# 数据库原理与应用期末复习提纲

## 1 绪论

### 1.1 数据库系统概述

#### 1.1.1 基本概念

数据库（DataBase, DB）

数据库管理系统（DataBase Management System, DBMS）

数据库系统（DataBase System, DBS）

#### 1.1.2 数据管理技术的发展

人工管理阶段 -> 文件系统阶段 -> 数据库系统阶段

#### 1.1.3 数据库系统的特点

① 数据结构化；

② 数据的共享性高、冗余度低且易扩充；

③ 数据独立性高：物理独立性（数据物理储存）、逻辑独立性（数据库逻辑结构）；

④ 数据由数据库管理系统统一管理和控制：数据安全性保护、数据完整性检查、并发控制、数据库恢复。

### 1.2 数据模型

#### 1.2.1 数据模型的三要素

① 数据结构；

② 数据操作；

③ 数据的完整性约束。

#### 1.2.2 概念模型

**基本概念**：实体、属性、码、实体型、实体集、联系。

**表示方法**：实体-联系方法（Entity-Relationship approach）使用E-R图描述现实世界的概念模型。E-R方法又称E-R模型。

实体1

实体2

联系

m

n

#### 1.2.3 逻辑模型

逻辑模型包括层次模型、网状模型、关系模型、面向对象数据模型等。它是按计算机系统的观点对数据建模，主要用于数据库管理系统的实现。

**关系模型的基本概念**：关系、元组、属性、码、域、分量。

**关系模式**：关系名（属性1，属性2，… ，属性n）。

### 1.3 数据库系统的结构

#### 1.3.1 三级模式结构

三级模式结构：外模式、模式、内模式。

**模式**：也称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。

**外模式**：也称子模式或用户模式，是数据库用户的局部数据视图。

**内模式**：也称储存模式，是数据物理结构和储存方式的描述。

数据库

内模式

模式

外模式a

外模式b

外模式c

应用组1

应用组2

应用组3

内模式/模式映像

/moshi

外模式/模式映像

/moshi

#### 1.3.2 二级映像

外模式/模式映像、内模式/模式映像保证了数据库系统中数据的逻辑独立性和物理独立性。

## 2 关系数据库

### 2.1 关系数据结构及形式化定义

#### 2.1.1 关系

**域**：一组具有相同数据类型的值的集合。

**笛卡尔积**：给定一组域，允许有相同项，则的笛卡尔积为：

其中每一行元素叫做一个**元组**，简称**元组**，元组中每一个值叫做一个**分量**。

**基数**：一个域允许的不同取值个数。几个域的笛卡尔积的元组数等于各个域的基数之积。

**关系**：的子集叫做在域上的关系，表示为。

**候选码**：能唯一标识一个元组，而其子集不能的属性组。若一个关系又多个候选码，则选取其中一个为**主码**。候选码的属性称为**主属性**；不在候选码中的属性称为**非主属性**或**非码属性**。

### 2.2 关系操作

#### 2.2.1 集合运算

**并**：。

**差**：。

**交**：。

**广义笛卡尔积**：。

#### 2.2.2 关系运算

关系运算运算符：大于，小于，大于等于，小于等于，等于，不等于，非，与，或。

**选择**：，其中为选择条件。

**投影**：，其中为中的属性列。

**连接**：，其中和分别为和上列数相等且可比的属性列，是比较运算符。连接运算从中选取关系在属性组上的值与关系在属性组上的值满足比较关系的元组。

为“”的连接运算称为**等值连接**。即从关系和的广义笛卡尔积中选取、属性值相等的那些元组。

**自然连接**是一种特殊的等值连接。他要求两个关系中进行比较的分量必须是同名属性组，并在结果中把重复的属性列去掉。记作。

### 2.3 关系的完整性

#### 2.3.1 实体完整性

若属性是基本关系的主属性，则非空。

#### 2.3.2 参照完整性

设是基本关系的一个或一组属性，但不是的码，是基本关系的主码。如果与相对应，则称是的外码，并称基本关系为参照关系，基本关系为被参照关系或目标关系。关系和不一定是不同的关系。

