

空间数据库2018秋季

# 空间查询语言1-1025

陈斌

北京大学地球与空间科学学院

[gischen@pku.edu.cn](mailto:gischen@pku.edu.cn)



# 空间查询语言

- 关系代数
- SQL语言
- SQL语言的空间扩展
- 空间查询
- 对象关系数据库及其空间扩展



# 关系代数

- 什么是关系？
  - 关系是各个对象之间的联系和对应
- 最常见到是两组对象之间的联系和对应
  - 职员-部门的隶属关系
- 也有三组或更多对象之间的联系和对应
  - 供应商-工程-零件的供应关系
- 从集合论看关系
  - 有序组.....笛卡儿积.....关系

# 有序组

- 二元有序组，或者二元组(2-tuple)，或者序偶(ordered pairs)
  - 设 $a, b$ 为任意对象，称集合族 $\{\{a\}, \{a, b\}\}$ 为二元有序组，简记为 $\langle a, b \rangle$
  - 称 $a$ 为 $\langle a, b \rangle$ 的第一分量， $b$ 为第二分量
  - “有序”的含义？
- 定理：对于任意序偶 $\langle a, b \rangle, \langle c, d \rangle$ ， $\langle a, b \rangle = \langle c, d \rangle$ 当且仅当 $a = c$ 且 $b = d$



# 有序组

- 有序的含义
  - 当 $a \neq b$ 时,  $\langle a, b \rangle \neq \langle b, a \rangle$ , 但 $\{a, b\} = \{b, a\}$
  - 利用元素和集合的两个不同层次实现两个对象的有序排列
- $n$ 元有序组( $n$ -tuple) $\langle a_1, \dots, a_n \rangle$ 
  - 递归定义:  $n=2$ 时,  $\langle a_1, a_2 \rangle = \{\{a_1\}, \{a_1, a_2\}\}$
  - $n>2$ 时,  $\langle a_1, \dots, a_n \rangle = \langle \langle a_1, \dots, a_{n-1} \rangle, a_n \rangle$
  - $a_i$ 称为 $n$ 元组的第 $i$ 分量
- 定理: 对于任意 $n$ 元组 $\langle a_1, \dots, a_n \rangle = \langle b_1, \dots, b_n \rangle$ 当且仅当 $a_1 = b_1, \dots, a_n = b_n$

# 集合的笛卡儿积

- 对任意集合 $A, A_2, \dots, A_n$ ,  $A_1 \times A_2$ 称作集合 $A_1, A_2$ 的笛卡儿积, 定义如下:
  - $A_1 \times A_2 = \{ \langle u, v \rangle \mid u \in A_1, v \in A_2 \}$
  - $A_1 \times A_2 \times \dots \times A_n = (A_1 \times A_2 \times \dots \times A_{n-1}) \times A_n$
- 例子:  $A = \{1, 2\}$ ,  $B = \{a, b\}$ ,
  - 则 $A \times B$ 等于 $\{ \langle 1, a \rangle, \langle 1, b \rangle, \langle 2, a \rangle, \langle 2, b \rangle \}$
  - $B \times A$ 等于 $\{ \langle a, 1 \rangle, \langle a, 2 \rangle, \langle b, 1 \rangle, \langle b, 2 \rangle \}$
  - $A \times \emptyset = \emptyset \times A = \emptyset$
  - $R^2 = \{ \langle x, y \rangle \mid x, y \text{ 是实数} \}$ ,  $R^2$ 为笛卡儿平面,  $R^3$ 为三维笛卡儿空间
  - 一般来说:  $A \times B \neq B \times A$ ,  $A \times (B \times C) \neq (A \times B) \times C$

# 集合的笛卡儿积

- 定理：对于任意有限集合 $A_1, \dots, A_n$ ，有 $|A_1 \times \dots \times A_n| = |A_1| * \dots * |A_n|$



# 关系的基本概念

- 采用二元组或者多元组的集合来表示关系
  - $ED = \{ \langle \text{张三}, \text{人事部} \rangle, \langle \text{李四}, \text{销售部} \rangle, \langle \text{王五}, \text{技术部} \rangle \}$
  - $SPJ = \{ \langle \text{公司甲}, \text{大楼}, \text{水泥} \rangle, \langle \text{公司甲}, \text{公路}, \text{水泥} \rangle, \langle \text{公司乙}, \text{大楼}, \text{钢筋} \rangle, \langle \text{公司丙}, \text{公路}, \text{沥青} \rangle \}$
- $R$ 称为集合 $A_1, A_2, \dots, A_{n-1}$ 到 $A_n$ 的 $n$ 元关系，如果 $R$ 是 $A_1 \times A_2 \times \dots \times A_n$ 的一个子集。
  - 当 $A_1 = A_2 = \dots = A_{n-1} = A_n$ 时，也称 $R$ 为 $A$ 上的 $n$ 元关系



