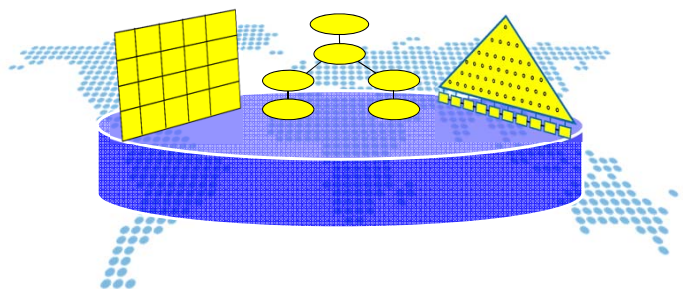


数据库系统 关系模型与SQL (1)

陈世敏
(中科院计算所)



Outline

- 关系模型复习
- SQL 初步
 - 表的定义、增删改
- 逻辑设计: ER图到关系模型

关系模型复习: Table/Relation (表)

- 列(Column): 一个属性, 有明确的数据类型
 - 例如: 数值类型 (e.g., int, double), 字符串类型(varchar), 类别类型(有些像编程语言中的enum)
 - 必须是原子类型, 不能够再进一步分割, 没有内部结构
- 行(Row): 一个记录 (tuple, record)
 - 表是一个记录的集合
 - 记录之间是无序的
- 通常是一个很瘦长的表
 - 几列到几十列
 - 成千上万行, 很大的表可以有亿/兆行

表的数学定义

- K列的表: $\{ \langle t_1, \dots, t_k \rangle \mid t_1 \in D_1, \dots, t_k \in D_k \}$
 - 整个表是一个集合 $\{ \langle t_1, \dots, t_k \rangle \}$
 - 集合的每个元素有这样的形式 $\langle t_1, \dots, t_k \rangle$
 - 第j个部分 t_j
 - D_j 是第j列的类型所对应的所有可能取值的集合 (域)

举例：学生信息表

列数称为度(arity 或 degree)
每一列是一个属性 (~10)

每一行是一个学生记录 (~10⁴)

记录的条数称为基(cardinality)

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
...
...
...
...
...
...
...

Key (键)

• Candidate key (候选键)

- 唯一确定一个记录
- 最少的列组合

• Primary key (主键)

- 选一个候选键为主键

• Foreign key (外键)

- 是另一个表的 Primary key
- 唯一确定另一个表的一个记录

举例：学生信息表

Primary Key (主键): 唯一确定一个记录

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
...
...
...
...
...
...
...

举例：选课信息表

Foreign Key (外键) Foreign Key (外键)

Couse ID	Student ID	Year	Semester	Grade
7001	131234	2014	春季	85
7012	145678	2014	夏季	80
7005	129012	2013	秋季	95
...
...
...
...
...
...
...

举例：Primary Key 与 Foreign Key

Course

ID	Name		
7001	体系结构		
7005	数据结构		
7012	大数据处理		

Student

ID	Name			
131234	张飞			
145678	貂蝉			
129012	孙权			

TakeCourse

Couse ID	Student ID			
7001	131234			
7012	145678			
7005	129012			

实际上，可以把外键理解为“指针”或“引用”把两个表关联起来

举例：选课信息表

什么列是Primary key? (CourseID, StudentID) 的组合

Couse ID	Student ID	Year	Semester	Grade
7001	131234	2014	春季	85
7012	145678	2014	夏季	80
7005	129012	2013	秋季	95
...
...
...
...
...
...

Outline

- 关系模型复习
- SQL 初步
 - 表的定义、增删改
- 逻辑设计：ER图到关系模型

SQL

- 1970s, system R的SEQUEL (Structured English QUery Language)
- 1980s-至今, 成为ANSI和ISO标准
 - SQL-86: 是IBM SQL实现的一个子集
 - SQL-89: 替代了SQL-86, 定义了基础常用的SQL功能
 - SQL-92: 更多数据类型、多种连接操作、断言、临时表、动态SQL等
 - SQL-99: 触发器、存储过程、用户定义函数、数据仓库扩展等
 - 之后的标准增加了对XML的支持等
 - 大部分数据库系统实现了SQL-92, 并在此基础上进行了扩展, 对于SQL-99和之后的标准大多只实现了标准的一部分
- 主流的关系型数据库语言
 - Declarative language (宣告式编程)
 - 4GL (第四代语言)
 - 3GL: 例如 C, C++, Java, C#, Javascript等