对于中每个元组在上的值必须为空值或等于中某个元组的主码值。

#### 2.3.3 用户定义的完整性

非空、唯一、满足特定条件等。

## 3 关系数据库标准语言SQL

### 3.1 SQL的特点

① 综合统一；

② 高度非过程化；

③ 面向集合的操作方式；

④ 以同一种语法结构提供多种使用方式；

⑤ 语言简洁，易学易用。

### 3.2 数据定义

|  |  |  |  |
| --- | --- | --- | --- |
| 操作对象 | 操作方式 | | |
| 创建 | 删除 | 修改 |
| 模式 | CREATE SCHEMA | DROP SCHEMA |  |
| 表 | CREATE TABLE | DROP TABLE | ALTER TABLE |
| 视图 | CREATE VIEW | DROP VIEW |  |
| 索引 | CREATE INDEX | DROP INDEX | ALTER INDEX |

|  |  |
| --- | --- |
| 常用数据类型 | 含义 |
| CHAR(*n*)  CHARACTER(*n*) | 长度为*n*的定长字符串 |
| VARCHAR(*n*)  CHARACTERVARYING(*n*) | 最大长度为*n*的变长字符串 |
| INT  INTEGER | 长整数（4字节） |
| SMALLINT | 短整数（4字节） |
| BIGINT | 大整数（8字节） |
| FLOAT(*n*) | 精度至少为*n*位数字的浮点数 |
| BOOLEAN | 逻辑布尔量 |
| DATA | 日期，YYYY-MM-DD |
| TIME | 时间，HH:MM:SS |
| TIMESTAMP | 时间戳类型 |

### 3.3 数据查询

SELECT [ALL|DISTINCT] <COLUMN\_NAME1> [, <COLUMN\_NAME2>] …

FROM <TABLE\_OR\_VIEW\_NAME> [, <TABLE\_OR\_VIEW\_NAME>] | (SELECT STATEMENT) [AS] <CNAME>

[WHERE <CONDITIONAL\_EXPREESION>]

[GROUP BY <COLUMN\_NAME> [HAVING <CONDITIONAL\_EXPREESION>]]

[ORDER BY <COLUMN\_NAME> [ASC | DESC];

### 3.4 数据更新

#### 3.4.1 插入数据

INSERT INTO <TABLE\_NAME> [(<COLUMN\_NAME1> [, <COLUMN\_NAME2>] …)]

VALUES (<CONSTANT1> [, < CONSTANT2>] …);

#### 3.4.2 修改数据

UPDATE <TABLE\_NAME>

SET <COLUMN\_NAME1> = <EXPREESION1> [, <COLUMN\_NAME2> = <EXPREESION2>] …

[WHERE <CONDITIONAL\_EXPREESION>];

#### 3.4.3 删除数据

DELETE FROM <TABLE\_NAME>

[WHERE <CONDITIONAL\_EXPREESION>];

### 3.5 空值的处理

#### 3.5.1 空值产生

缺省或 "NULL" 关键字。

#### 3.5.2 空值判断

使用 "IS NULL" 或 "IS NOT NULL" 判断是否为空值。

### 3.6 视图

#### 3.6.1 视图的建立与删除

CREATE VIEW <VIEW\_NAME> [(<COLUMN\_NAME1> [, <COLUMN\_NAME2>] …)]

AS <SELECT\_EXPREESION>

[WITH CHECK OPTION];

DROP VIEW <VIEW\_NAME> [CASCADE];

