第一章 引言

1.数据：描述事物的符号记录。

2.数据库：长期储存在计算机内，有组织的，可共享的大量数据集合。

3.数据库管理系统（DBMS）：介于用户与操作系统之间的数据管理软件系统。

4.数据库系统（DBS）：由一个互相关联的数据集合和一组用以访问这些数据的程序组成。由数据库、数据库管理系统、应用程序、数据库管理员和用户组成。

5.数据库管理员（DBA）：数据库设计、规划、协调的人员，最高特权的用户。

6.管理信息系统（MIS）：办公系统、决策系统、生产系统和信息系统面向数量的执行系统、面向价值的核算系统、报告监控系统，分析信息系统、规划决策系统。

7.数据管理发展的三个阶段特点：人工管理阶段（没有相应的文件系统，数据不共享，数据不具独立性）、文件系统阶段（操作系统中有了文件系统，打开、读、写和关闭操作。查询、修改、排序等处理都须编程解决。不支持并发访问。数据缺少统一管理）、数据库系统阶段（DBMS 统一管理，减少冗余；并发访问数据并一致性，数据安全性，故障情况下数据一致性的恢复）。

8.DBMS 的演变

9.DBMS 的特点：

（1） 用户接口，非过程数据库语言

（2） 查询处理与优化策略。

（3） 并发控制，解决冲突。

（4） 恢复功能，数据一致性状态。

（5） 完整性约束检查，实体完整性、引用完整性、域完整性、用户定义完整性。

（6） 访问控制。访问权限，安全性问题。

10.DBMS 的组成:（1）数据定义语言(DDL) （2）数据操纵语言(DML。过程化 DML：给出具体查询过程，例如关系代数；非过程化 DML：只给出查询需求，例如关系演算、SQL)（3）数据库运行控制程序(DCL)

11.DBMS 的功能部件:查询处理器（DML 编译器、DDL 解释器、查询求值引擎）与存储管理器（权限及完整性管理器、文件管理器、缓冲管理器）。

12.数据模型：概念数据模型（现实世界的数据抽象，描述一个单位的概念化结构,与 DBMS 无关，如 E—R 模型 ）、逻辑数据模型（DBMS 层面上的数据形式。如层次模型（实体用结点表示，实体之间的联系用树表示)、网状模型（实体由记录表示，记录间联系由系（set）表示，系是一个指针)、关系模型（实体由元组（记录）表示，实体集由关系（表）表示，实体之间的联系也通过关系表示))、物理数据模型（数据存储结构，如物理块、指针、索引方式）。

13.数据视图

逻辑数据模型、数据模式（数据模型下，对数据的逻辑结构的描述）、实例（数据模式下，数据的全体）。

14.“三个模式”：

逻辑层：称为模式或数据模式，数据库全体数据的逻辑结构和特征的描述。DBA 所看到的数据形式。

视图层：称为外模式、子模式，数据库部分数据的逻辑结构和特征的描述。用户所看到的数据形式。是模式的一个局部，称为视图。

物理层：称为内模式，磁盘存储方式中物理结构的描述，是 DBMS 设计人员看到的数据形式。

映射程序：外模式/模式映象：外模式和模式之间的对应关系，映象定义通常包含在各外模式中；模式/内模式映象：数据逻辑结构与存储结构之间的对应关系。

15.“数据独立性”：
物理独立性：指数据的物理结构的改变，修改模式/内模式映象，不影响数据库的逻辑结构，从而不致引起应用程序的变化。
逻辑独立性：数据库逻辑结构的改变，修改外模式/模式映象，不需要修改相应的应用程序。
16．文件系统的缺点：
数据的冗余和不一致性
数据访问困难
数据孤立
完整性问题
更新的原子性问题
多用户的并发访问
安全性问题
第二章 关系模型
第一节 关系代数
1.关系：二维表，由关系名标识。 关系是无序的。关系式笛卡尔积中有意义的子集。
关系的性质：
(1)每列同类型，同域，不同的列可以同域。
(2)列的顺序无关，交换列的次序，仍是同一个表
(3)行的顺序无关，交换行的次序，仍是同一个表
(4)任意两行不能全同，码的本质
(5)任意两列的列名不能相同，属性区分
(6)每一分量是不可分的数据项，关系数据库本质。
2.元组：表中的一行，称之为 n 元组，记录。
3.属性：表中的一列，属性名标识，字段。属性可能的取值范围叫做属性的域。域是原子的。
4.完整性约束：域完整性约束（属性值应是域中的值，属性的值能否为 null，由语义决定。域完整性约束是在确立关系模式时规定的，由 DBMS 负责检查）、实体完整性约束（每个关系应有一个主码，主码的值不能为 null，这就是实体完整性约束。如果在关系模式中说明了主码，DBMS 可以进行这项检查）、引用完整性约束（不同关系之间或同一关系的不同元组间的约束）、用户定义的完整性约束（用户定义的完整性约束是针对某一具体数据库的约束条件，由具体应用要求决定）。
5. 码：能够唯一确定一个元组的最小属性集
超码：包含码的属性集 。若 K 是一个超码，则 K 的任意超集也是超码。
候选码：最小的超码成为候选码。
主码：定义表时指定一个候选码，在一个关系中用来区分不同的元组。
外码：本关系的属性集，另一关系的主码。参照完整性约束：在参照关系中任意元组在特定属性上的取值必然等于被参照关系中某个元组在特定属性上的取值。
全码：表中找不出码，所有属性组成
主属性：任一候选码中的属性
非主属性：不在任何一个候选码中
6. 关系模式：关系名和属性名。
第三章 SQL
1.创建表结构 create table
2.删除表：drop table
3.更改表： alter table r add A D;

alter table r drop A;
alter table r modify A D;
4.建立索引：create index 索引名 on 表名
5.查询：select
（1）读取 FROM 子句的表或视图做笛卡尔积。
（2）WHERE 子句找出满足条件表达式的元组
（3）按 GROUP 子句指定列名分组，值相等的元组为一组，每个组产生结果表中的一条记录。可在每组中用集函数。如果 GROUP 子句带 HAVING 短语，则只有满足指定条件的组才予输出。
（4）按 select 子句中给出的列名或表达式求值输出
（5）按 ORDER 子句列名的值升或降序
缺省为保留重复元组，也可用关键字 all 显式指明。若要去掉重复元组，可用关键字 distinct 或 unique 指明。
6.更名运算
7.找出满足条件的字符串：列名 like “字符串”
%”匹配零个或多个字符
“__”匹配任意单个字符
8. 元组显示顺序
order by 列名 [asc | desc]
9.集合操作
集合并：union
集合交：intersect
集合差：except(minus)
注意：集合自动去除重复元组，如果需要保留，要用 all 显式说明。
10.分组和聚集函数
注意：group by 后面出现的那个属性也必须在 select 后面出现，否则报错。
Having 是对已经分组处理的结果进行条件选择，而不是对原始数据选择后再分组。
Distinct 去掉重复，
先 where,再 group，再 having
聚集函数忽略 null，count（*）除外
空值用 is 判断，where amount is null
11.空值测试
找出年龄值为空的老师姓名
select PNAME from PROF where PAGE is null
不可写为 where PAGE = null
12.嵌套子查询
<1>In 子查询：括号内外类型要匹配
<2>some/all 子查询：
表达式:比较运算符 some （子查询）
表达式的值至少与子查询结果中的一个值相比满足比较运算符
=SOME 等价于 IN
表达式：比较运算符 all （子查询）
表达式的值与子查询结果中的所有的值相比都满足比较运算符
<>ALL 等价于 NOT IN

