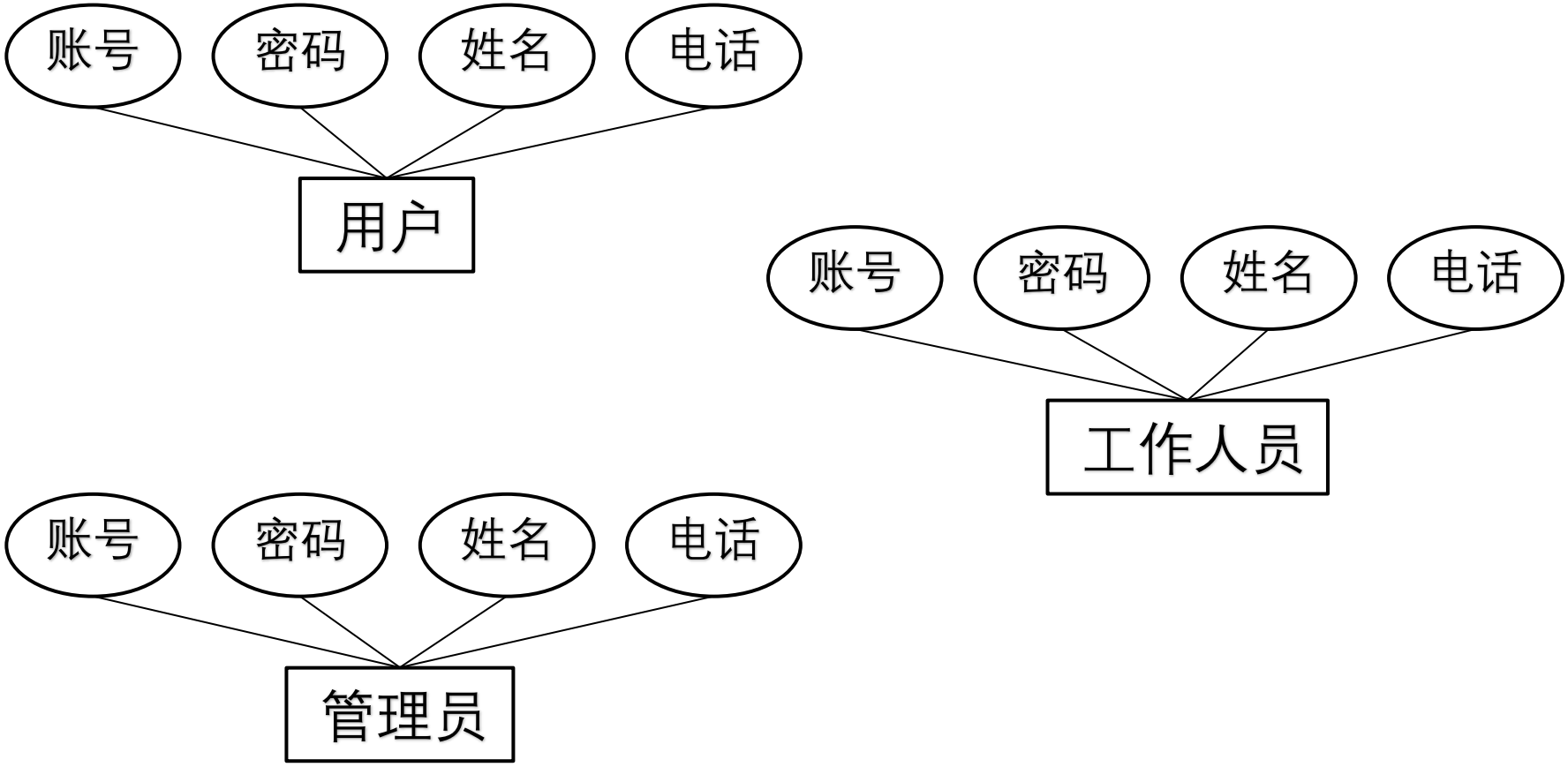
1 需求分析

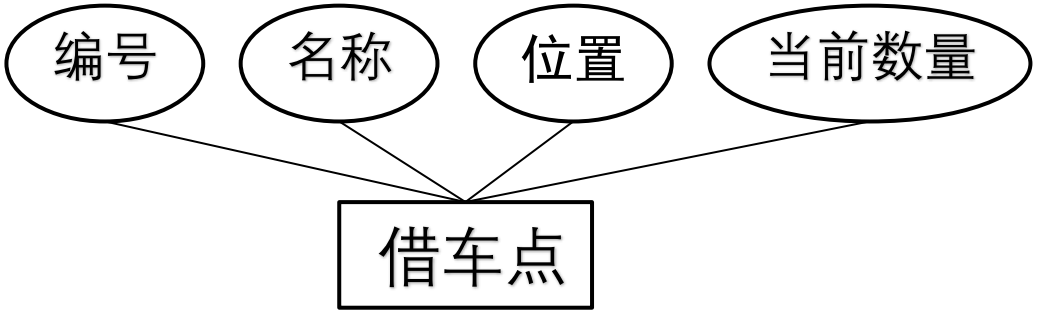
用户能够注册登录，能够根据借车点的名称查询借车点的位置，车辆剩余情况，能够借车。

