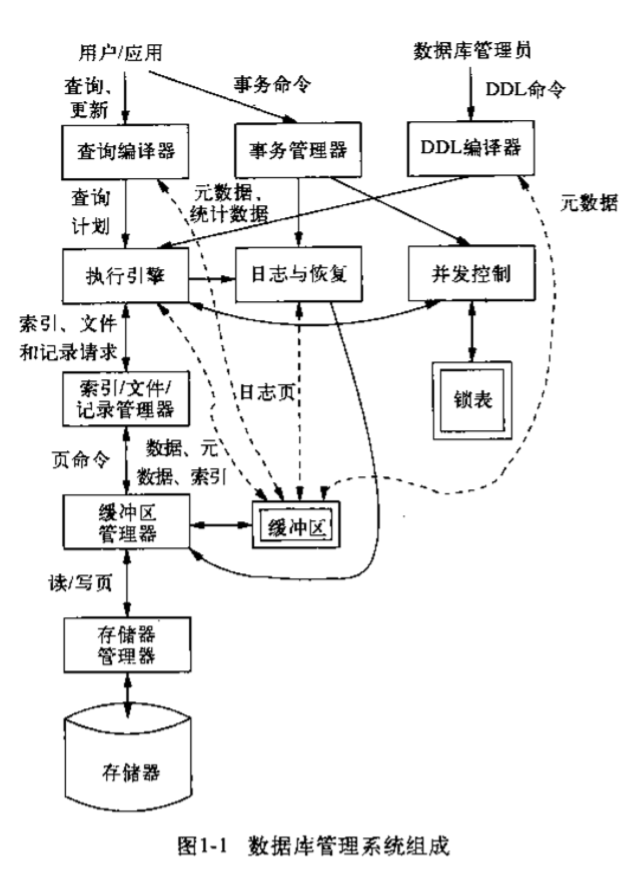
# Database

1.1.5/信息集成

信息集成：把包含在多个相关数据库中的信息连接到一起，成为一个整个数据库。

解决方案：1.创建数据仓库（data warehouse）2.实现协调器（mediator）或者中间件（middleware）

1.2/DBMS



## 第一部分

第二章：关系数据模型

二维表（也称作‘关系’）

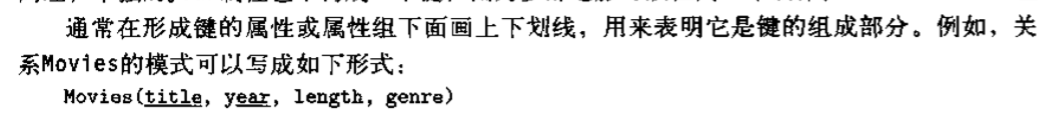
数据模型：数据结构、数据操作、数据上的一些约束

2.2（important）

关系名和其属性集合的组合叫做这个关系的模式（schema）。

2.2.7/键约束

键是唯一的，不可重复（若键约束多个属性组成，那么单个属性可以相同，但总体不同）



虚拟键：可以给其赋予ID，ID此时称为虚拟键。

2.3/在SQL（sequel）定义关系模式

2.3.2//数据类型 （关系中所有的属性必须要有一个数据类型）

|  |  |  |
| --- | --- | --- |
| 可变长度或固定长度字符串 | CHAR(n) | 表示最大为n个字符的固定长度字符串（自动填充至n） |
| VARCHAR（n） | 表示最多有n个字符的字符串（不填充，用标识符表示结束） |
| 固定或可变长度的位串 | BIT(n) | 表示长为n的位串 |
| BIT VARYING(n) | 表示最大长度为n的位串 |
| BOOLEAN  （具有逻辑类型的值） | TRUE | 真值 |
| FALSE | 假值 |
| UNKNOWN | 不确定 |
| 整数值 | INT/INTEGRE | 表示整数 |
| 浮点值 | FLOAT/REAL | 表示小数 |
| 日期和时间 | DATE/TIME | 表示日期和时间 |

# 数据库设计的第一范式

# 

数据库表中不能出现重复记录，每个字段是原子性的不能再分

不符合第一范式的示例

|  |  |  |
| --- | --- | --- |
| 学生编号 | 学生姓名 | 联系方式 |
| 1001 | 张三 | zs@gmail.com,1359999999 |
| 1002 | 李四 | [ls@gmail.com](mailto:ls@gmail.com),13699999999 |
| 1001 | 王五 | ww@163.net,13488888888 |