<3>测试集合是否为空：exists（子查询）
in 后的子查询与外层查询无关，每个子查询执行一次，而 exists 后的子查询与外层查询有关，需要执行多次，称之为相关子查询。
Eg：列出选修了 001 号和 002 号课程的学生的学号：
select SNO from SC as SC1 where SC1.CNO=001 and exists(select SNO from SC as SC2 where SC2.CNO=002 and SC2.SNO=SC1.SNO)
用 exists 表示超集：若 A 为 B 的超集，则 NOT EXISTS(B EXCEPT A)为 TRUE
表示“全部”概念：
列出选修了全部课程的学生姓名：
1） 任意：not exists(not exists)不存在一门课该同学没有选过：select SNAME from S where not exists (select CNO from C where not exists (select * from SC where SC.CNO=C.CNO and SC.SNO=S.SNO))
2） 超集:not exists(B except A)所求学生的选课集为所有课程的超集：select SNAME from S where not exists((select CNO from C)EXCEPT (select CNO from SC where SC.SNO=S.SNO))
3） ÷：not in(not in) 所求学生不在如下集合中：学生学号与任一课程组合不包含在 SC 中：select SNAME from S where SNO NOT IN (select SNO from C,S where (SNO,CNO) NOT IN(select SNO,CNO from SC))
<4>测试集合是否存在重复元组：unique（子查询）如果子查询结果中没有重复元组，则返回 true
Eg：找出只教授一门课程的老师姓名：
select PNAME from PROF where unique(selec PNO from PC where PC.PNO=PROF.PNO)
找出至少选修了两门课程的学生姓名
select SNAME from S where not unique (select SNO from SC where SC.SNO=S.SNO)
13.SQL 的数据修改功能
Insert into 表名 values (, , ,)
Delete from 表名 where …
Update 表名 set … =… where …
第四章 中级 SQL
1.连接表达式（了解，不考）
内连接：舍弃不匹配的元组
左外连接：内连接+左边失配的元组（缺少的右边关系属性用 null）
右外连接：内连接+右边失配的元组（缺少的左边关系属性用 null）
全外连接：内连接 + 左边失配的元组（缺少的右边关系属性用 null）+ 右边失配的元组（缺少的左边关系属性用 null）
2.视图（读懂即可，不必会写）
视图提供了一个对某些用户从视图中隐藏某些数据的机制。任何不是逻辑模型的一部分，但作为虚关系对用户可见的关系称为视图。
Create view ×××(属性 1，属性 2，…) as (select …)
• 视图更新
SQL 视图可更新的条件：
1>from 子句中只有一个表（关系）
2>select 中没有广义投影、聚集、distinct，只包含关系的属性名
3>任何没有出现在 select 子句中的属性可以取空值
4>不含 group by 或 having 子句

3.完整性约束
数据的正确性和相容性
·在属性值上的约束
非空约束：要求某属性取值不能为空 SNAME CHAR(8) NOT NULL
基于属性的检查子句 CHECK(AGE>15)
域约束子句 用 CREATE DOMAIN 定义域时，可以出现 CHECK
CREATE DOMAIN AGE SMALLINT
CHECK((VALUE >= 15) AND VALUE <= 25))
元组在所列属性上取值不能相同：unique(属性 1，属性 2，……)
·主码约束
主码值不允许空，也不允许出现重复
主码定义形式:
主码子句：PRIMARY KEY(Sno)
主码短语：Sno CHAR(4) PRIMARY KEY
·外码约束（参照完整性）
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。
如果关系 R2 的外部码 Fk 与关系 R1 的主码 Pk 相对应，则 R2 中的每一个元组的 Fk 值或者等于 R1 中某个元组的 Pk 值，或者为空值。
意义：如果关系 R2 的某个元组 t2 参照了关系 R1 的某个元组 t1，则 t1 必须存在。
例如:关系 S 在 Dno 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 Dno 值，表示该学生不可能分到一个不存在的系中。
定义形式：
在 SC 表中，定义 FOREIGN KEY (Sno) REFERENCES S(Sno)
·全局约束
全局约束涉及多个属性间的或多个关系间的联系
·对约束的命名、撤消和添加
命名 CONSTRAINT 约束名 <约束条件>
示例 Sno CHAR(4) CONSTRAINT S_PK PRIAMRY KEY
AGE SAMLLINT CONSTRAINT AGE_VAL CHECK(AGE >= 15 AND AGE <= 25)
撤消用 alter …drop…
添加用 alter …add…a
示例 alter table S drop constraint S_PK
alter table SC add constraint SC_CHECK check(Sno in select Sno from S)
4.断言（能看懂，会读）
断言是一个谓词，表达数据库总应该满足的条件
一旦定义了断言，系统验证其有效性，并且对每个可能违反该断言的更新操作都进行检查。这种检查会带来巨大的系统负载，因此应该谨慎使用断言。
create assertion ... check ...
对断言“所有 X, P(X)”，是通过检查“not exists X，非 P(X)”来实现的。
不允许男同学选修张老师课程
create assertion ASSE2 check (not exists (select * from SC where Cno in (select Cno from C where TEACHER='张')and Sno in(select Sno from S where SEX='M')))
每门课最多 50 名男同学选修 create assertion ASSE1 check(50 >= all (select

count(SC.Sno)from S,SC where S.Sno = SC.Sno and SEX = ‘M’group by Cno)))

