CS143 2020
Practice Final

Computer Science Department

With questions from Winter 2019 final

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

We are given the T(A,B,C,D,E)
(a) AD → CE
(b) BC → D
(c) AE → A

(d) B → E

Q1. is any of the previous FDs trivial? **Answer:** C is trivial

Q2. Is this schema BCNF? (To receive credit you must justify your answer.)

- **Answer:** To check whether it is BCNF, let us try the FDs. E.g check (b)
- B+= {B, E } not a key, so this is not BCNF.

Q3. Is this schema 3NF

Answer: we cannot consider (c) since this is trivial. Then, none of the given FDs ha E on the left side. So they cannot be part of a key.

Problem A: continued

We are given the T(A,B,C,D,E)
(a) AD → CE
(b) BC → D
~~(c) AE → A~~

(d) B → E

Q4: Transform the given FDs into an equivalent set of elementary FDs
(*no trivial FD, only one attribute on the right side, minimal left side).*

Answer: (a) is replaced by a1**:** AD → C and a2: AD → E

Q5. Compute a lossless decompositions of T into BCNF relations where no two relations share the same keys. (*Show the keys by underlining the attributes in the key. Use different underlining for different keys in the same relation)*

Problem A: continued

We are given the T(A,B,C,D,E)

(a) ~~AD → CE~~

(a1) AD → C

a2: AD → E

(b) BC → D

(~~c) AE → A~~

(d) B → E

Q5. Compute a lossless decompositions of T into BCNF relations where no two relations share the same keys. (*Show the keys by underlining the attributes in the key. Use different underlining for different keys in the same relation.*

Answer: We can start with B+= {B, E} and get: (B, E) (A, B, C, D)
Then using BC+= { B, C, D) we get (B,C, D) (B. C, A)
So: (B, E), (B,C, D) , and(B. C, A)

Q6. Can you reconstruct the original relation from those obtained in the decomposition? (Answer: Yes via natural joins). Is your decomposition FD preserving?

- **Answer: B->E, BC -> D are implied by the keys but AD -> C is lost.**

Problem B: 20 points—4 questions 5 points per question.