# 关系的例子

- 自然数的相等关系  $E_N = \{ \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 2 \rangle, \dots \}$ （列举法）
- 整除关系  $D = \{ \langle x, y \rangle \mid x \text{ 整除 } y \}$ （描述法）
- 小于关系  $L$ ：归纳法
  - 基础条款：  $\langle 0, 1 \rangle \in L$
  - 归纳条款：若  $\langle x, y \rangle \in L$ ，则  $\langle x, y+1 \rangle \in L$ ，  $\langle x+1, y+1 \rangle \in L$
  - 终极条款（略）

# 关系的基本运算

- R和S为A到B的二元关系,  $R, S \subseteq A \times B$
- 并:  $R \cup S = \{ \langle x, y \rangle \mid xRy \vee xSy \}$
- 交:  $R \cap S = \{ \langle x, y \rangle \mid xRy \wedge xSy \}$
- 差:  $R - S = \{ \langle x, y \rangle \mid xRy \wedge \neg xSy \}$
- 补:  $R^- = \underline{A \times B - R} = \{ \langle x, y \rangle \mid \neg xRy \}$ 
  - 并不是全集U-R, 而是全关系与R的差

# 用关系来描述实体和联系

- 实体=属性值的有序集合
  - 具有 $n$ 个属性的实体用 $n$ 元有序组描述
  - 所有属性值都相同的实体看作同一个实体
  - 实体的集合可以用关系来表示
    - 即所有属性对应的取值域集合 $A_1, A_2, \dots, A_n$ 的笛卡儿积 $A_1 \times A_2 \times \dots \times A_n$ 的子集
- 例如
  - 实体：<1001,张三,北京>
  - 实体集合：{<1001,张三,北京>, <1002,李四,上海>}



# 用关系来描述实体和联系

- 联系=实体+联系的属性值的有序集合
  - 用实体的标识属性（ID）代表实体
  - 联系也可以用关系来表示
- 例如：
  - 人员-部门联系<人员ID,部门ID,在部门年数>
    - {<1001,HR,0.5>,<1002,HR,3>,<1002,MGR,2>}



# 关系代数

- 什么是代数？
- 算术arithmetic
  - 研究整数、有理数、实数和复数的加、减、乘、除等运算
- 代数algebra
  - 算术的一般化，允许用字母等符号来代替数进行运算
- 代数结构algebraic structure
  - 在一个对象集合上定义若干运算，并设定若干公理描述运算的性质

# 代数结构

- 运算operator

- $S^n$ 到 $S$ 的一个函数，称为 $n$ 元运算（隐含了封闭性）
- 常用 $*$ 表示二元运算， $*(x,y)$ 常记做 $x*y$
- 常用 $\Delta$ 表示一元运算

- 二元运算的一般性质

- 结合律，如果二元运算满足：
  - $\forall x \forall y \forall z (x, y, z \in S \rightarrow x*(y*z) = (x*y)*z)$
- 交换律，如果满足
  - $\forall x \forall y (x, y \in S \rightarrow x*y = y*x)$
- $*$ 运算对 $\#$ 运算满足分配律
  - $\forall x \forall y \forall z (x, y, z \in S \rightarrow x*(y\#z) = (x*y)\#(x*z))$

# 代数结构：例子

- 加法、乘法是自然数集合上的二元运算
- 减法、除法不是自然数集合上的二元运算
- 除法甚至不是有理数、实数集合上的二元运算（除0无意义）
- 加法、乘法满足结合律、交换律；减法不满足结合律、交换律
- 乘法对加法、减法满足分配律



# 代数结构

- 代数结构的定义
  - 非空集合 $S$ ，称作代数结构的载体
  - 载体 $S$ 上的若干运算
  - 一组刻画载体上各运算性质的公理
- 例子
  - $\langle \mathbb{N}, + \rangle$ 是一个代数结构
  - 所有 $2 \times 2$ 实数矩阵 $M$ ，矩阵乘法 $*$ ， $\langle M, * \rangle$
  - $\langle \rho(A), \cup, \cap, \sim \rangle$ ， $A$ 幂集，并、交、补运算，是一个代数结构



# 关系代数Relation Algebra

- 是一个代数结构
- 运算对象是：关系；
- 六种基本运算
  - 选择select：得到子关系
  - 投影project：生成新的元数更少的关系
  - 笛卡儿积cross production：生成元数更多的关系
  - 并union：关系并集
  - 差difference：关系差集
  - 交intersection：关系交集

