

第一章 数据库发展史

- 数据管理技术的发展阶段：
 - 人工管理阶段
 - 文件系统阶段
 - 倒排文件系统阶段
 - 数据库阶段
- 文件系统的三个缺陷：
 - 数据冗余：同样的数据在多个文件中重复存储。
 - 不一致性：更新操作时，同样的数据在不同文件中不一样。
 - 数据联系弱
- 数据库管理技术进入数据库阶段的标志：
 - IMS系统
 - DBTG报告
 - E.F.Codd的文章
- 数据库提供四方面的数据控制功能：
 - 数据库的并发控制
 - 数据库的恢复
 - 数据的完整性
 - 数据的安全性

第二章 数据库系统结构

- 数据模型：结构数据模型应包含数据结构、数据操作和数据完整性约束。
 - 实体联系模型(ER模型)：P.P.Chen提出
 - 层次模型：用树形（层次）结构表示实体类型及实体间联系的数据模型。
 - 缺点：只能表示1：N联系。
 - 网状模型：用有向图结构表示实体类型及实体间联系。
 - 关系模型：用二维表格表达实体集。
 - 面向对象模型。
- 数据库的三级体系结构：外部级、概念级、内部级。
- 数据独立性：
 - 物理独立性：内模式的修改不影响概念模式。
 - 逻辑独立性：概念模式的修改不影响外模式。
- 数据库管理系统的主要功能：
 - 定义DDL
 - 操纵DML
 - 保护

- 维护
- 数据字典
- 数据库系统的组成：数据库、硬件、软件、数据库管理员(DBA)。

第三章 关系运算

- 一些概念
 - 关系模型：用二维表格表示实体集，关键码表示实体间联系的数据模型。
 - 非过程性语言：只需要指出干什么，不必指出怎么干。
 - 无限关系：元组个数为无穷多个的关系。
 - 无穷验证：验证公式真假时需要进行无限次验证。
 - 关键码
 - 候选键：唯一标识元组，不含多余属性的属性集。
 - 主键：从多个候选键中选择一个作为主键。
 - 超键：唯一标识元组的属性集。
 - 外键：关系R包含关系S中的主键所对应的属性组F，则F是R的外键，称S为参照关系，R为依赖关系。
- 关系模型的完整性规则：
 - 实体完整性规则：关系中元组的主键值不为空。
 - 参照完整性规则：外键只能为空值或参照关系中的某个主键值。
 - 用户定义完整性规则：针对数据满足的约束条件。如：工龄小于年龄、成绩大于等于0。
- 关系查询语言分为两类：
 - 关系代数语言：以集合操作为基础。
 - 关系演算语言：以谓词演算为基础。
 - 关系演算又分为：元组演算和域演算。
- 关系代数的五个基本操作：并、差、笛卡尔积、投影、选择。
- 关系代数的组合操作：交、连接、自然连接、除。
- 扩充的关系代数操作：外连接（分为左外连接、右外连接）、外部并（公共属性只取一次）、半连接（自然连接后在某一关系上的投影）。
- 要会写关系代数表达式。
- 要会将关系代数表达式转换为元组表达式。
- 要会将元组表达式转换为域表达式。
- 关系运算的安全性：
 - 对元组关系演算进行安全约束：对表达式中变量取值范围规定范围，使之不产生无限关系和无穷次验证，这种表达式称为安全表达式。
- **关系运算主要有关系代数、元组演算、域演算三种。**
- 关系优化的一般策略：
 - 尽早执行选择操作
 - 将笛卡尔积和之后的选择操作，改为F连接操作