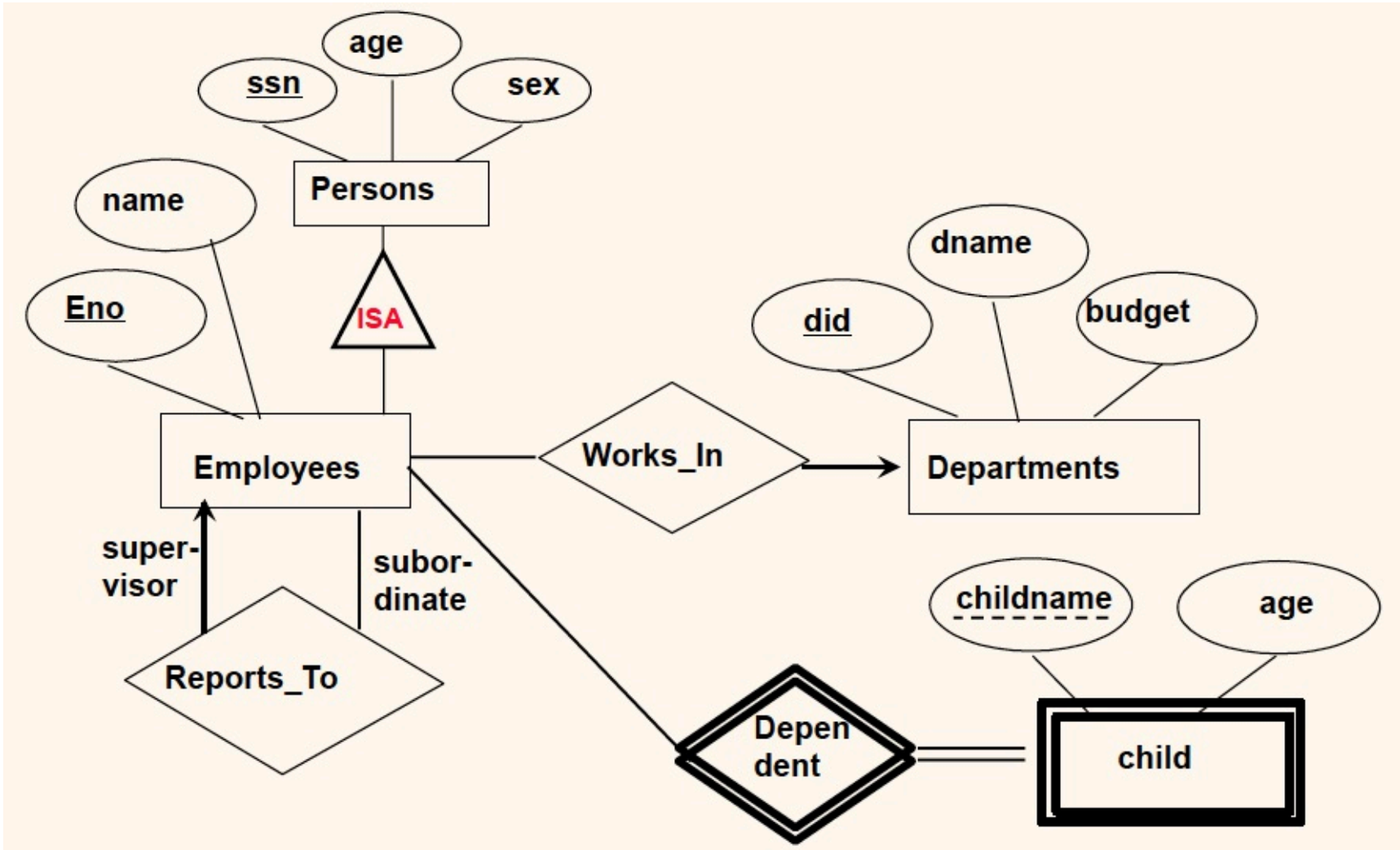
You must propose a relational schema for the DB described by the ER diagram below. You also know that there is complete information about our DB content, whereby all values are known for the attributes associated with each entity instance, and the diamond relationships are total since each employee works 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 with the following goals: (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. The priorities of the goals are in the order they are listed i.e. achieving minimum relation count is most important and so on. 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. 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).

B3. The arrow leading into Departments shows this entity's one-to-many relationship to Employees. Now assume that this relationship changes into a many-to-many relationship. Reconsider the tables that you derived in B1. Have their keys changed and how? Are those tables still in BCNF? Are they in 3NF? We will encounter some update anomalies using that old schema, and if so,? Please make your points by simple examples.

B4. Solve problems B1 and under the revised assumption made in B3.

