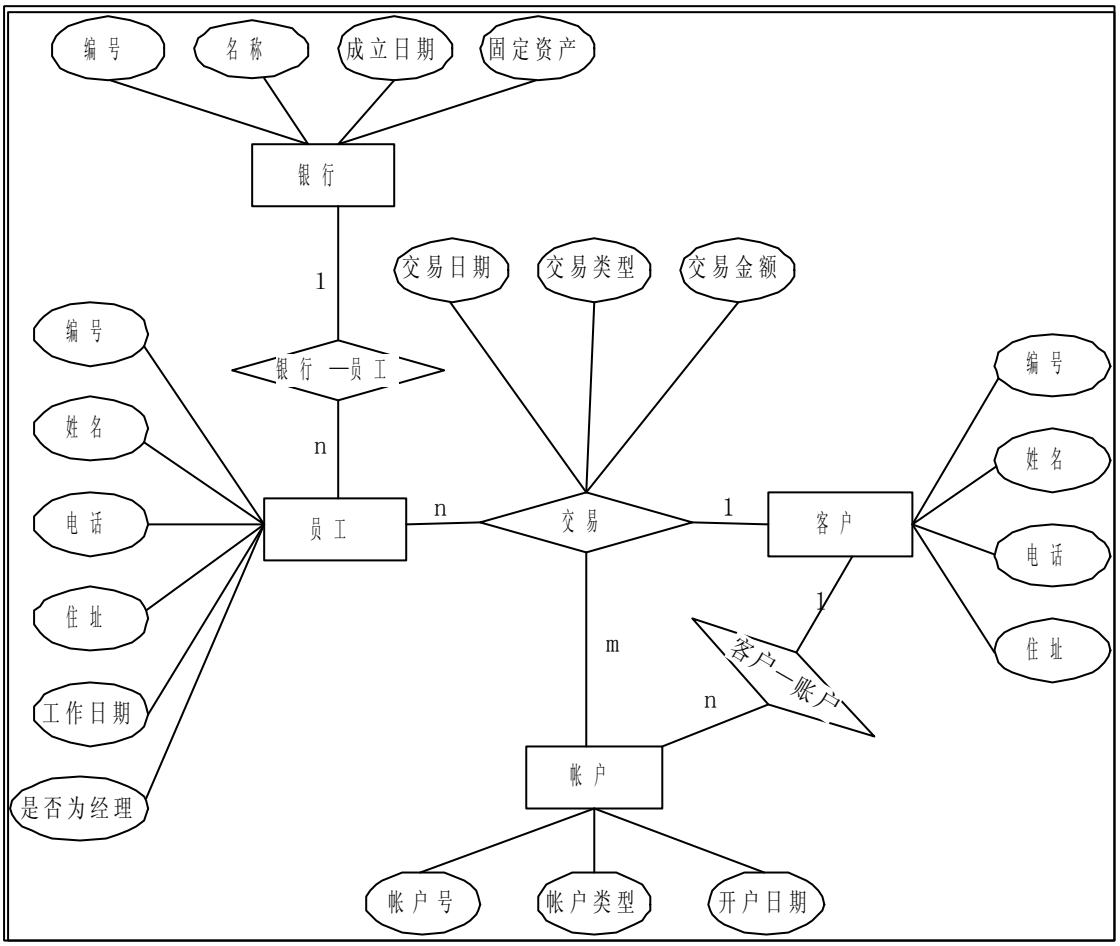
数据库系统原理 2005 试卷

一.

1 .概念模型设计：E-R 图



2 .逻辑模型设计：

银行(银行编号，银行名称，成立日期，固定资产)

Bank(Bno,Bname,Bdate,Bmoney)

员工(员工编号，银行编号，员工姓名，员工电话，员工住址，工作日期，是否为经理，归属经理)

Employee(Eno,Bno,Ename,Ephone,Eaddr,Edate,Ismanager,Manager)

客户(客户编号，客户姓名，客户电话，客户住址)

Client(Cno,Cname,Cphone,Caddr)

帐户(帐户号，客户编号，帐户类型，开户日期)

Account(Ano,Cno,Atype,Adate)

交易(员工编号，客户编号，帐户编号，交易日期，交易类型，交易金额)

Action(Eno,Cno,Ano,Acdate,Actype,Acmoney)

3.索引设计：

```
//查询银行情况
CREATE CLUSTER INDEX Bindex ON Bank(Bno);

//查询员工情况
CREATE CLUSTER INDEX Eindex ON Employee(Eno);

//根据客户姓名查询交易活动
CREATE VIEW CNview
AS
SELECT Cname,Action.* FROM Client,Action
WHERE Client.Cno=Action.Cno;
CREATE INDEX CNindex ON CNview(Cname);

//根据很行员工编号查询交易活动
CREATE INDEX ENindex ON Action(Eno);
```

二.

