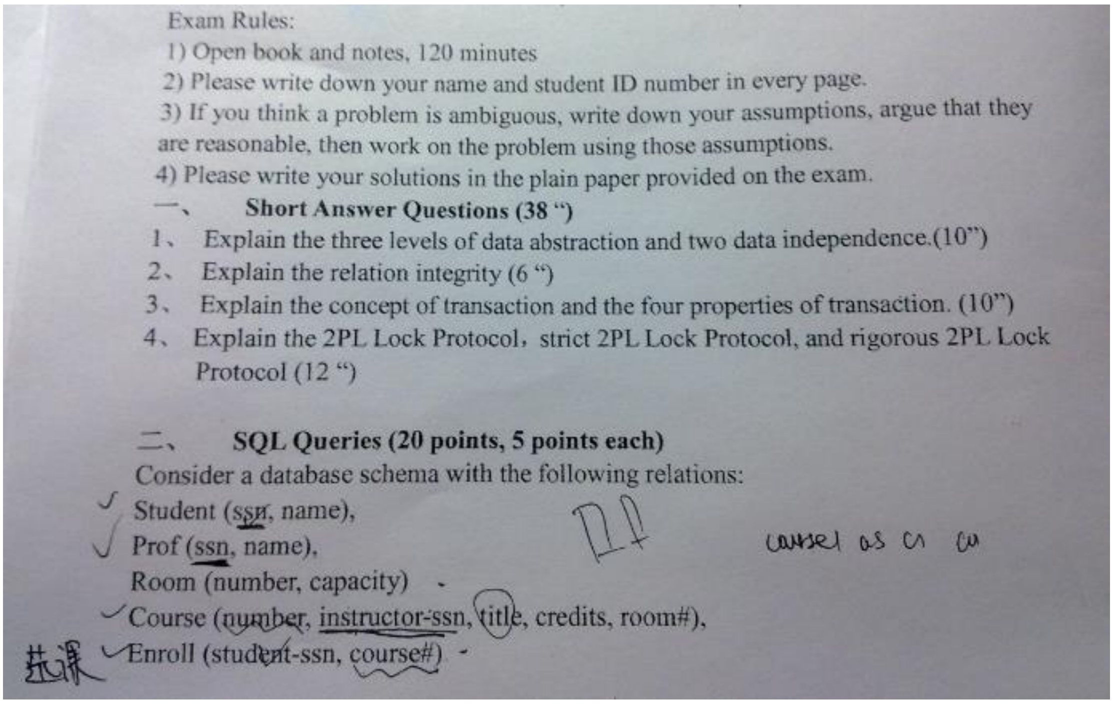


Exam Rules:

1) Open book and notes, 120 minutes

2) Please write down your name and student ID number in every page.

3) If you think a problem is ambiguous, write down your assumptions, argue that they are reasonable, then work on the problem using those assumptions.

4) Please write your solutions in the plain paper provided on the exam.

一、 Short Answer Questions (38 “)

1、 Explain the three levels of data abstraction and two data independence.(10”)

2、 Explain the relation integrity (6 “)

3、 Explain the concept of transaction and the four properties of transaction. (10”)

4、 Explain the 2PL Lock Protocol，strict 2PL Lock Protocol, and rigorous 2PL Lock Protocol (12 “)

二、 SQL Queries (20 points, 5 points each)

Consider a database schema with the following relations:

✓ Student (ssn, name),

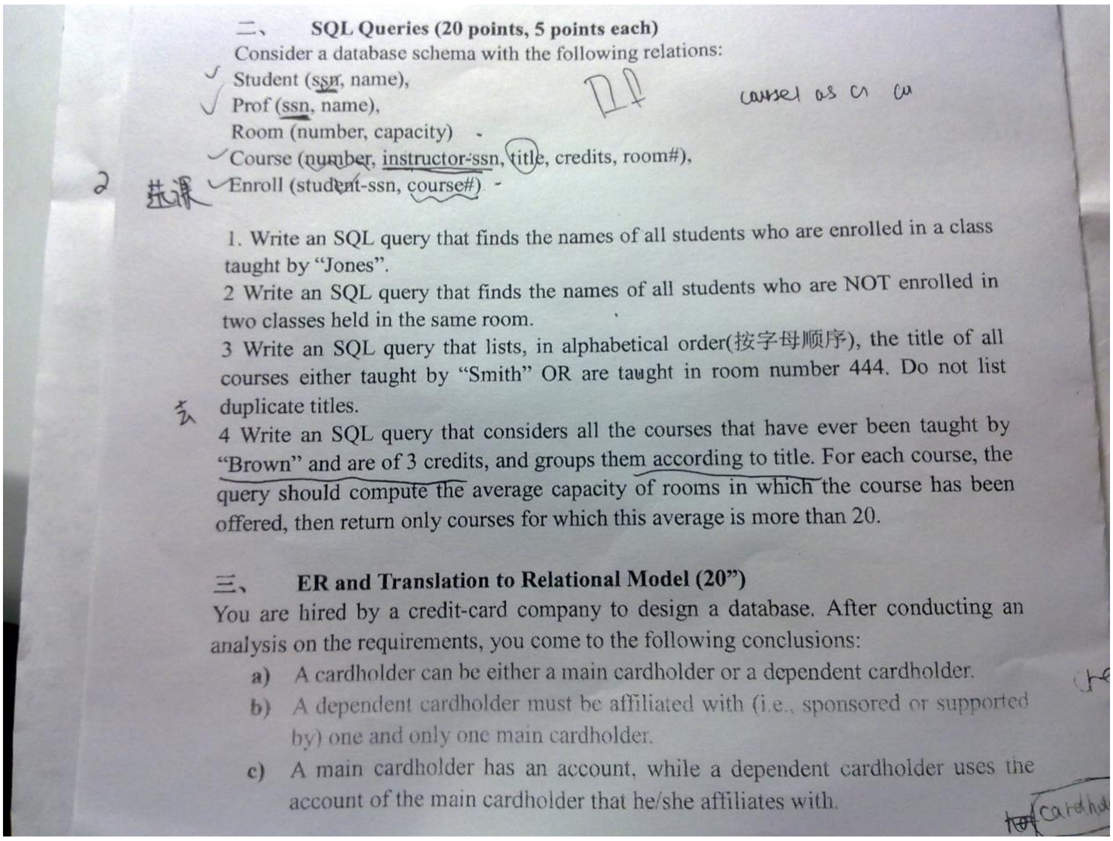
✓ Prof (ssn, name),

Room (number, capacity)

