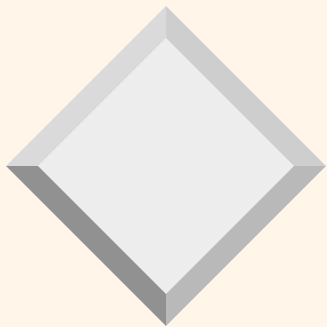


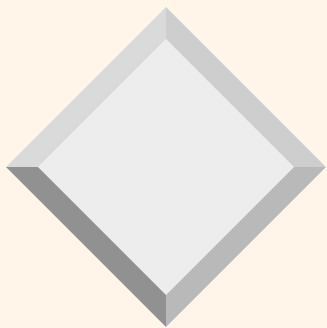
# 关系模型

## Introduction




## v 思考题

- 假如你被指派开发一个信息系统，你应该考虑哪些数据资源方面的因素？分析完成对应的ER图设计、数据库设计。



## v 思考题

- 假如你被指派开发一个信息系统，你应该考虑哪些数据资源方面的因素？分析完成对应的ER图设计、数据库设计。
- 假如让你实现一个数据存储系统，你应该考虑哪些因素？如何实现数据的存储？
- 如果让你实现一个数据查询系统，你应该考虑哪些因素？如何实现查询解析？



# 为什么学习关系模型？

- √ 目前广泛使用.
  - IBM DB2, Microsoft SQL Server, Oracle, etc.
- √ 存储方式:
  - 行存储: DB2, Oracle
  - 列存储: MonetDB, Vertica



# 关系数据库定义

- √ 关系数据库: 关系的集合
- √ 关系: 两部分组成:
  - 实例: 由行和列的表.
  - 模式: 定义关系的名字, 以及每一行的类型.
    - u E.G. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
- √ 可以把关系认为是行 (或者元组) 的集合.

# 例子

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

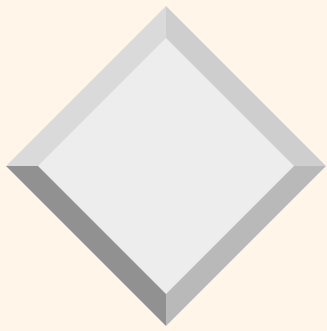


**关系模式：**一个关系的关系名和属性的集合被称为是关系的模式。

Student(sid,name,login,age,gpa)

**元组(Tuples)：**关系中的每一行被称为是一个元组，也称为记录(Records)。元组在每个属性上有相应的属性值。

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8



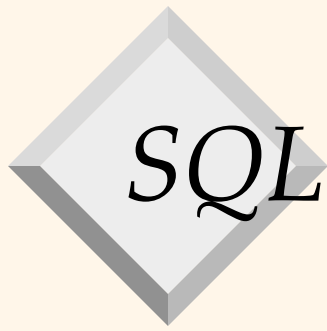
- √ **域(Domains):** 关系中每个属性都有一个特定的数据类型，被称为该属性的域。每条元组相对应的属性值必须包括在该域的范围內。属性A的域表示成 $DOM(A)$ 。
- √ **关系的实例(Instances):** 对于一个给定的关系，每一个元组的集合都被称为该关系的一个实例。通过对关系进行插入、删除、修改等操作可以获得该关系的不同实例。





# 关系的查询语言

- √ 关系模型的主要优势: 支持简单、有效的查询.
- √ 查询由用户来写, DBMS负责执行.
  - 关键: 关系查询简洁的语义.
  - 允许优化措施, 确保查询结果的正确性.



- v 由IBM (system R) 在1970s开发
- v 成为标准
- v 发展历程:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision, current standard)
  - SQL-99 (major extensions)



√ 查找所有18岁的学生:

```
SELECT *  
FROM Students S  
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

# 查询多个关系


v 下面这个查询如何执行?

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade="A"
```

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

S.name	E.cid
Smith	Topology112



# 利用SQL产生关系

- v 产生一个关系： Students.
- v 下个例子，表Enrolled是学生选课的信息.

```
CREATE TABLE Students
(sid: CHAR(20),
 name: CHAR(20),
 login: CHAR(10),
 age: INTEGER,
 gpa: REAL)
```

```
CREATE TABLE Enrolled
(sid: CHAR(20),
 cid: CHAR(20),
 grade: CHAR(2))
```



# 撤销关系

**DROP TABLE** Students

- √ 撤销关系之后，模式信息和元组都被删除.

**ALTER TABLE** Students

**ADD COLUMN** firstYear: integer

- √ 修改模式。加了新列firstYear;



# 增加或删除元组

v 加入元组:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

v 删除满足条件的元组 (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```



# 视图/Views

- v 一个 view 是一个关系, 但是我们存储的是定义 *definition*, 而不是元组的集合.

```
CREATE VIEW YoungActiveStudents (name, grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age<21
```

- v Views可以通过 **DROP VIEW** 命令撤销.