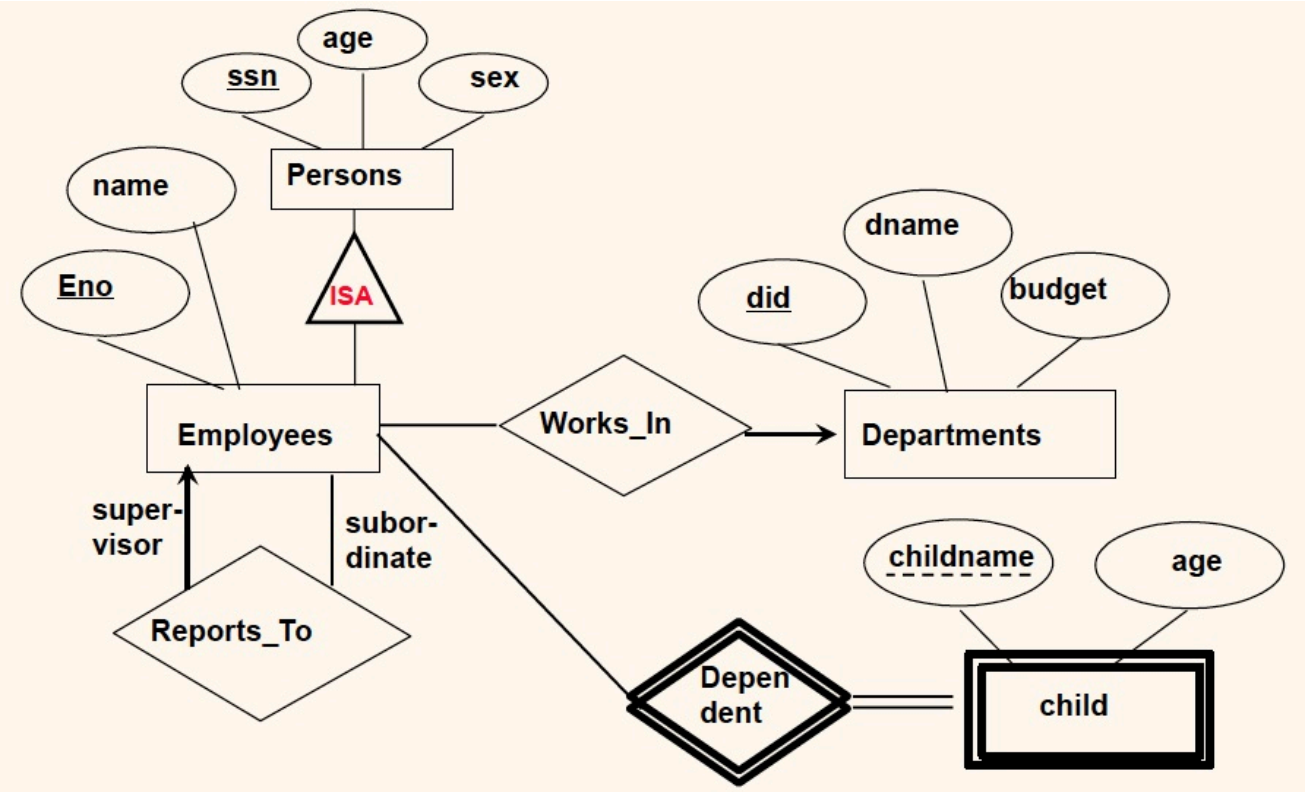

age
ssn
sex
Persons
name
Eno
ISA
dname
did
budget
Employees
Works_In
Departments
super-
visor
subor-
dinate
Reports_To
childname
age
Depen
dent
child

B1. We recognize the weak 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)**.



