MASTER

EECS-3421N: TEST #1

Electrical Engineering & Computer Science

Lassonde School of Engineering

York University

Family Name: _____

Given Name: _____

Student#: _____

EECS Account: _____

Instructor: Parke Godfrey
Exam Duration: 75 minutes
Term: Winter 2019

Instructions

- **rules**
  - The test is closed-note, closed-book. Use of a calculator is permitted.
- **answers**
  - Should you feel a question needs an assumption to be able to answer it, write the assumptions you need along with your answer.
  - If you need more room to write an answer, indicate where you are continuing the answer.
  - For multiple choice questions, choose *one* best answer for each of the following. There is no negative penalty for a wrong answer.
  - For E/R schema, keep the elements as *simple* as possible; e.g., an attribute is simpler than a relationship which is simpler than an entity.
- **notation**
  - For schema, the underlined attributes indicate a table's primary key (and are not nullable). Attributes with an "*" are not nullable. Foreign keys are indicated by FK.
- **points**
  - Each question is marked by the number of points it is worth.
  - There are five major parts worth 10 points each, for 50 points in total.

MARKING BOX		
1.	BCBBD BBBDA	/10
2.		/10
3.		/10
4.		/10
5.		/10
Total		/50

1. [10pt] **General.** *Jan-ken-po!* MULTIPLE CHOICE

(a) [1pt] *Relational database management systems* provide all but which of the following?

A. Facilities for creating and maintaining relational schema.
B. Automated conversion of E/R schema into relational schema.
C. Concurrency control via transaction management.
D. A declarative query language.
E. Integrity checking that forbids data changes that would violate integrity.

(b) [1pt] Consider a relation **R** with five attributes, A, B, C, D, and E. How many different keys could **R** have at most at the same time?

A. 1
B. 5
C. 10
D. 31
E. *The number is unbounded.*

(c) [1pt] NULL values can be used to

A. opt out of enforcement of the primary key.
B. opt out of enforcement of foreign keys.
C. add extra columns for some tuples.
D. achieve a normal form.
E. *All of the above.*

(d) [1pt] The XML data model is called *semi-structured* because

A. it not formally defined, in contrast to the relational model.
B. not all the data in an XML database needs to be *fully structured*, as it has to be in relational.
C. it is computationally simpler than the relational (*structured*) model.
D. there is no corresponding notion of *schema*, as there is for relational.
E. there is no *query language* for it.

(e) [1pt] In E/R, a *one-one* relationship set

A. can always be replaced in a logically equivalent way by two *one-many* relationship sets.
B. is an E/R construct, but it cannot be expressed in a relational schema.
C. may only (recursively) relate the same entity set to itself.
D. is rare, but is sometimes logically needed for the domain being modelled.
E. is an illegal construct.

(f) [1pt] In E/R, a *connecting* weak entity set

A. does not take its keys from other entity sets as does a "regular" weak entity set.
B. adds no key attributes of its own.
C. is weak only on *one* other entity set.
D. is always logically equivalent to a *sub-entity set* (via an *isa* hierarchy).
E. is an E/R construct that cannot be expressed in a relational schema.

(g) [1pt] *First Normal Form* (1NF) states that

A. every relation has more than one attribute.
B. every relation has a key prescribed.
C. every relation has a *singleton* key prescribed; that is, the key consists of a single attribute.
D. every relation has a key prescribed that consists of *all* its attributes.
E. no relation has more than one key prescribed.

(h) [1pt] The purpose of the normal forms is that

A. they help to locate anomalies in the data when they occur.
B. it is guaranteed that certain types of data anomalies cannot occur when the schema has been demonstrated to be in a given normal form.
C. refactoring a schema to be in a normal form guarantees the schema is *canonical*; that is, it is the one and only normalized schema that expresses the design correctly.
D. `SQL` queries will execute more efficiently the stronger a normal form the schema achieves.
E. they are a tool for checking whether the schema is attractive and easy to understand.

(i) [1pt] Suppose that we are told that relation **R** with attributes A, B, C, and D is in BCNF, and that *three* of the following *four* functional dependencies (FDs) hold for **R**. Which FD does not hold for **R**?