✓ Course (number, instructor-ssn, title, credits, room#),

选课 ✓ Enroll (student-ssn, course#)

coursel as c1 cu



二、 SQL Queries (20 points, 5 points each)

Consider a database schema with the following relations:

✓ Student (ssn, name),

✓ Prof (ssn, name),

Room (number, capacity)

✓ Course (number, instructor-ssn, title, credits, room#),

2 选课 ✓ Enroll (student-ssn, course#)

coursel as c1 cu

1. Write an SQL query that finds the names of all students who are enrolled in a class taught by “Jones”.

2 Write an SQL query that finds the names of all students who are NOT enrolled in two classes held in the same room.

3 Write an SQL query that lists, in alphabetical order(按字母顺序), the title of all courses either taught by “Smith” OR are taught in room number 444. Do not list duplicate titles.

4 Write an SQL query that considers all the courses that have ever been taught by “Brown” and are of 3 credits, and groups them according to title. For each course, the query should compute the average capacity of rooms in which the course has been offered, then return only courses for which this average is more than 20.

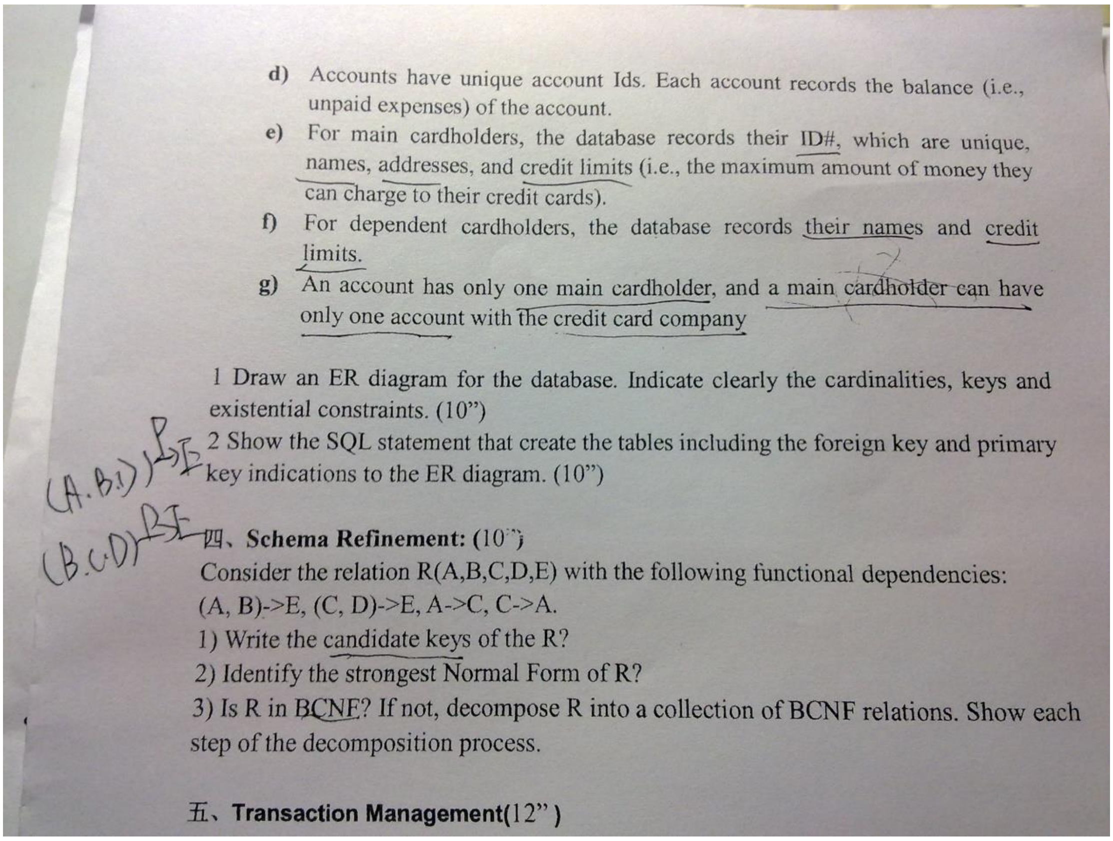
三、 ER and Translation to Relational Model (20”)

You are hired by a credit-card company to design a database. After conducting an analysis on the requirements, you come to the following conclusions:

a) A cardholder can be either a main cardholder or a dependent cardholder.

b) A dependent cardholder must be affiliated with (i.e., sponsored or supported by) one and only one main cardholder.

c) A main cardholder has an account, while a dependent cardholder uses the account of the main cardholder that he/she affiliates with.



d) Accounts have unique account Ids. Each account records the balance (i.e., unpaid expenses) of the account.

e) For main cardholders, the database records their ID#, which are unique, names, addresses, and credit limits (i.e., the maximum amount of money they can charge to their credit cards).

f) For dependent cardholders, the database records their names and credit limits.

g) An account has only one main cardholder, and a main cardholder can have only one account with the credit card company