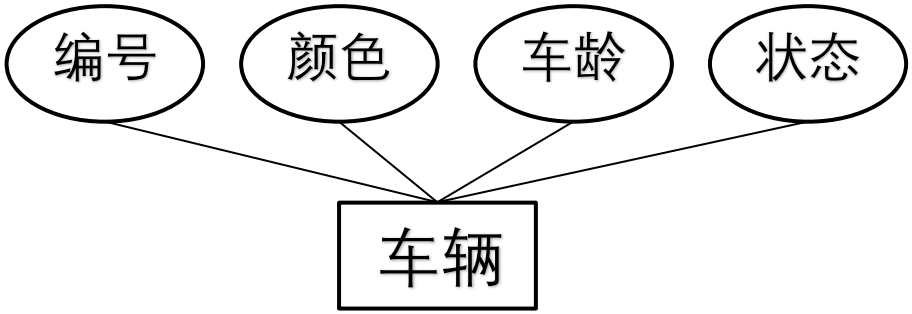
工作人员能够注册登录，能够对负责的借车点录入车辆情况，如编号，颜色，车龄，当前状态等；能够更新车辆归还情况。

对于多次借车未还的用户，管理员可以冻结其账号。

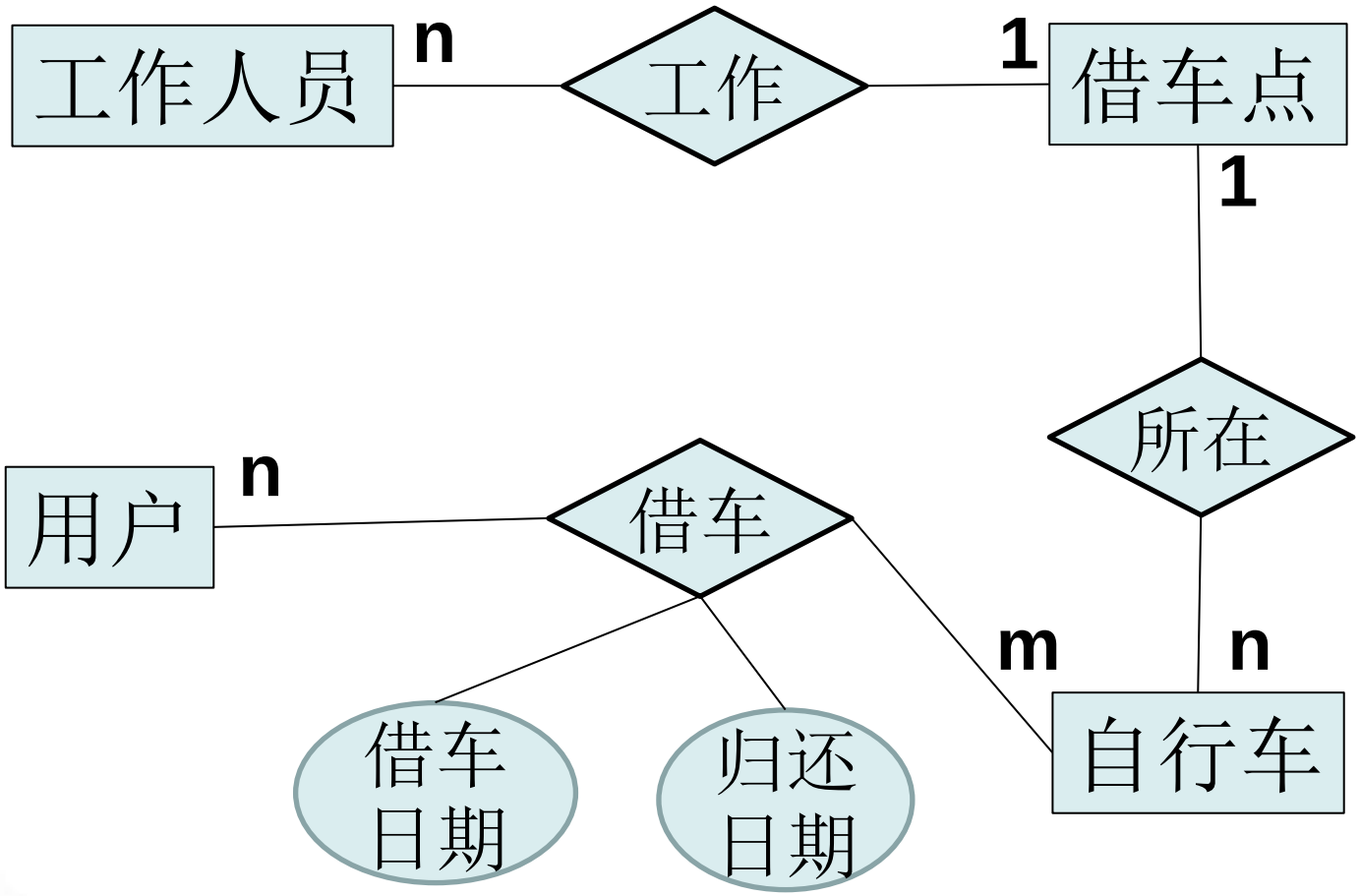
2 概念结构设计




编号
名称
位置
当前数量
借车点


编号
颜色
车龄
状态
车辆

系统总E-R图