#### 3.6.2 视图和表的区别

表是内模式，视图是外模式。

表只用物理空间而视图不占用物理空间，视图只是逻辑概念的存在，表可以及时对它进行修改，但视图只能有创建的语句来修改。

视图是查看数据表的一种方法，可以查询数据表中某些字段构成的数据，只是一些SQL语句的集合。从安全的角度说，视图可以不给用户接触数据表，从而不知道表结构。

表属于全局模式中的表，是实表；视图属于局部模式的表，是虚表。

视图的建立和删除只影响视图本身，不影响对应的基本表。

#### 3.6.3 视图的作用

① 简化用户操作；

② 使用户从多个角度看待同一数据；

③ 对重构数据库提供了一定程度的逻辑独立性；

④ 对机密数据提供安全保护；

⑤ 更清晰地表达数据。

## 4 数据库的安全性和完整性

### 4.1 安全性

#### 4.1.1 授权

GRANT <权限> [,<权限>]...

ON <对象类型> <对象名> [, <对象类型> <对象名>]...

TO <用户> [,<用户>]... [WITH GRANT OPTION]

#### 4.1.2 回收

REVOKE <权限> [,<权限>]...

ON <对象类型> <对象名> [, <对象类型> <对象名>]...

FROM <用户> [,<用户>]... [CASCADE|RESTRICT]

### 4.2 完整性

实体完整性；

参照完整性；

用户定义的完整性；

完整性约束命名子句：

CONSTRAINT <完整性约束条件名> [ PRIMARY KEY...|FOREIGN KEY...|CHECK...|NOT NULL|UNIQUE]

## 5 关系数据库理论

### 5.1 函数依赖

设是属性集上的关系模式。是的子集。若对于的任意一个可能的关系，中不可能存在两个元组在上的属性值相等,而在上的属性值不等,则称函数确定或函数依赖于，记作。称为这个函数依赖的决定属性组。