1 Draw an ER diagram for the database. Indicate clearly the cardinalities, keys and existential constraints. (10")

2 Show the SQL statement that create the tables including the foreign key and primary key indications to the ER diagram. (10")

(A.B.D) → E

(B.C.D) → BE

四、**Schema Refinement:** (10")

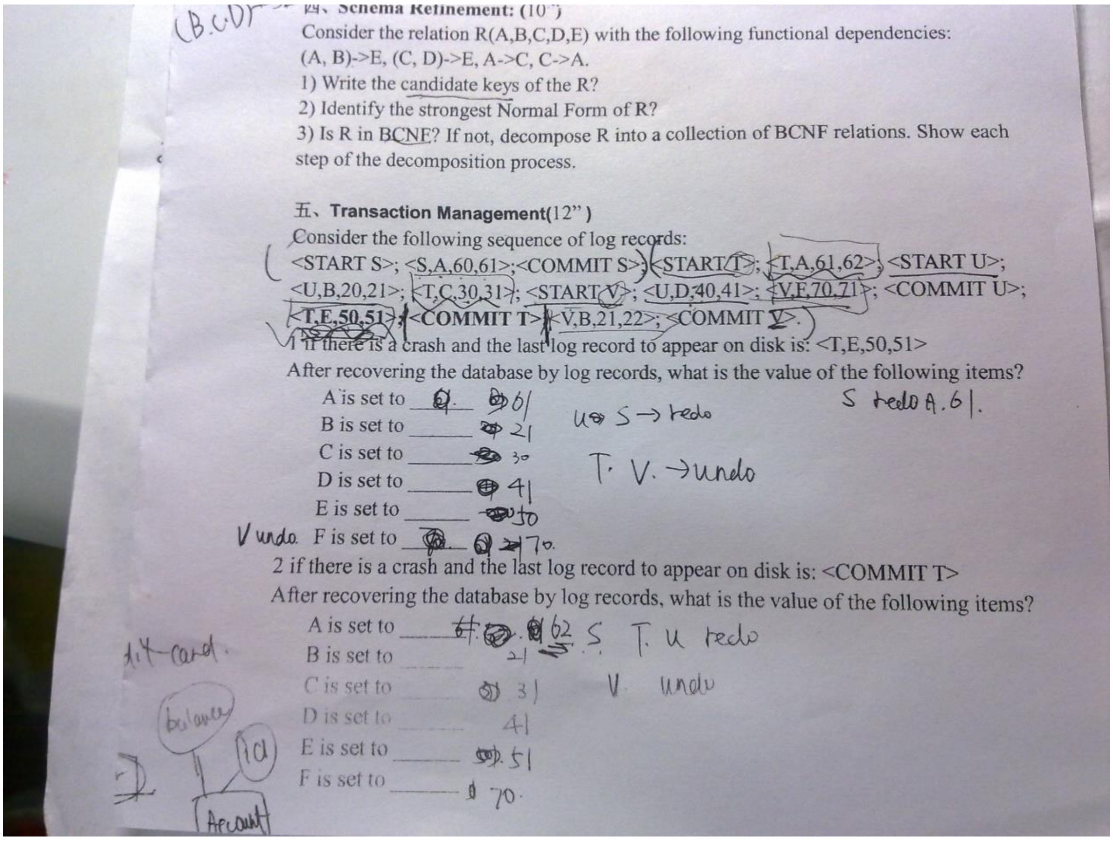
Consider the relation R(A,B,C,D,E) with the following functional dependencies:
(A, B)->E, (C, D)->E, A->C, C->A.

1) Write the candidate keys of the R?

2) Identify the strongest Normal Form of R?

3) Is R in BCNF? If not, decompose R into a collection of BCNF relations. Show each step of the decomposition process.

五、**Transaction Management**(12")



(B.C.D)

四、**Schema Refinement:** (10")

Consider the relation R(A,B,C,D,E) with the following functional dependencies:
(A, B)->E, (C, D)->E, A->C, C->A.

1) Write the candidate keys of the R?

2) Identify the strongest Normal Form of R?

3) Is R in BCNF? If not, decompose R into a collection of BCNF relations. Show each step of the decomposition process.

五、**Transaction Management**(12")

Consider the following sequence of log records:
<START S>; <S,A,60,61>;<COMMIT S>; <START T>; <T,A,61,62>; <START U>;
<U,B,20,21>; <T,C,30,31>; <START V>; <U,D,40,41>; <V,F,70,71>; <COMMIT U>;
<T,E,50,51>; <COMMIT T>; <V,B,21,22>; <COMMIT V>.

1 if there is a crash and the last log record to appear on disk is: <T,E,50,51>
After recovering the database by log records, what is the value of the following items?

A is set to ____ 61 S redo A.61.

B is set to ____ 21 U S → redo

C is set to ____ 30

D is set to ____ 41 T. V. → undo

E is set to ____ 50

V undo. F is set to ____ 70.

2 if there is a crash and the last log record to appear on disk is: <COMMIT T>
After recovering the database by log records, what is the value of the following items?