3 逻辑结构设计

系统的关系模式：

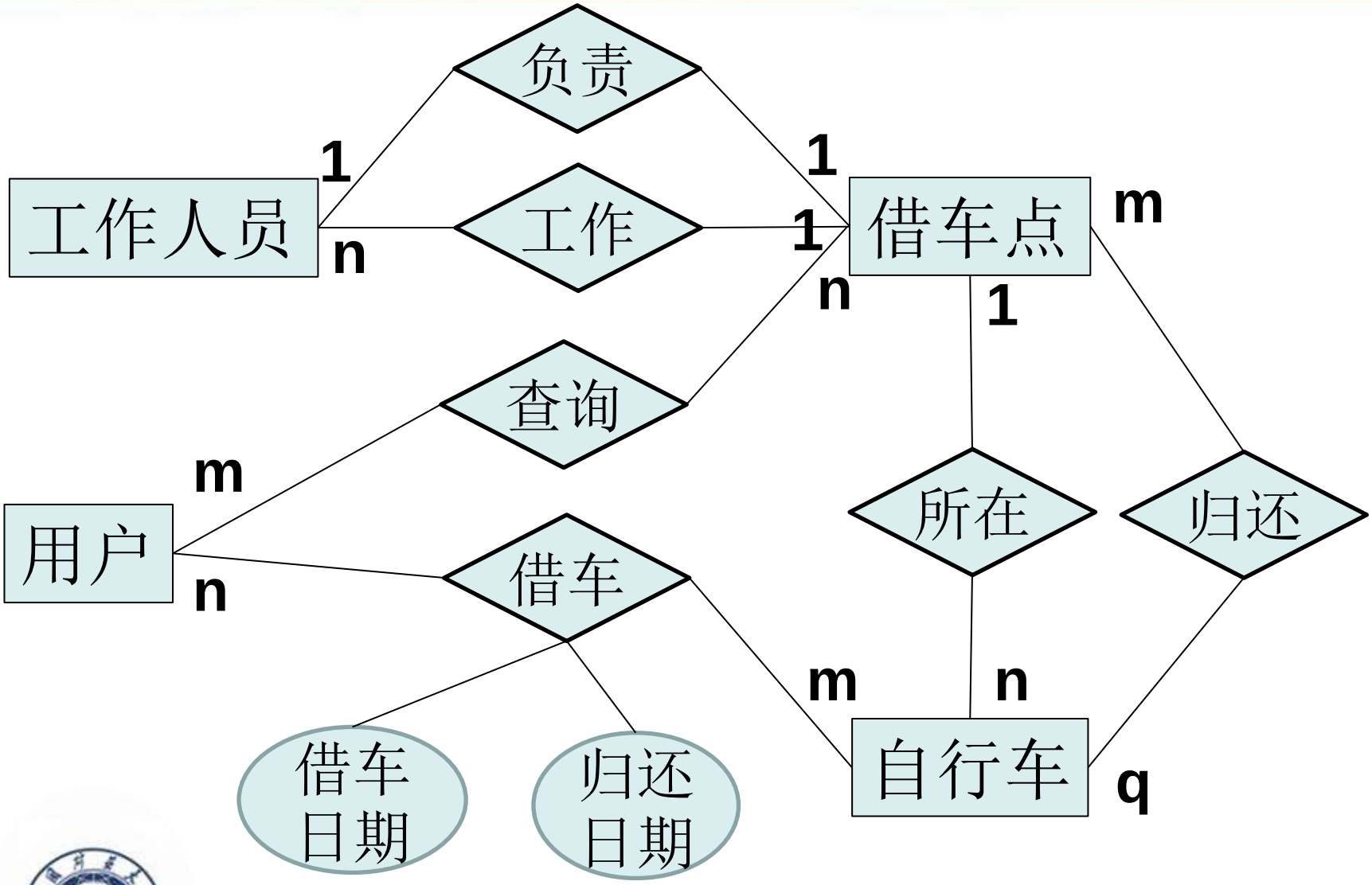
用户(账号，密码，姓名，电话，角色，状态，*借车点*)

借车点(编号，名称，位置，当前数量)

自行车(编号，颜色，车龄，状态，当前位置)

借车(自动编号，*账号*，*车号*，借车日期，归还日期)

系统总E-R图



3 逻辑结构设计

系统的关系模式：

用户(账号，密码，姓名，电话，角色，状态，*借车点*)

