Carleton University School of Computer Science

COMP 3005 Fall 2010 Mid-Term Test

(Write on the test paper — use the back if necessary;)

Name: __ANSWERS Total 35 Marks

Definitions

The following definitions are to be used for answering the related questions on the test. These definitions are consistent with those in your notes. Notice that the definition for prime attribute refers to **any** key, not just the primary key of a table schema.

Prime Attribute: If R=(A1,A2,…An) is a table, attribute Ai is prime if there exists a key K of R such that Ai is an element of K. If an attribute is not **prime** it is called **non-prime**.

Trivial Dependency: A dependency X->Y is **trivial** if Y is a subset of X.

Closure of F: if F is a set of functional dependencies, the closure of F, denoted F+, is {X->Y| F logically implies X->Y}.

Partial Dependency: Suppose X is a key of table R and Y is a proper subset of X, and A is an attribute not in Y. Then Y->A is a **partial dependency**.

2nd Normal Form: A table R with associated functional dependencies F is in 2nd normal form if F+ contains no partial dependencies Y->A where A is non-prime.

Transitive Dependency: Let Y be a set of attributes from table R and A be an attribute not contained in Y. The functional dependency Y->A is a **transitive dependency** if Y is neither a superkey of R nor a proper subset of a key of R.

3rd Normal Form: A table, with dependencies F, is in 3rd normal form if it is in 2nd normal form and if F+ contains no transitive dependencies Y->A where A is non-prime. (Equivalently, a table is in 3rd normal form if, for each non-trivial dependency Y->A, Y is a superkey or A is prime)

Boyce-Codd Normal Form: A table, with dependencies F, is in BCNF if F+ contains no partial or transitive dependencies. (Equivalently, a table is in BCNF if the left side of each non-trivial dependency in F+ is a superkey.)

Question 1) [15 marks] Consider the following attributes (facts) that are to be stored in a relational database about bicycle club members that participate in bicycle races.

```
memNo     //id number of an club member (unique)
bdate     //birthdate of an club member
name      //name of member
address   //address of member
email     //email address of member
race      //name of race (not unique)
raceDate  //date of race
official //race official or referee
route    //name of the route the race will follow
laps  //number of laps of the route racers must complete
courseSection //name of a road that is part of the route
sectionNumber //position of courseSection in the route
direction //direction to travel on route section (N,S,E,W)
time //time a racer took to complete a race
place //placement (1st, 2nd, 3rd, etc.) a racer got in a race
start //name of the start location of a race
finish //name of an end location of a race
```

Constraints
memNo is unique among members and members only have one name, birthdate, address, and email each.

A member can race in many races, and each race has many participants. A race name is not unique, but the race name and date combination is unique. The database must record the results that members get in each race that they enter.

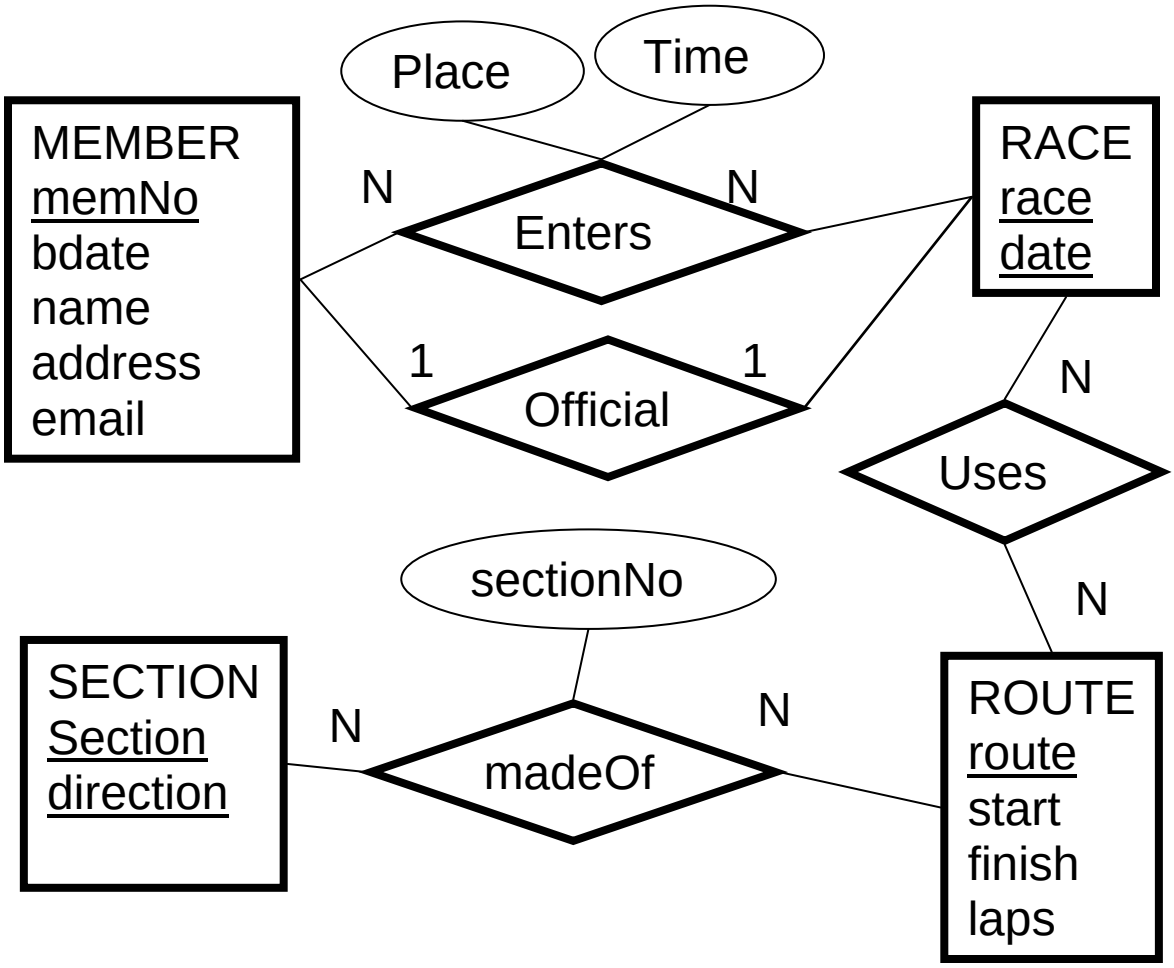
Each race is run on a particular race course or route that the racers must follow. A route has a start, finish location, a number of route sections (road segments) and a number of laps of the course that the racers must complete. Each section of the course consists of a road name and a direction to travel along that road (i.e. north, south, east, or west). In addition a sectionNumber is to keep track of how the course sections join up. That is, what sequence they are in to form the route. Each race event has an official, or referee, who is also a club member.