5.授权

• 权限图

• 结点是用户，根结点是 DBA，有向边 Ui→Uj，表示用户 Ui 把某权限授给用户 Uj

一个用户拥有权限的充分必要条件是在权限图中有一条从根结点到该用户结点的路径。

• 授权：grant …（如 select）on 表名 to …

后面加上 with grant option 表示获得权限的用户可以把权限再授予其它用户。

回收权限：revoke … on 表名 from … 收回权限时，若该用户已将权限授予其它用户，则也一并收回。授权路径的起点一定是 DBA

第五章 高级 SQL

1.嵌入的 SQL 语句

• 以 EXEC SQL 开始，分号结束。可以使用宿主语言的变量，不过要在前面加上冒号以区别于 SQL 变量。

• 宿主变量的声明

声明为通常的 C 变量，并将其放在下列标识语句之间

EXEC SQL BEGIN DECLARE SECTION

EXEC SQL END DECLARE SECTION

• 游标

在查询结果的记录集合中移动的指针

若一个 SQL 语句返回单个元组，则不用游标

若一个 SQL 语句返回多个元组，则使用游标

insert 语句

EXEC SQL insert into PROF values (:prof_no, :prof_name ,:salary , :dept_no , :salary);

delete 语句

EXEC SQL delete from PROF where PNO >:prof_no;

update 语句

EXEC SQL update PROF set SAL =:salary where PNO = : prof_no;

定义与使用游标的语句

declare 游标名 cursor for select..

open 打开游标，执行对应的查询

fetch 游标名 into 宿主变量表

close 关闭游标

2.触发器：触发器是一条语句，当对数据库做修改时，它自动被系统执行

触发器的定义：

指明什么条件下触发器被执行

指明触发器执行的动作是什么

触发器的作用：示警；满足特定条件时自动执行某项任务

触发器事件：Insert、 delete、update

职工工资增幅不得超过 10%

```
create trigger RAISE_LIMIT
after update of SAL on EMP
    referencing new row as nrow old row as orow
    for each row
    when (nrow.SAL > 1.1 * orow.SAL)
```

begin atomic signal SQLSTATE(‘7500’ ,’Salary increase 10%’)
end
3.OLAP 联机分析处理系统
OLTP 采用 E-R 模型和面向应用的数据库设计
OLAP 采用**星型或雪花模型**和面向主题的**多维数据立方体**设计
OLTP 主要功能是 DBMS
OLAP 主要功能是检索查询工具，多维数据分析工具、统计分析及数据挖掘工具。
• OLAP 的操作：
上卷（roll up)操作
沿维的概念分层向上攀升，然后在数据立方体上进行的聚集。
下钻（drill down)操作
沿维的概念分层向下延伸，然后在大的数据立方体上进行小立方体上值的细化。
切片（slice)操作
在数据立方体的一个维上选择。
切块（dice)操作
在数据立方体的两个或多个维上选择。
转轴（pivot)操作
转动数据的视角，一种目视操作
4.一些名词
API：用于程序和数据库服务器之间的交互
JDBC：定义了 Java 程序连接数据库服务器的应用程序接口
ODBC:开放数据库互连。定义了一个 API，应用程序用它来打开一个数据库连接，发送查询和更新，以及获取返回结果。
第六章 形式化关系查询语言
1.关系代数：过程化语言
2.基本运算：选择运算、投影运算（自动去重）、并运算（并运算有意义：两个关系属性数目必须相同，属性的域必须相同）、差运算（有意义的要求同并运算）
分配律：投影和并可以分配，投影和差不可分配
笛卡尔积：$r \times s \equiv \{tr \frown ts \mid tr \in r \wedge ts \in s\}$
说明:结果为关系是(n + m)目，前 n 列是 r 的元组，后 m 列是 s 的元组，区别相同属性加表名为前缀，元组连串，穷举性结合

更名: ρ x（A1，A2，…，An）(e) 赋名 x，属性更名为 A1，A2，…An

3.附加的关系代数运算：集合交运算 $r \cap s=r-(r-s)$
• 自然连接
• 赋值运算：表名←关系代数表达式
• 外连接：
1）左外连接
左侧表的元组全保留，与右侧表不匹配的元组，用 null 填充。
2）右外连接
右侧表的元组全保留，与左侧表不匹配的元组，用 null 填充。
3）全外连接
左右侧表的元组全保留，不匹配的元组，用 null 填充
• 除运算（除运算是包含判断，对语义“全部”、“所有”的处理）

• 广义投影：Π表达式 as 属性名，，．．．，表达式 as 属性名(e)
说明：e 是中间表，表达式没有属性名，用 as 短语缺确定一个属性名。
例：Π姓名，年龄+1 as 虚岁（学生）结果表的属性名为姓名，虚岁

• 聚集函数：𝒢聚集函数（属性）（e）

聚集函数有 sum(属性名)，avg(属性名)，min(属性名)，max(属性名)，count(属性名)，count-distinct(属性名)，结果是单属性单行的中间表，属性名：“聚集函数名（属性名）”，
若重新确定属性名，用 as 子句。

𝒢聚集函数（属性）as 属性名（e）

• 分组聚集：分组属性 1，分组属性 2，…𝒢聚集函数 1（属性 1），聚集函数 2（属性 2），…（e）
• 数据库的修改
删除：差运算与赋值运算实现删除运算。r←r - e
插入：并运算与赋值运算实现插入运算。r←r∪e
更新：广义投影与赋值运算实现更新运算。
第二节 关系演算
关系演算：用一阶谓词演算表示关系的操作叫关系演算

第七章 E-R 图
1.实体：客观存在并可相互区分的事物叫实体。
实体集：同型实体的集合称为实体集，如全体学生。
实体型：实体名和属性名组成实体型。对关系模型就是关系模式。例如，
学生(学号，姓名，性别，年龄，系别，入学时间)，是学生实体型。
属性：实体特征，通过其值区分不同的实体
实体对应表中一个元组（记录）
属性对应表中一个字段（数据项）：简单属性，复合属性，单值属性，多值属性（双椭圆），派生属性（虚椭圆），Null 属性
实体集对应一个表
2. 码(Key)
• 能唯一标识实体的属性或最小属性集，也称为实体键。具有唯一性的特点，即给出码属性的一个值，到实体集中或只找出一个实体或一个也找不到
•如果有多个码，每个码都称为侯选码，从中指定一个作为主码（在 E-R 图中用下划线标明）。例如职工实体集中，职工号为候选码，若姓名不重，也是候选码。
• 超码：候选码的超集，比候选码有多余的属性，也能唯一标识一个实体。
• 任一表都有码。
3.联系集：实体之间的相互关联，用联系名标识，分为两（多）实体集之间实体的关联，同一实体集内部实体之间的关联
联系的属性:实体有属性，联系也可以有属性。例如，选课联系中可以有成绩属性。
联系集:参与联系的实体组成的集合。联系一般也用表来表示，取各实体的主码和联系的属性组成表结构。
其码由各实体的主码组成。如选课表的码是学号和课程号。