SQL语句的分类

- 数据定义 (Data Definition Language, DDL)
 - 表的增删改、视图的操作等
- 数据操纵 (Data Manipulation Language, DML)
 - 记录的增删改查
- 数据控制 (Data Control Language, DCL)
 - 访问控制

Create Database

- 创建一个数据库
 - RDBMS可以管理多个数据库
 - 每个数据库可以包含多个表

- 语法

```
create database 数据库名;
```

- 举例

```
create database my_example_db;
```

创建表: Create Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer,  
  Name varchar(20),  
  Birthday date,  
  Gender enum(M, F),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

```
create table 表名 (  
  列名 类型,  
  列名 类型,  
  列名 类型,  
  .....  
);
```

有些像程序语言中的函数定义

名字

- 名字 (表名、列名等)
- 最长128个字符
- 通常情况不分大小写
 - 如果需要区分, 那么必须用“”把名字括起来
 - 例如: “HowMany”

类型

蓝色的是标准类型，其它类型在很多系统中都支持

• 数值类型

□ 整数

- tinyint: 8位
- smallint: 16位
- integer: 32位
- bigint: 64位

□ 浮点数

- real: 32位浮点数
- double或float: 64位浮点数

□ 精确小数

- decimal(size, d)或 numeric(size, d): size位十进制数，其中小数点右侧有d位小数
 - 所以，小数点左侧有size-d位数字

类型

• 字符串类型

- char(n): 定长字符串，长度为n
- varchar(n): 变长字符串，长度最大为n

• 日期和时间

- date: 日期，例如 date '2016-09-08'
- time: 时间，例如 time '13:50:12.5'
- timestamp: 时间戳，包含日期和时间

• 其它

- BLOB: Binary Large Object，例如最长4GB
- CLOB: Character Large Object，例如最长4GB
- enum(val1, val2, val3...): 枚举类型

默认值

• 当一个列的值缺失时默认的值: NULL

• 什么时候会缺失?

- 当插入一条新记录时，某列没有给具体的值
- 当修改表的定义，增加一个新的列时

☞ 可以定义非NULL的默认值

定义默认值: Default

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table 表名 (  
    列名 类型 default 默认值,  
);
```

```
create table Student (  
    ID integer,  
    Name varchar(20),  
    Birthday date default date '0000-00-00',  
    Gender enum('M', 'F'),  
    Major varchar(20) default 'Computer Science'  
);
```

声明主键Primary Key: 附加声明

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer primary key, ← 只适用于主键是一个  
  Name varchar(20),      属性的情况  
  Birthday date,  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

声明主键Primary Key: 独立声明

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer,  
  Name varchar(20),  
  Birthday date,  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float,  
  primary key (ID) ← 独立声明: 适用于任  
                      何主键, 主键可以包  
                      含多个属性  
);
```

声明候选键: 附加声明

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer primary key,  
  Name varchar(20) unique, ← 只适用于键是一个  
  Birthday date,          属性的情况  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

声明候选键: 独立声明

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer primary key,  
  Name varchar(20),  
  Birthday date,  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float,  
  unique (Name) ← 独立声明: 适用于任  
                  何候选键, 可以包含  
                  多个属性  
);
```

Primary Key和Unique的区别

• Primary Key

- 声明主键
- 唯一决定一个记录
- 组成主键的属性 **不为NULL**

• Unique

- 声明候选键
- 唯一决定一个记录
- 候选键的属性 **可以为NULL**

特别要求一列不为NULL

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer not null,  
  Name varchar(20),  
  Birthday date,  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

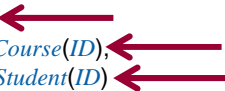


外键Foreign Key（又称为引用完整性）

TakeCourse

Couse ID	Student ID	Year	Semester	Grade
...