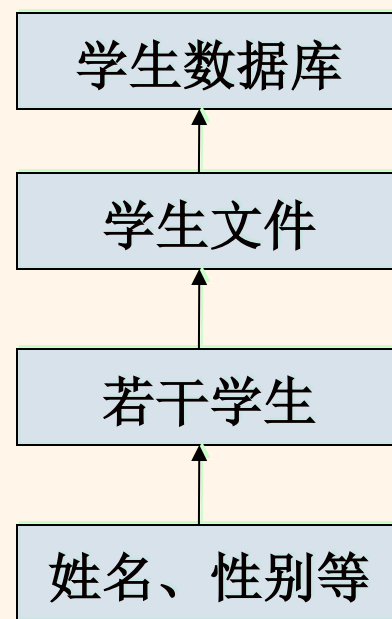
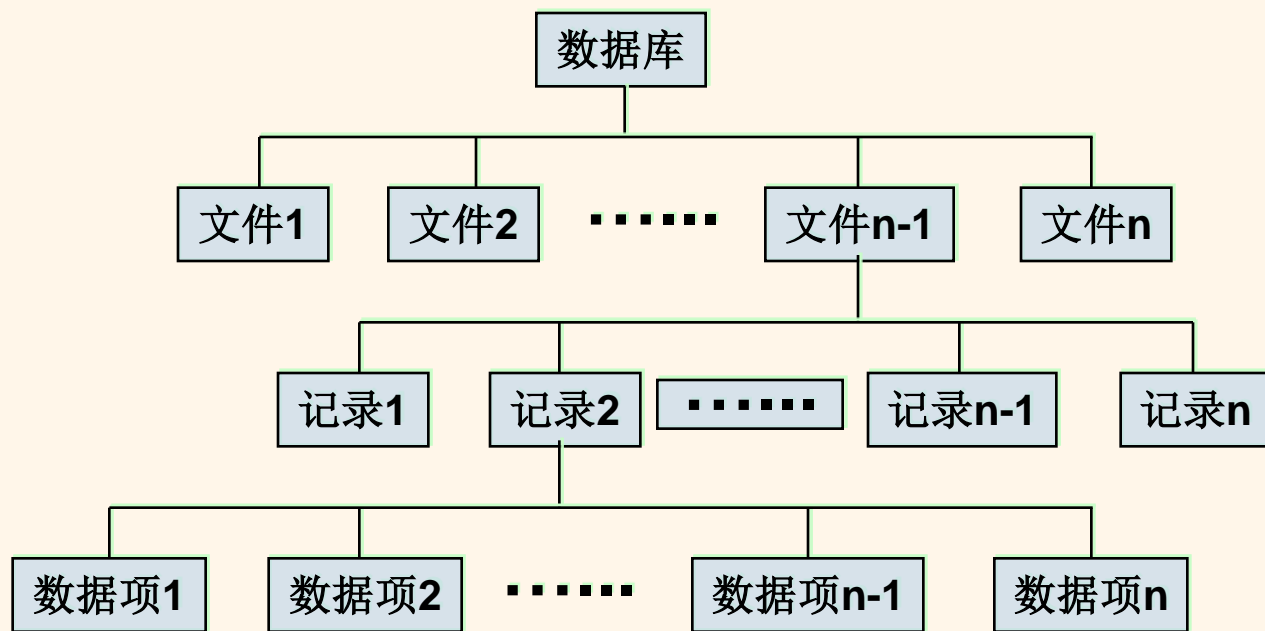




# *Views* 的安全性

- v *Views* 可以表示需要的信息 (或者总结信息), 从而隐藏了细节数据.
  - 例如YoungStudents, 我们可以从中知道谁注册了, 但是不知道他们注册了哪些课程.

# 数据组织的层次





# 数据组织的相关概念

## √ 数据组织的相关概念

### – 数据项

- u 是组成数据系统的有意义的最小基本单位。它的作用是描述一个数据处理对象的某些属性。

### – 记录

- u 与数据处理的某一对象有关的一切数据项构成了该对象的一条记录。标识记录的数据项称为关键项。

### – 文件

- u 相关（同类）记录的集合称为文件。

### – 数据库

- u 按一定方式组织起来的逻辑相关的文件集合形成数据库。

# 举例

## 主文件

记录地址

A

B

C

D

E

学号	姓名	成绩
870005	张三	456
870002	李四	645
870001	王五	587
870004	孙六	676
870003	钱七	565

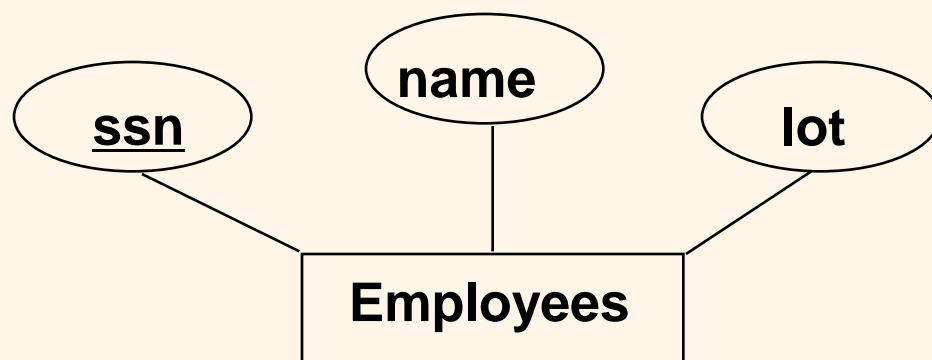
## 索引表

主关键字（学号）	记录地址
870001	C
870002	B
870003	E
870004	D
870005	A

关键字学号与学生记录地址的对应表

# 逻辑DB 设计: ER 到关系的转换

√ 实体——表.



```
CREATE TABLE Employees  
  (ssn CHAR(11),  
   name CHAR(20),  
   lot INTEGER,  
   PRIMARY KEY (ssn))
```