# 关系数据库回顾

- E.F.Codd于70年代初提出关系代数理论，他因此获得1981年的ACM图灵奖
- 关系理论是建立在集合代数理论基础上的，有着坚实的数学基础
- 早期代表系统
  - System/R：由IBM研制
  - INGRES：由加州Berkeley分校研制
- 目前主流的关系数据库管理系统
  - Oracle, Informix, SQL Server, DB2
  - MySQL, PostgreSQL
  - Access, Foxpro, Foxbase

# ●关系模型：数据结构

- 单一的数据结构——关系
- 实体集、联系都表示成关系

DEPT(D# , DN , DEAN)

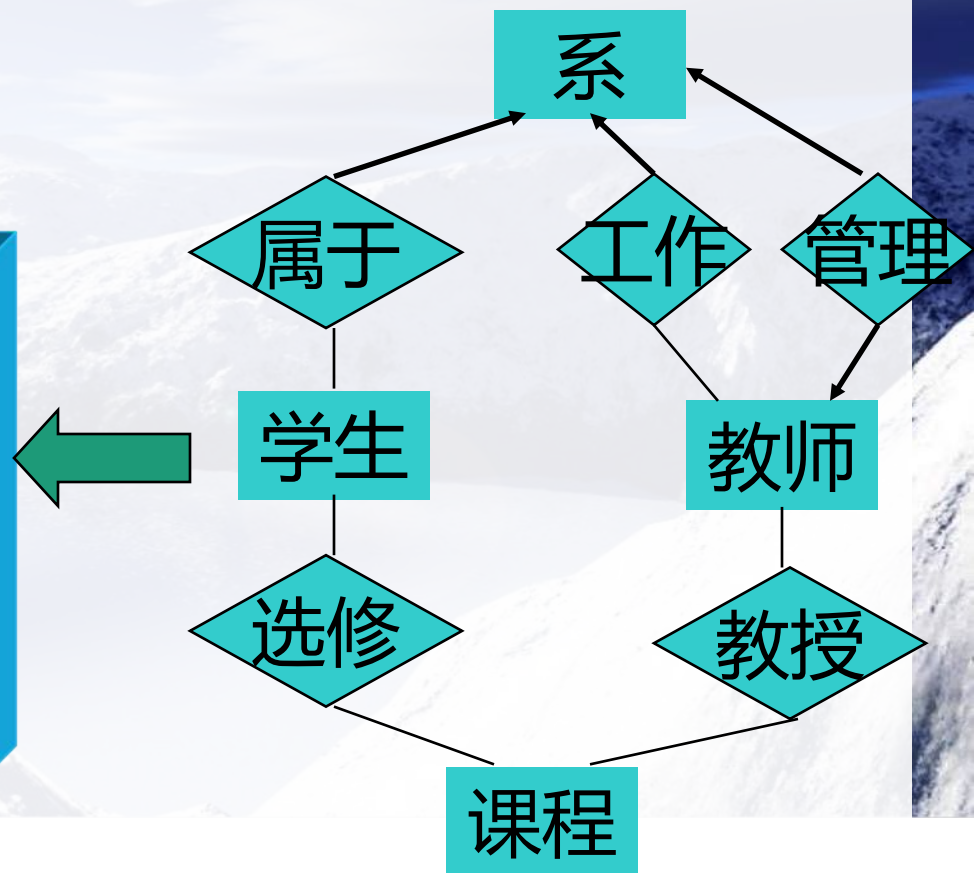
S(S# , SN , SEX , AGE , D#)

C(C# , CN , PC# , CREDIT)

SC(S# , C# , SCORE)

PROF(P# , PN, D# , SAL)

TEACH(P# , C#)



