UCLA
CS143—Spring 2016

Final Exam

Problem A: 30 points—6 questions: 5 points per question.

$T(A, C, B, D, E)$

(a) $AD \rightarrow CE$

(b) $BC \rightarrow D$

(c) $AB \rightarrow A$

(d) $B \rightarrow E$

Q1. is any of the previous FDs trivial?

Ans: (c) Is trivial and we eliminate it.

Q2: *Transform the given FDs into an equivalent set of* ***elementary*** *FDs (no trivial FD, only one attribute on the right side, minimal left side).*

(a1) AD → C
(a2) AD → E
(b) BC → D
(d) B → E

T(A,C,B,D,E) with (a1) AD → C
(a2) AD → E
(b) BC → D
(d) B → E

Q3: is **T(A,C,B,D,E)** BCNF?

Answer: *To check whether it is BCNF, check the non-trivial FDs, starting with* (a1)

(a1): AD+ = {A,D,C, E }. *Since* B *is missing* AD *is not a key, or a superset of a key: This is a BCNF violation.*

T(A,C,B,D,E) with

(a1) AD → C

(a2) AD → E

(b) BC → D

(d) B → E

Q4: is **T(A,C,B,D,E)** 3NF?

Answer: *We have seen that a1 and a2 violate BCNF, thus a1 (a2) violate 3NF unless* there exist an elementary X->A where X is a key for T and X contains A (E for a2). But E is not in the left side of ant elementary FD.

T(A,C,B,D,E) with

(a1) AD → C
(a2) AD → E
(b) BC → D
(d) B → E

Q5: is **T(A,C,B,D,E)**

Compute a lossless decompositions of T into BCNF relations where no two relations share the same keys.

Using a1, a2: T1(A,D, C, E), T2(A,B,D) *These are BCNF*

But we can try another decomposition.

e.g. (d): R1(B, E) R2(A, B, C, D) *Now R2 is not BCNF, let us decompose it into:* R21(B,C,D), R22(A, B, C). *R1, R21, R22 is BCNF.*

T(A,C,B,D,E) with

(a1) AD → C
(a2) AD → E
(b) BC → D
(d) B → E

Q6: *Can you reconstruct the original relation from those obtained in the decomposition? What RA operation will you be using for that?*
Yes, using natural joins.

Q7. *Is your decomposition FD preserving? To receive credit you must justify your answer.*

With: T1(A,D, C, E), T2(A,B,D): we loose B → E and BC → D

With: R1(B, E), R21(B,C,D), R22(A, B, C). We loose a1 and a2

Problem B: 20 points—4 questions 5 points per question.
You must propose a relational schema for the DB described by the ER diagram below. You also know that there is complete information about our DB content, whereby all values are known for the attributes associated with each entity instance, and the diamond relationships are total since each employee works in some department and has a supervisor. (Of course, the ISA relationship is an exception, inasmuch as each employee is a person, but not every person is an employee.) Please solve the following problems:

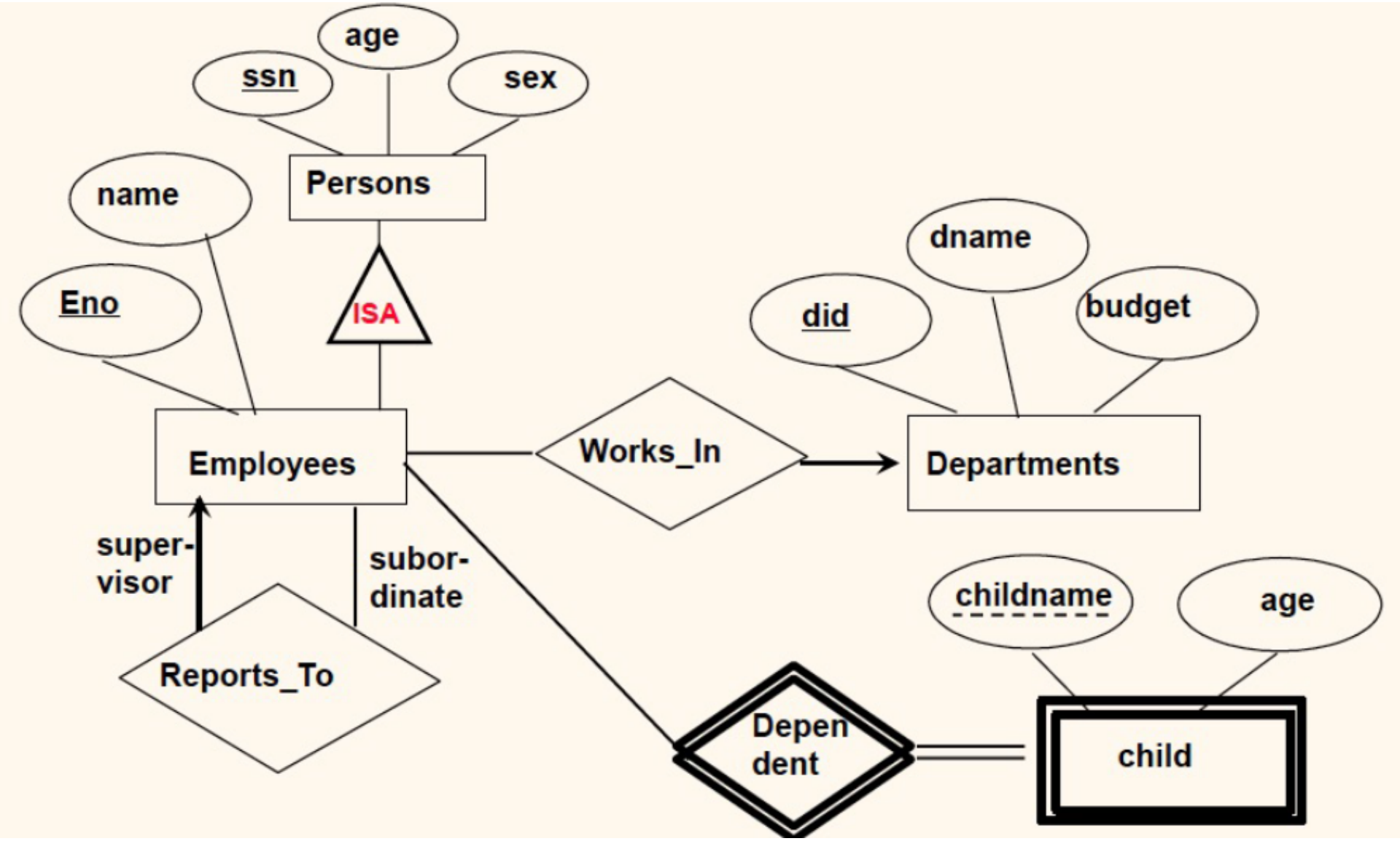
B1. For the DB just described, design a BCNF schema that must have (i) a minimum relation count, (ii) a minimum count of attribute in each relation, and (iii) it must be such that, owing to the completeness of our DB, all columns in all relations can be declared with the "nulll not allowed" option. Show the keys of the resulting relations by underscoring the key attributes and using different underscoring styles for different keys in the same relation.

B2, B3, B4, B5

Problem B: 20 points—4 questions 5 points per question.
You must propose a relational schema for the DB described by the ER diagram below. You also know that there is complete information about our DB content, whereby all values are known for the attributes associated with each entity instance, and the diamond relationships are total since each employee works in some department and has a supervisor. (Of course, the ISA relationship is an exception, inasmuch as each employee is a person, but not every person is an employee.) Please solve the following problems:

B1. For the DB just described, design a BCNF schema that must have (i) a minimum relation count, (ii) a minimum count of attribute in each relation, and (iii) it must be such that, owing to the completeness of our DB, all columns in all relations can be declared with the "nulll not allowed" option. Show the keys of the resulting relations by underscoring the key attributes and using different underscoring styles for different keys in the same relation.

B2, B3, B4, B5


age
ssn
sex
Persons
name
Eno
ISA
dname
did
budget
Employees
Works_In
Departments
super-
visor
subor-
dinate
Reports_To
childname
age
Depen
dent
child

B1. *For the DB just described, design a BCNF schema that must have (i) a minimum relation count, (ii) a minimum count of attribute in each relation, and (iii) it must be such that, owing to the completeness of our DB, all columns in all relations can be declared with the "nulll not allowed" option. Show the keys of the resulting relations by underscoring the key attributes and using different underscoring styles for different keys in the same relation.*

We recognize the weqk entity. Thus, we import Eno from Employees and get:

1. dependentchild(Eno, childname, age)

Then for Departments we get**:** **2.** departents(did, dname, budget)

Then, from Persons we get: **3.** persons(SSN, age, sex)

We can now turn to Employees where, with the help of an additional attribute **SuperEno** we break the Reports to cycle and, by importing ssn through ISA we obtain. **4.** employees(Eno, name, did, superEno, SSN).