A is set to ____ 62 S T. U redo

B is set to ____ 21

C is set to ____ 31 V. undo

D is set to ____ 41

E is set to ____ 51

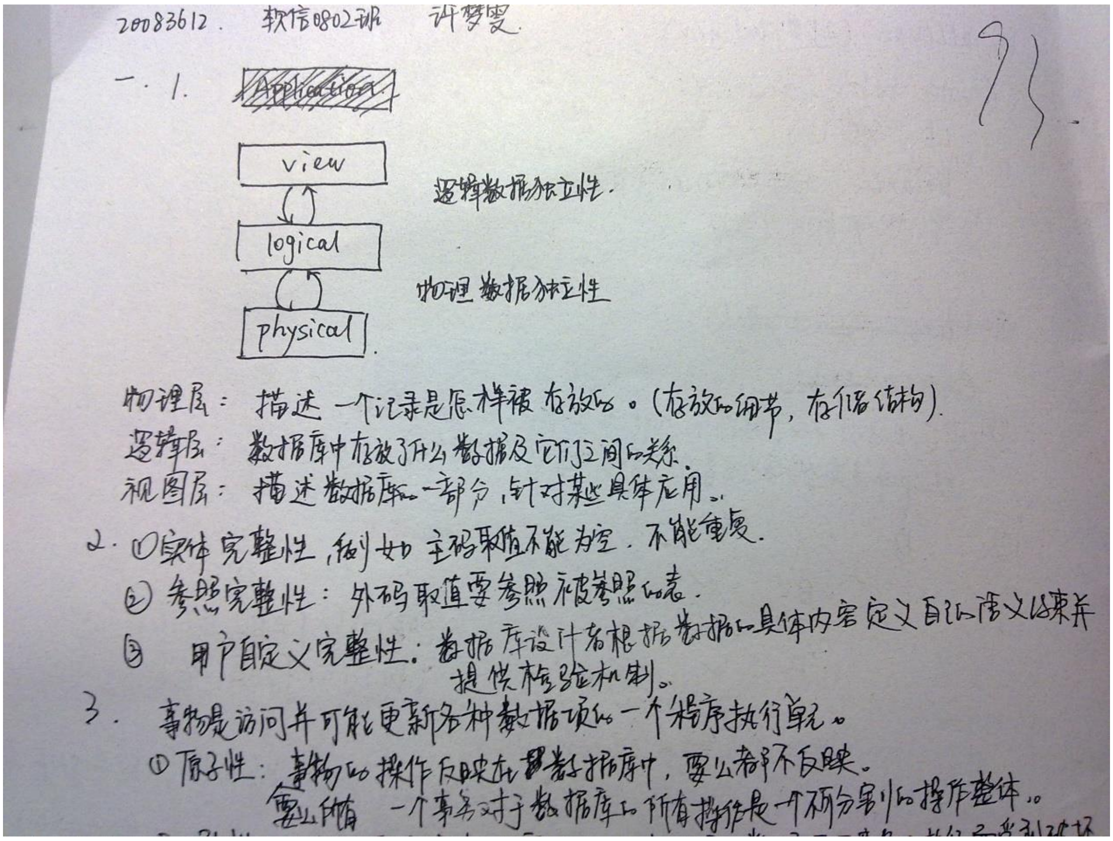
F is set to ____ 70.

dit-card.

balance

id

Account



20083612. 软信0802班 许梦雯.

93

一. 1. ~~Application~~

view

逻辑数据独立性.

logical

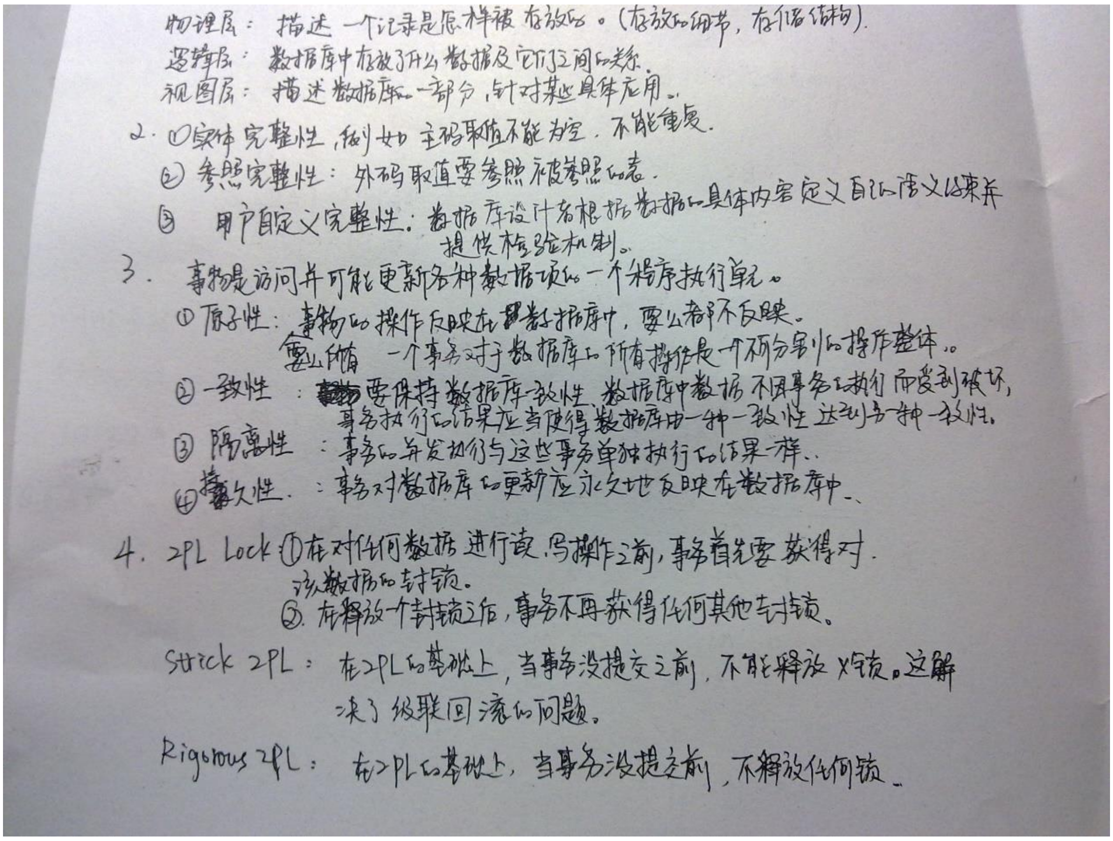
物理数据独立性

physical

物理层：描述一个记录是怎样被存放的。(存放的细节，存储结构).
逻辑层：数据库中存放了什么数据及它们之间的关系.
视图层：描述数据库的一部分，针对某些具体应用。

2. ①实体完整性，例如：主码取值不能为空，不能重复.
② 参照完整性：外码取值要参照被参照的表.
③ 用户自定义完整性：数据库设计者根据数据的具体内容定义自己的语义约束并提供检验机制。