如果，，则称为**平凡的函数依赖**，否则为**非平凡的函数依赖**。

关系模式中，如果，而且不依赖于的任何一个真子集，则称**完全函数依赖**于，记作。若不完全函数依赖于，则称**部分函数依赖**于，记作。

在中，如果(不属于且不依赖于)，,且不属于，则称**传递函数依赖**于。记作。

设为关系模式中的属性或属性组合。若，则称为的一个侯选码。若关系模式有多个候选码，则选定其中的一个作为主码。

### 5.2 范式

如果一个关系模式的所有属性都是不可分的基本数据项，则。

若关系模式，且每一个非主属性都完全函数依赖于的码，则。

若，且每个非主属性都不传递依赖于码，则。

对关系模式，若且不属于时，必含有码，则。

如果，且只有一个候选码，则。

## 6 数据库设计

需求分析阶段：分析客户的业务和数据处理需求。

概念结构设计阶段：设计数据库的E-R模型图，确认需求信息的正确和完整。

逻辑结构设计阶段：应用范式审核数据库结构。

物理结构设计阶段：物理实现数据库。

数据库实施。

数据库运行和维护。

其中前两个阶段独立于DBMS，后四个阶段与DBMS有关。

## 7 数据库恢复与并发

### 7.1 事务

事务是系统或用户定义的一组操作序列。这些操作要么全做，要么全不做，是一个不可分割的工作单位。事务是恢复和并发控制的基本单位。

事务的四大特性（ACID）：

① 原子性(Atomicity)：事务是不可分割的数据库逻辑工作单位。事务中的所有操作要么成功执行，要么都不执行。

② 一致性(Consistency)：事务执行的结果应保证DB从一个一致性状态（正确状态）转到另一个一致性状态。

③ 隔离性(Isolation)：一个事务的执行不能被其它事务干扰。

④ 持续性(Durability)：一旦事务成功提交，对数据库的影响应是持久的。

### 7.2 故障种类

① 事务内部故障：溢出、发生死锁等。

② 系统故障：操作系统或DBMS代码错误、硬件错误、断电等。

③ 介质故障：磁盘损坏、磁头碰撞、瞬时强磁场干扰等。

④ 计算机病毒。

### 7.3 数据转储

转储是指DBA将整个数据库复制到磁带、磁盘或其他存储介质上保存起来的过程。备用的数据称为后备副本或后援副本。

**静态转储**：系统中无运行事务时执行的转储。转储期间不允许对数据库有任何存取、修改活动。得到的一定是一个数据一致性的副本。降低了数据库的可用性。

**动态转储**：转储操作与用户事务并发进行。转储期间允许对数据库进行存取或修改。不能保证副本中的数据正确有效。

利用动态转储得到的副本进行故障恢复：需要把动态转储期间各事务对数据库的修改活动登记到日志文件，后备副本加上日志文件才能把数据库恢复到某一时刻的正确状态。

**海量转储**：每次转储全部数据库。

**增量转储**：只转储上次转储后更新过的数据。

### 7.4 日志文件

日志文件是用来记录事务对数据库**更新操作**的文件。

事务故障恢复和系统故障恢复必须用日志文件。

在动态转储方式中必须建立日志文件，后备副本和日志文件结合起来才能有效地恢复数据库。在静态转储方式中，也可以建立日志文件。

### 7.5 故障恢复策略

事务故障：

撤消事务（UNDO）

系统故障：

事务未提交：撤消未完成事务（UNDO）

事务已提交：重做已完成事务（REDO）

介质故障：

重装后备副本

重做已完成事务（REDO） 恢复策略

### 7.6 并发操作

并发操作带来的三类数据不一致性：

① 丢失修改：写-写冲突；

② 不可重复读：读-写冲突；

③ 读脏数据：写-读冲突。

### 7.7 封锁类型与协议

**排它锁**：若事务T对数据对象A加上X锁，则只允许T读取和修改A，其它任何事务都不能再对A加任何类型的锁，直到T释放A上的锁。保证T释放A上的锁之前其它事务不能再读取和修改A。

**共享锁**：若事务T对数据对象A加上S锁，则事务T可以读A但不能修改A，其它事务只能再对A加S锁，而不能加X锁，直到T释放A上的S锁。共享锁保证其它事务可以读A，但在T释放A上的S锁之前不能对A做任何修改。

**一级封锁协议**：事务T在修改数据R之前必须先对其加X锁，直到事务结束才释放。一级封锁协议可防止丢失修改。一级封锁协议中，如果仅仅是读数据不对其进行修改，是不需要加锁的，所以它不能保证可重复读和不读“脏”数据。

**二级封锁协议**：一级封锁协议加上事务T在读取数据R之前必须先对其加S锁，读完后即可释放S锁。二级封锁协议可以防止丢失修改和读“脏”数据。二级封锁协议中，由于读完数据后即可释放S锁，所以它不能保证可重复读。

**三级封锁协议**：一级封锁协议加上事务T在读取数据R之前必须先对其加S锁，直到事务结束才释放。三级封锁协议可防止丢失修改、读脏数据和不可重复读。

**活锁**：某个事务永远处于等待封锁的状态。

活锁的避免：采用“先来先服务”策略。

**死锁**：当事务出现循环等待时，如不加干预，则会一直等待下去，形成死锁。

死锁的诊断与解除：超时法、等待图法。

### 7.8 串行调度和并发调度

**串行调度**：属于同一事务的操作紧挨在一起，对n个事务，可以有n!个有效调度。

**并行调度**：来自不同事务的操作可以交叉执行。

**可串行化调度**：多个事务的并行执行是正确的，当且仅当其结果与按某一次序串行执行这些事务时的结果相同。

**两段封锁协议**：所有事务必须分两个阶段对数据项加锁和解锁：

① 对任何数据进行读写操作之前，事务首先要申请并获得对该数据的封锁；

② 释放一个封锁之后，事务不再申请和获得任何其他封锁。

两段锁的含义：事务分为两个阶段，第一阶段是获得封锁，也称为扩展阶段；第二阶段是释放封锁，也称为收缩阶段。

遵守两段锁协议的事务也可能会发生死锁。