# 关系模型

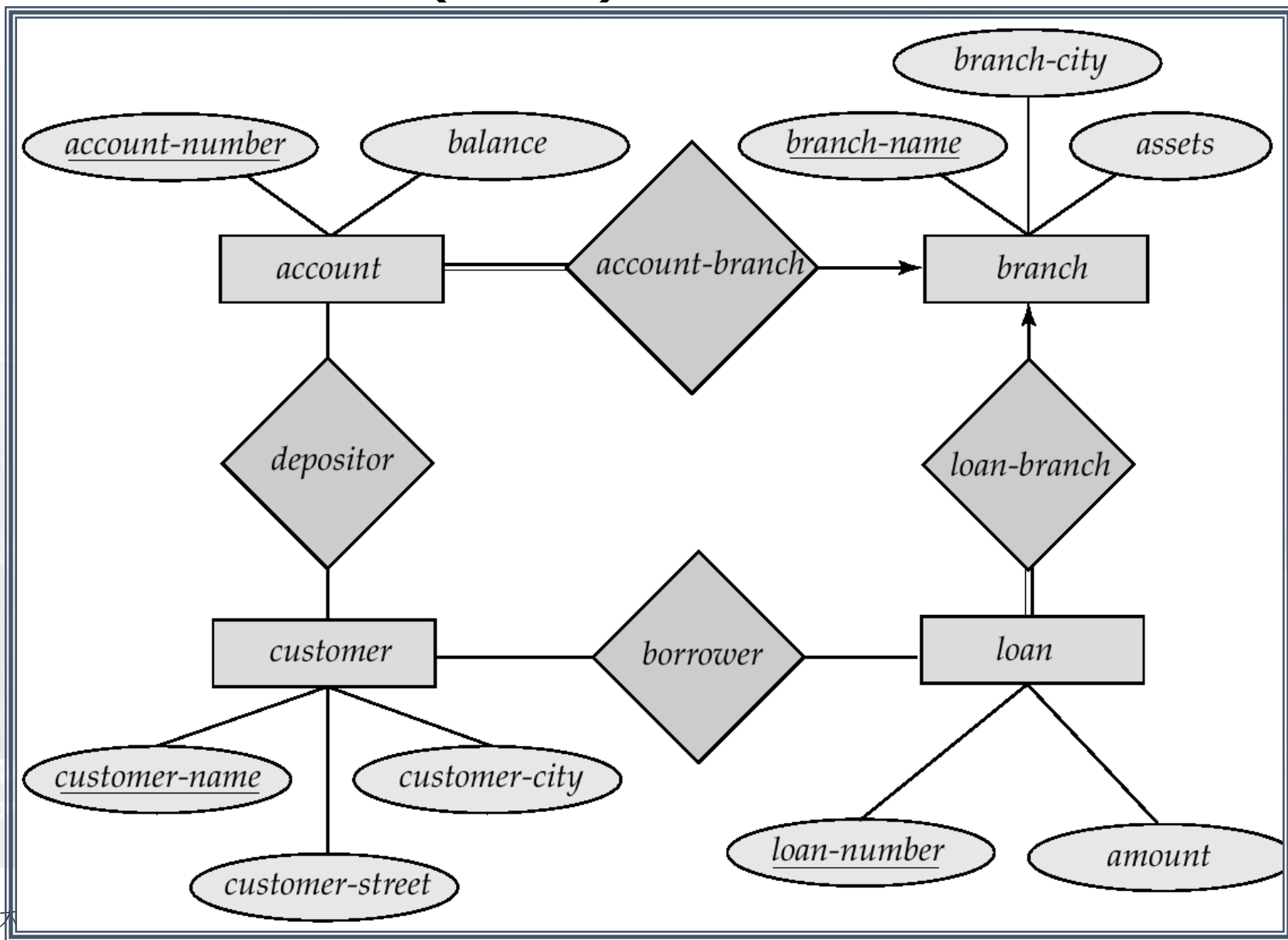
- 候选码 (Candidate Key)
  - 关系中的一个属性组，其值能唯一标识一个元组。若从属性组中去掉任何一个属性，它就不具有这一性质了，这样的属性组称作候选码
    - 如DEPT中的D#，DN都可作为候选码
  - 任何一个候选码中的属性称作主属性
    - 如SC中的S#，C#