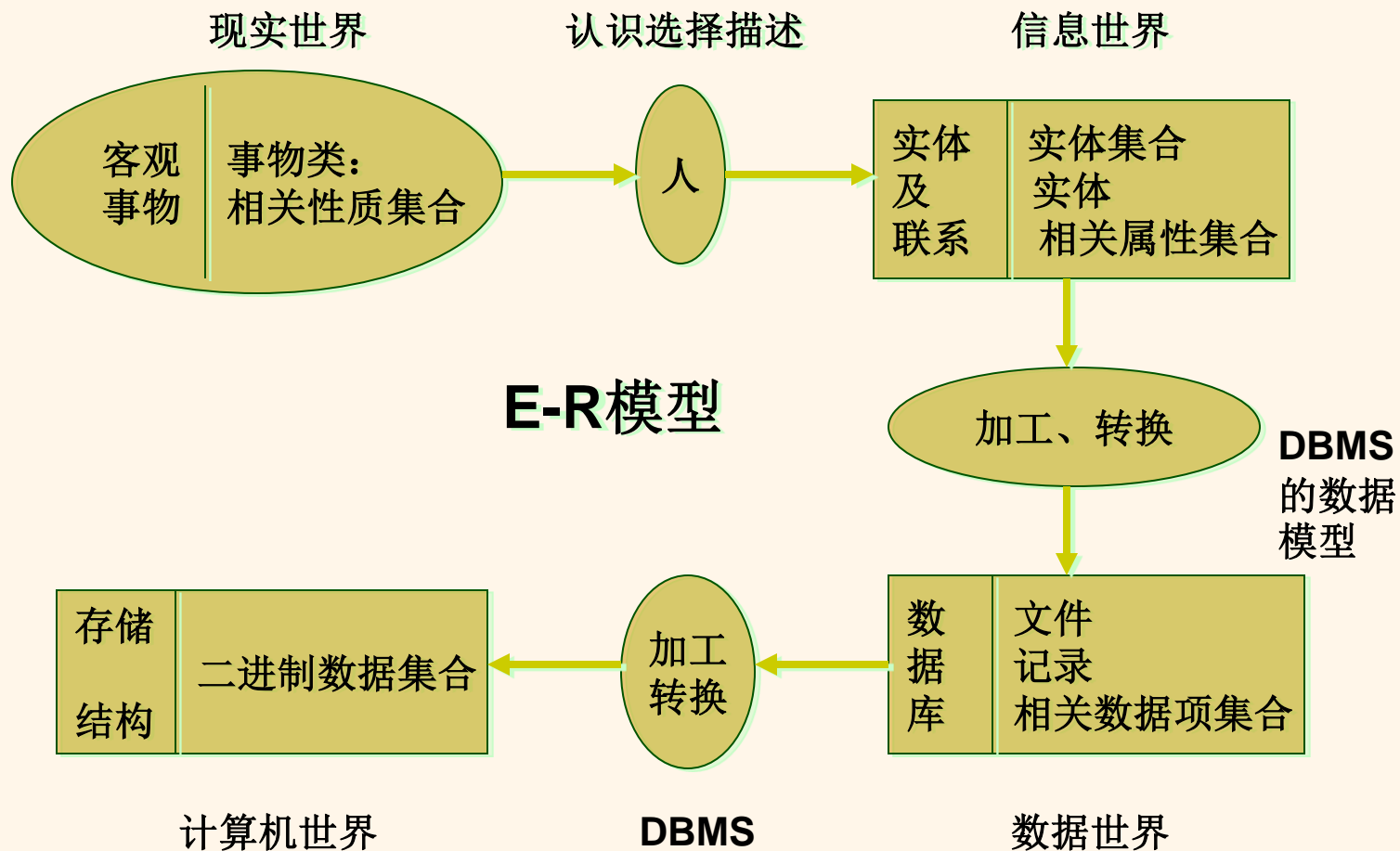


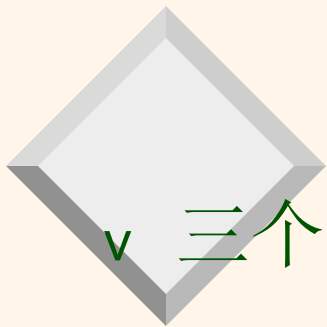
# 把ER 图转换到表

```
CREATE TABLE Manages(  
    ssn CHAR(11),  
    did INTEGER,  
    since DATE,  
    PRIMARY KEY (did),  
    FOREIGN KEY (ssn) REFERENCES Employees,  
    FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE Dept_Mgr(  
    did INTEGER,  
    dname CHAR(20),  
    budget REAL,  
    ssn CHAR(11),  
    since DATE,  
    PRIMARY KEY (did),  
    FOREIGN KEY (ssn) REFERENCES Employees)
```

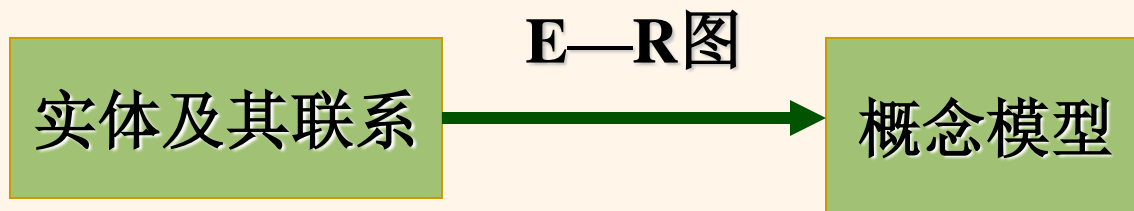
# 信息的转换





## v 三个不同世界术语

客观世界	信息世界	数据世界
组织（事物及其联系） 事物类（总体） 事物（对象、个体） 特征（性质）	实体及其联系 实体集 实体 属性	数据库（概念模型） 文件 记录 数据项