**存在问题：**

1. 最后一条记录和第一条重复（不唯一，没有主键）
2. 联系方式字段可以再分，不是原子性的

|  |  |  |  |
| --- | --- | --- | --- |
| 学生编号(pk) | 学生姓名 | email | 联系电话 |
| 1001 | 张三 | zs@gmail.com | 1359999999 |
| 1002 | 李四 | [ls@gmail.com](mailto:ls@gmail.com) | 13699999999 |
| 1003 | 王五 | [ww@163.net](mailto:ww@163.net) | 13488888888 |

关于第一范式，每一行必须唯一，也就是每个表必须有主键，这是我们数据库设计的最基本要求，主要通常采用数值型或定长字符串表示，关于列不可再分，应该根据具体的情况来决定。如联系方式，为了开发上的便利行可能就采用一个字段了。

# 数据库设计的第二范式

# 

第二范式是建立在第一范式基础上的，另外要求所有非主键字段完全依赖主键，不能产生部分依赖

**示例：**

|  |  |  |  |
| --- | --- | --- | --- |
| 学生编号 | 学生姓名 | 教师编号 | 教师姓名 |
| 1001 | 张三 | 001 | 王老师 |
| 1002 | 李四 | 002 | 赵老师 |
| 1003 | 王五 | 001 | 王老师 |
| 1001 | 张三 | 002 | 赵老师 |

**确定主键：**

|  |  |  |  |
| --- | --- | --- | --- |
| 学生编号(PK) | 教师编号(PK) | 学生姓名 | 教师姓名 |
| 1001 | 001 | 张三 | 王老师 |
| 1002 | 002 | 李四 | 赵老师 |
| 1003 | 001 | 王五 | 王老师 |
| 1001 | 002 | 张三 | 赵老师 |

以上虽然确定了主键，但此表会出现大量的冗余，主要涉及到的冗余字段为“学生姓名”和“教师姓名”，出现冗余的原因在于，学生姓名部分依赖了主键的一个字段学生编号，而没有依赖教师编号，而教师姓名部门依赖了主键的一个字段教师编号，这就是第二范式部分依赖。

**解决方案如下：**

**学生信息表**

|  |  |
| --- | --- |
| 学生编号（PK） | 学生姓名 |
| 1001 | 张三 |
| 1002 | 李四 |
| 1003 | 王五 |

**教师信息表**

|  |  |
| --- | --- |
| 教师编号（PK） | 教师姓名 |
| 001 | 王老师 |
| 002 | 赵老师 |

**教师和学生的关系表**

|  |  |
| --- | --- |
| 学生编号(PK)  fkà学生表的学生编号 | 教师编号(PK) fkà教师表的教师编号 |
| 1001 | 001 |
| 1002 | 002 |
| 1003 | 001 |
| 1001 | 002 |

如果一个表是单一主键，那么它就复合第二范式，部分依赖和主键有关系

以上是一种典型的“多对多”的设计

# 数据库设计的第三范式及三大范式总结

# 

建立在第二范式基础上的，非主键字段不能传递依赖于主键字段。（不要产生传递依赖）

|  |  |  |  |
| --- | --- | --- | --- |
| 学生编号（PK） | 学生姓名 | 班级编号 | 班级名称 |
| 1001 | 张三 | 01 | 一年一班 |
| 1002 | 李四 | 02 | 一年二班 |
| 1003 | 王五 | 03 | 一年三班 |
| 1004 | 六 | 03 | 一年三班 |

从上表可以看出，班级名称字段存在冗余，因为班级名称字段没有直接依赖于主键，班级名称字段依赖于班级编号，班级编号依赖于学生编号，那么这就是传递依赖，解决的办法是将冗余字段单独拿出来建立表，如：

**学生信息表**

|  |  |  |
| --- | --- | --- |
| 学生编号（PK） | 学生姓名 | 班级编号（FK） |
| 1001 | 张三 | 01 |
| 1002 | 李四 | 02 |
| 1003 | 王五 | 03 |
| 1004 | 六 | 03 |

**班级信息表**

|  |  |
| --- | --- |
| 班级编号（PK） | 班级名称 |
| 01 | 一年一班 |
| 02 | 一年二班 |
| 03 | 一年三班 |

以上设计是一种典型的一对多的设计，一存储在一张表中，多存储在一张表中，在多的那张表中添加外键指向一的一方的主键

**三范式总结**

第一范式：有主键，具有原子性，字段不可分割

第二范式：完全依赖，没有部分依赖

第三范式：没有传递依赖

数据库设计尽量遵循三范式，但是还是根据实际情况进行取舍，有时可能会拿冗余换速度，最终用目的要满足客户需求。

一对一设计，有两种设计方案：

第一种设计方案：主键共享

第二种设计方案：外键唯一