A. $A \mapsto BCD$
B. $BC \mapsto A$
C. $CD \mapsto B$
D. $D \mapsto C$
E. *There is not enough information to determine this here.*

(j) [1pt] If all a relation **R**'s keys are *singleton*—that is, each key is a set of just one attribute—then we know that **R**

A. must be in 2NF, but might not be in 3NF.
B. must be in 3NF, but might not be in BCNF.
C. must be in BCNF, 3NF, and 2NF.
D. must be in BCNF but cannot be in 2NF.
E. *None of the above.*

---

2. [10pt] **Entity/Relationship Modelling.** *Quick, draw!* DESIGN

Perfectionist Cruise Lines is well known for planning everything down to the last detail. They need a database to keep track of cruise passengers and their scheduled activities.

Requirements

A *passenger* is a cruise customer, identified by an assigned *p#*, and we keep the passenger's *name*. A passenger will be on the cruise ship for a number of days. Call each day that the passenger is on the ship a *booking*. For each *day* (*booking*), he or she *occupies* a *cabin*. (Model "day" in this case, as it will make the design cleaner.) A cabin is identified by a *room#* along with which *deck* it is on. Each *deck* has a name (*dname*), and a *level* which identifies it.

Perfectionist offers many *activities*. An activity is identified by *aname*, and has a *fee*. Not all activities are available every day. So we must keep track of which activity is available on any given day; call an activity occurring on a day an *event*. For each event, we should keep an attribute *capacity*, which says how many people at most can be scheduled for it. We need also to record on which *deck* the event takes place. There can be any number of activities on a given day (events), of course. And a given activity may be repeated on different days.

We need to keep track of which events—activities on given days—a passenger has *scheduled*. Of course, we would like our schema to ensure that an event that a passenger has signed up for is, in fact, for a day that the passenger is *booked* on the ship!

(a) [2pt] For each *event*, we keep a *capacity* which tells us how many people at most can participate. Should we also keep another attribute *#scheduled* that records how many passengers are scheduled for the event?

Why or why not?

No. This would be redundant. It can be computed already from the data kept in the schema.

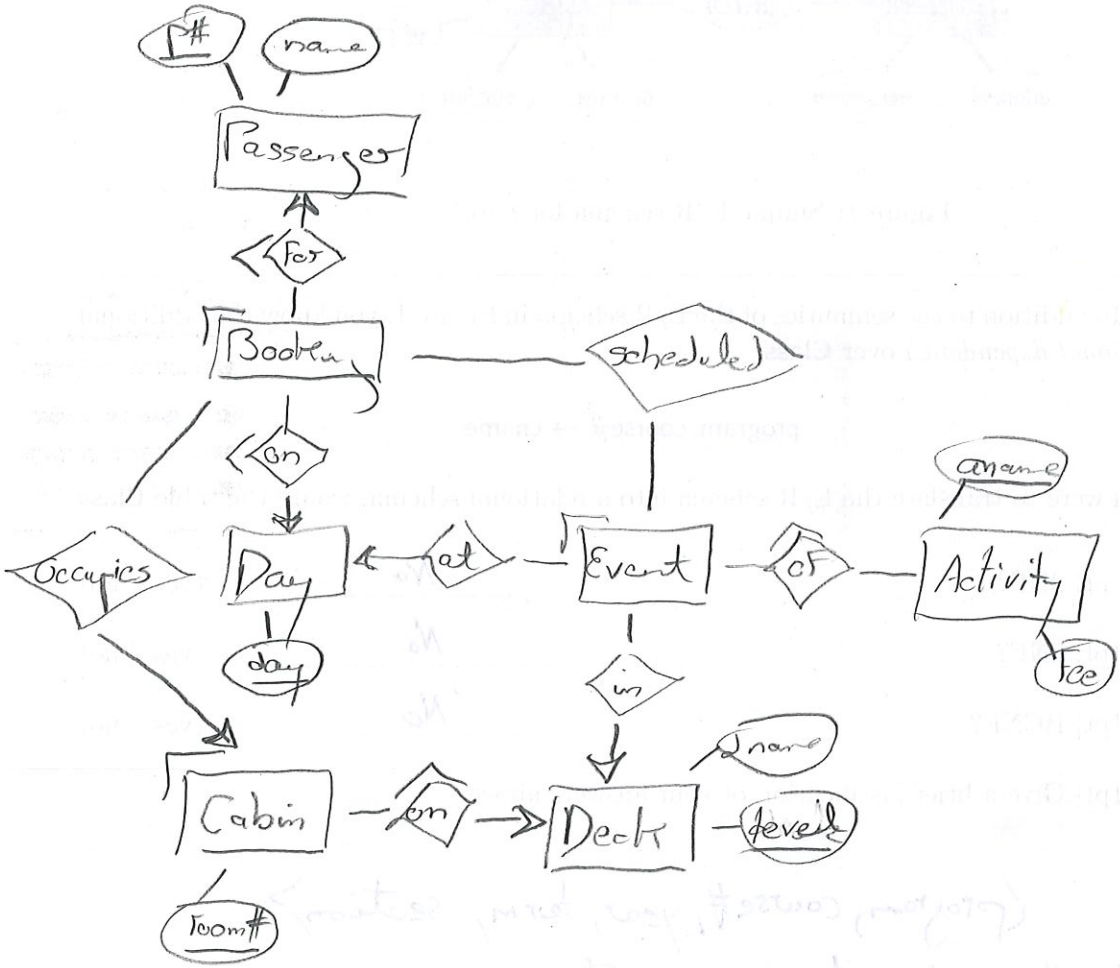
2: No, with explanation
1: No, but explanation flawed.
∅: Otherwise