借车点(编号，名称，位置，当前数量，*负责人*)

自行车(编号，颜色，车龄，状态，*当前位置*)

借车(自动编号，*账号*，*车号*，借车日期，归还日期)

查询(自动编号，*账号*，*车号*，查询时间)

归还(自动编号，*车号*，*借车点号*，归还日期)

子模式（视图）

未归还车辆的用户名单

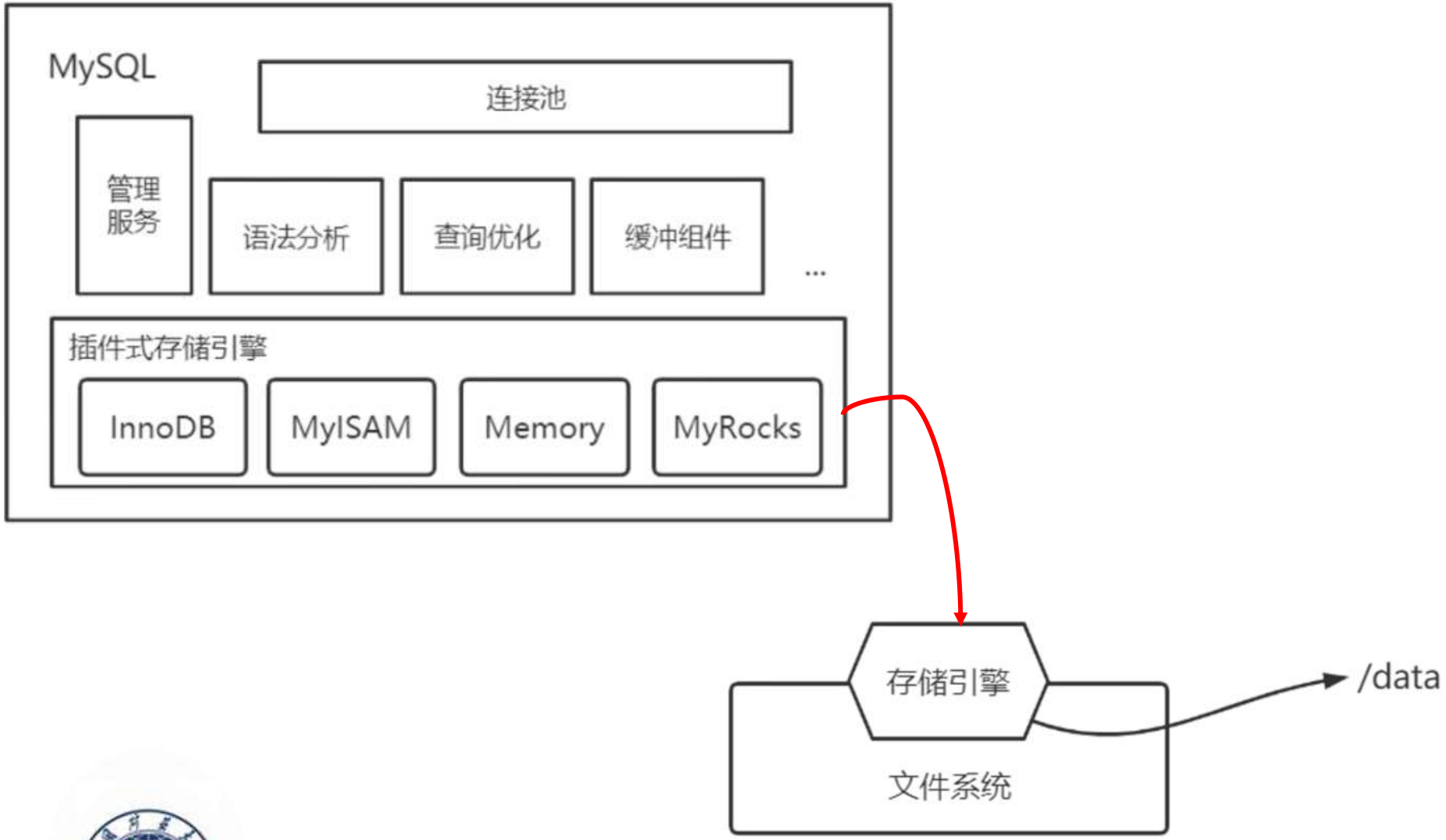
```
create view No_return(用户姓名, 电话, 借车日期)
as
select 姓名, 电话, 借车日期
from 用户, 借车
where  用户.账号=借车.账号
    and datediff(curdate(),借车日期)>1
    and 还车日期 is null;
```

