数据库系统的特点：

1.数据结构化

2.数据的共享度高，冗余度低，易扩展

数据共享可以大大减少数据的冗余

3.数据独立性高

数据独立是指数据的使用与数据的说明分离。

数据独立性用来描述应用程序与数据结构之间的依赖程度，包括数据的物理独立性和数据的逻辑独立性。

物理独立性是指用户的应用程序在一定程度上独立于数据库中数据的物理结构。

逻辑独立性是指用户的应用程序在一定程度上独立于数据库中数据的全局逻辑结构。

4.数据由数据库管理系统统一管理和控制

数据库管理系统提供的数据控制功能：

（1）数据的安全性保护；

（2）数据的完整性检查；

（3）并发控制；

（4）数据库恢复。

数据模型的分层：

1.概念模型（概念层，第一层）

用于描述某个特定组织所关心的信息结构。

概念模型的特点：语义表达能力强；易于理解；独立于任何 DBMS；容易想 DBMS 所支持的逻辑数据模型转换。

常用的概念模型：实体-联系模型（E-R 模型）和面向对象模型（OO 模型）

2.逻辑模型（逻辑层，第二层）

用于描述数据库数据的整体逻辑结构。

传统的逻辑数据模型：层次模型，网状模型和关系模型；

非传统的逻辑数据模型：面向对象模型，XML 模型。

3.物理模型（物理层，最低层）

用于描述数据的物理存储结构和存取方法。

数据模型的三要素：

1.数据结构

描述数据库的组成对象（实体）以及对象之间的联系。

内容包括：与对象的类型、内容、性质有关的；与对象之间的联系有关的。

2.数据操作

是指对数据库中各种对象的实例允许执行的操作吉赫，包括操作及有关的操作规则。

数据库主要有查询和更新两大类操作。

数据操作是对系统动态特性的描述。

3.数据完整性约束

是一组完整性规则。

数据完整性规则包括数据结构完整性规则和数据操作完整性规则。

数据库的三级模式结构：（数据库->内模式->模式->外模式）

1.内模式

也称存储模式，对应于物理层数据抽象，它是数据的物理结构和存储方式的描述，是数据在数据库内部的表示方式。

DBMS 提供内模式描述语言来严格定义内模式。

2.模式

也称逻辑模式，对应于逻辑层数据抽象，是数据库中全体的逻辑结构和特征的描述。

DBMS 提供数据定义语言（DDL）来严格定义模式。

3.外模式

也称子模式或用户模式，对应于视图层数据抽象，它是数据库用户能够看见和使用的局部数据的逻辑结构和特征描述。

DBMS 提供子模式定义语言来严格定义子模式。

两层映像与数据独立性：

1.外模式/模式映像

模式描述的是数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。

一个模式对应于多个外模式。

当模式改变时，有数据库管理员对各个外模式/模式的映像作相应的改变，可以使外模式保持不变。应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据的逻辑独立性。

2.模式/内模式映像

数据库中只有以一个模式，也只有一个内模式，所以模式/内模式映像是唯一的，它定义了数据全局逻辑结构与存储结构之间的对应关系。

当数据库的存储结构改变了，由数据库管理员对模式/内模式映像作相应的改变，可以使模式保持不变，从而应用程序也不必修改，保证了数据的物理独立性。

数据库系统组成：

从 DBMS 角度来看（内部结构）：数据库系统结构是外模式/模式/内模式的三级模式；

从用户角度来看（外部结构）：数据库系统结构分为单用户结构、主从式结构、分布式结构、客户/服务器、浏览器/应用服务器/数据库服务器等结构。

数据库管理系统（DBMS）：

1.DBMS 的功能

数据定义；

数据组织、存储和管理；

数据操纵；

数据库的事务管理和运行管理；

数据库的建立和维护；

其他功能

2.DBMS 的组成

模式更新；

查询；

更新；

查询处理器；

存储管理器;
事务管理器

数据库系统的相关人员:

1.数据库管理员

决定数据库中的信息内容和结构;
决定数据库的存储结构和存取策略;
定义数据的安全性要求和完整性约束;
监控数据库的使用和运行;
数据库的改进和重组重构。

2.系统分析员和数据库设计人员

3.应用程序元员

4.用户

关系数据结构:

1.域

域是一组具有相同数据类型的值的集合。

空值是所有可能的域的一个取值，表明值未知或不存在。

2.关系

对一个关系而言，一个最基本的要求是它每个属性的域必须是原子的。如果域中的每个值都被看做是不可再分的单元，则域是原子的。

3.关系模式

关系的描述称为关系模式

4.超码

对于关系 r 的一个或多个属性的集合 A，如果属性集 A 可以唯一地标识关系 r 中的一个元组，则称属性集 A 为关系 r 的一个超码。一个集合中如果包含超码，那么那个集合也是超码。

5.候选码和主码

对于关系 r 的一个或多个属性的集合 A，如果属性集 A 是关系 r 的超码，且属性集 A 的任意真子集都不能成为关系 r 的超码，则称属性集 A 为候选码。

若一个关系有多个候选码，则可以选定其中的一个候选码作为该关系的主码。

6.外码

设 F 是关系 r 的以一个属性（或属性集），K_s 是关系 s 的主码。如果 F 和 K_s 相对应，则称 F 是关系 r 参照关系 s 的外码。

关系完整性约束:

关系模式中有 3 类数据完整性约束：实体完整性、参照完整性和用户自定义完整性。其中实体完整性和参照完整性是关系模型必须满足的数据完整性约束，被称作是关系的两个不变性。

关系操作:

关系操作有查询操作（最主要）和更新操作两大类。

关系操作的特点是集合操作方式，即操作的对象和结果都是集合。这种操作方式也称为

一次一集合的方式。相应的，非关系数据模型的数据操作方式称为一次一记录的方式。

关系模型的相关概念：

关系模型的数据结构为关系，每一个表（关系）有唯一的名字。关系数据库是表的集合，即关系的集合。表中一行代表的是若干值之间的关联，即表的一行是由有关联的若干值构成。非正式地说，一个表是一个实体集，一行就是一个实体，它由共同表示一个实体的有关联系的若干属性的值所构成。由于一个表是这个有关联的值的集合（即行的集合），而表这个概念和数学上的关系概念密切相关，因此称为关系模型。

完整性规则：

1.实体完整性规则

若属性集 A 是关系 r 的主码，则 A 不能取空值 null。

如果竹马的属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这是不允许的。

2.参照完整性规则

若关系 r 的外码 F 参照关系 s 的主码，则对于关系 r 中的每一个元组在属性 F 上的取值，要么为空值 null，要么等于关系 s 中某个元组的主码值。

基本关系代数运算：

1.传统的集合运算

并、差、交、笛卡儿积

2.专门的关系运算

选择

在数据库中，查找 2015 级的所有班级情况

$$\sigma_{grade=2015}(Class)$$

投影

在数据库中查找所有学生的姓名和民族

$$\prod_{studentName,nation}(Student)$$

连接

$$r \rhd\lhd_{\theta} s = \sigma_{\theta}(r \times s)$$

除运算

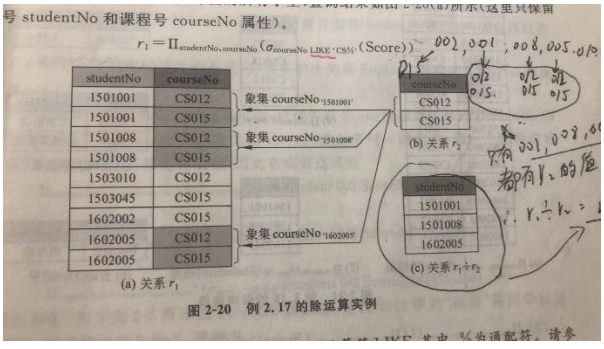


图 2-20 例 2.17 的除运算实例

SQL 特点：

1.综合统一；

2.高度非过程化；

3.面向集合的操作方式；

4.同一种语法结构提供两种使用方式；

5.语言简洁，易学易用。

索引基本概念:

(1) 顺序索引: 索引中的记录基于搜索码值顺序排列。包括索引顺序文件和 B+树索引文件等。

顺序索引主要用于支持快速地对文件中的记录进行顺序或随机地访问。

顺序索引的结构: 在索引中按搜索码值的顺序存储索引项, 并将索引项与包含该索引项中搜索码值的文件记录 (或文件块) 关联起来。即一个索引项由两部分组成: 一个搜索码值, 一个指向文件中包含该搜索码值的记录的指针。