(b) [1pt] In the Stanford dialect of E/R, could we capture in our E/R schema that a passenger can participate in up to at most five events a day?

No. The Stanford dialect does not give us a way to state this as part of the ~~sch~~ E/R schema.

1: No
∅: Otherwise

(c) [7pt] Draw an E/R schema that captures the requirements.



Semantics

+1 : Models Booking correctly

+1 : Event as a bridge entity btwn Day & Activity

+1 : scheduled handled well

+1 : Others ... Cabin on Deck, etc.

Syntax

+1 : Weak, etc handled correctly

+1 : Everything (Ent) has key declared

+1 : E/R language used correctly

3. [10pt] **E/R Logic.** *Highly illogical.* ANALYSIS

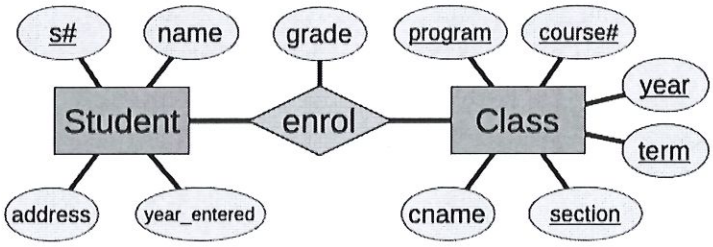


Figure 1: Simple E/R schema for *enrol*.

(a) [4pt] In addition to the semantics of the E/R schema in Figure 1, you know the additional *functional dependency* over **Class**:

$$\text{program}, \text{course\#} \mapsto \text{cname}$$

If you were to translate the E/R schema into a relational schema, would the table **Class** be in

i. [1pt] 2NF? No (yes / no)

ii. [1pt] 3NF? No (yes / no)

iii. [1pt] BCNF? No (yes / no)

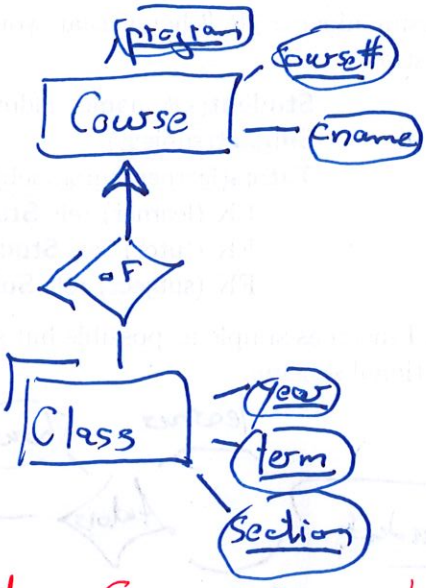
iv. [1pt] Give a brief justification of your answers above.