```
create table TakeCourse (  
  CourseID integer,  
  StudentID integer,  
  Year year,  
  Semester enum('Spring', 'Summer', 'Fall'),  
  Grade float,  
  primary key (CourseID, StudentID),  
  foreign key (CourseID) references Course(ID),  
  foreign key (StudentID) references Student(ID)  
);
```



删除表：Drop Table

• 删除一个表

drop table 表名;

• 举例


drop table Student;

修改表：Alter Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

删除列



- 删除列

`alter table 表名 drop 列名;`

- 举例


`alter table Student drop GPA;`

修改表：Alter Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

增加列



- 增加列

`alter table 表名 add 列名 类型;`

`alter table 表名 add 列名 类型 default 默认值;`

- 举例

`alter table Student add GPA float;`

`alter table Student add GPA float default 100;`

表的增删改

- DDL

□ 增：create table

□ 删：drop table

□ 改：alter table

□ 注意：alter table 可能是一个很昂贵的操作

Outline

- 关系模型复习

- SQL 初步

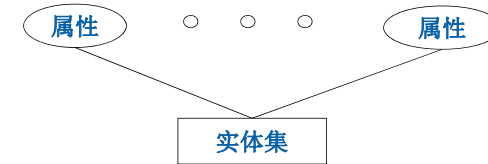
□ 表的定义、增删改

- 逻辑设计：ER图到关系模型

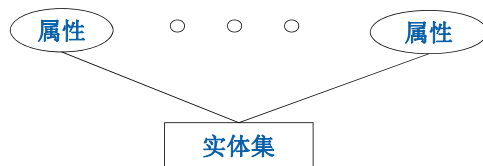
数据库设计过程

- 1) 需求分析
- 2) 概念设计 (ER模型等)
- 3) 逻辑设计 (ER模型→关系模型)
- 4) 模式细化 (规范化等)
- 5) 物理设计 (物理模式, 性能等)
- 6) 应用与安全设计 (定义外部模式等)

ER图实体集和属性的表示

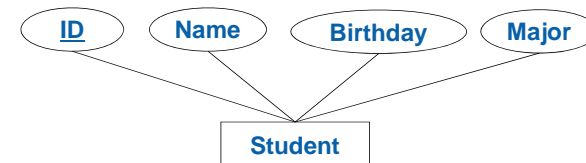


ER图的实体集→关系表

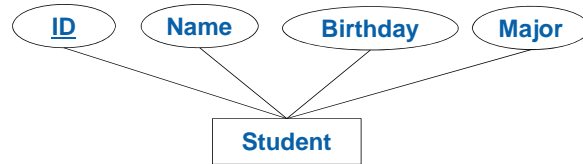


```
create table 实体集名 (  
    属性1 类型,  
    属性2 类型,  
    ...  
);
```

ER图的实体集→关系表：举例

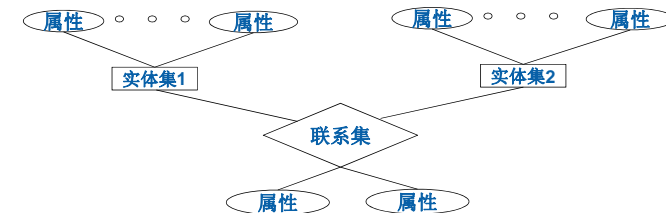


ER图的实体集→关系表：举例



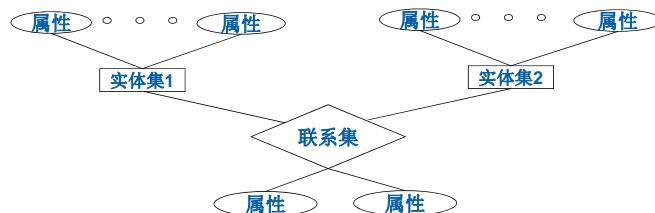
```
create table Student (
  ID integer primary key, ← 下划线对应主键
  Name varchar(20),
  Birthday date,
  Major varchar(20)
);
```

ER图的联系集



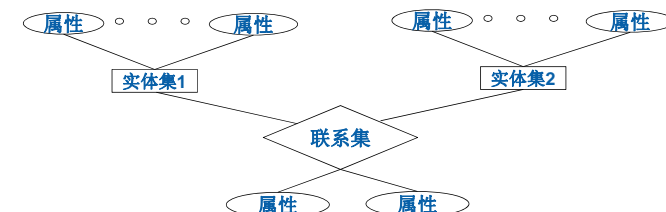
ER图的联系集→关系表

- 联系集也可以映射为关系表



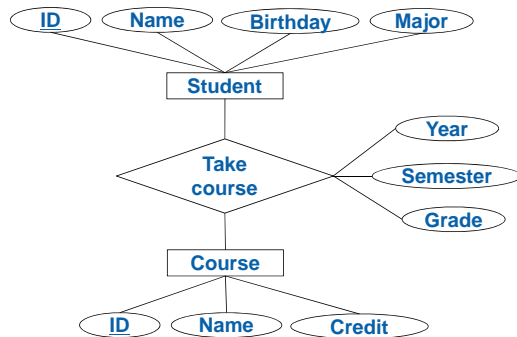
- 包含来自参与实体的主键，作为外键
- 包含联系集的属性

ER图的联系集→关系表

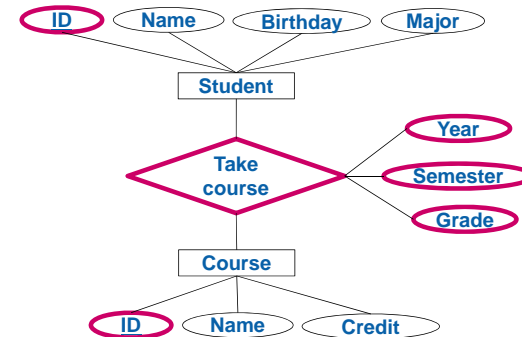


