EECS-3421A: TEST #1

"Design"

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:	Fall 2016

Instructions

- 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 schema, the underlined attributes indicate a table's primary key (and are, hence, not nullable). Attributes appended with "$*$" are not nullable. Foreign keys are indicated by FK.
- The number of points a given question is worth is marked; it is worth one point, if not marked.
- There are five major parts worth 10 points each, for 50 points in total.

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

1. [10pt] **Entity/Relationship Modelling.** *With modelling, you pose.* Exercise

Requirements for the NSFW Database.

The *Nova Scotia Forestry Works* (NSFW) oversees logging[1] of trees in the province. They have commissioned you to do an E/R design for a database to help them track logging.

There are two types of entities that register with the NSFW: logging *companies* that intend to log in Nova Scotia; and *owners* who own forested *plots* of land that are zoned for logging. For a company to log, or for an owner to "sell" logging rights on a plot or his or hers, each needs to be licensed by the NSFW. A *licensee* is assigned a unique *licence#* by the NSFW, has an *issued* date, and has an *address* (either the company's address or the owner's address, depending). For an owner, we are to keep additionally the owner's *name*. For a company, we are to keep additionally the *title* of the company (essentially, its name, but the NSFW wants this called "title") and the *year* that it was *founded*.

A *plot* of land is a forested area that is zoned for logging. It is identified by a unique *plot#* and we record (exactly) one owner of the plot.

The NSFW keeps track of *types* of trees—e.g., oak, maple, pine, and spruce—that are available for logging. Each type is identified by a *type* name and has a description (*desc*). For each type in a plot, we record an estimate of the *number* of trees of that type *contained* on the plot and an estimate of the *tonnage* of wood of that type (that is, how much wood there would be if we logged all the trees of that type from the plot).

A company may enter a *contract* with an owner to log a specific type of tree from a given plot of that owner's. We should ensure that the plot is known to *contain* that type of tree in order to create a contract. The contract should record a logging *fee* (a base fee for the contract), a *quota* (which is the annual maximum tonnage amount that the company is allowed to log of that type of tree on that plot), and a *rate* (the cost per tonnage that they log).

NSFW keeps track per *month*—that is, a given *month* in a given *year*—for each contract the *haul*; that is, the *tonnage* of wood of that type that the company logged from the plot in that month.

---

a. [2pt] We want to be able to check in the database whether the amount that a company has logged annually in a plot for a given type of tree is more than the quota that its contract allows.

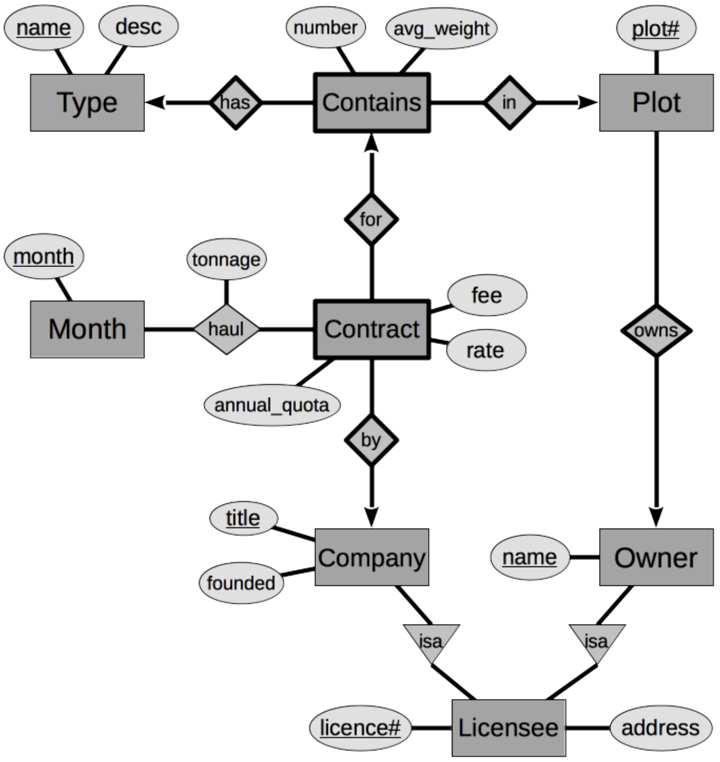
Should we add to our design an attribute annual_tonnage (wherever we placed quota) to accommodate this?

Why or why not?

No. It is already implicit in the design. The sum of monthy hauls for a year is this value.