# E-R信息模型的设计与应用

## √ E-R图描述现实世界的概念模型

### - 实体:

实体集

### - 属性:

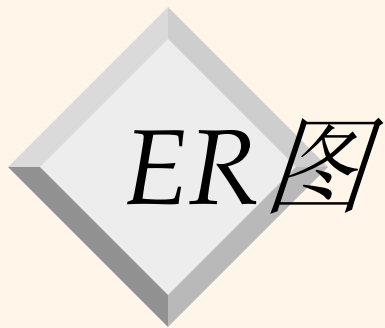
属性

### - 联系:

- u 一对一联系
- u 一对多联系
- u 多对多联系

联系集

线段

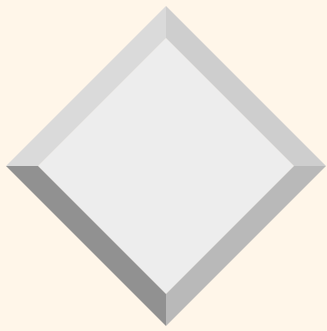


## √ 实体（Entity）：

- 是一个有着一系列显著的、易辨认的属性的对象。
- 实体可以是具体的（物体、人物等。）
- 实体也可以抽象的（事、概念、事物之间的联系）

## √ 确定实体的指导

- 找出问题中的大模块
- 问题陈述中的名词



## v 属性（Attribute）

- 实体的特性,它描述了实体的一个部分。
- 一个实体可由若干个属性来刻画。
  - u 如学生（学号，姓名，性别，……）

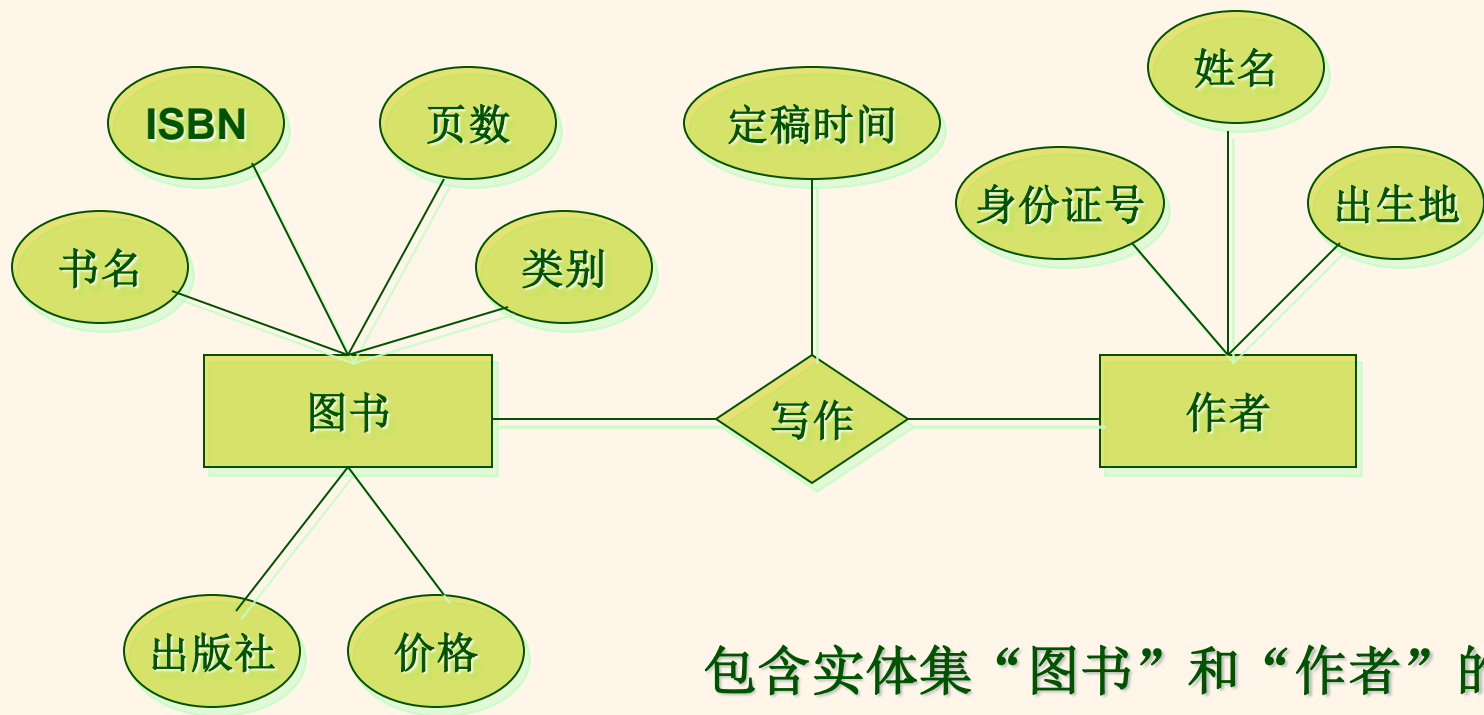
## v 码（Key）：

- 唯一标识实体的属性集。
  - u 如：学号为学生的码。

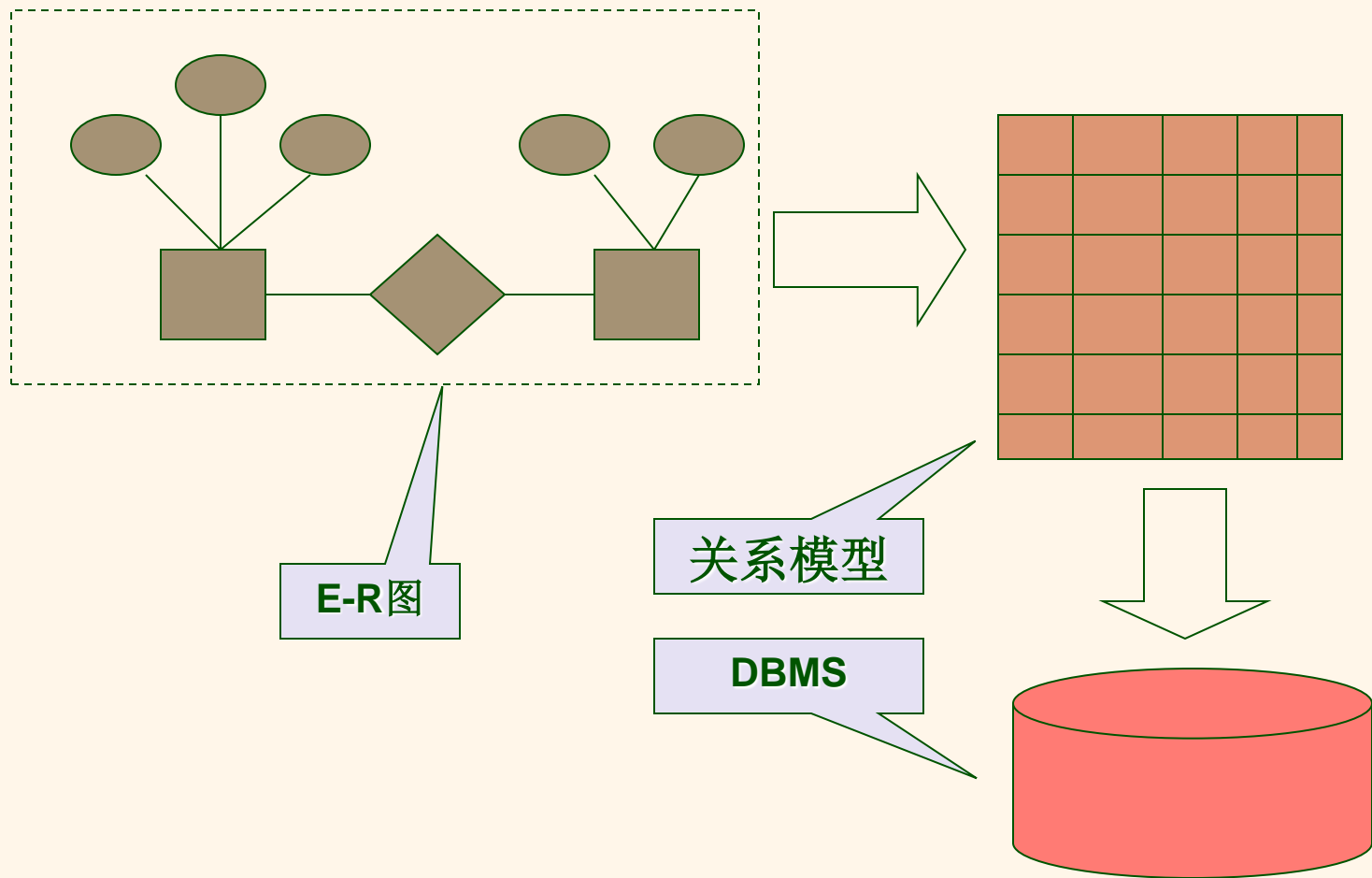
## v 确定属性的指导

- 可能将名词当作属性,但不应将其和实体混淆.