(注：题目中的学生关系应该用 S (SNO,SN,AGE,SEX,SDEPT))

用 SQL,关系代数和元组关系演算语言完成操作：

1 .SQL: SELECT SN
FROM S
WHERE SDEPT='计算机' AND AGE<20;

关系代数：$\pi_{SN}(\sigma_{SDEPT='计算机' \wedge AGE<20}(S))$

元组关系演算：GET W(S.SN):S.SDEPT='计算机'∧S.AGE<20

2 .SQL: SELECT CN
FROM C,T,SC
WHERE C.CNO=SC.CNO AND T.ENO=SC.ENO AND T.EN='李明';

关系代数：$\pi_{CN}(\sigma_{EN='李明'}(T) \triangleright\triangleleft SC \triangleright\triangleleft C)$

元组关系演算：

```
RANGE SC SCX
RANGE T TX
GET W(C.CN): ∃SCX(SCX.CNO=C.CNO ∧
                ∃TX(TX.ENO=SCX.ENO AND TX.EN='李明'))
```

3 .SQL: SELECT SN FROM S SX
WHERE NOT EXISTS(
SELECT * FROM C CX
WHERE NOT EXISTS(
SELECT * FROM SC
WHERE SNO=SX.SNO AND CNO=CX.CNO AND G='A')) ;

关系代数：$\sigma_{SN}((S \triangleright\triangleleft \pi_{G='A'}(SC)) \div \sigma_{CNO}(C))$

元组关系演算：

RANGE SC SCX

RANGE C CX

GET W(S.SN): ¬∃ CX(¬∃ SCX(SCX.SNO=S.SNO ∧ SCX.CNO=CX.CNO ∧ SCX.G='A'))

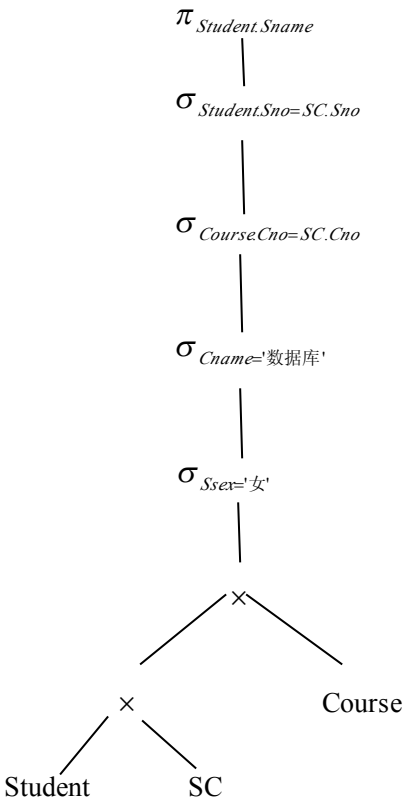
用 SQL 语言完成操作：

```
4．  CREATE VIEW COMPUTER_VIEW
     AS
     SELECT  EN,CN,SC.* FROM SC,T,C
     WHERE T.DEPT='计算机' AND SC.ENO=T.ENO AND SC.CNO=C.CNO;
     GRANT SELECT ON COMPUTER_VIEW
     TO U1;
5.   CREATE TRIGGER STRIGGER ON SC
     FOR INSERT,UPDATE   AS
     DELETE FROM SC
     WHERE EXISTS( SELECT  * FROM INSERTED
     WHERE SC.SNO=INSERTED.SNO AND
     INSERTED.G<60 AND INSERTED.SNO LIKE '2007%8_ _ _');
6.    CREATE PROCEDURE QT
      @AGE VARCHAR(2),@DEPT VARCHAR(10)
      AS
        DECLARE QC CURSOR FOR
        SELECT  EN,SEX FROM T
        WHERE AGE=@AGE AND DEPT=@DEPT;
        OPEN QC;
        DECLARE @EN VARCHAR(2);
        DECLARE @SEX VARCHAR(2);
        FETCH NEXT FROM QC
        INTO @EN,@SEX;
        WHILE @@FETCH_STATUS=0
        BEGIN
             DECLARE @CALL VARCHAR(10);
             SET @CALL=SUBSTRING(@EN,1,1);
             SET @CALL=@CALL+CASE @SEX
             WHEN '男'   TEHN '先生'
             WHEN '女'   TEHN '女士'
             END;
             PRINT @CALL;
             FETCH NEXT FROM QC
             INTO @EN,@SEX;
       CLOSE QC;
        END
```

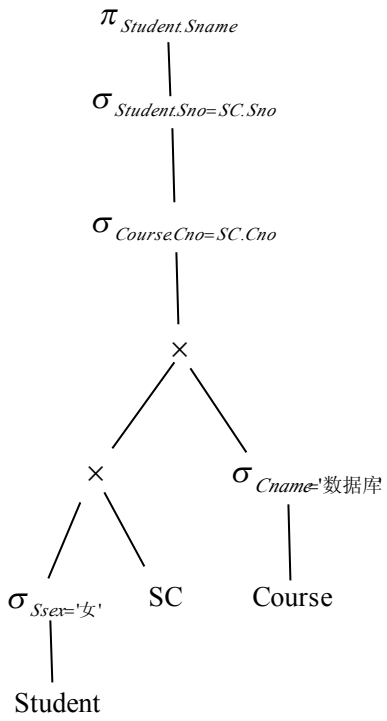
7. 关系代数表达式：

$$\pi_{Student.Sname}(\sigma_{Student.Sno=SC.Sno\wedge Course.Cno=SC.Cno\wedge Cname='数据库'\wedge Ssex='女'}(Student\times SC\times Course))$$

关系代数语法树



优化后关系代数语法树



优化后的关系代数表达式：

$$\pi_{Student.Sname}(\sigma_{Student.Sno=SC.Sno\wedge Course.Cno=SC.Cno}(\sigma_{Ssex='女'}(Student)\times SC\times\sigma_{Cname='数据库'}(Course)))$$

三.