The same route can be used by more that one race. Different routes might use the same road sections as well.

It must be possible from the information in your database to construct the route. (That is, to determine the order of the course sections that make up a route.)

The race has a start location and a finish location. (This would be the name of the closest landmark to the start and finish line respectively.)

Question 1a) [10 marks] Construct an E-R diagram for the situation described above observing the constraints and the following rules. Your attributes can only be the ones listed above (you cannot make up any new attributes) Your design must not repeat any information. Any facts should only appear once when the tables are constructed from the E-R diagram.
Your E-R diagram should clearly show the key for each entity and the cardinality for any relationships and weak vs. strong entities. Make up useful names for your Entities and Relationships.



Marking:
Start with 10 marks, deduct a mark for any error, e.g. missing key, missing cardinality etc.
Deduct 2 mark per attribute for any new attributes that were added (they were not supposed to add attributes.
Deduct 2 marks if you cannot reconstruct the course (order of sections) from the design.
Deduct 1 mark each for any unnecessary items they have (entities that are not needed, relationships that are not needed) It is OK if they used multivalued attributes, but make sure they were properly mapped to tables in the next part.

Question 1b) [5 marks] Construct a set of tables that corresponds exactly to your E-R diagram. Show the keys for each table and any foreign keys referencing other tables.

Table: MEMBER
MemNo, Name, Bdate,Address, Email

Table ROUTE
ROUTE, Start, Finish, Laps

Table SECTION
Section, Direction

Table: RACE
Race, RaceDate, Official
//Offical is a foreign key referring to MEMBER TABLE: MemNo.

Table: RACECOURSE
Race, RaceDate, Route //Route is Foreign Key Referincing Table RACE

Table SECTIONOF

Route, Section, Direction, SectionNumber
//Route is foreign key referencing table ROUTE
//Section,Direction is foreign key referencing table SECTION

Table ENTRYS
MemNo,Race,RaceDate,Time,Place
//MemNo is foreign key referencing table MEMBER
//Race, RaceDate is foreign key referencing table RACE

Marking:
Start with 5 marks and deduct 1 mark for any mistake. E.g. mistakes include:

-Table that does not correspond directly to E_R diagram.
-Attributes that are added that do not correspond to part of the diagram
-Error in keys or forgotten keys
-Errors in, or omitted, foreign keys.

Question 2) [4 marks] Consider the set of attributes R={A,B,C,D,E,F} and the following set of functional dependencies proposed by the table designer.

Fd= {ABD->AC, B->E, BA->E, C->BE, AD->FB, C->E }

A colleague suggests that they use the following dependency set instead

Fm = { AD->CF, C->B, B->E}

To do this we would have to prove that the two sets are equivalent.