- 同时计算一连串的选择和投影
- 语法树中，叶子是关系，非叶子节点是关系代数操作。

第四章 结构化查询语言

- 外模式对应于视图，模式对应于基本表，元组称为行，属性称为列。内模式对应于存储文件。
 - 基本表是实际存储在数据库中的表，视图是从基本表或其他视图中导出的表，本身不存储在数据库中。
- SQL的四个组成部分：
 - 数据定义(DDL): 定义SQL模式、基本表、视图、索引。
 - 数据操纵(DML): 分为数据查询和数据更新两类。数据更新包括插入、删除、修改三种操作。
 - 数据控制(DCL): 包括对基本表和视图的授权、完整性规则的描述、事务控制语句。
 - 嵌入式SQL。
- SQL中的基本数据类型：
 - INTEGER: 长整数(INT)。
 - SMALLINT: 短整数。
 - FLOAT(n): 精度n位的浮点数。
 - NUMERIC(p,d): 定点数, p位数字, d位小数。
- 基本表的创建：
 - 如何定义主键, 外键, 约束:

```
create table spj
(sno char(4) not null,
pno char(4) not null,
jno char(4) not null,
price numeric(7,2),
qty smallint,
primary key(sno,pno,jno),
foreign key(sno) references s(sno),
foreign key(pno) references p(pno),
foreign key(jno) references j(jno),
constraint c_qty check (qty between 0 and 10000));
```

- 基本表的修改：
 - 添加属性:

```
alter table s add address char(12);
```

- 删除属性:

```
alter table b drop [属性名] cascade
```

cascade: (包括删除视图)/restrict(没有视图或约束引用时删除,否则拒绝删除)。

restrict: 没有视图才会删除列属性。

- like谓词:

```
select sname form s where saddr like '上海%'。
```

- %表示与任意长度(可以为0)字符串匹配。
- _表示可以与任意单个字符匹配。

- 连接操作:

- 检索每个工程使用P3零件的情况:

```
select * from j full outer join (  
    select * from spj  
    where pno='P3') spj1 on j.jno=spj1.jno;
```

- 数据分组: 将某一列的值相同的元组分为一组, 对每一个分组进行聚合操作。

- 集合操作:

- 并: union 使用all选项, 返回结果不消除重复元组。
- 交: intersect
- 差: except
- 比较操作: 资格比较: in / not in
 - 求不使用编号为P3零件的工程编号JNO和名称JNAME:

```
select jno, jname from j  
where jno not in (  
    select jno from spj  
    where j.jno = spj.jno and pno='p3');
```

- 求至少不使用p3和p5两种零件的工程编号jno

```
select jno from j  
where jno not in (  
    select jno from spj  
    where pno in ('p3','p5'))  
order by jno;
```

- 算术比较: some, all, in 可用=some代替

- 求不使用编号为p3零件的工程编号jno和名称jname(not in/ not exists/ <>all有时可以等价):

```
select jno,jname from j where jno <> all(select jno from spj where pno = 'p3');
```

- 查询最昂贵的商品价格:

```
select distinct pno, price from spj where price > all(select price from spj);
```

或者用max聚合函数:

```
select distinct pno, price from spj where price =(select max(price) from spj);
```

- 查询至少使用了“东方配件厂”一种零件的工程编号：

```
select distinct jno from spj where sno = some(select sno from s where sname='东方配件厂');
```

- 数据插入：

```
insert into j(jno,jname,jcity,balance)
values('j8','地铁二号线','上海','689');
```

- 数据删除：

```
delete from <基本表> where();
```

用于删除基本表满足条件的元组。

- 数据修改：

```
update <基本表> set 列名=<值表达式> where();
```

使用where可修改指定元组的列值。

- 当供应商S3提供的零件P5的单价低于该零件的平均单价时，将其提高6%：

```
update spj
set price = price * 1.06
when sno = 's3'
and pno = 'p5'
and price < (select avg(price) from spj when pno = 'p5');
```

- 对视图的更新：

- 用了连接，不行；用了分组聚合操作，不行；只有行列子集视图可以。

- DCL语言：

- 授权：

```
grant select,insert on <表> to <用户>
```

给指定用户授予某个基本表的两个权限。常见权限：update, delete.

- 撤销权限：

```
revoke insert on <表> from <用户>
```

- 嵌入式SQL：

- 声明：

```
EXEC SQL <SQL语句>END_EXEC
```

C语言中，可使用/结束：

```
EXEC SQL <SQL语句>;
```

- 常见状态码：
 - 02000：已取完所有结果。
 - 00000：取数据出错。
 - 01000：警告信息，但成功执行。