F={A->B,A->D,E->D,D->B,BC->D,BC->E,E->B,E->G,G->A}

=>F={A->D,D->B,BC->D,BC->E,E->G,G->A}<最小覆盖>

1. R 的候选码：

{C,E,F,H,I,J} {B,C,F,H,I,J} {D,C,F,H,I,J} {A,C,F,H,I,J} {G,C,F,H,I,J}

2. 由 BC->E,E->G,G->A 得 BC->A

再由 D->B 可得 DC->A

3. R 不属于 3NF,因为存在传递函数依赖，

由 E->D,D->B 可得 E->B， 再由 E->BG 得 E->B

分解成 3NF,并具有无损连接性和保持函数依赖：

F 的最小覆盖为：{A->D,D->B,BC->D,BC->E,E->G,G->A}

按具有相同左部的原则进行分组：{A,D} {B,C,D,E} {E,G} {G,A}

再并上 R 的码{C,E,F,H,I,,J}

最后的分解为：{A,D} {B,C,D,E} {E,G} {G,A} {C,E,F,H,I,,J}

四.

1.在动态转储过程中，为保证数据的一致性，应将转储过程中各个事务对数据库的操作记录到日志文件中。因些应该将事务 T1 的 O11,O12 的操作以及事务 T2 的 O21,O22 操作记录到日文件中。

2.若在记录的最后出现故障：

正向扫描日志文件，查找出已完成的事务，并将其放入 REDO 队列，查找出未完成的事务，并将其放入 UNDO 队列。

结果如下：

REDO={T1,T3,T4,T6}

UNDO={T2,T5,T7}

正向扫描日志文件，对 REDO 队列里的每一个事务重新执行它的所有操作

反向扫描日志文件，对 UNDO 队列里的每一个事务执行它的操作的逆操作

3.在 B 点设置了检查点：

对在检查点 B 处还未完成的事务以及在 B 后才开始的事务进行操作，将已完成的事务放入 REDO-LIST 队列,将未完成的事务放入 UNDO-LIST 队列。

结果如下：

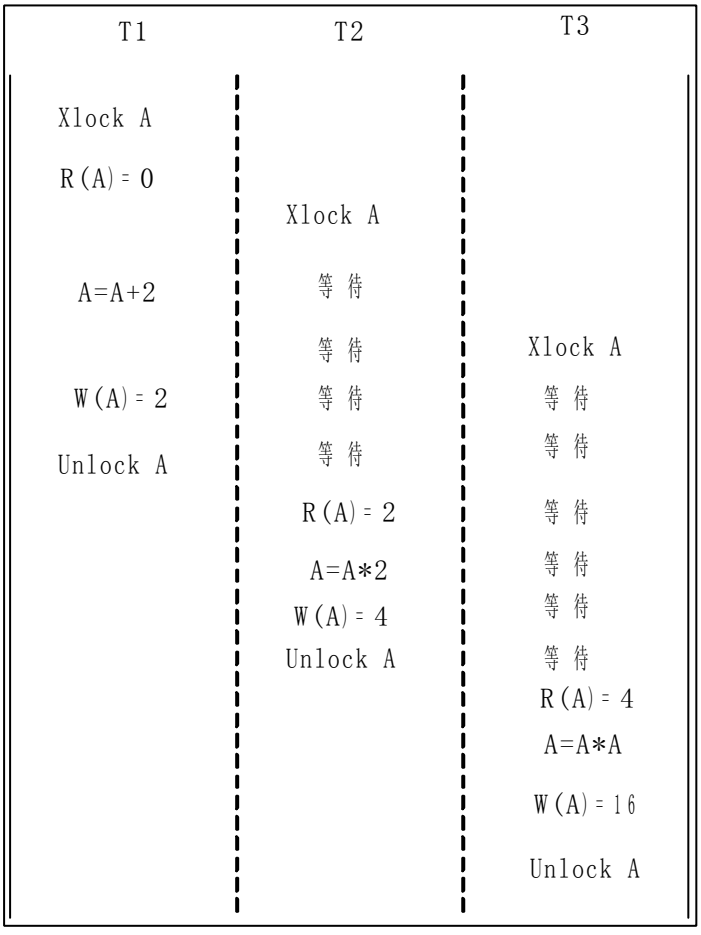
REDO-LIST={T6}

UNDO-LIST={T2,T5,T7}

正向扫描日志文件，对 REDO-LIST 队列里的每一个事务重新执行它的所有操作

反向扫描日志文件，对 UNDO-LIST 队列里的每一个事务执行它的操作的逆操作

4.



最终结果：A＝16

5.