(2) 散列索引: 索引中的记录基于搜索码值的散列函数 (也称哈希函数) 的值平均地、随机地分布到若干个散列通中。

(3) 搜索码: 用于在文件中查找记录的属性或属性集。

(4) 索引文件: 建立了索引的文件。

(5) 主索引 (聚集索引): 如果索引文件中的记录按照某个搜索码值指定的顺序物理存储, 那么该搜索码对应的索引就称为主索引。

(6) 辅助索引 (非聚集索引): 搜索码值顺序与索引文件中记录的物理顺序不同的那些索引称为辅助索引。

(7) 对索引技术评价的考虑因素: 访问类型、访问时间、插入时间、删除时间、空间开销。

索引的作用:

索引是加快数据检索的一种工具。

一个基本表可以建立多个索引, 从不同角度加快查询速度;

如果索引建立的较多, 会给数据维护带来较大的系统开销。

常用索引的特点和比较:

(1) 索引顺序文件: 建立了主索引的索引文件。

1.稠密索引: 对应索引文件中搜索码的每一个值在索引中都有一个索引记录 (或索引项)。每一个索引项包含搜索码值和指向具有该搜索码值的第一个数据记录的指针。

2.稀疏索引: 稀疏索引只为索引文件中搜索码的某些值建立索引记录 (或索引项)。

3.多级索引

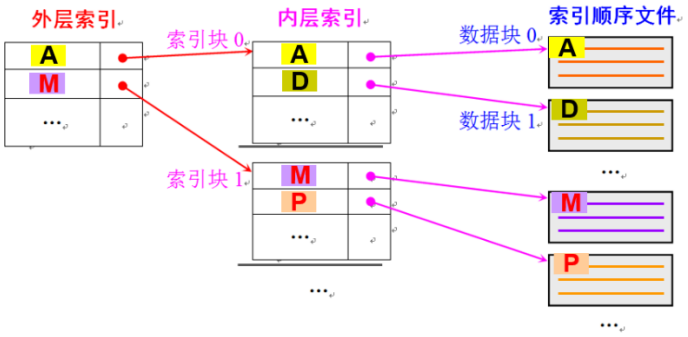
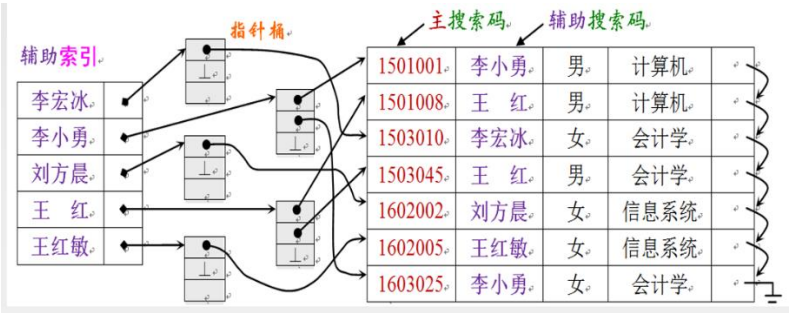


图 8-12　二级稀疏索引结构

4.辅助索引：辅助索引必须是稠密索引。将数据文件中具有该搜索码值的所有记录的指针存放在一个指针桶中，索引项中的指针域再存放指向指针桶的指针。



（2）B+树索引：

叶节点的结构：每个叶节点最多可存放 n-1 个搜索码值，最少也要存放 n/2（向上取整）-1 个搜索码值，并要使 B+树索引称为稠密索引。

非叶节点的结构：B+树索引中的非叶节点形成叶结点上的一个稀疏索引。搜索码值数和叶节点相同。

（3）散列索引：

散列索引将搜索码值及其相应的文件记录指针组织成一个散列索引项。

构建方法：①将散列函数作用于一条文件记录的搜索码值，以确定该文件记录所对应的散列索引项的散列桶；②将由该搜索码值以及相应文件记录指针组成的散列索引项存入散列桶中。