3. 事物是访问并可能更新各种数据项的一个程序执行单元。
① 原子性：事物的操作反映在~~[illegible]~~数据库中，要么都不反映。
要么所有 一个事务对于数据库的所有操作是一个不可分割的操作整体。



物理层：描述一个记录是怎样被存放的。(存放的细节，存储结构).
逻辑层：数据库中存放了什么数据及它们之间的关系.
视图层：描述数据库的一部分，针对某些具体应用。

2. ①实体完整性，例如：主码取值不能为空，不能重复.
② 参照完整性：外码取值要参照被参照的表.
③ 用户自定义完整性：数据库设计者根据数据的具体内容定义自己的语义约束并提供检验机制。

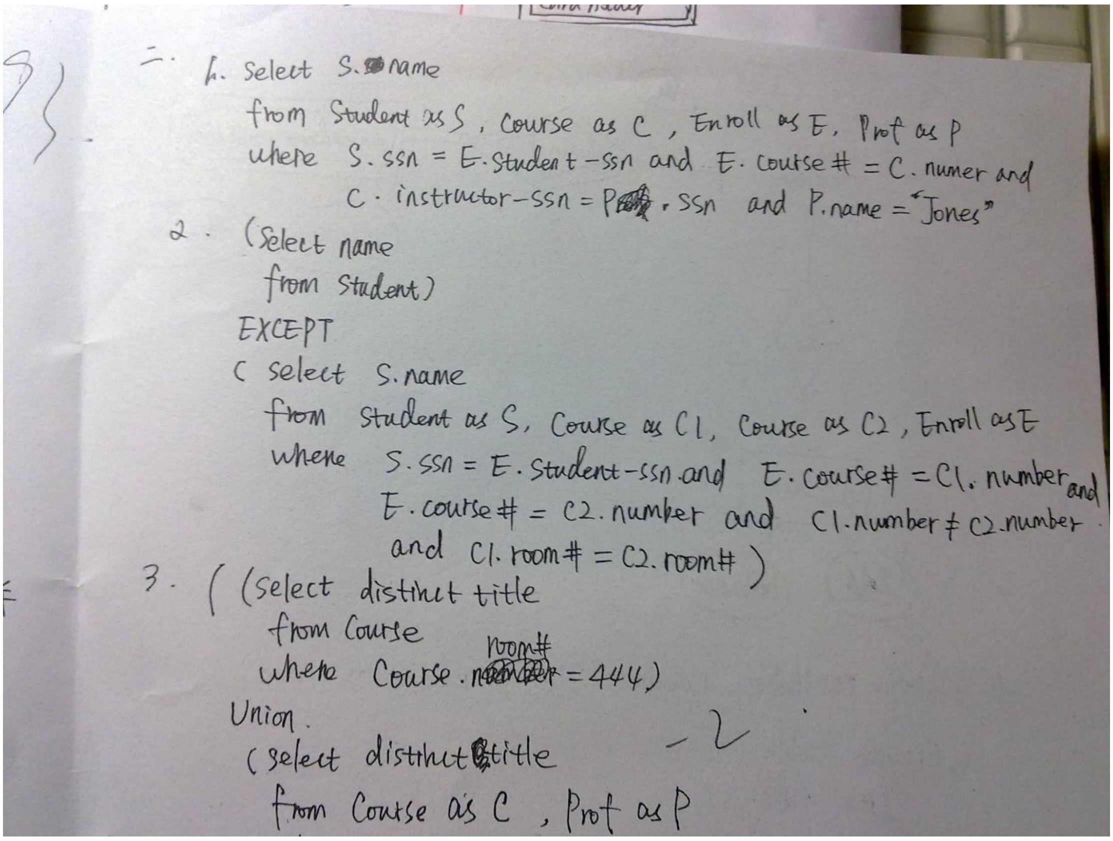
3. 事物是访问并可能更新各种数据项的一个程序执行单元。
① 原子性：事物的操作反映在~~[illegible]~~数据库中，要么都不反映。
要么所有 一个事务对于数据库的所有操作是一个不可分割的操作整体。

② 一致性：~~[illegible]~~要保持数据库一致性，数据库中数据不因事务的执行而受到破坏，事务执行的结果应当使得数据库由一种一致性达到另一种一致性。
③ 隔离性：事务的并发执行与这些事务单独执行的结果一样..
④ 持久性：事务对数据库的更新应永久地反映在数据库中..