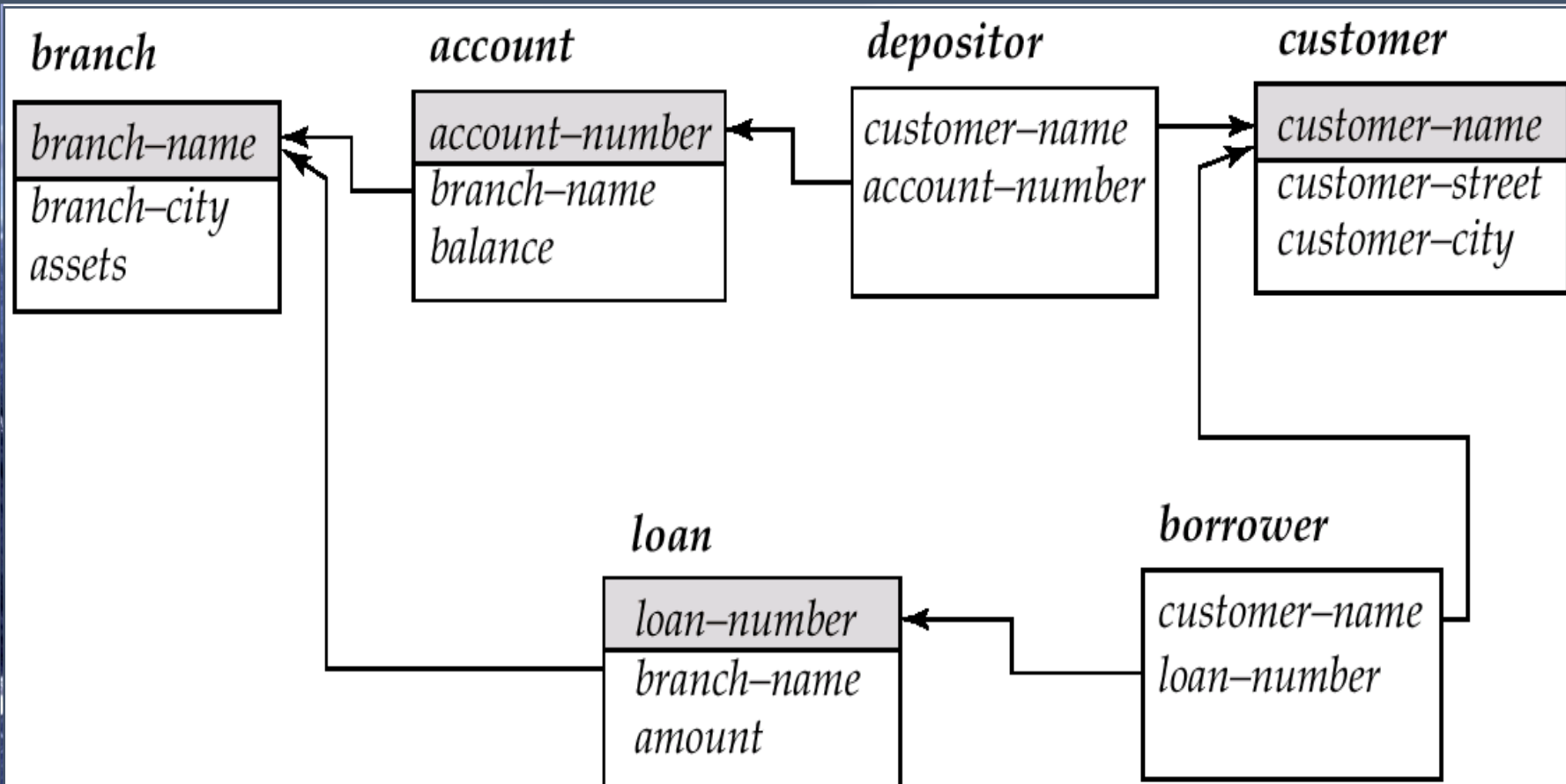
# 关系模型

- 主码 (Primary Key)
  - 进行数据库设计时, 从一个关系的多个候选码中选定一个作为主码
    - 如可选定D#作为DEPT的主码
- 外部码 (Foreign Key)
  - 关系R中的一个属性组, 它不是R的码, 但它与另一个关系S的码相对应, 则称这个属性组为R的外部码
    - 如S关系中的D#属性

# 关系模型（图）



# 关系模型（图）



# 关系模型：关系模式

- 关系的描述称作关系模式，包括关系名、关系中的属性名、属性向域的映象、属性间的数据依赖关系等，记作 $R(A_1, A_2, \dots, A_n)$
- 属性向域的映象一般直接说明为属性的类型、长度等
- 某一时刻对应某个关系模式的内容(元组的集合)称作关系
- 关系模式是型，是稳定的；关系是某一时刻的值，是随时间不断变化的



# 关系模型：关系数据库

- 其型是关系模式的集合，即数据库描述，称作数据库的内涵(Intension)
- 其值是某一时刻关系的集合，称作数据库的外延(Extension)

# 关系模型：关系操作

- 关系操作是集合操作，操作的对象及结果都是集合，是一次一集合（Set-at-a-time）的方式而非关系型的数据操作方式是一次一记录（Record-at-a-time）
- 关系操作可以用关系代数和关系演算两种方式来表示，它们是相互等价的  
如用关系代数来表示关系的操作，可以有选择、投影、连接、除、交、差、并等

# 关系模型：关系模式完整性

- 实体完整性
  - 关系的主码中的属性值不能为空值
  - 空值：不知道或无意义
  - 意义：关系对应到现实世界中的实体集，元组对应到实体，实体是相互可区分的，通过主码来唯一标识，若主码为空，则出现不可标识的实体，这是不容许的



# 关系模型：关系模式完整性

- 参照完整性

- 如果关系 $R_2$ 的外部码 $F_k$ 与关系 $R_1$ 的主码 $P_k$ 相对应，则 $R_2$ 中的每一个元组的 $F_k$ 值或者等于 $R_1$ 中某个元组的 $P_k$ 值，或者为空值
- 意义：如果关系 $R_2$ 的某个元组 $t_2$ 参照了关系 $R_1$ 的某个元组 $t_1$ ，则 $t_1$ 必须存在
- 例如关系 $S$ 在 $D\#$ 上的取值有两种可能
  - 空值，表示该学生尚未分到任何系中
  - 若非空值，则必须是 $DEPT$ 关系中某个元组的 $D\#$ 值，表示该学生不可能分到一个不存在的系中