⟨program, course#, year, term, section⟩ is the only key for Class.

⟨program, course#⟩, the LHS of the FD, is a proper subset of that key; hence, this is a violation of 2NF. So of 3NF and BCNF too.

1: proper explanation
0: otherwise

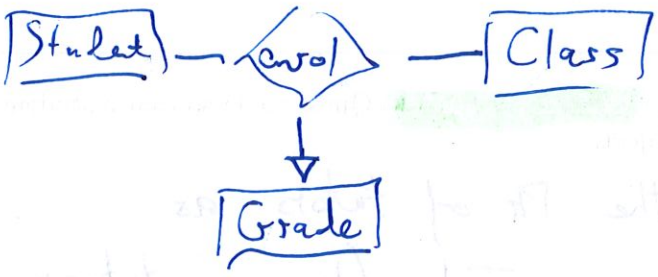
(b) [3pt] Refine (redraw) the E/R schema in Figure 1 to fix any issues that you found in answering Question 3a.



3: Make Class a weak ent on Course; Course has cname.
2: Weak ent, but some mistake
1: Evidence student understood question ∅: Otherwise.

(c) [3pt] Dr. Datta Bas recommends redoing the *Enrol* E/R schema as in Figure 1 to promote the grade attribute of **enrol** to be an entity set **Grade**—that way, *which* grades can be assigned are controlled by what goes into the entity-set **Grade**—and to make the **enrol** relationship *ternary* instead of *binary*.

Redraw the E/R schema with **Student**, **Class**, **Grade**, and **enrol** which implements Dr. Bas's recommendation.



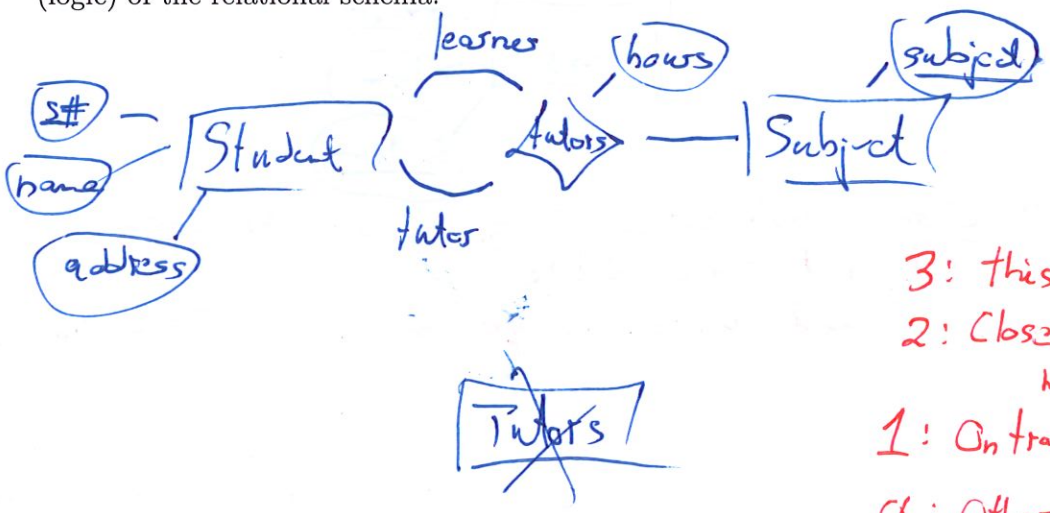
3: Ternary, ∅-1 arrow to Grade
2: Ternary, but mistake, such as 1 arrow to Grade
1: Evidence ...
∅: Otherwise

4. [10pt] **Translating Schema.** *Where's my Babel fish?!* EXERCISE

(a) [3pt] Draw a parsimonious E/R schema that would result in the following relational schema on translation.

Student(s#, name, address)
Subject(subject)
Tutors(learner, tutor, subject, hours)
FK (learner) refs **Student** (s#)
FK (tutor) refs **Student** (s#)
FK (subject) refs **Subject**

By *parsimonious*, I mean as simple as possible but still capturing properly the semantics (logic) of the relational schema.