[1] *logging.* The cutting down (harvesting) of trees for commercial use.

b. [8pt] Design an E/R diagram capturing the requirements for the NSFW database.



marking guide

+2 keys are correct
+1 rest of E/R logic w/ additional attr's correct
+1 Contains is a connecting entity
+1 that Contract connects to Contains
+1 rel-ship hauls and entity Month
+1 Contract is a connecting entity for rel-ship w/ Month
+1 ISA used for Company and Owner as Licensee

2. [10pt] **General.** *Luck of the draw.* Multiple Choice

---

a. [1pt] Functions of a relational database management system include all *except*

A. to ensure integrity constraints are not violated by updates to the data.
[✓ **B.**] to support general programming functionality through `SQL`, or another relational query language.
C. to support application programs accessing its databases through `SQL`.
D. to ensure that the changes of each transaction are committed in entirety or not at all.
E. to support the creation and altering of new databases.

---

b. [1pt] The rule of *data independence* is that

A. all information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.
B. all views that are theoretically updatable must be updatable by the system.
[✓ **C.**] changes that are made to the physical storage representations or access methods must not require changes be made to application programs.
D. changes that are made to tables that do not modify any of the data already stored in the tables must not require changes be made to application programs.
E. data in different tables must not be related.

---

c. [1pt] A *key* that is created just for the purpose of the database to distinguish tuples in the table is called

A. proper.
B. compound.
[✓ **C.**] surrogate.
D. relational.
E. diplomatic.

---

d. [1pt] NULL values can be used

[✓ **A.**] to opt a tuple out of enforcement of a foreign key.
B. to opt a tuple out of enforcement of the primary key.
C. to make a tuple to be non-updatable.
D. to add extra columns for a specific tuple.
E. to delete a tuple from the table.

---

e. [1pt] *Multiway relationships*

A. can never be equivalently replaced by (several) binary relationships.
B. have keys like entities.
C. are used to relate weak entities.
[✓ **D.**] relate more than two entities.
E. are not a part of the E/R model.

---

f. [1pt] A *weak entity*

[✓ **A.**] inherits part of its key from the “parent” entities to which it is related.

B. is an entity with *no* key.

C. is an entity with *no* attributes besides its key.

D. is *never* mapped to a table in conversion to a relational schema.

E. is the same thing as ISA in E/R.

g. [1pt] A weak entity set that contributes no attributes of its own to its key is called

A. a super entity set.

B. a sub-class.

C. a super-class.

D. a lame entity set.

[✓ **E.**] a connecting entity set.

h. [1pt] Relational schema differ from E/R diagrams in that

A. all tables are inherently equivalent to weak entities.

[✓ **B.**] attributes / columns are sometimes repeated between tables, unlike entities.

C. the concept of relationship cannot be expressed.

D. the concept of entity cannot be expressed.

E. the concept of multiway relationship cannot be expressed.

i. [1pt] Why are the normal forms useful?

A. They help us find anomalies in the data.

B. They are just a tool for checking whether our relational design makes sense or not.

C. If the schema is in BCNF, we are guaranteed that queries will execute faster than if it were not in BCNF.

[✓ **D.**] By having a relational schema in a given normal form, it guarantees that certain types of data anomalies cannot occur.

E. They are useless, but earn database consultants lots of money. (Don’t tell anyone!)

j. [1pt] The XML data model is called *semi-structured* because

A. it is computationally simpler than the relational (a *structured*) model.

B. it not formally defined, in contrast to the relational model.

C. there are no query languages for it.

[✓ **D.**] not all the data in an XML database needs to be *fully structured*, as it has to be in relational.

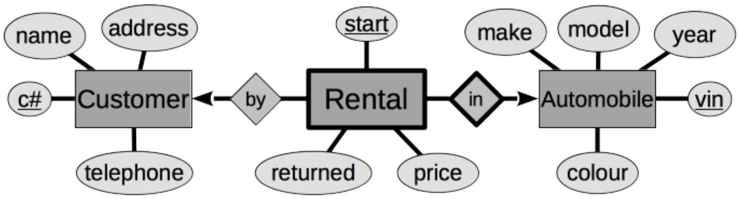
E. there is no corresponding notion of *schema*, as there is for relational.