# 关系模型：关系模式完整性

- 用户定义的完整性
  - 用户针对具体的应用环境定义的完整性约束条件
  - 如S#要求是8位整数，SEX要求取值为“男”或“女”
- 系统支持
  - 实体完整性和参照完整性由系统自动支持
  - 系统应提供定义和检验用户定义的完整性的机制

# 关系模型（例）

供应商关系S（主码是“供应商号”）

供应商号	供应商名	所在城市
B01	红星	北京
S10	宇宙	上海
T20	黎明	天津
Z01	立新	重庆

零件关系P（主码是“零件号”，外码是“供应商号”）

零件号	颜色	供应商号
010	红	B01
312	白	S10
201	蓝	T20

今要向关系P中插入新行，新行的值分别列出如下。哪些行能够插入？

- A. ( '037' , '绿' , null )
- B. ( null , '黄' , 'T20' )
- C. ( '201' , '红' , 'T20' )
- D. ( '105' , '蓝' , 'B01' )
- E. ( '101' , '黄' , 'T11' )

# 关系数据语言基本概念

- 关系数据语言的特点

- 一体化

- 一般关系系统的数据语言都同时具有数据定义、数据操纵和数据控制语言，而不是分为几个语言。对象单一，都是关系，因此操作符也单一。而非关系型系统，如DBTG，有对记录的操作，有对系的操作

- 非过程化

- 用户只需提出“做什么”，无须说明“怎么做”，存取路径的选择和操作过程由系统自动完成

- 面向集合的存取方式

- 操作对象是一个或多个关系，结果是一个新的关系（一次一关系）。非关系系统是一次一记录的方式



# 抽象的查询语言

- 关系代数
  - 用对关系的运算来表达查询，需要指明所用操作
- 关系演算
  - 用谓词来表达查询，只需描述所需信息的特性
- 元组关系演算
  - 谓词变元的基本对象是元组变量
- 域关系演算
  - 谓词变元的基本对象是域变量