• 联系的类型
两实体集之间 1：1 联系:如果对于实体集 A 中的每一个实体，实体集 B 中至多有一个实体与之关联，反之亦然。注意：至多一个，意味着可以没有。一对一不是一一对应。
两实体集之间 1： n 联系：如果对于实体集 A 中的每个实体，实体集 B 中有 n 个实体(n≥0)与之关联，反之，对实体集 B 中的每个实体，实体集 A 中至多有一个实体与之关联。
两实体集之间 m： n 联系:如果对于实体集 A 中的每个实体，实体集 B 中有 m 个实体(m≥0)与之关联，反之，对于实体集 B 的每个实体，实体集 A 中也有 n 个实体(n≥0)与之关联。
多实体集之间 m:n:p 联系
4.映射基数在 ER 图中的表示
用箭头或线段来表示联系的映射基数。
一个实体集内的二元表示：
一对一，一对多，多对一，多对多。
5.联系的参与度(participation)：
每个实体有一个参与联系的次数，取最小、最大的参与次数 min 和 max，定义实体集的参与度。
min=0，称联系为部分参与；如果 min＞0，称联系为全参与（在 ER 图中用双线表示）。Max 限制每个实体的参与次数。
6.全部参与与存在依赖
设 A R B，如果 A 存在依赖于 B，则 A 全部参与联系 R。
7.弱实体集（在 ER 图中以双边框矩形表示）
如果一个实体集的所有属性都不足以形成主码，则称这样的实体集为弱实体集。
弱实体集与其拥有者之间的联系称作标识性联系（双边框菱形表示）
弱实体集与强实体集之间是一对多的联系
弱实体集必然存在依赖于强实体集
分辨符（也叫部分码，用下划虚线表示）弱实体集中用于区别依赖于某个特定强实体集的属性集合。弱实体集的主码由该弱实体集所存在依赖的强实体集的主码和该弱实体集的分辨符组成。
从联系集用双线（全部参与）连接弱实体集，用箭头（一对多联系）指向强实体集。
8.拓展 ER 特性
• 特殊化：用标记为 ISA 的三角形来表示
ISA = “is a”，表示高层实体和低层实体之间的“父类－子类”联系
• 概括:特殊化的逆
• 属性继承：高层实体集的属性被低层实体集自动继承
• 聚集：聚集是一种抽象，通过它联系被作为高层实体集。实体集 A 与 B 以及它们的联系可被看成另一实体集 C
8．ER 模型设计要点
• 一般说来，按数据粒度确定实体与属性，能形成元组的设计成实体，只是单一数据项的设计成属性。
• 实体有多方面性质，属性没有，如城市为属性，只一个，为实体，可以多个属性。
• 实体中，多值属性，还要其它若干属性，则将该多值属性定义为另一实体。
9.ER 向关系模式的转换

- 实体 → 关系
- 属性 → 关系的属性
- 复合属性 → 将每个组合属性作为复合属性所在实体的属性。划分成更小的部分。

- 多值属性 → 新的关系+所在实体的码。用一个单独的关系模式表示。Inst_phone = (ID , phone_number)
- 弱实体集：所对应的关系的码由弱实体集本身的分辩符再加上所依赖的强实体集的码
- 概括：高层实体集和低层实体集分别转为表；低层实体集所对应的关系包括高层实体集的码。

一．实体转换

•强实体集，将每个强实体集直接转换为一个关系，实体的码作为关系的码，实体的属性作为关系的属性。Student (ID,name,tot_cred)

•弱实体集，将每个弱实体集直接转换为一个关系，关系中增加强实体集的码。

多值属性单独建表

•超类实体建表，属性包括所有共有属性。

子类实体建表，属性包括超类的码和子类所独有的属性。如果概括是不相交的且全部的，不为超类建表，只为子类建表。

聚集可以单独建表，也可以不建表，用联系对联系表达。

二．实体间联系的转换（可看 PPT 上的例子）

总原则：N：M 联系单独建表，其他，增加属性。

•两实体间 1：N 的联系。将 1 方实体的码纳入 N 方实体对应的关系中，作为 N 方实体关系的外码。如果联系本身有属性，则将属性也纳入 N 方实体对应的关系中。

•同一实体间 1：N 的联系。在实体对应的关系中增加属性，属性包括：与 N 个体相联系的 1 个体的码，以及联系的属性。

•两实体间 M：N 的联系。联系单独转换为一个关系，属性包括：两实体的码、联系的属性。

•同一实体间 M ：N 的联系。联系单独转换为一个关系，属性包括：个体的码、与之有联系的另一个体的码、以及联系的属性。

•多实体间 M：N 的联系。联系单独转换为一个关系，属性包括：各实体的码、联系的属性。

•两实体间 1：1 的联系。将任一方实体的码纳入另一方实体对应的关系中，作为外码。如果联系本身有属性，则将属性也纳入对方实体对应的关系中。

第八章 关系数据库设计

1.1NF：R 的所有属性域是原子的，即可。无其他要求。

2.函数依赖：X→Y。叫作“X 函数确定 Y”或“Y 函数依赖于 X”， X 为决定因素，Y 为依赖因素。R 下的所有函数依赖组成 R 的函数依赖集，记为 F。

类型：平凡函数依赖，部分函数依赖，完全函数依赖，传递函数依赖

部分函数依赖和完全函数依赖：如果 X→Y，且不存在 $X' \subseteq X$ ，使 X′→Y，则称 Y 完全函数依赖 X,否则称 Y 部分函数依赖

3. 全码： R=(U)中不存在任何依赖，所有属性组成全码

4.2NF：在 R∈1NF 的基础上，每个非主属性都完全依赖于 R 的码。若存在非主属性部分依赖于码，则降为 1NF。

5.函数依赖集的逻辑蕴涵：若 A→B，B→C，则 A→C

被 F 逻辑蕴涵的所有函数依赖的集合叫做 F 的闭包 F^+。如果 F=F^+，则称 F 是函数依赖的完备集。

Armstrong 公理系统：自反律、增补律、传递律、合并律、分解律、伪传递律。

属性闭包：设 X⊆U ，则属性集 X 关于函数依赖集 F 的属性闭包为 **X^+_F= {A | A∈U 且 X→A 能由 F 根据推理规则推出}**

把所有的属性闭包合在一起就是 F 的闭包。

典型题目：1>属性闭包的求法

2>候选码的求法：一个属性集函数决定 U，若它的任一个子集都不能函数决定 U，则属性集可作为候选码。候选码从箭头左边的属性中找。仅在箭头左边的属性一定在候选码中。

6. 函数依赖集的等价

定义 设 R=(U)的两个函数依赖集 F 和 G，如果 F+=G+，则称 F 和 G 是等价的，如果 F 和 G 是等价的，则称 F 覆盖 G 且 G 覆盖 F。