```
create table 联系集名 (
  实体集1主键 类型, 实体集2主键 类型,
  ...
  联系集属性1 类型, 联系集属性2 类型,
  ...
  foreign key ...
);
```

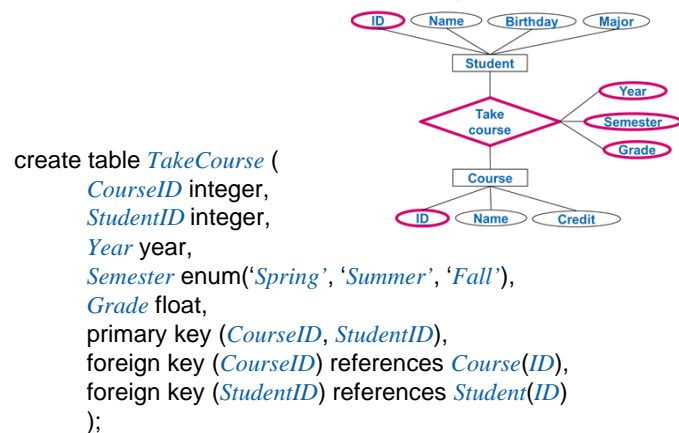
ER图的联系集→关系表：举例



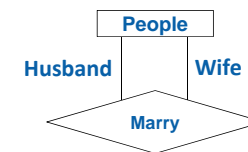
ER图的联系集→关系表：举例



ER图的联系集→关系表：举例



单一实体集内部的联系

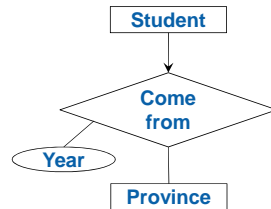


```
create table Marry (
    HusbandID integer,
    WifeID integer,
    primary key (HusbandID, WifeID),
    foreign key (HusbandID) references People(ID),
    foreign key (WifeID) references People(ID)
);
```

ER图的联系集：Key Constraint

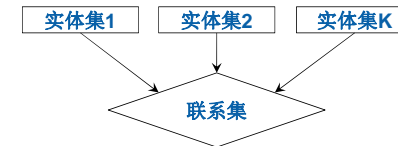
- 每个Student至多参与一个ComeFrom联系

□ 于是Student的主键也是ComeFrom的候选键



```
create table ComeFrom (
    StudentID integer,
    ProvinceID integer,
    Year integer,
    primary key (StudentID),
    foreign key (StudentID) references Student(ID),
    foreign key (ProvinceID) references Province(ID)
);
```

ER图的联系集：Key Constraint

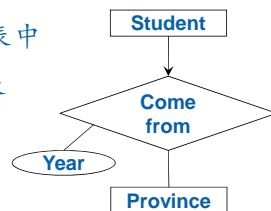


主要注意的是联系集的键

- 每个键约束的实体集，其主键都是联系集的候选键
- 选择其中一个为联系集的主键 (primary key)
- 其它为联系集候选键 (unique)

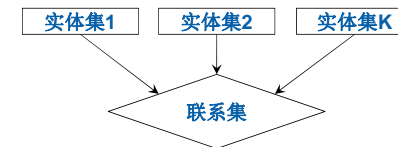
ER图的联系集：Key Constraint, 方法2

- 把ComeFrom归并到Student表中
- 不需要单独建立ComeFrom表



