目录

[1 题型 1](#_Toc105667621)

[2 基本概念 1](#_Toc105667622)

[2.1 三层模式-两级映射 1](#_Toc105667623)

[2.2 连接 3](#_Toc105667624)

[2.3 各类并发控制协议 3](#_Toc105667625)

[3 总结 3](#_Toc105667626)

[3.1 根据功能需求设计ER图 3](#_Toc105667627)

[3.2 写关系代数表达式 4](#_Toc105667628)

[3.3 ER图转关系模型 4](#_Toc105667629)

[3.4 候选码 4](#_Toc105667630)

[3.5 最小函数依赖集 5](#_Toc105667631)

[3.6 无损连接性的判断方式 5](#_Toc105667632)

[3.7 保持函数依赖的判断方式 7](#_Toc105667633)

[3.8 BCNF分解 7](#_Toc105667634)

[3.9 3NF分解 7](#_Toc105667635)

[3.10 代数优化 7](#_Toc105667636)

[3.11 SQL语句例题 8](#_Toc105667637)

1. 题型

* 单选题（2分\*10）
* 简答题（5分\*4）
  + 主码、候选码、超码的区别
  + 某一种特殊数据库的特点分析
* 综合应用题（6分\*5）
  + 根据功能需求绘制ER图
  + ER图转关系模型
  + 根据功能需求写SQL语句（实验一）
  + 无损分解和保持函数依赖的判断
  + BCNF、3NF的分解
  + 代数优化三大法则的应用
  + 事务并发控制，三大技术，六小技术的应用分析
  + 日志恢复技术

1. 基本概念
   1. 三层模式-两级映射

图示

描述已自动生成

**外模式：**

1）数据库的用户使用的局部数据的逻辑结构和特征的描述

2）数据库用户的数据视图，是与某一应用程序有关的数据的逻辑表示。（如应用程序A只能看见其相对于的外模式1，应用程序B只能看见其相对于的外模式2，不能看见不属于自己的外模式。相当于是模式的一个子集）

外模式的地位：介于模式与应用之间。

模式与外模式的关系：一对多

1）外模式是模式的子集

2）一个数据库可以有多个外模式，反应了不同的用户的应用需求、看待数据的方式、对数据保密的要求。

3）对于模式中的同一个数据，不同外模式可以对数据的长度、类型等有不同的定义。

外模式与应用的关系：一对多。

1）同一外模式可以为某一个用户的多个应用系统所使用

2）但一个应用程序只能使用一个外模式

外模式的用途：

1）保证数据库安全，每个用户只能看见自己对应外模式的数据

2）保证数据独立性。

**模式概念，也称为模式、逻辑模式**：

模式的定义就是对数据的逻辑结构进行定义（包括数据项的名字、类型、取值范围等）。

1)它是数据库中全体数据的逻辑结构和特征的描述；（例如在学生选课数据库模式中，包含学生记录、课程记录和学生选课记录，这就是一个逻辑模式）

2)模式是所有用户的公共数据视图。因为数据库是多人共享使用的，模式就是大家都看到的样子，综合了所有用户的数据需求。

注意：

1）一个数据库只有一个模式。

2）与数据库的物理存储无关。

3）与具体的应用程序、开发工具和开发语言无关。（意思就是可以使用各种开发工具和开发语言来使用数据库）

**内模式：**

1）数据物理结构和存储方式的描述

2）是数据在数据库内部的表示方式

Ⅰ.记录的存储方式：如顺序存储，按B树结构存储，Hash存储）

Ⅱ.索引的组织方式：B+树索引，hash索引，Join index索引

Ⅲ.数据是否压缩存储

Ⅳ.数据是否加密

注：一个数据库只有一个内模式。

**外模式/模式映射：定义外模式与模式之间的对应关系。**

每一个外模式都对应一个外模式/模式映射，在外模式中规定了从模式中如何分离出。

外模式/模式映像的作用：

保证了数据的逻辑独立性。当模式改变时，数据库管理员修改有关的外模式/模式**映射**，使外模式保持不变；而应用程序是根据数据的外模式编写的，从而应用程序不必修改，保证了数据与应用程序的逻辑独立性，简称为数据的逻辑独立性。