典型题目：判断 F 和 G 是否等价：

检查 F 中的每个函数依赖 X→Y，检查 Y 是否属于 X^+_G。

检查 G 中的每个函数依赖 X→Y，检查 Y 是否属于 X^+_F。

7.正则覆盖（极小函数依赖集）

F 的正则覆盖是一个与 F 相等的最小函数依赖集。

特点：每个函数依赖的右端是一个属性；

函数依赖集没有多余的依赖（A→B，C→D，两个都不可以删掉）；

每个函数依赖左端决定因素没有多余属性。（AB→C，A，B 都不可以删掉）

典型题目：计算函数依赖集 F 的正则覆盖（正则覆盖未必唯一）

8.3NF：在 R∈2NF 的基础上，每个非主属性都不部分依赖（2NF 已经达到的条件）也不传递依赖于 R 的码。

典型题目：3NF 的判断：步骤的关键是确定候选码。

(1) 确定 R 的候选码，主属性，非主属性，决定因素。

(2) 如果 R 的每个属性都是基本数据项，则 R∈1NF。

(3) 如果 R∈1NF,检查所有非主属性是否对所有候选码都不存在部分函数依赖。若是，R∈2NF。当候选码都是单一属性时，R 必然属于 2NF。

(4) 如果 R∈2NF，检查所有非主属性是否对所有候选码都不存在传递函数依赖，若是，R∈3NF。当 R 没有非主属性时，R 必属于 3NF。

步骤的关键是确定候选码

9.BCNF 范式：R=（U）的所有非平凡函数依赖 α → β ， α 部分都包含 R 的一个候选码，则 R∈BCNF。

3NF 与 BCNF 的差别：3NF 要消除的是非主属性对码的部分依赖和传递依赖，而 BCNF 要消除的是主属性对码的部分依赖或传递依赖。如果 R∈3NF，检查所有决定因素是否都包含候选码，若是，R∈BCNF

10.模式分解

• 分解的准则

(1) 分解具有“无损连接性”

(2) 分解要“保持函数依赖”

(3) 分解既“保持函数依赖”，又具有“无损连接性”。

• 无损连接分解

• **典型题目：判断一个分解是否是无损分解。**

方法一（分解为 2 个子模式的）：

将 R 分解成 R1，R2，若F^+中至少存在以下一个依赖，则是无损分解：

(U1∩U2)→(U1-U2)∈F+ 或(U1∩U2)→(U2-U1)∈F +

方法二（分解成 2 个以上子模式的）：(1) 构造 n 列（属性），k 行（子模式）的矩阵 M
(2) 逐个检查 F 中函数依赖 X→Y,修改表中的元素。过程如下：按 X 列值相等确定各行，修改 Y 列对应行的值；如果其中之一为 aj，则全改为 aj，否则全改为 bmj，m 为这些行中的最小行号。
(3) 对 F 的每个函数依赖按(2)处理一遍后，若有某一行为 a1,a2,…,an 则判定分解是无损分解。算法终止。否则按(2)的过程再来一遍，直到一行为 a1,a2,…,an，判定为无损分解，或者 M 矩阵没有变化，判定为不是无损连接。

• **典型题目：保持函数依赖的分解的判断。**

投影的定义：F 在子模式 Ri=（Ui）上的投影为：ΠRi (F)={X→Y｜X→Y∈F+∧XY 属于 Ui}
注意：从 F +中找。
保持函数依赖的分解的定义：设ρ是 R 的一个分解，F 是 R 的函数依赖集，如果 G= ΠR1(F)∪ΠR2(F)∪…Π Rk(F)逻辑蕴涵 F，则称ρ是保持函数依赖分解，或简称保持依赖。
11.3NF 的模式分解算法（且保持函数依赖）
方法：
(1)求 Fc
(2)如果 Fc 中有一函数依赖 X→A,且 XA=U，则ρ={R},输出ρ，结束。
(3)对 Fc 中的每一个 X→Ai 都构成一个关系模式 Ri=(X , Ai)。如果 Fc 中有 X→A1,X→A2,…,X→An(左部决定因素相同)则以 R=(X,A1,A2,…,An)构成一个模式。

典型题目：求分解关系模式为 3NF 且保持函数依赖性和无损连接性的分解。

12.BCNF 的无损连接分解
方法：
(1) 初始化ρ={R}。
(2) 如果ρ中所有关系模式均已是 BCNF 转(4)，否则，从ρ中找出非 BCNF 的关系模式 S，在 S 中必有 X→A, X 不包含 S 的码，A 也不在 X 中。将 S 分解为 S1=(X，A)，S2=(Us-A)，Us 为 S 的属性组。
(3) 在ρ中以{S1,S2}代替 S，转(2)。
(4) 结束，输出ρ

典型题目：求 R 为 BCNF 的无损连接分解。

分解结果不是唯一的，这与选择分解次序有关。
结论：分解算法保证分解是无损的，但不能保证一定是保持依赖的。
13.多值依赖的三种定义
1>设 R=（U），X，Y，Z 是 U 的子集，Z=U-X-Y，对于 R 的任一 r, 如果有（x,y1,z1）和(x,y2,z2)属于 r, 则有（x, y1,z2）和（x, y2,z1）属于 r, 那么称 Y 多值依赖 X，Z 多值依赖 X。记为 X→→Y，X→→Z。
2>设 R=（U），X，Y，Z 是 U 的子集，Z=U-X-Y，X→→Y 成立当且仅当对于 R 的任一 r, 给定一对（x,z）值有一组 Y 的值，这组值仅仅决定于 x 的值而与 z 值无关。
3>关系模式 R(U)，X、Y、Z⊂U，Z=U–X–Y，对于 R(U)的任一关系 r，若存在元组 t1，t2，使得 t1[X] = t2[X]，那么就必然存在元组 t3，t4，使得：
t3[X] = t4[X] = t1[X] = t2[X]
t3[Y] = t1[Y]， t4[Y] = t2[Y]
t3[Z] = t2[Z]， t4[Z] = t1[Z]
则称 Y 多值依赖与 X，记作 $X \rightarrow\rightarrow Y$
14.4NF:设 R=（U），D 是 R 上的函数依赖和多值依赖集，如果 D 的每个 X→Y， X→→Y ，X 必是 R 的超码，那么 R 属于 4NF。

第十二章 事务管理

1. 事务(transaction)：访问并可能更新数据的逻辑工作单位，一组操作序列，一个或一段应用程序。

2.事务的特性：原子性、一致性、隔离性和持续性（ACID）。

原子性(atomicity)

一个事务是一个不可分割的工作单位。

一致性(consistency)

事务必须是使数据库从一个一致性状态变到另一个一致性状态。当数据库成功事务提交的结果时，就说数据库处于一致性状态。

隔离性(isolation)

一个事务的执行不能被其它事务干扰。即一个事务内部的操作及使用的数据对并发的其它事务是隔离的，并发执行的各个事务之间不能相互干扰。如有干扰，要加隔离机制。

持续性(durability)

持续性也称永久性，是指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。后面的其它操作或故障不应对其有任何影响。