子模式（视图）

不诚信名单

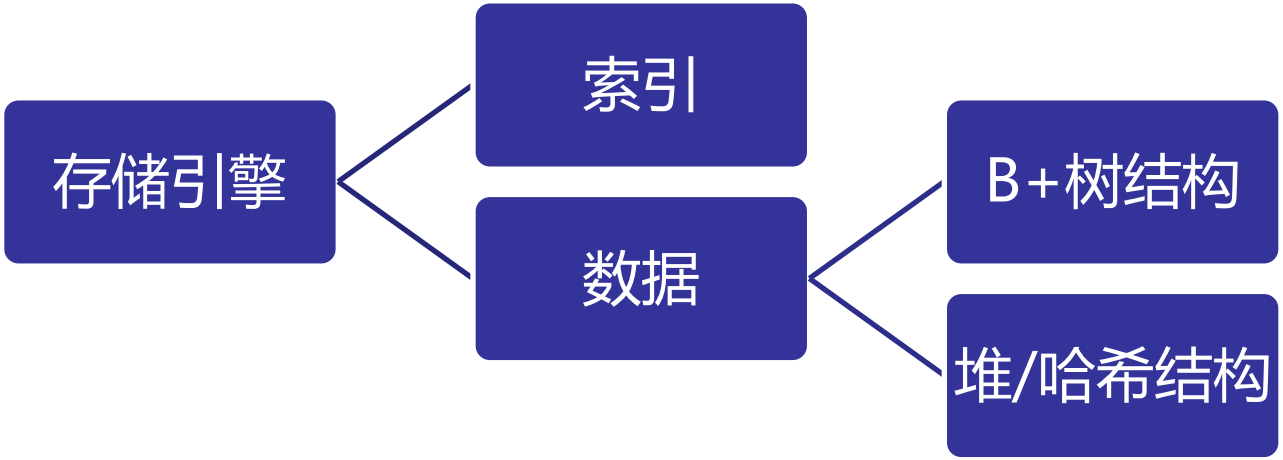
```
create view blacklist(账号，姓名)
as
select 账号，姓名
from 用户, 借车
where  用户.账号=借车.账号
    and datediff(curdate(),借车日期)>1
    and 还车日期 is null
group by 账号
having count(*)>=5;
```

数据库的物理设计

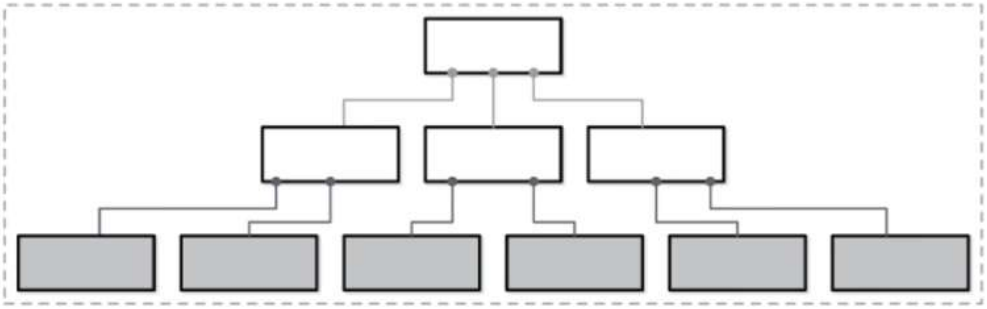

MySQL
连接池
管理服务
语法分析
查询优化
缓冲组件
...
插件式存储引擎
InnoDB
MyISAM
Memory
MyRocks
存储引擎
/data
文件系统