# E-R图



# E-R图向关系数据模型的转换转换过程

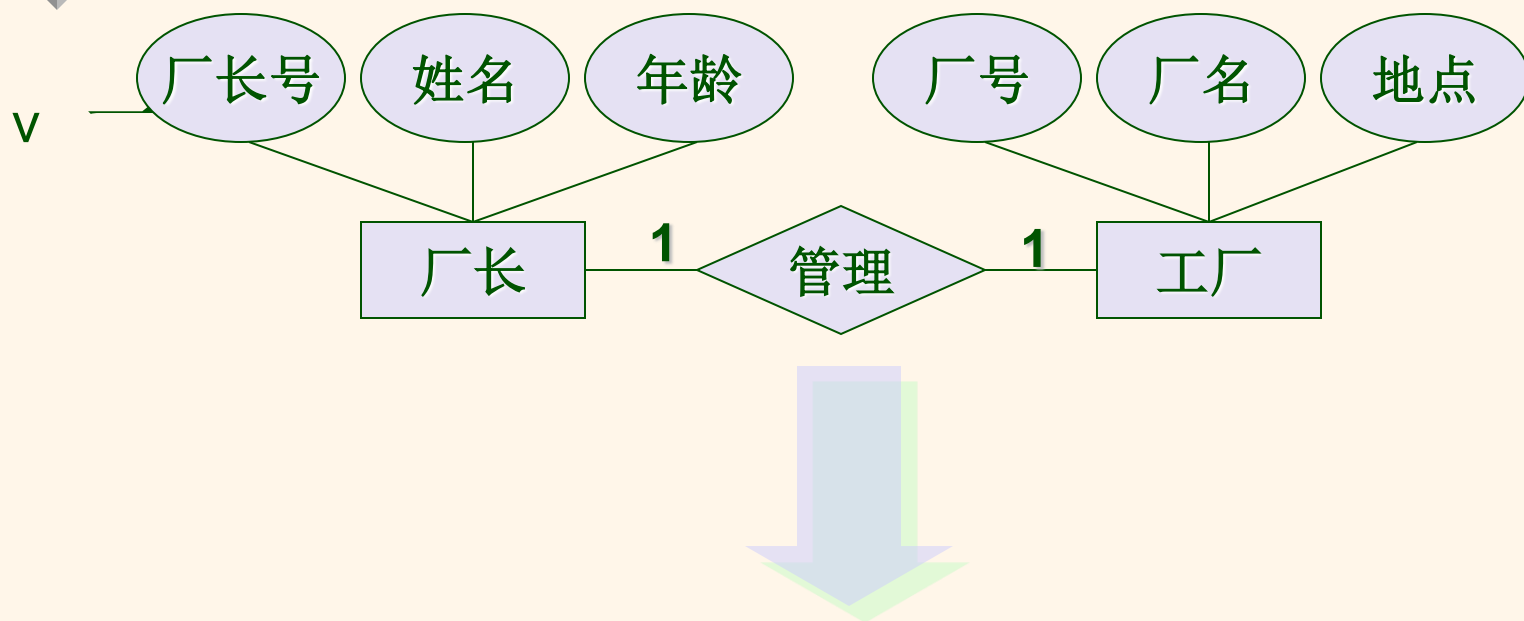




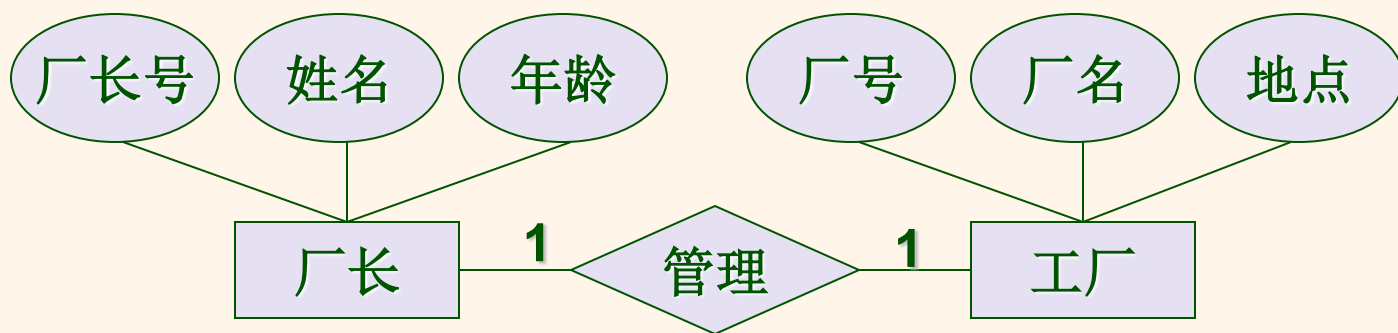
# E-R图的转换

- 将E-R图转换成相应的表
  - u 表是行和列的集合，实体被表示成表的形式。
  - u 用列标题表示实体的属性
  - u 用行表示关于实体的实际数据
- v 关于表和属性的命名规则
  - 属性名和表名中不能包含空格
  - 表名对实体的描述应该是有意义的。
    - u 如 student(cStuID,cStuName,nStuAge,.....)
  - 表名只能描述一个主题

# 转换举例



# 关系模式的生成

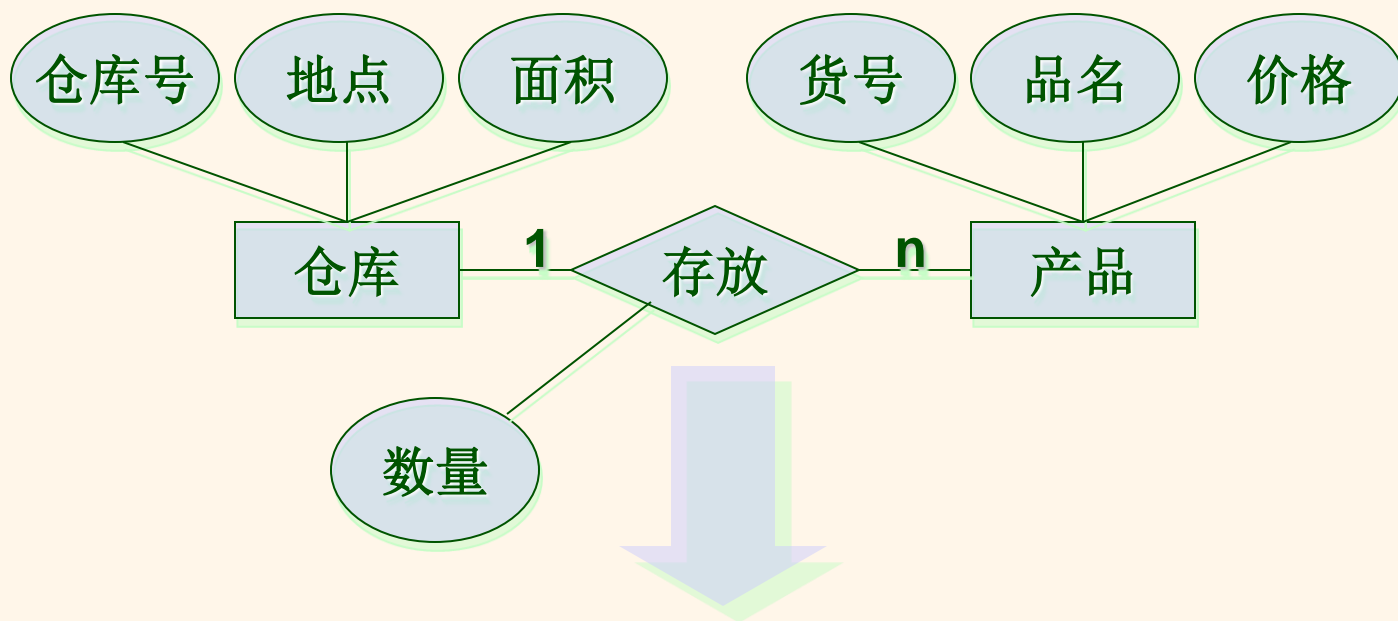


厂长 (厂长号, 厂号, 姓名, 年龄)