散列索引只能是一种辅助索引结构。

（4）比较：

散列其实就是一种不通过值的比较，而通过值的含义来确定存储位置的方法，它是为有效地实现等值查询而设计的。

基于散列技术不支持范围检索。

基于 B+树的索引技术能有效地支持范围索引，并且它的等值检索效果也很好。

但是，散列技术在等值连接等操作中是很有用的，尤其是在索引嵌套循环连接方法中，基于散列地索引和基于 B+树地索引在代价上的差别会很大。

查询处理的过程：

（1）基本步骤：①语法分析与翻译；②查询优化；③查询执行。

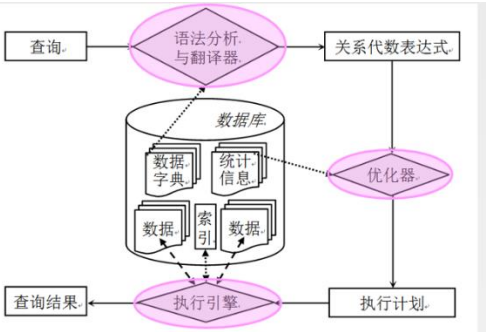


图 8-22　查询处理的过程

（2）语法分析与翻译器：

①检查用户查询地语法，利用数据字典验证查询中出现的关系名和属性名等是否正确；

②构造该查询语句地语法分析树表示，并将其翻译成关系代数表达式。

（3）查询执行计划与查询优化器：

执行一个查询，不仅需要提供关系代数表达式，还要对该关系代数表达式加上注释来说明如何执行每个关系运算。生产查询执行计划。

不同地查询执行计划会有不同的代价。构造具有最小查询执行代价的查询执行计划称为查询优化，由查询优化器来完成。

查询优化是影响 RDBMS 性能的关键因素。

（4）查询执行引擎

查询执行引擎根据输入的查询执行计划，调用相关算法实现查询计算，并将计算结果返回给用户。

有效地对内存缓冲区进行管理是影响查询执行性能的非常重要的方面。

查询代价的度量：

磁盘存取时间（最重要的代价）、执行一个查询所用 CPU 时间以及在并行/分布式数据库系统中的通信开销

查询优化的过程：

①逻辑优化，即产生逻辑上与给定关系代数表达式等价的表达式；

②代价估计，即估计每个执行计划的代价；

③物理优化，即对所产生的表达式以不同方式做注释，产生不同的查询执行计划。

逻辑优化和物理优化的概念：

逻辑优化：系统尝试找出一个与给出的查询表达式等价，但执行效率更高的查询表达式；

物理优化：为处理查询选择一个详细的策略。

启发式优化规则：

尽早执行选择操作；

尽早执行投影运算；

索引的设计：

常见的存储方式：索引方法、聚集方法和 Hash 方法。目前使用最普遍的是 B+树索引。

特点：

（1）存取路径和数据是分离的；

（2）存取路径可以由用户建立和删除，也可以由系统动态建立和删除；

（3）存取路径的物理组织可以采用顺序文件、B+树文件或 Hash 文件。

数据库安全性：

数据库安全保护的目标是确保只有授权用户才能访问数据库，而所有未被授权的人员则无法接近数据库。通常，安全措施是指计算机系统中用户直接或通过应用程序访问数据库所要经过的安全认证过程。