Thus we obtain a total of 4 relations, each containing a minimum number of attributes.

B2. Complete your schema declarations by showing the foreign keys in each relation (you can do that by either using the SQL declarations or simply drawing a picture where foreign keys references are displayed as arrows across relations).

Answer: We have the following foreign key references:

dependentchild.Eno → employees.Eno

employees.ssn → persons.ssn

employees.did → departents.did

employees.SuperEno → employees.Eno

B3. With the many-to-many relation the FD Eno → did disappears, and the pairs (Eno, did) and (ssn, did) become the keys of our table 4. Now, we still have say Eno → name, superEno with Eno no longer a key. Thus our table employees is no longer BCNF, and we see update anomalies. For instance, if an employee works in N>1 departments, then his/her reassignment to a different supervisor will require updating N records. This is an anomaly.

Answer: Since the many-to-many relation the FD **Eno → did** disappears, the pairs **(Eno, did)** and **(ssn, did)** become the keys of our table 4. Now, we still have say **Eno → name, superEno** and **Eno** no longer a key. Thus our table employees is no longer BCNF, and we see update anomalies. For instance, if an employee works in N>1 departments, then his/her reassignment to a different supervisor will require updating N records. This is an anomaly.

B4. Solve problems B1 and under the revised assumption made in B3.
Answer: We will thus have to decompose table 4 into a pair of tables:

4.a employees(Eno, name, superEno, SSN).

4.b works in(Eno, did).

Problem C: 30 points– 6 questions, 5 points per question. Whenever applicable show the appropriate graph and always justify your answers.

Consider the following schedule, where T1 has an earlier timestamp than T2:

T1	T2
start	
write(A)	
	start
	read(A)
read (C)	
	write(C)
write(C)	

Please answer the following questions:

Q1. Is this schedule conflict-serializable ?

T→T2 on A and C
T2 → T1 on C

NO. according to the graph.

Problem C, cont

NO: 2PL schedules are serializable

Q2. Can this schedule be generated under a 2PL protocol?

Q3. Show what will happen to T1 and T2 if they try to execute this schedule under a timestamp-based scheduling protocol (with Thomas write rule).

If t1 and t2 are respectively the timestamps of T1 and T2, then:

T1	T2	
start		
write(A)		wts(A):= t1
	start	
	read(A)	ok since t2>wts(A)
read (C)		rts(C):=t1
	write(C)	ok since wts(C) >t1; wts(C):=t2
write(C)		t1<wts(C) so do nothing–Thomas

Q4. Show what will happen to T1 and T2 if they try to execute this schedule under a strict 2PL locking strategy and no deadlock prevention (assume that a transaction locks a resource just before it needs it)?

T1	**T2**
start	
write(A)	
	start
	R-L(A) and wait-for T1
read (C)	
write(C)	
unlock A, C ; commit	
	read(A)
	write(C)

Q5. What will happen to T1 and T2 if they instead execute using a strict 2PL strategy and a wait-die deadlock prevention scheme (assume that a transaction locks a resource just before it needs it)?

T1	**T2**
start	
write(A)	
	start
	R-L(A) and die
read (C)	
write(C)	
unlock A, C ; commit	
	restart
	read(A)
	write(C)

Q6. In the wait-die deadlock prevention protocol, transactions that are rolled back keep their old timestamp. What is the reason for that?

In a conflict younger transactions die but older transaction wait and finally complete: thus keeping the old timestamp reduces starvation.

Extra Credit [4 points]. To receive credit you must justify your answers.

- Does the protocol used in C.Q3 guarantee freedom from deadlocks?
- Does the protocol used in C.Q4 guarantee cascadeless schedules?

C.Q3 uses a timestamp based serializability protocol. Such protocol is always deadlock free.

C.Q.4: Under strict 2PL a transaction T1 holds its X-loxka locks until commit, so there is no way that a transaction T2 can read T1's dirty data.