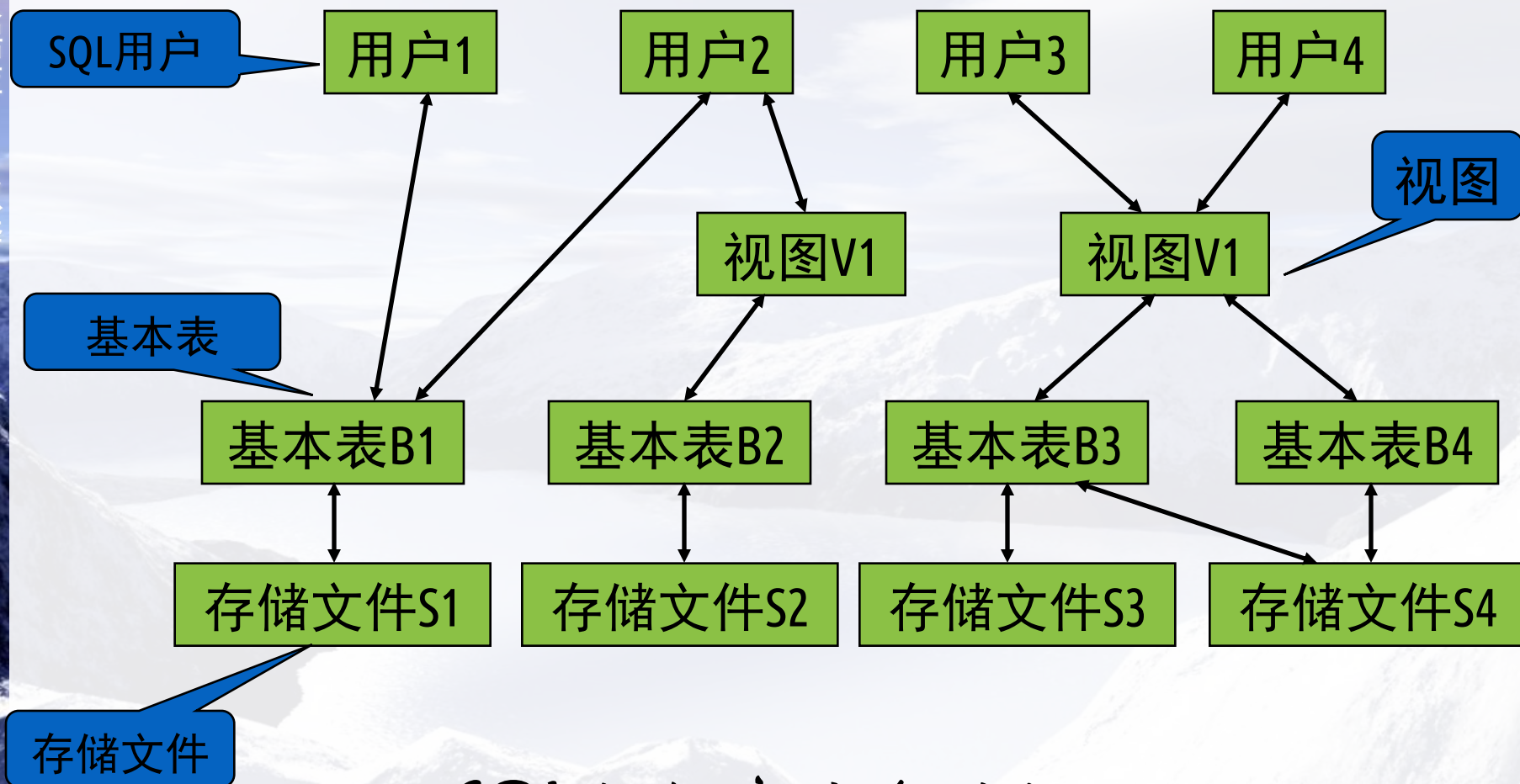
# 具体系统中的实际语言

- SQL (Structural Query Language)
  - 介于关系代数和关系演算之间，由IBM公司在研制System R时提出
- QUEL (QUERy Language)
  - 基于Codd提出的元组关系演算语言ALPHA，在INGRES上实现
- QBE (Query By Example)
  - 基于域关系演算，由IBM公司研制

# SQL概述：历史

- SQL: Structured Query Language
- 1974年，由Boyce和Chamber提出
- 1975-1979年，在System R上实现，由IBM的San Jose 研究室研制，称为Sequel

# SQL概述（图）



## SQL数据库体系结构

# SQL概述-标准化

- 有关组织
  - ANSI(American National Standard Institute)
  - ISO(International Organization for Standardization)
- 有关标准
  - SQL-86
  - SQL-89: “具有完整性增强的数据库语言SQL”, 增加了对完整性约束的支持
  - SQL-92: “数据库语言SQL”, 是SQL-89的超集, 增加了许多新特性, 如新的数据类型, 更丰富的数据操作, 更强的完整性、安全性支持等。
  - SQL-3: 最新标准, 增加对面向对象模型的支持



# SQL概述-特点

- 一体化
  - 集DDL, DQL, DML, DCL于一体
  - 单一的结构---关系, 带来了数据操作符的统一
- 面向集合的操作方式set-at-a-time
  - 一次一集合, 数据操纵的对象和结果都是集合
- 高度非过程化
  - 用户只需提出“做什么”, 无须告诉“怎么做”, 不必了解存取路径
  - 减轻用户负担, 提高数据独立性

# SQL概述-特点

- 两种使用方式，统一的语法结构
  - SQL既是自含式语言（用户使用），又是嵌入式语言（程序员使用）
- 语言简洁，易学易用
  - SQL语言完成数据定义、操纵、控制和查询的核心功能只用了9个动词
    - CREATE, DROP, ALTER, SELECT, INSERT, UPDATE, DELETE, GRANT, REVOKE
  - SQL语言语法简单，接近自然语言，容易学习和使用

# SQL：数据定义语言DDL

- 数据定义的对象
  - 关系数据库的基本对象是基本表（table）
  - 并可以在基本表上定义视图（view）作为数据外模式
  - 定义索引（index）用于向系统提示多种存取路径，由系统自动选择，以加快查找速度。
  - （用户自定义数据类型）



# SQL：数据定义语言DDL

操作对象	操作方式		
	创建	删除	修改
基本表Table	CREATE TABLE	DROP TABLE	ALTER TABLE
视图View	CREATE VIEW	DROP VIEW	
索引Index	CREATE INDEX	DROP INDEX	



# SQL：数据定义语言DDL

- SQL中，任何时候都可以执行一个数据定义语句，随时修改数据库结构。
  - 而在非关系型的数据库系统中，必须在数据库的装入和使用前全部完成数据库的定义。若要修改已投入运行的数据库，则需停下一切数据库活动，把数据库卸出，修改数据库定义并重新编译，再按修改过的数据库结构重新装入数据
- 数据库定义随时修改
  - 不必一开始就完全合理
- 可进行增加索引、撤消索引的实验，检验其对查询效率的影响

# SQL：数据查询语言DQL

- select语句基本结构

- select  $A_1, A_2, \dots, A_n$  --查询输出  
from  $r_1, r_2, \dots, r_m$  --查询操作表  
where P --查询条件

- 与关系代数式的等价性

$$\prod_{A_1, A_2, \dots, A_n} (\sigma_p(r_1 \times r_2 \times \dots \times r_m))$$

- 示例：查询所有学生的姓名

- select name from student

# SQL：数据查询语言DQL

- 整个select语句的语义：
- 从from子句中指定的表名或者视图名所建立的连接关系中
- 按照where子句所指定的逻辑条件选取目标元组
- 再根据select子句所指定的目标列表表达式，选出目标元组中的属性值形成结果关系输出