3.事务状态

活动状态，部分提交状态，失败状态，中止状态，提交状态

4.并发

并发可能导致丢失修改、不可重复读、读脏数据。

这三种情况导致的原因：并发操作破坏了事务的隔离性。要采用正确的调度方式，使一个事务不受其它事务的干扰。达到可串性化的结果

5. 可串行化

• 调度：按某运行顺序来排定事务运行步骤。写成读写操作的序列。

• 串行调度：一个事务运行完后，运行下一个事务，时间上不交叉、不重叠。

• 并发执行的可串行化调度：几个事务的并行执行的结果，与这几个事务任一种串行调度的结果相同。

6. 并发执行可串行化调度的两种方法：基于锁(locking)的协议、基于时间戳(timestarmping)的协议。

7.冲突可串行化

• 冲突操作：两个事务 Ti,Tj 对同一个数据 Q 的读写、写读、写写操作，有 6 种情况：Ri(Q)Wj(Q),Rj(Q)Wi(Q)， Wi(Q)Rj(Q)， Wj(Q)Ri(Q)， Wi(Q)Wj(Q)， Wj(Q)Wi(Q)

• 非冲突操作：Ti,Tj 对不同数据的读写操作，Ti,Tj 对同一个数据 Q 的读操作 Ri(Q)Rj(Q),Rj(Q)Ri(Q)

冲突的操作不能交换，交换产生的结果不同，非冲突操作可以交换，不影响结果。

调度 S 交换其中的非冲突操作经形成新的调度 S’，S 与是 S’等价。若 S’是一个串性调度，则 S 是可串性化的（并行调度）。

• 冲突可串性化的优先图判断法

判断调度 S 是否是冲突可串性化:

T1，T2，…Ti,…Tj…表示为结点

如果 Ri(Q)之后有 Wj(Q),或 Wi(Q)之后有 Rj(Q) ，或 Wi(Q)之后有 Wj(Q), 则 Ti、Tj 之间加有向边 Ti→Tj

若图无环，S 是冲突可串性化的。

等价的串性调度是，按有向边出口依次截取结点。

8.视图可串行化

• 两个调度 S、S’满足：1>对每个数据项 Q 的初值，两个调度是同一个事务读。

2>事务 Ti 读取的 Q 值，在两个调度中，都是由同一个事务产生的。
3>Q 的终值，两个调度是同一个事务写入。
则 S 与 S’等价。
若 S’是一个串性调度，则 S 是视图可串性化的。

• 视图可串性化的扩展优先图（带标记）判断
T1，T2，…Ti,…Tj…表示为结点
对调度 S 增加两个虚事务: 头事务 Tb 只有写操作 Wb（Q）,尾事务 Tf 只有读操作 Rf（Q）
调度 S 写成读写操作序列 S’=Wb(Q) … … Rf(Q)
找出所有写读对 Wi(Q)Rj(Q), 加 0 标记有向边 Ti-0→Tj
对调度中间的每个 Wk(Q):
对应头写读对 Wb(Q)Ri(Q), 加 0 标记有向边 Ti-0→Tk
对应尾写读对 Wj(Q)Rf(Q), 加 0 标记有向边 Tk-0→Tj
对应中间的写读对 Wi(Q)Rj(Q), 加 p 标记有向边 Tk-P→Ti 和 Tj -P→Tk（两条边）

如果无环，是视图可串性化的调度。
如果去掉 P 标记边对的一条，无环，也是视图可串性化调度。
等价的串性调度是，按有向边出口依次截取结点。
9.并发控制
•锁的概念：事务 T 对某个数据对象操作之前，先向系统发出请求，对其加锁。加锁成功后，事务 T 开始对该数据对象进行操作，在事务 T 释放它的锁之前，其它的事务不能更新此数据对象。加锁不成功，事务 T 等待。
• 封锁协议：事务运用 X 锁和 S 锁对数据对象加锁时的约定。
• 锁的类型
X 锁：排它型锁(exclusive lock)，又称为写锁（为写操作而加得锁）。
若事务 T 对数据对象 Q 加上 X 锁，则只允许 T 读取和修改 Q，直到 T 释放 Q 上的锁，其他任何事务都不能对 Q 加上任何类型的锁，提出加锁，只能等待。
S 锁：共享型锁(share lock)，又称为读锁（仅为读操作而加的锁）。
若事务 T 对数据对象 Q 加上 S 锁，其它事务还能再对 Q 加上 S 锁，但加不上 X 锁。
保证了 T 释放 Q 上的 S 锁之前，其它事务可以读 Q，不能对 Q 做任何修改。
• 一级封锁协议
事务 T 在修改数据 Q 之前必须先对其加 X 锁，直到事务结束才释放。
说明：事务只读数据时，协议不要求加锁。
事务结束包括正常结束(commit)和非正常结束(rollback)。
一级封锁协议可防止丢失修改。但不能保证可重复读和不读“脏”数据。
• 二级封锁协议
事务 T 在修改数据 Q 之前必须先对其加 X 锁，直到事务结束才释放。
事务 T 在读取数据 Q 之前必须先对其加 S 锁，读完后即释放 S 锁。
二级封锁协议可防止丢失修改，防止读“脏”数据，但不能保证可重复读。
• 三级封锁协议
事务在修改数据 Q 之前必须先对其加 X 锁，直到事务结束才释放。事务在读取数据 Q 之前必须先对其加 S 锁，直到事务结束才释放 S 锁。
三级封锁协议可防止丢失修改，防止读“脏”数据，保证可重复读。
• 两阶段封锁协议

对任何数据读写之前，先要获得对该数据的封锁，释放一个封锁之后，事务不再加任何锁。
两段：增长阶段，申请加锁阶段；缩减阶段，释放锁阶段。事务开始为增长阶段，一旦释放锁，进入缩减阶段。
所有事务都遵守两阶段协议，并发调度是可串性化的。（充分条件，不是必要条件）。
事务不遵守两阶段协议，并发调度是能是可串性化的，也可能不是。
• 死锁和活锁
两个或两个以上的事务都在等待其中另一个事务解除封锁，它才能继续执行下去，结果任何一个事务都无法继续执行，这种现象称为死锁(dead lock)。
封锁技术可能产生某个事务可能永远处于等待状态，得不到执行的机会，这种现象称为活锁(live lock)。
解决活锁的一种简单方法是采用“先来者先执行”的控制策略，也就是简单的排队方式。
DBMS 中有一个“死锁测试程序”，每隔一段时间检查并发的事务是否发生死锁，发现死锁，只能撤消某个事务，恢复该事务到初始状态。
两阶段封锁协议可能发生死锁。
10.基于时间戳的协议
时间戳定义：事务启动时间。
事务 Ti 的时间戳记为 TS（Ti）
若 TS（T1）＞TS（T2），则称 T1 是年轻的事务，或称“后到的事务”，T2 是年长的事务，或称“先来的事务”。
时间戳用计数器实现，初值为 0，第一个到来的事务为 1，依次加 1。
数据项 Q 的时间戳
W-TS（Q）：所有执行了写 Q 操作的事务中最年轻事务的时间戳（值最大）。
R-TS（Q）：所有执行了读 Q 操作的事务中最年轻事务的时间戳（值最大）。
• 时间戳协议的意义
1>尽管对多个事务并发调度，交叉执行，但总的调度结果与事务先后到来的顺序所确立的串性调度等价。
2>交叉执行中，
“先到的事务”Ti 要读被“后到的事务”Tj 改过的数据，不符合 Ti→ Tj 串行等价，就发生冲突。
“先到的事务” Ti 要修改被“后到的事务” Tj 读过或改过的数据，不符合 Ti→ Tj 串行等价，也发生冲突
3>发生冲突，撤消并重新启动该事务，使它成为最后到来的事务，赋予新的时间戳（最大）。
事务启动序列发生变化，等价的串性调度顺序也随之变化。
4>事务对数据的更新都是在最后 COMMIT 时起作用，执行到中间，撤消事务不影响结果。