4. 2PL Lock ①在对任何数据进行读、写操作之前，事务首先要获得对该数据的封锁。
②. 在释放一个封锁之后，事务不再获得任何其他封锁。

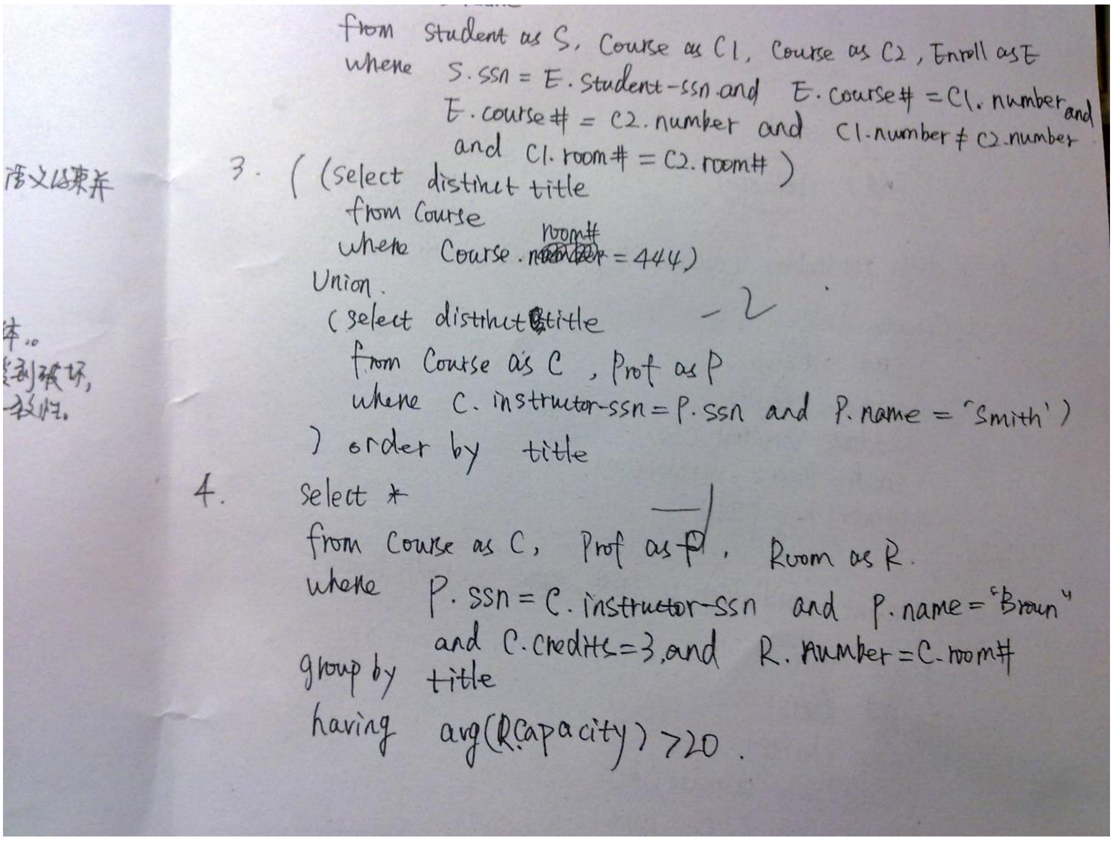
Strick 2PL：在2PL的基础上，当事务没提交之前，不能释放X锁。这解决了级联回滚的问题。

Rigorous 2PL：在2PL的基础上，当事务没提交前，不释放任何锁.



二.

```
1. Select S.name
   from Student as S , Course as C , Enroll as E, Prof as P
   where  S.ssn = E.student-ssn and E.course# = C.numer and
          C.instructor-ssn = P.ssn  and P.name = "Jones"
2. (Select name
    from Student)
   EXCEPT
   ( select  S.name
     from  Student as S, Course as C1, Course as C2 , Enroll as E
     where  S.ssn = E.student-ssn and  E.course# = C1.number and
            E.course# = C2.number and  C1.number ≠ C2.number
            and  C1.room# = C2.room# )
3. ( (select distinct title
      from Course
      where  Course.room# = 444)
   Union
    (select  distinct title
      from Course as C , Prof as P
```

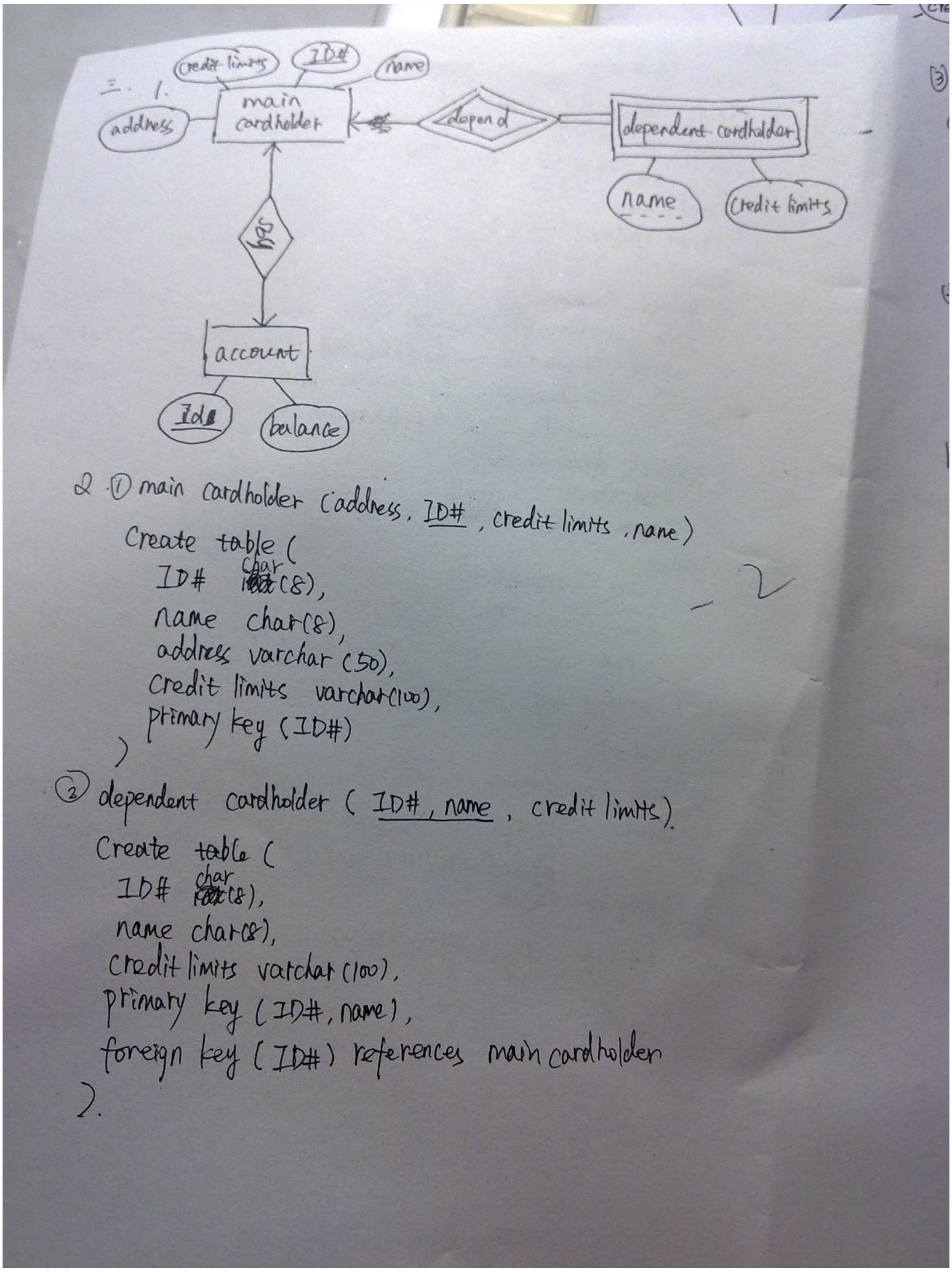


```
     from  Student as S, Course as C1, Course as C2 , Enroll as E
     where  S.ssn = E.student-ssn and  E.course# = C1.number and
            E.course# = C2.number and  C1.number ≠ C2.number
            and  C1.room# = C2.room# )
3. ( (select distinct title
      from Course
      where  Course.room# = 444)
   Union
    (select  distinct title
      from Course as C , Prof as P
```

```
      where  C.instructor-ssn = P.ssn and P.name = 'Smith')
   ) order by  title
4.    Select *
      from Course as C,  Prof as P,  Room as R.
      where   P.ssn = C.instructor-ssn  and  P.name = "Braun"
              and C.credits = 3, and  R.number = C.room#
      group by  title
      having  avg(R.capacity) > 20 .
```