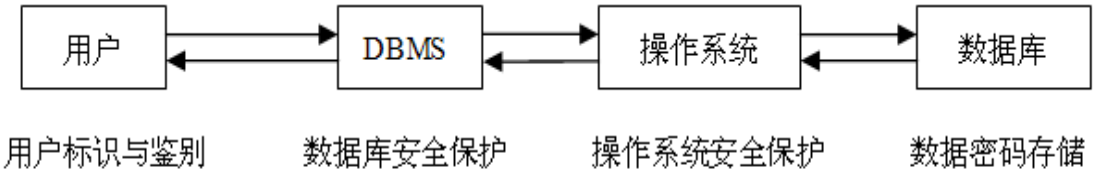


图 9-1　数据库安全认证过程

（1）用户标识与鉴别

当用户访问数据库时，要求先将其用户名与口令或密码提交给数据库管理系统进行认证。只有在确定其身份合法后，才能进入数据库进行数据存取操作。

（2）数据库安全保护

①存取控制：存取控制机制主要包括两部分：定义用户权限，并将用户权限登记到数据字典中；合法权限检查。

②视图：视图机制能隐藏用户无权存取的数据，从而自动地对数据库提供一定程度的安全保护。

视图主要功能提供数据库的逻辑独立性。

③审计：审计是一种监视措施，用于跟踪并记录有关数据的访问活动。审计追踪把用户对数据库的所有操作自动记录下来，放在审计日志中。

审计日志内容：操作类型，操作终端标识与操作者标识，操作日期和时间，操作所涉及的相关数据，数据库的前映像和后映像。

（3）操作系统安全保护

通过操作系统提供的安全措施来保证数据库的安全性。

（4）数据密码存储

数据加密是防止数据库中数据存储和传输失密的有效手段。

基本思想：明文->密文，以密文方式存储和传输。

（5）安全标准

①主体和客体

主体是指数据库的访问者，包括用户、进程和线程等。

客体是指数据库中的数据和载体，如基本表、视图、存储过程和数据文件等。

②自主存取控制

是一种基于存取矩阵的存取控制模型。

3 元素组成：主体、客体和存取操作。

③强制存取控制

④隐蔽通道

⑤数据库安全的形式化

⑥访问监控器

⑦安全标准简介

目标权限授权与回收：

[例9.14] 将基本表Score的若干权限分别授予用户u1、u2、u3、u4、u5和u6。

- 将表Score的所有权限授予用户u1，且可以转授权限

```
GRANT all ON Score TO u1 WITH GRANT OPTION
```

- 用户u1将表Score的所有权限授予用户u2，且可以转授权限

```
GRANT all ON Score TO u2 WITH GRANT OPTION
```

- 用户u2将表Score的查询和插入权限授予用户u5，且不可转授权限

```
GRANT select, insert ON Score TO u5
```

- 用户u2将表Score的所有权限授予用户u4，且可以转授权限

```
GRANT all ON Score TO u4 WITH GRANT OPTION
```

- 用户u4将表Score的查询和删除权限授予用户u6，且可以转授权限

```
GRANT select, delete ON Score TO u6 WITH GRANT OPTION
```

[例9.15] 用户u2将转授给用户u4的对表Score的修改和查询权限收回：

```
REVOKE select, update ON Score FROM u4 CASCADE
```

- 本例必须级联收回，因为用户u4将对表Score的查询和删除权限转授给了用户u6。

■ **[例9.16]** 用户u4将转授给用户u6的对表Score的查询权限收回：

```
REVOKE select ON Score FROM u6
```

数据库完整性概念：

(1) DBMS 必须提供的功能：

完整性约束条件定义机制；

完整性检查方法；

违约处理措施。

(2) 完整性约束作用对象：关系、元组、列。

(3) 完整性约束可分为：

①静态约束和动态约束

②或立即执行约束和延迟执行约束

数据库完整性实现：

（1）实体完整性：实体完整性要求基本表的主码值唯一且不允许为空值。在 CRETE TABLE 中使用 PRIMARY KEY 或者在 ALTER TABLE 中使用 ADD PRIMARY KEY 实现。

```
CREATE TABLE Score (
    ...
    /* 主码由3个属性构成，必须作为元组约束进行定义 */
    PRIMARY KEY (studentNo, courseNo, termNo)   -- 定义匿名的元组约束
```

（2）参照完整性：参照完整性为若干个表中的相应元组建立多对一的参照关系。在 CRETE TABLE 中使用 FOREIGN KEY 或者在 ALTER TABLE 中使用 ADD FOREIGNKEY 实现。

```
CREATE TABLE Score (
    ...
    /* 外码约束只能定义为表约束，studentNo是外码，被参照表是Student */
    FOREIGN KEY (studentNo) REFERENCES Student (studentNo), --匿名的表约束
    /* 外码约束只能定义为表约束，courseNo是外码，被参照表是Course */
    FOREIGN KEY (courseNo) REFERENCES Course (courseNo),    --匿名的表约束
    /* 外码约束只能定义为表约束，termNo是外码，被参照表是Term */
    FOREIGN KEY (studentNo) REFERENCES Term (studentNo)     --匿名的表约束
)
```