第十三章 数据仓库与数据挖掘
1.数据仓库中数据组织的基本原则是面向主题性表示，DW 中的所有数据都是围绕着某一主题组织、展开的。
数据仓库的数据仍具有集合性，意味着必须以某种数据集合的形式存储起来。
2.目前 DW 所采用的数据集合方式主要是：
1>关系模式，以关系数据库方式存储
2>多维模式，以多维数据库方式存储
3>混合模式，两者相结合的方式存储
3.元数据: 仓库结构描述,模式,视图,维,导出数据的定义;汇总算法,数据库到数据仓库的映射,

系统数据,元数据作目录用,保存在当前磁盘上,仓库中.
4. 数据库是长期储存在计算机存贮介质上，有一定组织形式、可共享的数据集合。
数据仓库是一个面向主题的、集成的、非易失的且随时间变化的数据集合，用来支持管理人员的决策。
数据库的主要任务是 OLTP（联机事务处理）
数据仓库的主要任务是 OLAP（联机分析处理）
OLTP 面向客户、办事员、操作员的事务和查询处理
OLAP 面向经理、主管、分析人员的数据分析
OLTP 处理的数据是基本表中的元组数据
OLAP 处理的数据是不同粒度的汇总数据
OLTP 采用 E-R 模型和面向应用的数据库设计
OLAP 采用星型或雪花模型和面向主题的多维数据立方体设计
OLTP 主要功能是 DBMS
OLAP 主要功能是检索查询工具，多维数据分析工具、统计分析及数据挖掘工具
5.数据挖掘
数据挖掘就是应用一系列技术从大型数据库或数据仓库中提取人们感兴趣的信息和知识，这些知识或信息是隐含的，事先未知而潜在有用的，提取的知识表示为概念、规则、规律、模式等形式。
6.分类的概念
已知训练集中的对象属于某个类的情况下，把给定的数据分到一定的类中。
分类的两步过程：
首先，在已知训练数据集上，根据属性特征，为每一种类别找到一个合理的描述或模型，即分类规则，形成分类器。其次根据规则对新数据进行分类。
ID3 方法——决策树分类
贝叶斯分类算法
7.决策树分类器
决策树分类器
决策树是一种用于产生分类规则的树结构
树中的内节点表示在一个属性上的测试
每个树叶代表类或类分布
树中的每个分支代表一个测试输出，即一条规则。
样本的属性为结点，信息量最大的为根节点。
属性的值为分支。
8. ID3 算法
基本思想是贪心算法，采用自上而下、分而治之的方法构造决策树。
首先检测训练数据集的所有属性，选择信息增益最大的属性 A 建立决策树根节点。由该属性特征的不同取值建立分支。
对各分枝的实例子集递归，用该方法建立树的节点和分枝，直到某一子集中的数据都属于同一类别，或者没有特征可以再用于对数据进行分割。
• 属性信息增益（互信息）计算：
I（U，V）=H（U）-H（U/V）=类别熵－条件熵
其中，U 为类别集合，{N，P}V 为属性的值集合
• 类别熵（先验熵）的计算：
H（U）=∑P(ui)log2(1/P(ui))=-∑P(ui)log2(P(ui))，P(ui)=属于 ui 类的样本数/全体样本数

• 条件熵（后验熵）的计算：
H（U/V）=∑P(vj)∑P(ui｜vj)log2(1/P(ui|vj)))，P(vj)=属性取值为 vj 的样本数/全体样本数；
P(ui｜vj)=属性取值为 vj 的样本中属于 ui 类的样本/属性取值为 vj 的样本数
• 决策树算法的基本策略：
树以代表训练样本的单个结点开始（根）（所有的数据都在根结点）
如果样本都在同一类。则该结点成为树叶，并用该类标记；
递归划分停止的条件：给定结点的所有样本属于同一类。
9.朴素贝叶斯分类：统计学分类方法,给定样本,求属于每个类的概率,按最大值分配样本到该类
• 朴素:假定一个属性值对给定类的影响独立于其他属性的值.
• 贝叶斯定理:
设 X 是未知类的样本,H 是 X 属于 C 的一个假定,X 的分类问题变为求 P(H｜X),即给定 X,H 成立的概率
P(H｜X)称为条件 X 下 H 的后验概率:P(H｜X)=P(X｜H)P(H)/P(X)
其中:P(X｜H)称为条件 H 下 X 的后验概率;P(H)是 H 的先验概率,独立于 X;P(X)是 X 的先验概率;
例如:样本的类是各种水果,X 表示是红色和圆的样本,H 假定 X 是苹果类,P(H｜X)表示:当 X 是红色和圆的样本时,X 属苹果类的确信度。
• 朴素贝叶斯分类算法:
每个样本表示为 n 维向量:X={x1,x2,…,xn},属性 A1,A2,…An 的值
设定 m 个类 C1,C2,…Cm,对未知类的样本 X,当且仅当 P(Ci｜X)> P(Cj｜X) ,(1≤j ≤m, i≠j), 将 X 分配给类 Ci, 即 P(Ci｜X)是最大的,按贝叶斯公式:P(Ci｜X) = P(X｜Ci) P(Ci)/ P(X)
其中: P(X)对每个类是常数,只需求 P(X｜Ci) P(Ci)
P(Ci)=Ci 中的训练样本数/训练样本总数
如果 P(Ci)未知,假定 P(C1)= P(C2)=…= P(Cm)按属性值相互条件独立的朴素假定 P(X｜Ci)= ∏nk=1 P(Xk｜Ci)，其中: P(Xk｜Ci)=类 Ci 中具有 Xk 值的样本数/类 Ci 中样本数
10.聚类：
是在要划分的类未知的情况下，将数据对象分成不同的类 ，无指导学习。
• 聚类中的数据表示
对象---属性矩阵:N×P 矩阵
N 行：对象个数，P 列：属性个数
其中 x(i ,j)是对象 i 的属性 j 的值
相似度矩阵： N×N 矩阵 ，其 d(i ,j)是对象 i 与对象 j 之间的相似度
• 相似度计算