数据库的物理设计

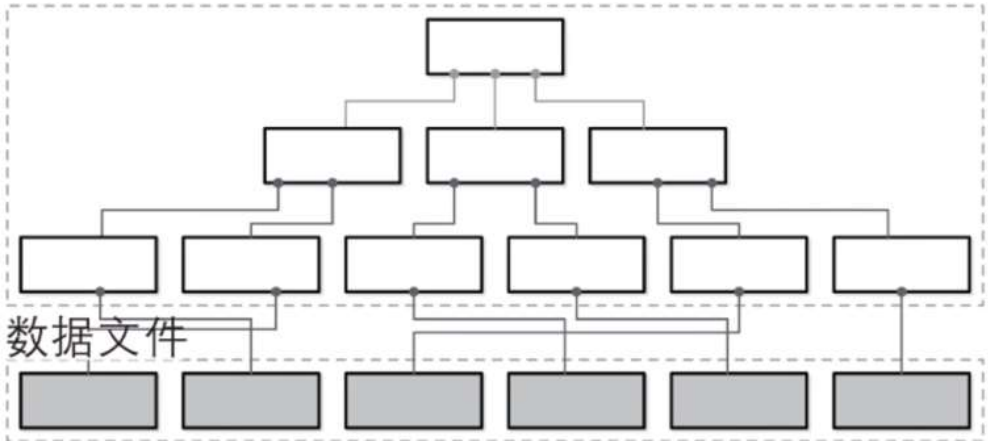
存储引擎存储什么？



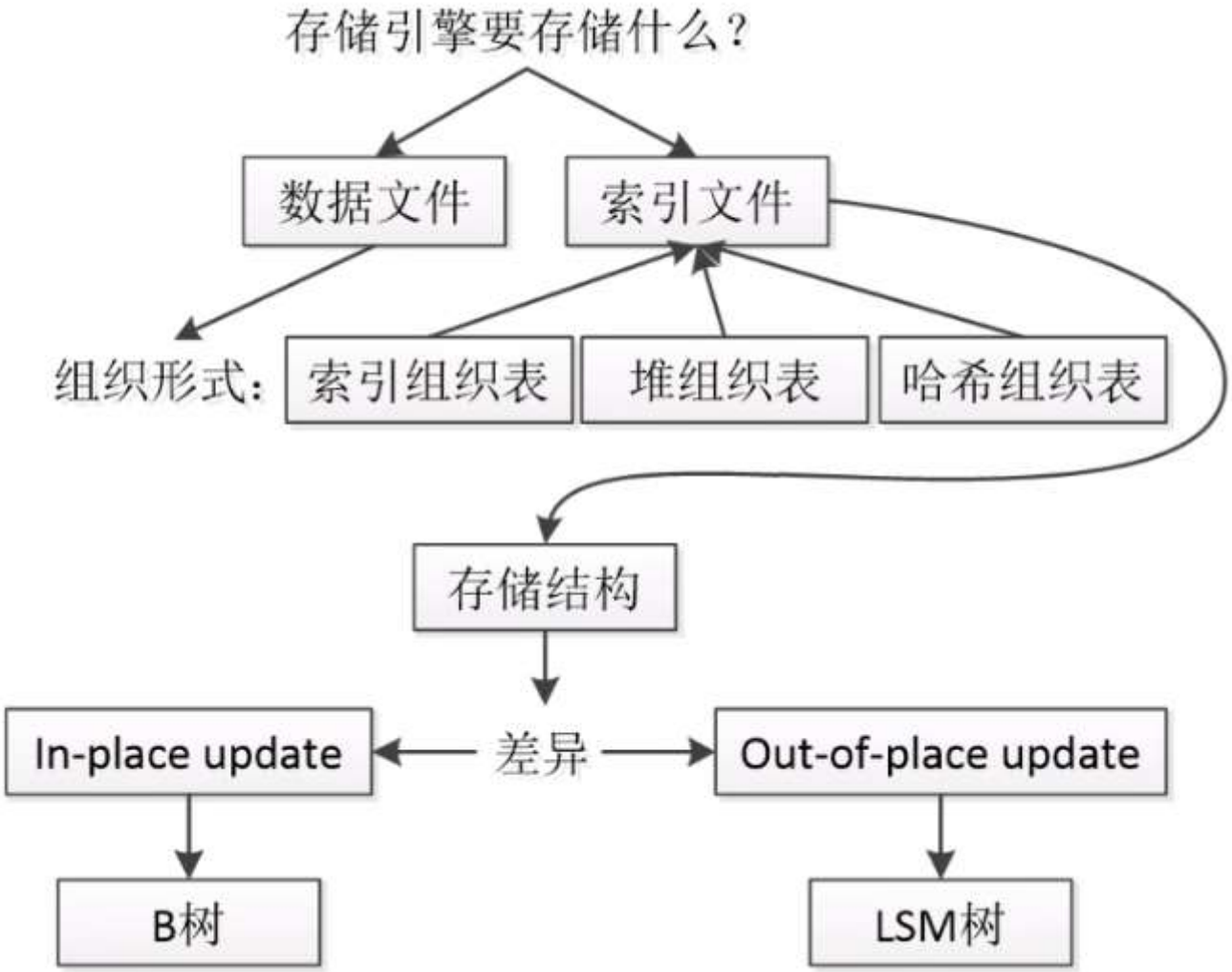
B+树结构



堆结构



数据库的物理设计


存储引擎要存储什么？
数据文件
索引文件
组织形式：
索引组织表
堆组织表
哈希组织表
存储结构
In-place update
差异
Out-of-place update
B树
LSM树