-2

体。
到破坏,
一致性。



三. 1.
credit limits
ID#
name
address
main cardholder
depend
dependent cardholder
name
credit limits
has
account
Id
balance

2 . ① main cardholder (address, ID#, credit limits, name)

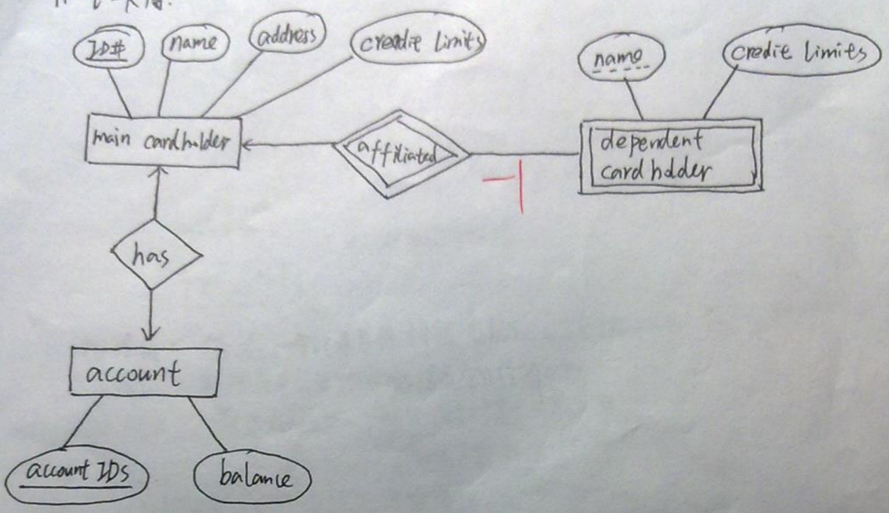
Create table (
  ID# ~~int~~ char(8),
  name char(8),
  address varchar(50),
  credit limits varchar(100),
  primary key (ID#)
)

② dependent cardholder (ID#, name, credit limits).

Create table (
  ID# ~~text~~ char(8),
  name char(8),
  credit limits varchar(100),
  primary key (ID#, name),
  foreign key (ID#) references main cardholder
).

20083613 软信0802班 张新

三、1. E-R图.



2. main_card holder (ID# , name , address , credit_limits)

```
SQL语句:  creat  table  main_cardholder
          (    ID#     ~~int~~char(10),
               name    char(20),
               address varchar(30),
               credit_limits   char(20),
               primary key ( ID# )
          )
```

-2 .

account (accountIDs , balance)

```
SQL语句:   creat  table  account
           (
              account_IDs     ~~int~~ char(10),
              balance        char(20)
              primary key (account_IDs)
           )
```

dependent cardholder (ID#, name, credit_limits)

SQL语句:

create table dependent
(ID# ~~int~~ char(10),
name char(20),
credit_limits char(30),
primary key (ID#, name),
foreign key (ID#) references main_cardholder)

IV. 1. candidate key : (ABD) 和 (BCD)

2. 第一范式，因为 AB→E 存在非主属性对码的部分函数依赖，所以不是 2NF.