3: this
2: Close, small mistake
1: On track
∅: Otherwise.

(b) [2pt] Show a change to the *relational schema* in Question 4a so that a student may have *at most one* tutor per subject.

Change the PK of tutors as
Tutors (learner, tutor*, subject)

2: this
1: Evidence student understands
∅: Otherwise

(c) [5pt] Consider the E/R schema in Figure 2.

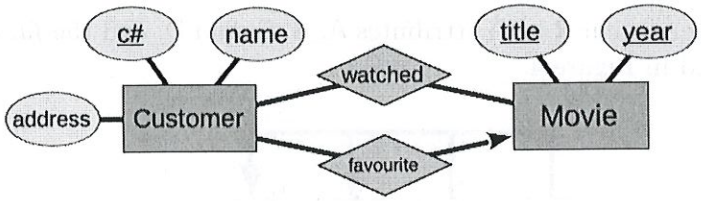


Figure 2: Simple *favourite-movie* schema.

You complain that this does not ensure that the customer has actually watched the movie that he or she claims is his or her favourite. So Dr. Mark Dogfury redraws it as in Figure 3.

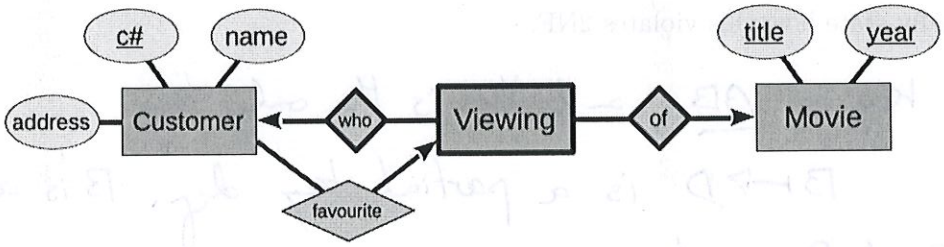


Figure 3: A more complicated *favourite-movie* schema.

Translate the Dr. Dogfury's E/R schema in Figure 3 into a relational schema—using the abbreviated style as used in Question 4a—showing *primary* and *foreign* keys. Ensure in your relational schema that a customer's favourite movie is, indeed, a movie that he or she has watched (viewed), as Dr. Dogfury intended in his E/R schema.

Customer (c#, name, address, title*, year*)
Fk (c#, title, year) refs Viewing -- Favourite
Movie (title, year).
Viewing (c#, title, year)
Fk (c#) refs Customer
Fk (title, year) refs Movie

+2: Viewing is correct
+1: Movie
+1: Customer
+1: Fk From Customer to Movie For Favourite

5. [10pt] **Design Theory.** *Find that key!* EXERCISE

(a) [2pt] Consider a relation **R** with attributes A, B, C, and D, and the *functional dependencies* as indicated in Figure 4.

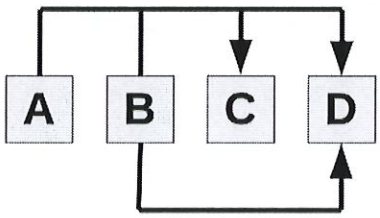


Figure 4: A functional-dependency diagram.

Formally state how this violates 2NF.

Key is AB, and this is the only key.
B ↦ D is a partial-key dep. B is a proper subset and D is not prime

2: proper subset & not prime
1: on track, but missing something
∅: Otherwise

(b) [2pt] Add one functional dependency that uses just *two* attributes (e.g., $X \mapsto Y$) to the set of functional dependencies in Figure 4 for **R** in Question 5a that would result in **R** being in 3NF.

D ↦ B.
Then AD is a key too.
And D then is prime, so not a violation of 3NF or BCNF.

2: A correct answer
1: Something really compelling...
∅: Otherwise

(c) [3pt] Consider relation **R** with attributes A, B, C, D, E, and F, with the following functional dependencies.

$$AB \mapsto C \qquad DE \mapsto F$$
$$BC \mapsto D \qquad F \mapsto A$$
$$CD \mapsto E$$

Calculate the keys of **R**.