存储结构

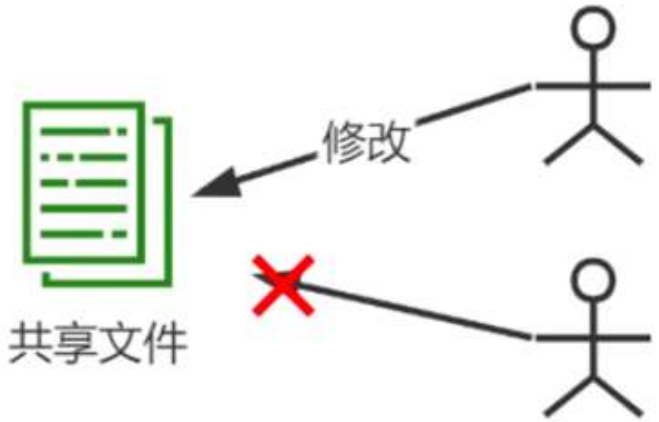
存储结构具备的特点：

（1）适合磁盘存储

IO尽量少，且一次读取连续的空间

（2）允许并发操作

增删改对存储结构的影响尽量小



存储结构

不适合做存储结构

- 数组
- 链表
- 哈希表
- 跳跃表
- 二叉搜索树
- 平衡二叉树
- 红黑树

适合做存储结构

- B树 → **In-place update**
- LSM树 → **Out-of-place update**

数据库的物理设计

- **In-place update structure**：

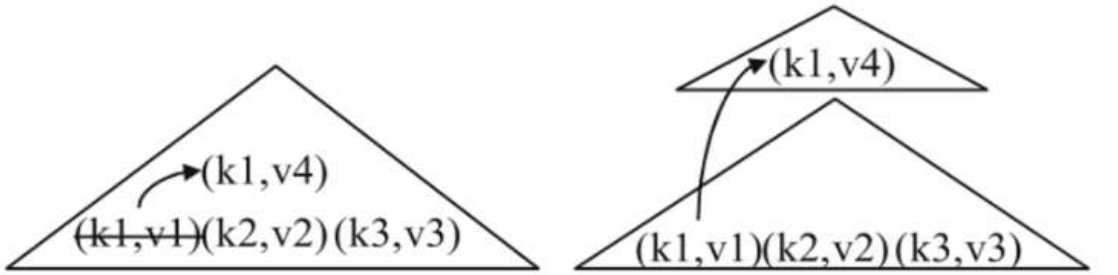
  就地更新结构

  直接覆盖旧记录，存储更新内容

- **Out-of-place update structure**：

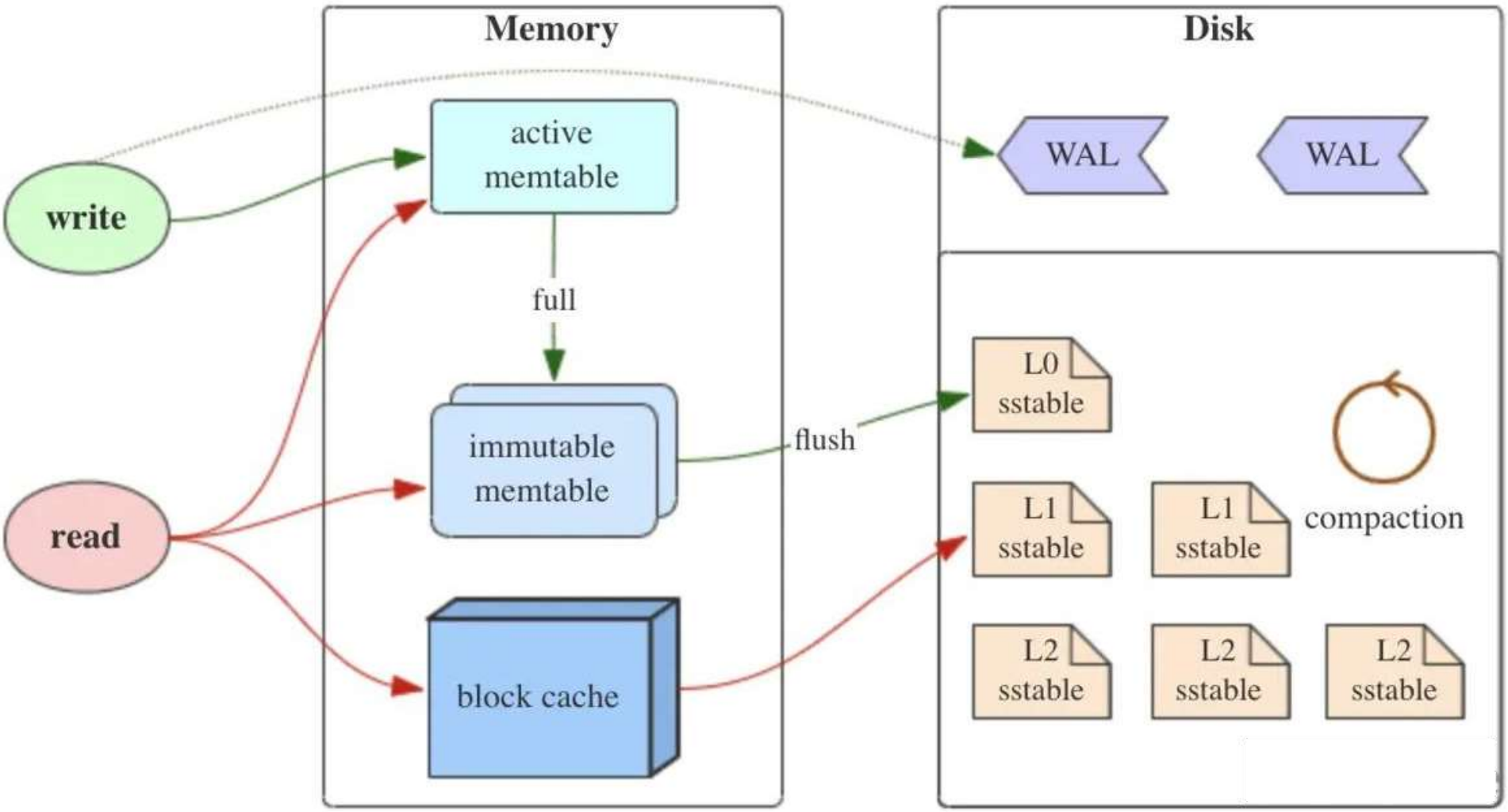
  异位更新结构

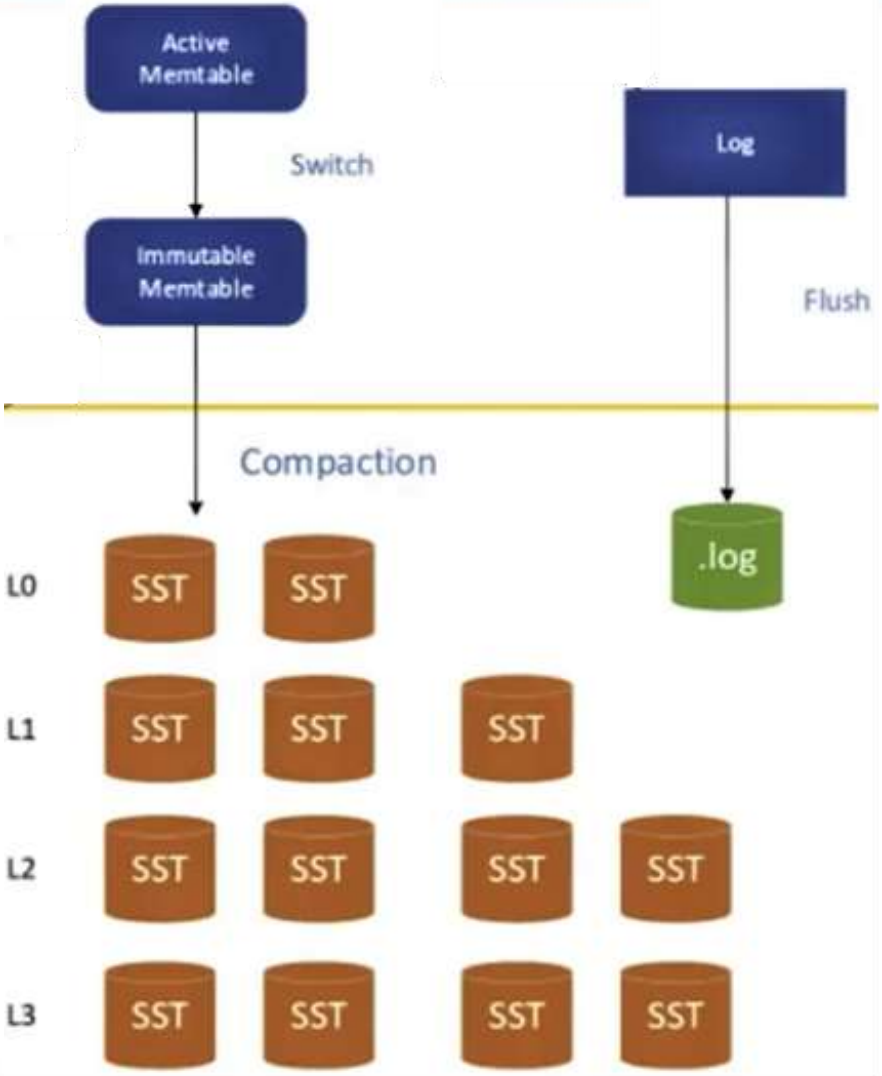
  更新的内容存储到新的位置，而不是覆盖旧的记录



(a) In-place update structure **(b)** Out-of-place update structure

LSM树的结构


Memory
Disk
write
read
active
memtable
full
immutable
memtable
block cache
flush
WAL
WAL
L0
sstable
L1
sstable
L1
sstable
compaction
L2
sstable
L2
sstable
L2
sstable


Active Memtable
Switch
Immutable Memtable
Log
Flush
Compaction
.log
L0
L1
L2
L3
SST

数据库设计总结

- 需求分析
- 概念结构设计
- 逻辑结构设计
- 数据库的物理设计
- 数据库的实施
- 数据库运行和维护

概念结构设计小结

集成局部视图

1、合并分E-R图，生成初步E-R图

- 消除冲突
  - 属性冲突
  - 命名冲突
  - 结构冲突

2、修改与重构

- 消除不必要的冗余，设计生成基本E-R图
  - 分析方法
  - 规范化理论

逻辑结构设计小结

- 任务
  - 将概念结构转化为具体的数据模型

- 逻辑结构设计的步骤

  (1)将概念结构转化为特定DBMS支持下的数据模型转换

  (2)对数据模型进行优化

  (3)设计用户子模式

- 重点：**E-R模型向关系模型的转换原则**

物理结构设计小结

- 重点：B+树结构