3. R 不是 BCNF.

分解: (1) $(AB)^+ = ABCE$，所以 AB→E 违反 BCNF

$R_1 = ABE$

$R_2 = ABCD$

(2) $(CD)^+ = ACDE$. 所以 (CD)→E 违反 BCNF

$R_3 = CDE$

(3) ~~A~~⇒ $A^+ = AC$，所以 A→C 违反 BCNF

$R_4 = AC$

$R_5 = ABD$

−2

∴ BCNF分解: R1, R3, R4, R5 即 (ABE)∪(CDE)∪(AC)∪(ABD)

V. 1. A is set to 61

B 21

C 30

D 41

E 50

F 70

redo : U S

undo : T V

2. A is set to 62

B 21

C 31

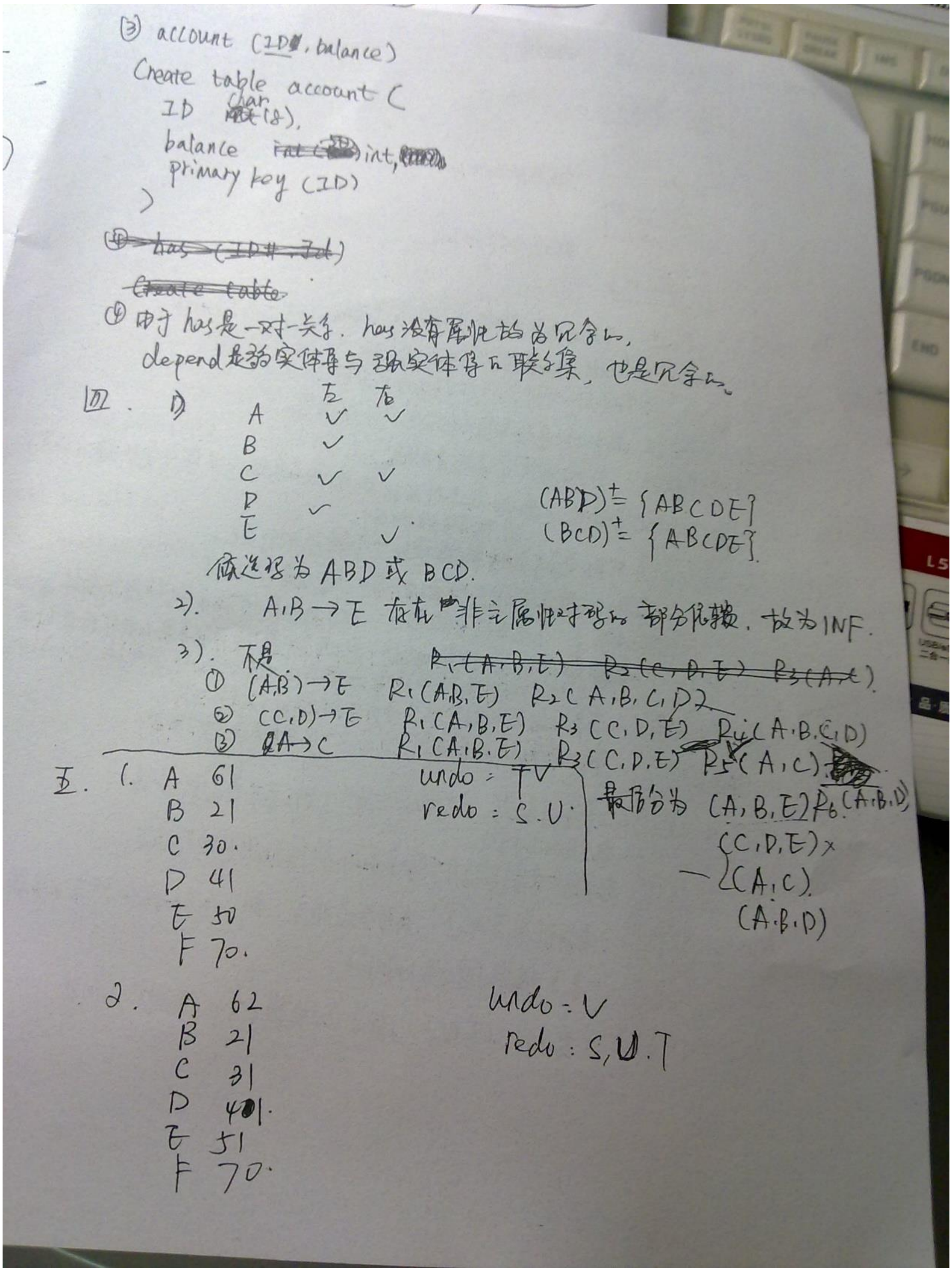
D 41

E 51

F 70

redo : S T U

undo : V



③ account (<u>ID</u>, balance)

```
Create table account (
  ID  char(8),
  balance  ~~int (20)~~ int,
  primary key (ID)
)
```

~~④ has (ID#, Id)~~

~~Create table~~

④ 由于 has 是一对一关系. has 没有属性的为冗余的,
depend是弱实体集与强实体集联系集, 也是冗余的。

四. 1)

	左	右
A	✓	✓
B	✓	
C	✓	✓
D	✓	
E		✓

$(ABD)^+ = \{ABCDE\}$

$(BCD)^+ = \{ABCDE\}$

故选码为 ABD 或 BCD.

2). A,B → E 存在非主属性对码的部分依赖, 故为1NF.

3). 不是 ~~R1 (A, B, E) R2 (C, D, E) R3 (A, C)~~

① (A,B) → E R1 (A,B, E) R2 (A, B, C, D)

② (C,D) → E R1 (A, B, E) R3 (C, D, E) R4 (A, B, C, D)

③ A → C R1 (A, B, E) R3 (C, D, E) R5 (A, C) R6 (A, B, D)

最后分为 (A, B, E)
(C, D, E) ×
(A, C).
(A, B, D)

五. 1.

A	61
B	21
C	30.
D	41
E	50
F	70.

undo : T V

redo : S . U .

2.

A	62
B	21
C	31
D	41.
E	51
F	70.

undo : V

redo : S, U . T