欧几里得距离

$$d(i,j)=sqt(|x_{i1}-x_{j1}|^2+|x_{i2}-x_{j2}|^2+\cdots+|x_{ip}-x_{jp}|^2)$$

加权的欧几里得距离

$$d(i,j)=sqt(w_1|x_{i1}-x_{j1}|^2+w_2|x_{i2}-x_{j2}|^2+\cdots+w_p|x_{ip}-x_{jp}|^2)$$

• 主要聚类方法
划分方法
1) 初始：确定 k 个划分， k 个类
2)每个类至少一个对象，每个对象只属于一个类
3）按距离，把对象划分到不同类中
典型的方法是 k-平均算法、 k-中心点算法

• k-平均算法

算法： k-平均

输入：类数目 k 和包含 n 个对象的数据库

输出：k 个类

方法：

1）任选 k 个对象作为初始类中心

2）repeat

3) 计算类中对象的平均值，将每个对象重新赋给最类似的类

4）更新类的平均值，

5）until 不再发生变化

说明：初始随机选择 k 个对象，代表类的平均值（类中心），其余对象计算与每个类中心的距离，划分到最近距离的类中，重新计算类中心，重复。

• k-中心点算法

算法： k-中心点

输入：类数目 k 和包含 n 个对象的数据库

输出：k 个类

方法：

1）任选 k 个对象作为初始类中心点

2）repeat

3) 将每个对象指派给离中心点最近的类

4）随机选择非中心点 h, 计算 h 替换中心点 i 的总代价 RChi , 如果 RChi < 0 , 用 h 替换中心点，然后对非中心点对象按与新的中心点的距离，指派类

5）until 不再发生变化

说明：$RC_{hi}=\sum^{k}_{i=1}\sum_{p\in Ci}(|p-m_i'|^2-|p-m_i|^2)$

其中： RChi 为 h 替换中心点 i 的总代价，p 是给定对象，mi 是类 C i 的中心点，mi’是类 C i 用 h 替换中心点 i 的后的中心点。

基本思想是设置初始中心点后，反复用非中心点对象替换中心点，以改进聚类质量。

因为 k-平均对孤立点是敏感的

11.关联规则：

若两个或多个变量间存在着某种规律性，就称为关联。关联分析的目的就是找出数据库中隐藏的关联网。

项集：项的集合称为项集（itemset）。设 I={Il,I2,..,In}是一个项集，其中 Ii(I=1,2,3,…,n)可以是购物篮中的物品，也可以是保险公司的顾客。

K 项集---包含 K 个项的项集被成为 K 项集，表示项集中项的数目。

事务：事务是项的集合，设有事务 T，则 T⊆I。对应每个事务有唯一的标识，如事务号记为 ID。设 X 是 I 中项的集合，如果 X⊆T，则称事务 T 包含 X。

事务集：事务的集合称为事务集。设某事务集为 D，则 D={T1，T2，…,Tn},

D={Ti|Ti∈ D,i=1,2,…,n}.

关联规则是如下形式的逻辑蕴涵：A⇒B，其中 A，B 是项集，A ∈ I，B∈ I，A ∩ B=Φ。

• 一般用两个数据描述关联规则的属性。

设 A，B 是项集，对于事务集 D，A∈D，B∈D，A ∩ B=Φ，

（1 ）可信度（置信度）：“值得信赖性”。

可信度定义为：

可信度（A→B）=包含 A 和 B 的元组数/包含 A 的元组数

可信度表达的就是在出现项集 A 的事务集 D 中，项集 B 也同时出现的概率。
支持度（Support）
支持度（A→B）=包含 A 和 B 的元组数/元组总数。
支持度描述了 A 和 B 这两个项集在所有事务中同时出现的概率。
• Apriori 算法（找频繁项集算法）：
基本概念：
频繁项集——所有支持度大于最小支持度的项集称为频繁项集，或简称项集。
强关联规则——同时满足最小支持度阈值和最小可信度阈值的规则称为强规则。
找出所有的频繁项集，由频繁项集产生强关联规则。
• 关联规则“S⇒(L-S)”的可信度计算：
可信度（A⇒B）=包含 A 和 B 的元组数/包含 A 的元组数
则可信度（S⇒(L-S)）=包含 S 和 L-S 的元组数/包含 S 的元组数=包含 L 的元组数/包含 S 的元组数=support_count(L)/support_count(S)
12.文本挖掘
TF-IDF 加权标引：
TF(term frequency):词条频率,在文档中出现的次数,定义为: $tf_j(Di)$
IDF(inverse document frequency):文档频率倒数,出现词条 tj 的文档占整个文档集合的倒数的对数,定义为: $Log(N/n_j)$
N 是文档总数, n_j 是出现词条 tj 的文档数
TF-IDF 权= $tf_j(Di)* Log(N/n_j)$
显然,TF 越大说明该词条在文档中出现越多,越重要
n_j 越大,$Log(N/n_j)$越小,说明 tj 出现的文档越多,越不重要,例如“的”出现在所有文档中,不重要.
IDF 取对数的目的是使权对 N 变化不敏感

第十四章 基于对象的数据库
1. **复合类型**：结构类型：不同类型元素的有序集合 示例：日期由日、月、年组成（3，June，2001）数组类型：同类元素的有序集合 示例：人名数组（Annie, Bob, Jerry, Tom)
多集类型(multiset)：同类元素的无序集合，并且允许一个成员多次出现示例：成绩（80，90，75，80，90，80）集合类型(setof)：同类元素的无序集合，一个成员最多只能出现一次
示例：课程（MATHS，PHYSICS，OS，DB）

传统关系模型：属性只能是基本数据类型
对象-关系模型：属性可以是复合数据类型。可以直接表达 E-R 中的复合属性和多值属性
数据类型可以嵌套：课程成绩单{(MATHS，80)，(PHYSICS，90)，(OS，85)，(DB，95)}
外层是集合类型，内层是结构类型
2.与复杂类型有关的查询
路径表达式
点号不但用来引用复合属性，还可以用于引用类型
使用引用隐藏了表之间的连接操作，简化了查询
定义：将一个嵌套关系转换为 1NF 的过程称为解除嵌套
3. 类型引用：类型的属性可以是对属于指定类型的对象的应用
如声明：author-list setof (ref(Person))
表明 author-list 是一个对 Person 对象引用的集合
表引用：author-list setof (ref(people))

用表的主码或元组的标识符来实现对元组的引用

表引用 Create table depatments (name varchar (20)head ref (people)) 等价 Create type Depatment (Name varchar (20)Head ref (person) scope people Create table departments of type Depatment