3. [10pt] **Relational Schema.** *You don't choose your relations!* ANALYTIC

Note that attributes appended with "*" below are *not* nullable.

a. [3pt] Show an E/R diagram that captures

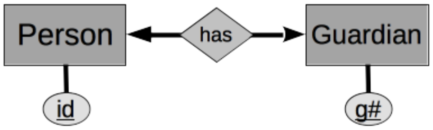
Customer(c#, name, address, telephone)
Automobile(vin, make, model, year, colour)
Rental(vin, start, c#*, returned, price)
FK (vin) refs **Automobile**
FK (c#) refs **Customer**



marking guide
+1 correct syntax & elements (attr's & 3 "concepts")
+1 recognition that Rental is an entity
+1 correct **weak** and arrowheads

b. [2pt] Reverse-engineer the following relational schema to an appropriate E/R diagram.

Guardian(g#)
Person(id, g#*)
FK (g#) refs **Guardian**
UNIQUE (g#)



marking guide
+1 correct entities & attr's
+1 correct 1-1 & arrowheads

c. [3pt] Write a relation for *Employee* which includes attributes for *emp#* (which uniquely determines an employee) the employee's *name*, *office#*, *phone#*, and department (*dept*). Also, this should include the employee's *boss*, who is another employee. (Assume each employee has no boss or one boss.)

Follow the style of the relational schema presented in Question 3a. Write any additional relations, if needed.

Employee*(emp#, name, office#, phone#, dept, boss)*
FK (boss) refs **Employee** *(emp#)*

marking guide
+1 right rel'n & attr's
+1 has a 'boss' attr
+1 FK cast correctly

d. [2pt] Is it possible to capture any E/R model correctly in a relational schema without needing any *compound* key (that is, a key consisting of several attributes)?

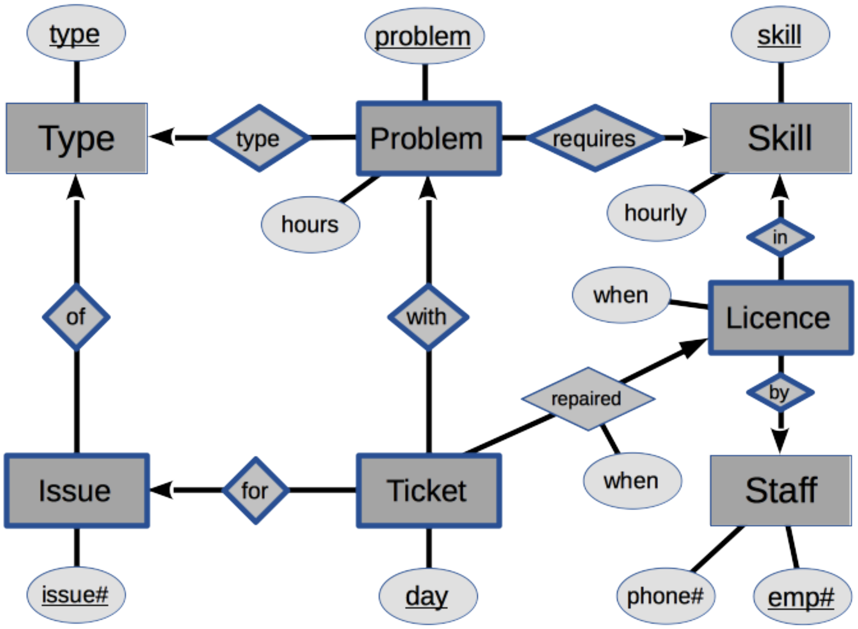
Why or why not?

No. Because many-many rel-ships must be converted into tables. These tables' keys are the union of the keys' attributes of the entities that the rel-ship relates.
Likewise for weak entities to tables.

marking guide
+1 No.
+1 Decent explanation.
or
+1 For yes, if states surrogate keys and UNIQUE.

4. [10pt] **Conceptual to Schema.** *You don't choose your relations!* Exercise

Translate the following E/R diagram faithfully to a relational schema. Follow the style of the relational schema presented in Question 3a. Use a *restrictive* interpretation.