（3）用户自定义完整性：使用 CHECK 实现。

```
age         tinyint   DEFAULT 16   NULL
  CONSTRAINT ageCK CHECK ( age > 0 AND age < 60 ),
```

（4）完整性约束的修改

■ 删除约束：

```
ALTER TABLE <tableName>
      DROP CONSTRAINT <constraintName>
```

■ 添加约束：

```
ALTER TABLE <tableName>
   ADD CONSTRAINT <constraintName>
     < CHECK | UNIQUE | PRIMARY KEY | FOREIGN KEY >
          (<constraintExpr>)
```

■ 其中

- <tableName>为欲修改约束所在的基本表的名称；
- <constraintName>为欲修改的约束的名称。

事务概念：

对于用户而言，事务是具有完整逻辑意义的数据库操作序列的集合。

对于数据库管理系统而言，事务则是一个读写操作序列。

事务提交：将成功完成事务的执行结果永久化，并释放事务占有的全部资源。

事务回滚：中止当前事务、撤销其对数据库所做的更新，并释放事务占有的全部资源。

事务特性：

原子性，一致性，隔离性，持久性。

原子性也称为故障原则性或故障可靠性；

一致性也称为并发原子性或事务正确性；

隔离性也称执行原子性或并发正确性或可串行化；

持久性也称为恢复原子性或恢复可靠性。

事务实现方式：

事务开始：BEGIN TRANSACTION

事务提交：COMMIT TRANSACTION，COMMIT WORK

事务回滚：ROLLBACK TRANSACTION，ROLLBACK WORK

事务并发带来的问题：

（1）读脏数据：如果事务 T2 读取事务 T1 修改但未提交的数据后，事务 T1 由于某种原因中止而撤销，这时事务 T2 就读取了不一致的数据。

（2）不可重复读：是指事物 Ti 两次从数据库中读取的结果不同。（三种情况，增删改）

（3）丢失更新：两个或多个事务都读去了同一数据值并修改，最后提交的事务的执行结果覆盖了前面事务提交的执行结果，从而导致前面事务的更新被丢失。

事务调度正确性准则：

（1）事务调度定义：

将由多个事务操作组成的随机执行序列称为一个调度。

对由一组事务操作组成的调度序列而言，应满足下列条件：

①该调度应包括该组事务的全部操作；

②属于同一个事务的操作应保持在原事务中的执行顺序

（2）冲突可串行化：

如果一调度 S 与一串行调度是冲突等价的，则称 S 是冲突可串行化的。

冲突可串行化调度执行结果一定是正确的，而正确的调度不一定都是冲突可串行化的。

（3）冲突可串行化判别准则：

如果优先图中无环，则 S 是冲突可串行化的，反之 S 是非冲突可串行化的。

死锁的概念：

持有锁的事物出现相互等待都不能继续执行。

备份的作用：

数据保护：防止数据丢失，如硬件故障、软件错误、人为操作失误、病毒攻击或恶意破坏等情况导致的数据丢失。

灾难恢复：在发生自然灾害、系统崩溃或其他意外事件导致数据库不可用时，备份可以用于恢复数据，确保业务的连续性。

版本控制：备份可以用来保存不同时间点的数据状态，有助于追踪数据的变更历史，或者在需要时恢复到特定的历史状态。

法律和合规要求：某些行业或组织可能需要遵守特定的数据保留政策，备份可以帮助满足这些法律或合规要求。

数据迁移和测试：在数据库迁移到新系统或在开发环境中进行测试时，备份可以作为数据源，确保迁移或测试的准确性。

性能优化：对于一些只读或历史数据，可以通过备份将数据转移到辅助存储系统中，从而优化在线事务处理系统的性能。

安全和审计：备份可以作为安全措施的一部分，帮助检测和恢复数据篡改。同时，备份日志可以用于审计和监控数据库的变更。

日志概念:

日志是 DBMS 记录数据库全部更新操作的序列文件。

日志作用（特点）:

日志文件记录了数据库的全部更新顺序。

日志文件是一个追加（append-only）文件。

DBMS 允许事务的并发执行导致日志文件是“交错的”。

属于单个事务的日志顺序与该事务更新操作的执行顺序是一致的。

日志记录通常是先写到日志缓冲区中，然后写到稳固存储器（如磁盘阵列）中。