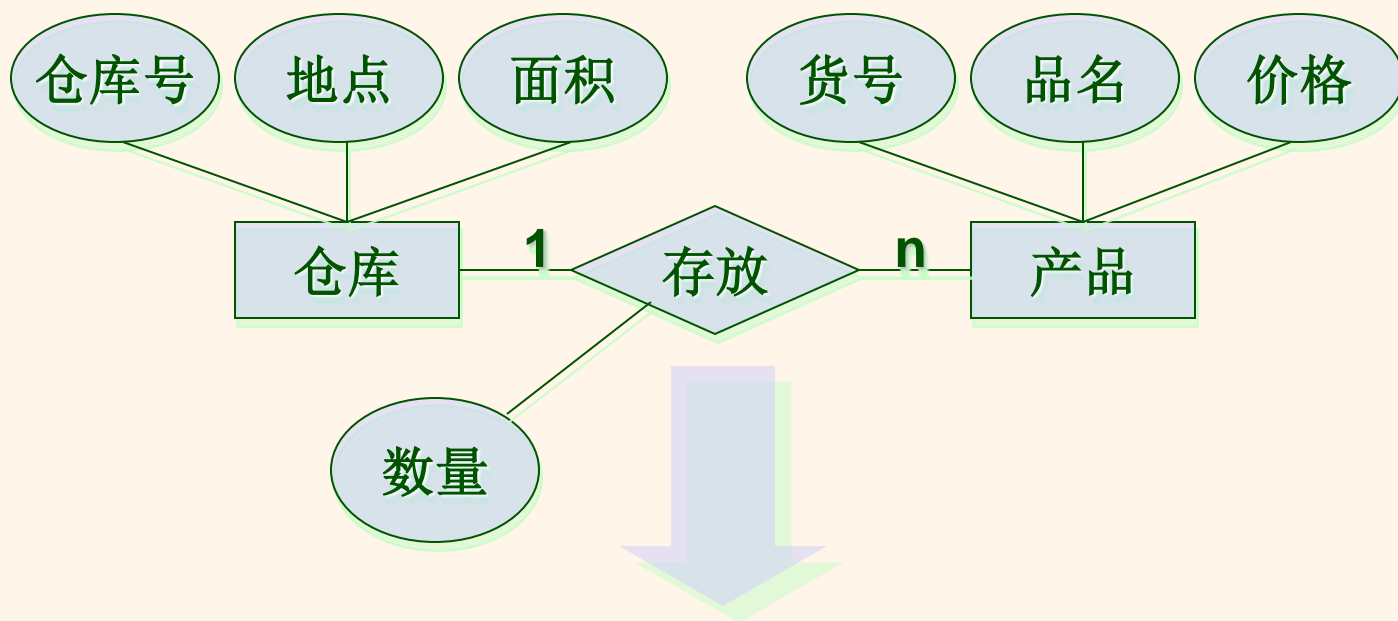
工厂 (厂号, 厂名, 地点)