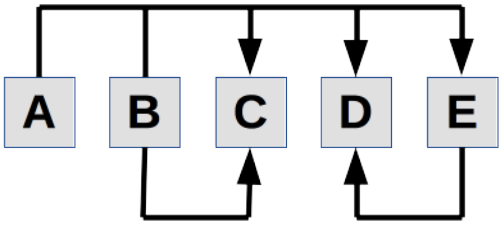


Type(*type*)
Issue(*issue#, type*)
  FK (type) refs **Type** *– of*
Skill(*skill, hourly*)
Problem(*problem, type, skill, hours*)
  FK (type) refs **Type** *– type*
  FK (skill) refs **Skill** *– requires*
Staff(*emp#, phone#*)
Licence(*skill, emp#, when*)
  FK (skill) refs **Skill** *– in*
  FK (emp#) refs **Staff** *– by*
Ticket(*issue#, problem, day, type, skill*, emp#, when*)
  FK (issue#) refs **Issue** *– for*
  FK (problem) refs **Problem** *– with*
  FK (skill, emp#) refs **Licence** *– repaired*

marking guide
+3 right syntax
+2 capture all tables needed
+3 have correct FKs
+1 got the additional attr's
+1 'repaired' is 0-1 and 'skill' is not null for Problem FK

5. [10pt] **Design Theory.** *Who's normal?!* Short Answer

a. [5pt] Consider the following relation **R** with attributes A, B, C, D, and E and with functional dependencies (FDs) as marked.



i. [1pt] What is the key?

AB

ii. [1pt] What is the nickname for the type of FD that violates 2NF?

partial key

iii. [1pt] What is an FD from above that violates 2NF?

B ↦ C

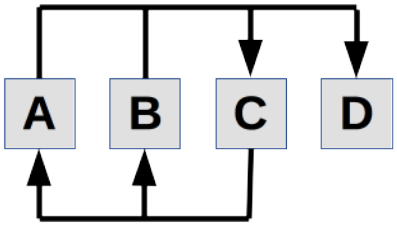
iv. [1pt] What is the nickname for the type of FD that violates 3NF (but not 2NF)?

transitive

v. [1pt] What is an FD from above that violates 3NF (but not 2NF)?

E ↦ D

b. [3pt] Consider the following relation **R** with attributes A, B, C, and D and with functional dependencies (FDs) as marked.



State *yes* or *no* for each of the following. For a *no* answer, state a violating FD. (E.g., "Yes." Or, "No, E ↦ F.")

i. [1pt] Is **R** in BCNF? yes

ii. [1pt] Is **R** in 3NF? yes

iii. [1pt] Is **R** in 2NF? yes

---

c. [2pt] If a relation is *not* in BCNF but it is in 3NF—so it has a "back" dependency—does the relation *necessarily* have more than one key?

State *yes* or *no*, and explain *briefly*.

Yes, it does.
Consider a relation with the set of attributes $\mathcal{R}$. A back dependency is where a set of attributes, $\mathcal{A}$, that are not *part of a key, a set of attributes $\mathcal{K}$ such that $\mathcal{K} \mapsto \mathcal{R}$, which functionally determines an attribute (or attributes) that* are *part of that key $\mathcal{K}$; i.e., $\mathcal{J} \subset \mathcal{K}$, $\mathcal{A} \nsubseteq \mathcal{K}$, and $\mathcal{A} \mapsto \mathcal{J}$. By* Augmentation, *$\mathcal{A} \cup (\mathcal{K} - \mathcal{J}) \mapsto \mathcal{J} \cup (\mathcal{K} - \mathcal{J})$; that is, $\mathcal{A} \cup (\mathcal{K} - \mathcal{J}) \mapsto \mathcal{K}$. By* Transitivity *then, $\mathcal{A} \cup (\mathcal{K} - \mathcal{J}) \mapsto \mathcal{R}$. $\mathcal{A} \cup (\mathcal{K} - \mathcal{J})$ is also a key of the relation, which is distinct from $\mathcal{K}$.*
$\therefore$ The relation has more than one key.

Extra Space

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 $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then either

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

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

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

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

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

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

Figure 1: The Normal Forms.

REFERENCE

E/R diagram hand-drawing guide.

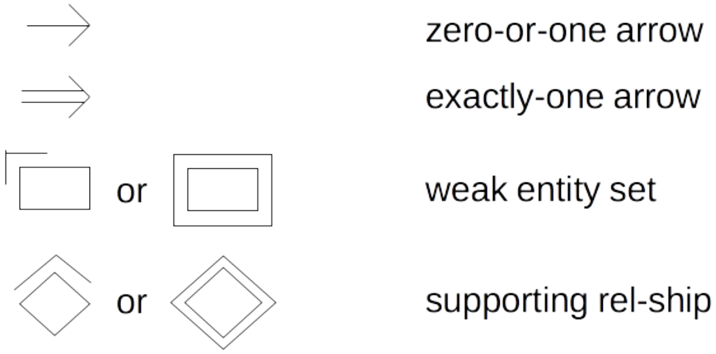


Figure 2: E/R drawing guide.