**模式/内模式映像：**

定义了数据全局逻辑结构与存储结构之间的对应关系（例如说明逻辑记录和字段在内部是如何表示的）。

数据库中模式/内模式映像是唯一的，定义在模式当中。

用途：保证了数据的物理独立性。

1）当数据的存储结构改变时（如本来我们用堆存储，改成了B+树），数据库管理员修改模式/内模式映像，使模式保持不变。

2）应用程序不受影响，保证了数据与程序的物理独立性，简称为数据的物理独立性。

**总结：外模式/模式映像保证了当模式改变时，外模式不用变 （逻辑独立性），模式/内模式映像保证了当内模式改变时，模式不用变 （物理独立性）。**

* 1. 连接

文本, 信件

描述已自动生成

等值连接：

select \* from R,S where R.A=S.B;

select \* from R join S on R.A=S.B;

自然连接

select \* from R natural join S;

select \* from R join S using (A);

外连接

select \* from R natural left outer join S;

select \* from R natural right outer join S;

select \* from R natural full outer join S;

笛卡尔积：

select \* from R cross join S;

* 1. 各类并发控制协议
* **两阶段锁定：**用于可接受单一粒度的简单应用程序。如果存在大型只读事务，多版本两阶段协议会做得更好。此外，如果必须不惜一切代价避免死锁，则树协议将更可取
* **具有多个粒度锁定的两阶段锁定：**用于应用程序组合，其中某些应用程序访问单个记录，而其他应用程序访问整个关系或其主要部分。上面提到的2PL的缺点也适用于这个。
* **树形协议：**如果所有应用程序都倾向于以与特定部分顺序一致的顺序访问数据项，则使用此协议。该协议没有死锁，但交易通常必须锁定不需要的节点才能访问所需的节点。
* **时间戳排序：**如果应用程序需要等效于特定串行排序（例如，到达顺序）的并发执行，而不是任何串行排序，请使用。但是，冲突是通过回滚事务而不是等待来处理的，并且计划是不可恢复的。为了使它们可恢复，必须容忍额外的开销和增加的响应时间。如果存在长只读事务，则不适合，因为它们会挨饿。不存在死锁。
* **有效性检查：**如果两个并发执行事务冲突的可能性较低，则可以有利地使用此协议，以获得更好的并发性和良好的响应时间，同时降低开销。不适合在高争用的情况下，当大量浪费的工作时会完成。
* **多版本时间戳排序：**如果时间戳排序合适，但读取请求希望从不等待，请使用。但是具有时间戳排序协议一样的缺点。
* **多版本两阶段锁定：**此协议允许只读事务始终提交而无需等待。更新事务遵循 2PL，因此允许通过等待而不是回滚来解决冲突的可恢复计划。但是死锁的问题又回来了，虽然只读事务不能参与其中。但是，保留多个版本会增加空间和时间开销，因此在低冲突情况下，纯 2PL 可能更可取。

1. 总结
   1. 根据功能需求设计ER图

文本, 信件

描述已自动生成

黑板上的卡通

中度可信度描述已自动生成

* 1. 写关系代数表达式

文本, 信件

描述已自动生成 p193

* 1. ER图转关系模型

图示

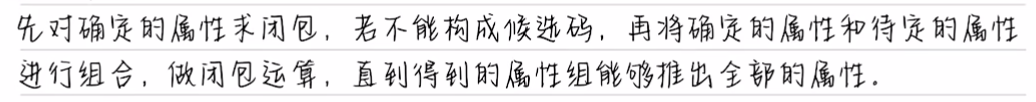
描述已自动生成

* 1. 候选码

注意：候选码不止一个，需要找完，即将所有可能的最小超码全部进行闭包尝试。

文本, 信件

描述已自动生成



* 1. 最小函数依赖集

手机屏幕截图

描述已自动生成

方法二：

①合并右侧

②左右侧最小化

* 1. 无损连接性的判断方式

证明无损连接：如果 R1 ∩ R2 → R1 或 R1 ∩ R2 → R2，则分解 {R1， R2} 是无损连接分解。

证明不是无损连接或当分解为多个模式时：

文本, 信件

描述已自动生成

日历

中度可信度描述已自动生成

日历

低可信度描述已自动生成

* 1. 保持函数依赖的判断方式

方法一：计算子模式的函数依赖集闭包并集，看是否等于父模式的函数依赖集闭包。

方法二：找一个反例（书P208-8.13）

一个更简单的参数如下：F1 不包含箭头右侧的 D 依赖项。F2 不包含箭头左侧的 B 的依赖项。因此，要保留 B → D，必须有一个 FD， B → a在F1+中，a→ D在F2+中（即B→D可用传递性推导）。由于两个方案的交集是 A，因此 a = A.观察 B → A 不在 F1+中，故不满足函数依赖。