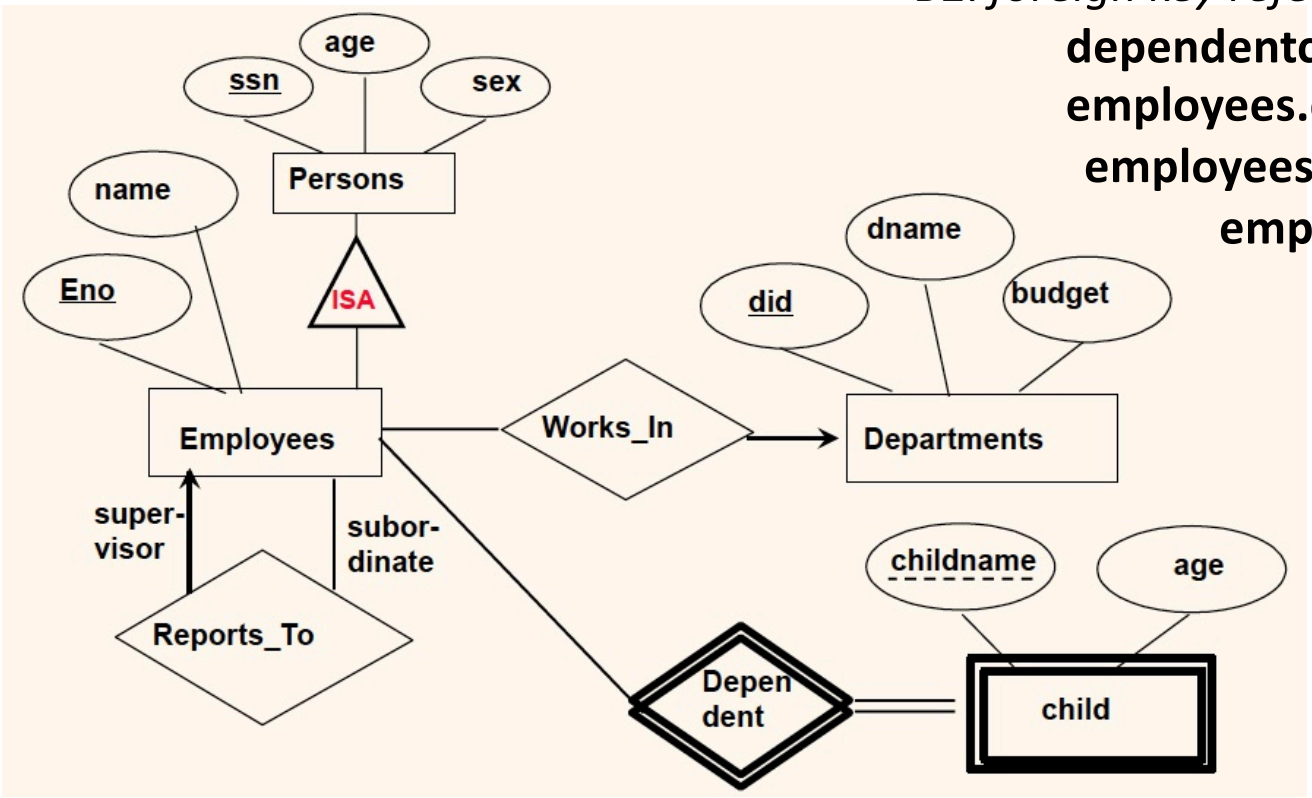
We can now turn to Employees where, with the help of an additional attribute **SuperEno** we break the **Reports_to** cycle and, by importing **ssn** turough ISA we obtain.