# SQL：数据查询语言DQL

- group by <列名>{,<列名>} [having <条件表达式>]
  - 按照group by子句指定的分组属性列分组，每个组产生结果关系中的一条记录
  - 可以指定多个分组的属性列，而对于分组属性列以外的属性列，select子句中都需要作用以聚集函数才能输出
  - group by子句中的having子句则用来指定和聚集函数有关的输出条件，只有符合条件的分组记录才输出。



# SQL：数据查询语言DQL

- order by <列名> [asc|desc]{,<列名> [asc|desc]}
  - 输出结果关系按照order by子句中指定的输出属性列来排序，可以依次指定多个排序属性列
- 条件表达式构成
  - 包含各属性列和运算函数，以及用于各种数据类型值的比较符
  - 用逻辑联结词（NOT、AND、OR）联结表达式项，形成更加复杂的复合条件

# SQL：数据查询语言DQL

- 还提供了集合判定条件
- IN操作符判定一个值是否属于某个集合
- EXISTS操作符判定某个集合是否非空
- 而这里的集合都可以用显式枚举或者嵌套子查询来表示
  - where color in ('red', 'yellow', 'blue')
  - where exists(select id from part where color='red')

# SQL：数据查询语言DQL

- 提供了一阶谓词的基本表达和支持
- 可以在集合前面加上ANY或者ALL的前缀，分别表示集合中的任意一个元素和所有元素，并可以参与各种比较运算符的比较
- 例：找出平均成绩最高的学生
  - ```
select student_id
from student_course
group by student_id
having avg(score) >= all (select avg(score) from student_course
group by student_id)
```



# SQL：数据查询语言DQL

- 提供了集合运算符
- 用于联结多个SELECT语句，进一步生成新的关系
- 这些集合运算符包括并运算UNION，交运算INTERSECT和差运算EXCEPT
- 例：求选修了001号课程但是没有选修002号的学生
  - (select student\_id from student\_course  
where course\_no='001')  
**except**  
(select student\_id from student\_course  
where course\_no='002')



# SQL：数据查询语言DQL

- SELECT语句充分体现了SQL语言的特征
- 非过程化：仅需要指明查询结果所符合的条件，而不需要指定如何得到查询结果
- 面向集合：可以对整个元组集合进行操作，而不需要枚举和以某种次序遍历集合中的每个元组

# SQL：数据操纵语言DML

- SQL语言中数据操纵语句包括：
- 插入INSERT
- 更新UPDATE
- 删除DELETE
- 用于数据库内数据的维护，这三个语句也是非过程化和面向集合的

# SQL：数据控制语言DCL

- 关系数据库提供统一的数据控制功能
  - SQL语言提供了数据控制功能，能够在一定程度上保证数据库中数据的安全性、完整性，并提供了一定的并发控制及恢复能力
- 数据库的完整性
  - 数据库中数据的正确性和相容性
  - SQL语言定义完整性约束条件的语言成分主要体现在基本表、视图和索引的定义语句（CREATE TABLE/VIEW/INDEX）中，可以在表的定义中定义主码、外码、取值唯一和其它的属性列级和表级约束条件



# SQL：数据控制语言DCL

- 数据库的安全性
- 保护数据库，防止不合法的使用所造成的数据泄漏和破坏。
- 数据库系统中保证数据安全性的主要措施是进行存取控制
  - 规定不同用户对于不同的数据对象所允许执行的操作，
  - 控制各用户只能存取有权存取的数据
- SQL语言提供了由DBA和数据对象所有者决定的权限定义和收回的手段（grant/revoke）
  - 权限的对象可以定义到数据库、基本表、视图或属性列上



# SQL：数据控制语言DCL

- 并发控制和恢复：
- 当多个用户**并发**地对数据库进行操作的时候，对他们加以控制、协调，以保证并发操作能够正确按照每个用户的操作语义进行，并保持数据库的**一致性**
- 恢复则是在发生各种类型的故障和错误，数据库处于不一致状态时，将数据库**恢复**到一致的状态

# SQL：数据控制语言DCL

- SQL语言提供了并发控制和恢复的功能
- 定义事务：begin transaction
- 事务的提交：commit work
- 事务的回滚：rollback work

# 作业练习

- 阅读相关文档
  - <https://blog.csdn.net/a2011480169/article/details/72862308>
  - [https://blog.csdn.net/qq\\_39607437/article/details/79620383](https://blog.csdn.net/qq_39607437/article/details/79620383)
- 构造一个SQL模拟功能，对OSM数据进行管理和查询
  - 点数据（如GPS轨迹）
  - 导入点数据，保存为一个文本文件，每行一个点
  - 进行简单的查询