* 1. BCNF分解

分解：若U → W不满足BCNF，则分解R为UW和R-W

* 1. 3NF分解

1. 求正则覆盖

记得用两种方法验证

2. 分解子模式

3. 验证任何一个候选码是否在任何子模式中，如果没有，新增子模式

4. 删除为某子模式的子集的子模式

* 1. 代数优化

查询：“张三”老师所在学院的预算和地址。

手机屏幕截图

描述已自动生成 表格

描述已自动生成

select budget, building

from instructor join department using (depe\_name)

where name=”张三”

查询优化：选择下移策略

* 1. SQL冷知识

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

文本

中度可信度描述已自动生成

* 1. 实验一SQL代码

|  |
| --- |
| 实验一 |
| -- 1）创建表  CREATE TABLE sqllab.student  (SID Varchar(20) NOT NULL,   SNAME Varchar(30),   DEPT varchar(20),   AGE int ,   GENDER char(1) check(gender='M' or gender='M' or gender='F') not null default 'F' ,   PRIMARY KEY (SID)  );  -- 为属性选择合适的域、合适的主码和外键约束，并为他们插入所列出数据；  insert into  teacher (TEID,TENAME,DEPT,SALARY) values  ('14001','Katz','CS',75000),  ('14002','Crick','Biology',72000);  -- 2）在student表中，为姓名为’Zhang’且系信息错填为’Physics’的同学修改信息，将其系信息修改为’CS’;  update student  set DEPT='CS'  where SNAME='Zhang' and DEPT='Physics';  -- 3)删除teacher表中，属于Finance学院的教师信息;  delete \* from teacher where DEPT='Finance';  -- 4）在teacher表中，为工资低于或等于70000的教师增长10%的工资，为工资高于70000的教师增长5%的工资。  update teacher  set SALARY = case  when SALARY<=70000 then SALARY\*1.1  else SALARY\*1.05  end;  /\* 2.基本数据查询 \*/  -- 1）基于teacher表，找出“物理系Physics和生物系Biology”所有教师的名字和工资；  (select TENAME, SALARY from teacher where DEPT='Physics')  Union  (select TENAME, SALARY from teacher where DEPT='Biology');  -- 2）基于teacher表，列出教师所在的各个系名，要求每个系名仅出现一次；  select DEPT from teacher group by DEPT;  -- 3）基于teacher表，希望查看“若将每位教师的工资提高20%后”各位教师的姓名和工资是多少；  select TENAME,SALARY\*1.2 as SALARY from teacher;  -- 4）基于表student、SC和course，查看到计算机系CS的每位学生所选课程的所有信息，包括学生姓名、所在系、课程名称、课程学分的情况。  select SNAME,student.DEPT,CNAME,CREDITS from (student natural join sc) join course using (CID);  /\* 3. 复杂数据查询 \*/  -- 1）查询全体学生的姓名、年龄；  select SNAME,AGE from student;  -- 2）查询所有选修过课的学生的学号；  select SID from sc;  -- 3）查询考试成绩低于60分的学生的学号；  select SID from sc where GRADE < 60;  -- 4）查询年龄在20至23之间的学生姓名、性别和年龄；  select SNAME,GENDER,AGE from student where AGE between 20 and 23;  -- 5）查询所有姓liu的学生的学号、姓名和年龄；  -- select SID,SNAME,AGE from student where charindex('liu',SNAME COLLATE Latin1\_General\_CI\_AS)=0;  -- 不缺分大小写  select SID,SNAME,AGE from student where lower(SNAME) like 'liu%';  -- 6) 查询学习C1课程的学生最高分数；  select max(GRADE) from sc where CID='C1';  -- 7) 查询各个课程号与相应的选课人数；  select CID,count(SID) from sc group by CID;  -- 8) 查询选修C3课程的学生的姓名；  select SNAME from (student natural join sc) where CID=C3;  -- 9) 查询每一门课程的平均成绩。  select CID,avg(GRADE) from sc group by CID;  /\* 4、学生只能选择自己学院开设的课程。发现CS学院有的同学选择了其他学院开设的课程。在SC表中删除这些错选的记录。 \*/  -- 查询  select \* from sc where  SID in (select SID from ((student natural join sc) join course using (CID)) where student.DEPT!=course.DEPT)  and  CID in (select CID from ((student natural join sc) join course using (CID)) where student.DEPT!=course.DEPT);  -- 删除  delete from sc where  SID in (select SID from ((student natural join sc) join course using (CID)) where student.DEPT!=course.DEPT)  and  CID in (select CID from ((student natural join sc) join course using (CID)) where student.DEPT!=course.DEPT);  -- 最后检查  select \* from (student natural join sc) join course using (CID); |