```
create table Student (
    .....
    ProvinceID integer,
    Year integer,
    foreign key (ProvinceID) references Province(ID)
);
```

ER图的联系集：Key Constraint, 方法2



- 选择任何一个键约束的实体集，假设其关系为R
- 扩展R，增加下述列
 - 联系集的属性
 - 参与联系的其它实体集的主键，增加外键

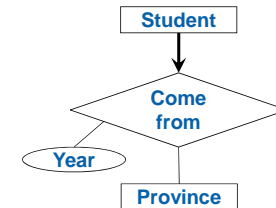
ER图的联系集：

Key Constraint + Total Participation

- Key Constraint + Total Participation
 - Key Constraint: 至多参与1次
 - Total Participation: 至少参与1次
 - 实体集每个实体参与1次且仅参与1次联系
- 那么可以在上述方法2上增加not null

ER图的联系集：举例

Key Constraint + Total Participation



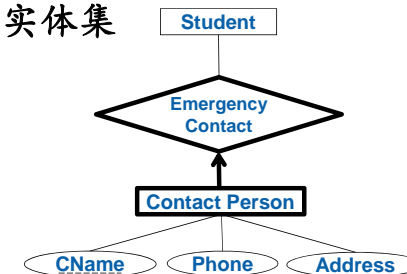
create table *Student* (

```
.....
ProvinceID integer not null,
Year integer not null,
foreign key (ProvinceID) references Province(ID)
);
```

ER图的联系集：普通Total Participation？

- 普通的完全参与，无法简单表达
- 只能用Check或Assertion来表达
 - 将在后续的课程中介绍
 - 代价很高
- 通常就不勉强了

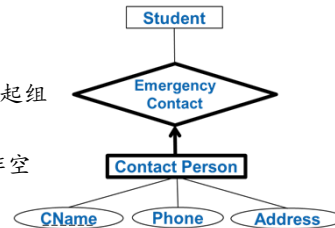
ER图的弱实体集



- 弱实体满足key constraint + total participation
 - 所以可以用前面的方法
- 不同点
 - 部分键
 - 删除主实体时，也删除相应的弱实体

ER图的弱实体集

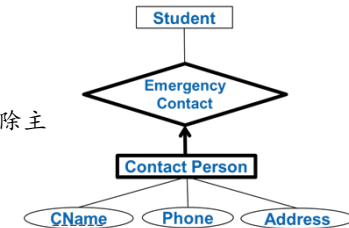
- 部分键必须和主实体的主键一起组成主键
- Primary key 保证了StudentID 非空



```
create table ContactPerson (
    StudentID integer,
    CName varchar(20),
    Phone integer, Address varchar(100),
    primary key (StudentID, CName),
    foreign key (StudentID) references Student(ID)
    on delete cascade
);
```

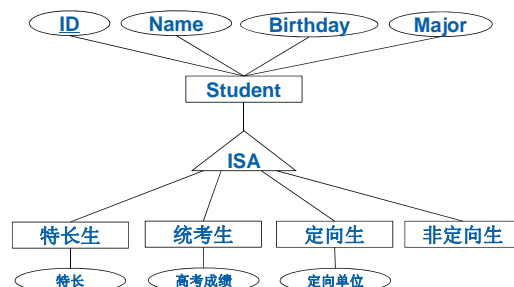
ER图的弱实体集

- StudentID 外键引用采用了 on delete cascade，保证在删除主实体的同时删除弱实体



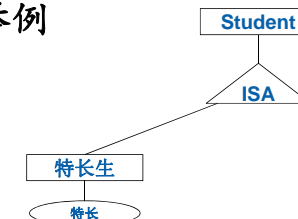
```
create table ContactPerson (
    StudentID integer,
    CName varchar(20),
    Phone integer, Address varchar(100),
    primary key (StudentID, CName),
    foreign key (StudentID) references Student(ID)
    on delete cascade
);
```

ER图类层次



- 把子实体创建为一个关系
- 在子实体的关系中引用父实体的主键

ER图类层次：举例



```
create table StudentWSpecialty (
    StudentID integer,
    Specialty varchar(20),
    primary key (StudentID),
    foreign key (StudentID) references Student(ID)
    on delete cascade
);
```

小结

- 关系模型复习
- SQL 初步
 - 表的定义、增删改
- 逻辑设计：ER图到关系模型