# 转换举例



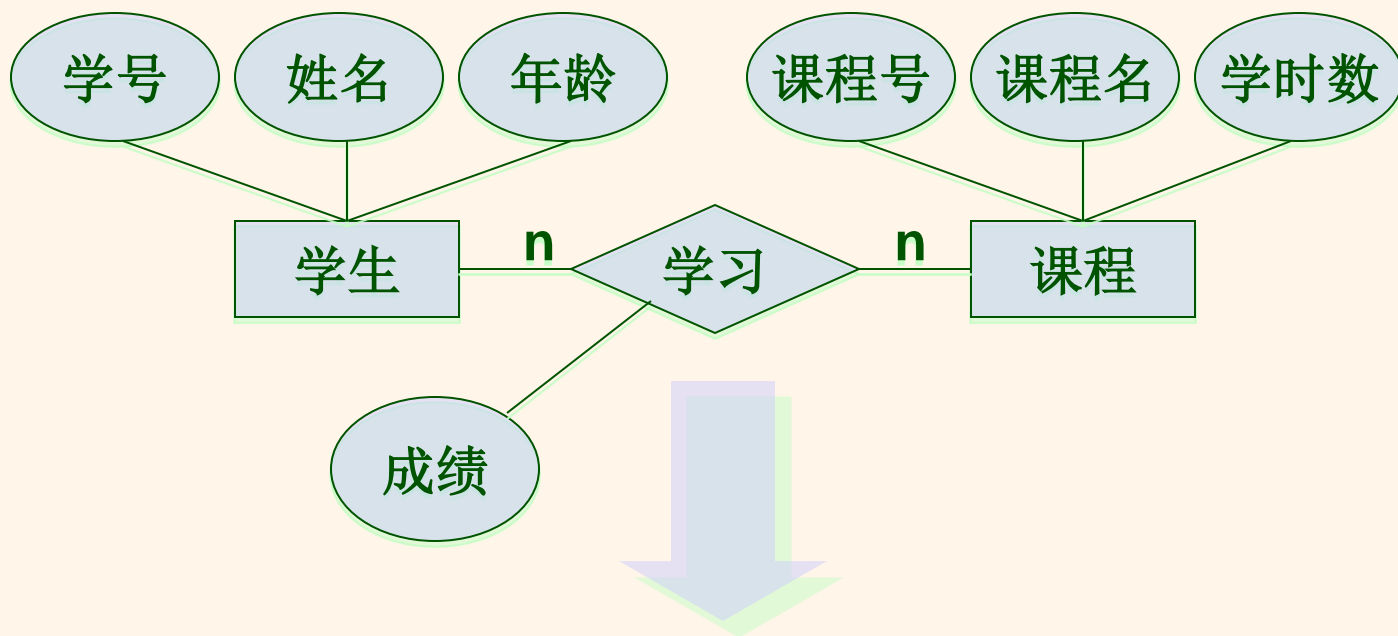
# 转换举例



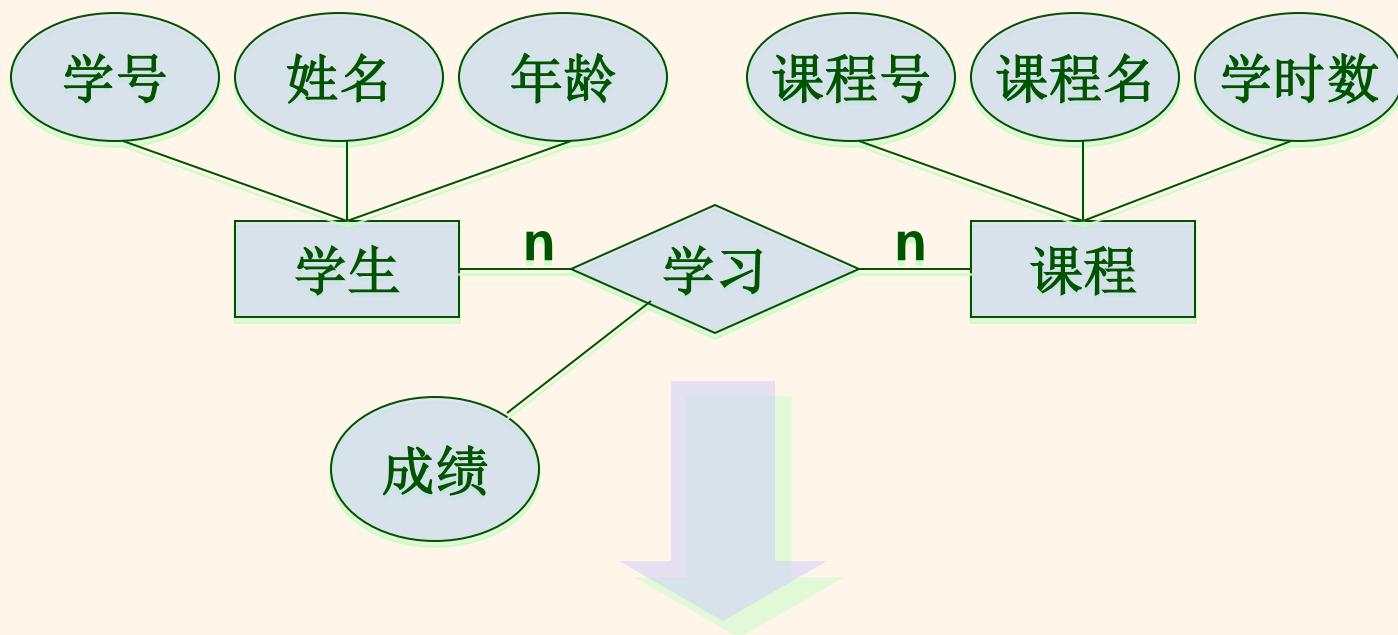
仓库（仓库号，地点，面积）

产品（货号，品名，价格，仓库号，数量）

# 转换举例三



续



学生 (学号, 姓名, 年龄)

课程 (课程号, 课程名, 学时数)

学习 (学号, 课程号, 成绩)

# 数据库设计规范化

- √ 数据库设计规范化——范式理论篇
- √ 必要性：解决在关系模式设计时，存在的数据存储异常现象：数据冗余、修改异常、插入异常、删除异常等。

ISBN	书名	页数	价格	出版社	作者
7-04-001968-O.719	概率论	403	5.8	高教出版社	盛聚
7-04-001968-O.719	概率论	403	5.8	高教出版社	谢式千
7-111-06887-4	可靠性模型与应用	270	19.0	机械工业出版社	蒋仁言
7-111-06887-4	可靠性模型与应用	270	19.0	机械工业出版社	左明健
7-5327-1224.9/1.717	基督山伯爵	1428	18.0	止海译文出版社	大仲马
7-5237-1224-9/1.321	三个火枪手	982	16.7	上海译文出版社	大仲马

续

√ 第一规范型：

- 如果一个关系模式中的属性都是单纯的（即不可再分为更小的属性），则称该模式是属于第一规范型（First Normal Form, 即 1NF）。

姓名	地址			
	省	市	街道	邮编
甲	江苏	南京	卫岗	210095

续

## √ 第二规范型:

- 如果一个关系模式属于1NF，并且所有的非关键字都完全地依赖于关键字（即不存在部分依赖），则称该关系模式属于第二规范型，即 2NF。