AB BC BF BDE

3: perfect
2: Missing one, or an extra
1: On track, but more mistakes
0: Otherwise

(d) [3pt] Consider relation **S** with attributes A, B, C, D, E, and F, with the functional dependencies as in Figure 5.

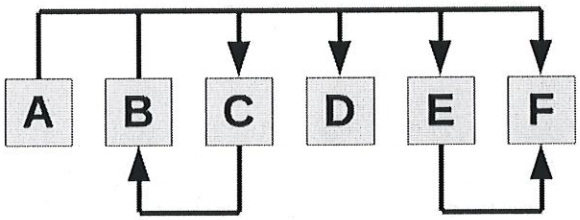


Figure 5: Another functional-dependency diagram.

Is there a *lossless-join* decomposition of **S** that is BCNF *and* that is *dependency preserving*?
Explain why or why not.

There isn't! Consider just ABC

This is not in BCNF! And it cannot be decomposed and still have us be dependency preserving)

3: This
2: Close to this
1: Evidence understood question.
0: Otherwise

EXTRA SPACE

RELAX. TURN IN YOUR TEST. RETURN TO THE WILD.

REFERENCE *(Detach this page for convenience, if you want.)*

The Normal-Form Definitions.

1NF: Domain of each attribute is an *elementary* type; that is, not a *set* or a *record structure.*

2NF: Whenever $\mathcal{X} \mapsto \mathsf{A}$ is a functional dependency that holds in relation $\mathbf{R}$ and $\mathsf{A} \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is not a proper subset of any key for $\mathbf{R}$.

3NF: Whenever $\mathcal{X} \mapsto \mathsf{A}$ is a functional dependency that holds in relation $\mathbf{R}$ and $\mathsf{A} \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is a key or a super-key for $\mathbf{R}$.

BCNF: Whenever $\mathcal{X} \mapsto \mathsf{A}$ is a functional dependency that holds in relation $\mathbf{R}$ and $\mathsf{A} \notin \mathcal{X}$, then

- $\mathcal{X}$ is a key or a super-key for $\mathbf{R}$.

An attribute A is called *prime* if A is in any of the candidate keys.

Figure 6: The Normal Forms.

REFERENCE

E/R diagram hand-drawing guide.

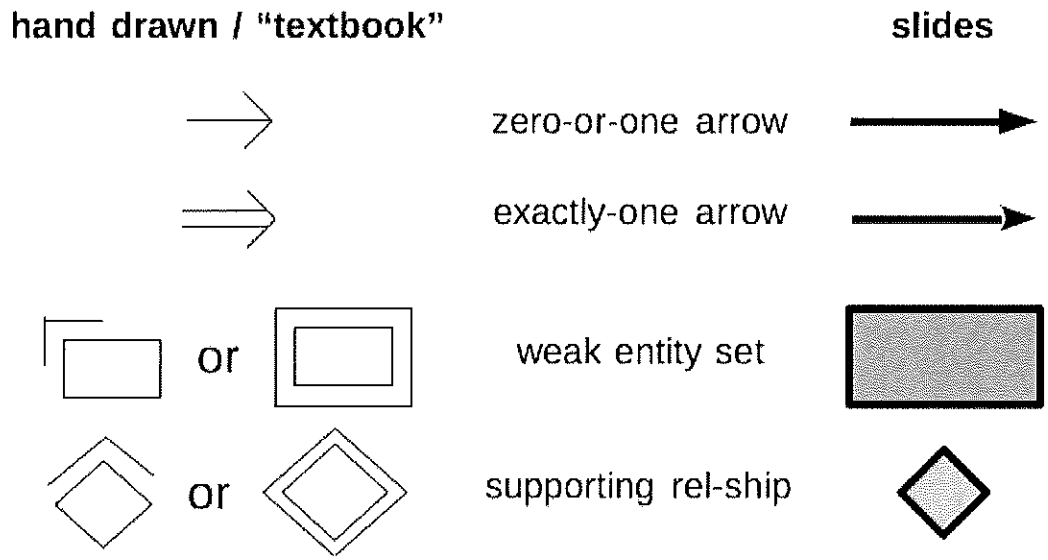


Figure 7: E/R drawing guide.