4**. employees(Eno, name, did, superEno, ssn).**

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

1. dependentchild(Eno, childname, age). **2. departents(did, dname, budget)**

3. persons(ssn, age, sex). **4. employees(Eno, name, did, superEno, ssn).**



B2. *foreign key references:*

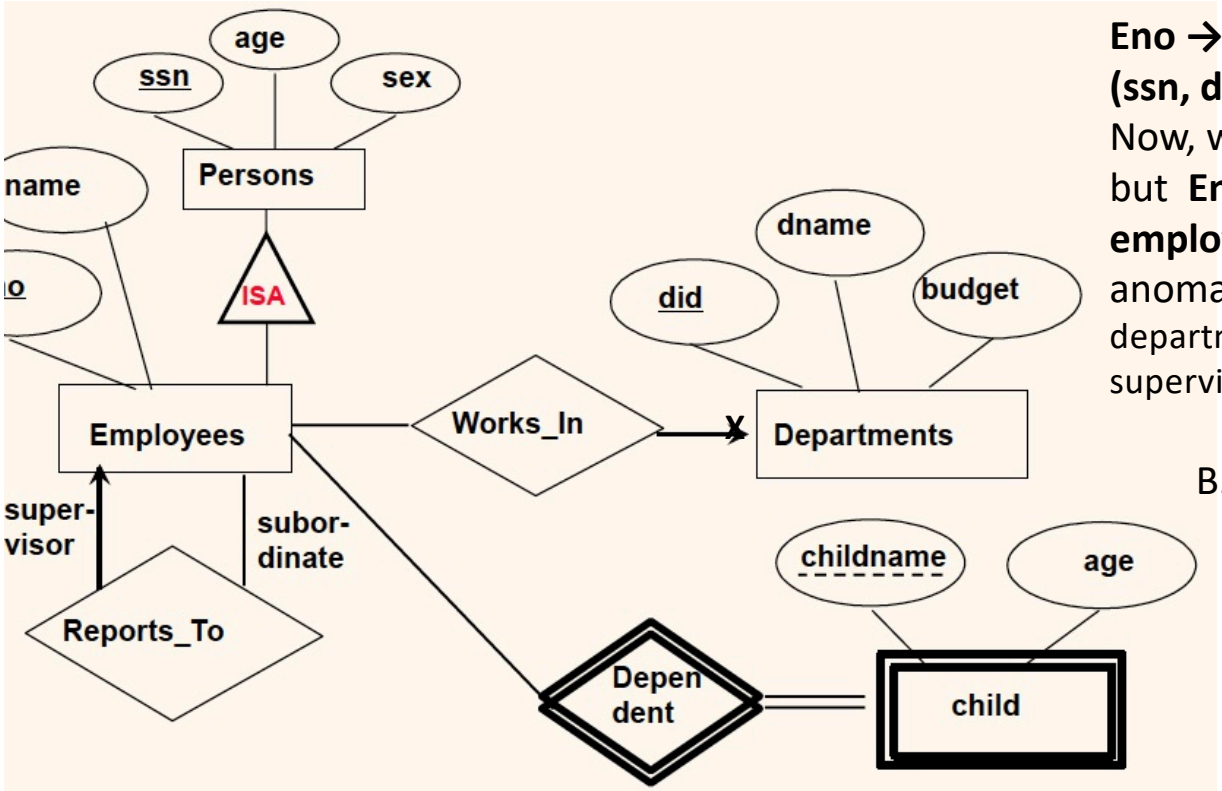
dependentchild.Eno → employees.Eno

employees.did → departents.did

employees.SuperEno→employees.Eno

employees.ssn → persons.ssn

1. dependentchild(Eno, childname, age).
2. departents(did, dname, budget)
3. persons(ssn, age, sex).
4**. employees(Eno, name, did, superEno, ssn).**



B3: answer

Eno → did disappears, and both (**Eno, did**) and **(ssn, did)** become the keys of table 4.
Now, we still have **Eno → name**, **superEno**
but **Eno** is no longer a key. Thus our table **employees** is no longer BCNF, and we have 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.)

B.4 We will thus have to decompose table 4 into a pair of tables:
employees(Eno, Name, SuperEno, ssn).
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.

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

Q1. Is this schedule conflict-serializable ? Justify your answer.

- NO. according to the precedence graph (draw it).

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

- NO. this would ensure serializability.

Problem C -cont.

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).*

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

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

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 rts(C) < t2 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
read (C)	
write(C)	
unlock A, C ; commit	
	resume
	read(A)
	write(C)

Q5. *Show 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. *For each protocol discussed in Q3, Q4, and Q5 states whether it avoids or does not avoid dirty reads. State the reasons for your answer and illustrate it with examples*

Answer: strict **2PL** prevents dirty reads the other algorithms do not. Thus Q4 and Q*5* avoid dirty reads. Q3 does not.

```
1st Checkpoint  ( ) ---
T0 start
T0  C, 900, 777
T1 start
2nd Checkpoint: (T0, T1) ---
------------log page boundary
T2 start
T1 C, 777 , 955
T1 B, 400, 800
T1 commit
------------log page boundary
T3 start
3rd Checkpoint: (?)
T2 A, 100, 200
T3 B, 800, 623
T4 start
T0 Commit
------------log page boundary
T4 C, 955, 2100
T5 start
T4 commit
T5 C, 2100, 4000
------------crash
```

Problem D: 20 points—4 questions: 5 points each

Recovering from a crash using the **immediate database modification** protocol, the database system finds the following log.

>Q1: Specify the transaction list for the 3rd checkpoint with the correct transaction numbers that would have been written by the system.
Answer: It should be Checkpoint: (T0, T2, T3)

> Q2: How many pages of the log will the system read for recovery
Answer: Walking back from the end, at the last checkpoint T0, T2 and T3 are still uncommitted. To find their star we walk back to the first page for a total of four pages.

> Q3. Which transactions will be undone and which will be redone?
Answer: Undone (no commit): T2, T3, T5 .
Redone (commit after last checkpoint): T0, T4
Ignored (commit before last checkpoint): T1

> Q4. What are the values of A, B and C at the end of a successful recovery (explain the reason for your statement)?
Answer: A=100 because (T2 A, 100, 200) which was then undone;
B=800, because (T3 B, 800, 623) which was then undone;
C= 2100, because (T4 C, 955, 2100) was redone
while (T5 C, 2100, 4000) was undone.