A 零 件 号	B 仓 库 号	C 零 件 数 量	D 仓 库 地 址
1	1	100	北 区 1 号
2	1	150	北 区 1 号
3	1	200	北 区 1 号
4	2	150	南 区 1 号

续

v

### 第三规范型：

- 如果一个关系模式属于2NF，并且不存在非关键字传递地依赖于关键字，则称该关系模式属于第三规范型（3NF）。

职工号	职工	职务	工资
1001	张三	工程师	200
1002	李四	技术员	120
1005	王五	高工	350

- 鲍依斯-科得范式（BCNF）：在第三范式的基础上，数据库表中如果不存在任何字段对任一候选关键字段的传递函数依赖则符合第三范式。



续

v 适当的规范化

规范化程度越高

↓ 数据冗余

↓ 更新异常

↑ 连接运算时间

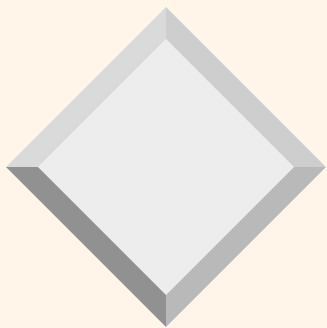
↑ 查询时间

↓ 效率



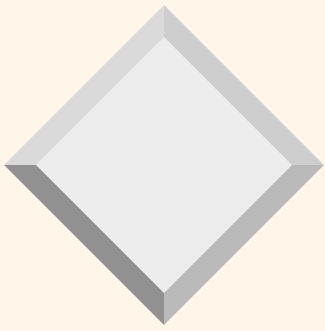
# 关系模型: 总结

- √ 数据的表的描述形式.
- √ 简单直观, 目前广为使用.
- √ 完整性约束可以由DBA设定, 一般是有一定的应用语义. DBMS 负责有效性检验.
  - 两种重要的约束: 主键约束和外键约束
  - 另外, 还有域约束.
- √ 强大的查询语言.
- √ 数据库设计的工具: ER图



## v 思考题

- 假如你被指派开发一个信息系统，你应该考虑哪些数据资源方面的因素？分析完成对应的ER图设计、数据库设计。
- 假如让你实现一个数据存储系统，你应该考虑哪些因素？如何实现数据的存储？
- 如果让你实现一个数据查询系统，你应该考虑哪些因素？如何实现查询解析？



v Q&A?