2a)[2 marks] Using Armstrong's axioms or the six inference rules of functional dependencies, show that the dependency AD->CF in Fm is logically implied by those in Fd (Show clearly each time one of the rules is applied)

AD>-CF implied because :

AD->BF, given in Fd
AD->F decomposition rule

ABD-AC given in Fd
ABD->C decompostion rule

AD->B given in in Fd
AD->ABD augmentation rule
AD- >C transitive rule : AD- >ABD, ABD->C

AD->CF union rule, AD->C, AD->F

2b)[2 marks] (Now the reverse) Using Armstrong's axioms or the six inference rules of functional dependencies, show that the dependency ABD->AC in Fd is logically implied by those in Fm (Show clearly each time one of the rules is applied)

ABD>-AC implied because :

AD->CF, given in Fm
AD- >ACF augmentation rule (add A to both sides)
ABD->ABCF augmentation rule (add B to both sides)
ABD- >AC decomposition rules (ABD->AC, ABD->BF)

Question 3 [4 marks]

For each question below a relation R has been defined over attributes A,B,C,D,E,F along with a set of functional dependencies that apply to them. State the highest normal form the table R=ABCDEF would currently satisfy

3a) F={ AB-> CDEF, EF ->C }

Current Normal Form = 2nd NF [1 mark]

3b) F={ AB-> CDEF, BC->D, }

Current Normal Form = 2nd NF [1 mark]

3c) F={ AB-> CDEF, B->C , D->C}

Current Normal Form = 1nf //B->C is partial that violates 2nf [1 mark]

3d) F={ AB-> CDEF, EF->B , D->B }

Current Normal Form = 3nf [1mark]

Question 4) [12 marks] The following tables have been implemented to represent the telephone switching example described in the course and your assignments. The primary keys are underlined in the table descriptions. Table attributes are shown in [] brackets.

```
FACILITIES[PORT_ID]
 //represents line, trunks or treatments by portid

LINES[PORT_ID, AREA_CODE, OFFICE_CODE, STATION_CODE, STATE]
//represents lines of the switch including their directory number
//STATE can be "busy" or "idle"

TRUNKS[PORT_ID, OTHER_SWITCH]
//trunks that connect to other (foreign) switches

TRUNK_CHANNELS[PORT_ID,CHANNEL,STATE]
//use of trunk channels, channel = 0…23, STATE = "busy" or "idle"

SUBSCRIBERS[PORT_ID, NAME, ADDRESS]
//customers who rent lines

SERVICES[SERVICE_CODE, SERVICE_NAME]
//services that customer lines can subscribe to

SERVICE_SUBSCRIBERS[PORT_ID,SERVICE_CODE]
//services that lines currently subscribe to

TREATMENTS[TREATMENT_CODE,PORT_ID,TREATMENT_NAME]
//treatments, like busy tone, that can be applied if a call fails

TRUNK_ROUTES[PORT_ID, AREA_CODE, OFFICE_CODE]
//area codes and office codes served by particular trunks
//Sample trunk_routes data:
```

PORTID	AREA_CODE	OFFICE_CODE
100	613	235
100	613	232
101	613	000
101	416	765
101	416	763
101	905	555
101	905	543
101	416	238
102	905	000
102	416	000
103	802	000
103	514	000
103	000	000

```
CALLS[CALL_ID,ORIG_PID, ORIG_CHANNEL, TERM_PID,
      TERM_CHANNEL,AREA_CODE,OFFICE_CODE,STATION_CODE,STATE]
//describes calls in progress
//Area, office, station codes represent the dialed digits
//state can be "idle", "dialing", "ringing", "talking", "disconnected"
//ORIG_PID, and TERM_PID are foreign keys referring to PORT_ID's
```

For the following questions provide a **relational algebra** query that would produce a table with the required information

4a) [3 marks] Write a relational algebra expression that will give the name and address of all the subscribers that are the originator of a call.

PROJ[name,address](subscribers JOIN[port_ID=Orig_PID] calls)

3marks
1 for project, 1 for joining the correct tables, 1 for correct join condition

Their answers might look different but should be logically equivalent.

4b) [3 marks] Write a relational algebra expression that will list the names and address of those subscribers that originated a call to Toronto. (Toronto numbers are those with a 416 or 905 area code.)

```
PROJ[name,address](
    SEL[area_code = '416' or area_code = '905'] (
        subscribers JOIN[port_ID=Orig_PID] calls
        )
)
```

3 marks:
1 for correctly finding originators
1 for selecting only the calls dialed to Toronto
1 for projecting the right columns

4c) [6 marks] We want to find all available trunk channels to use for a call going to Toronto (area code 416 or 905). Write a relational algebra query that will list the port ID and channel number of **all** the idle channels that can be used to route the call to Toronto.

PROJ[portID, channel](
  SEL[state="idle AND (area_code = 416 OR area_code = 905 OR area_code =000)]
(TrunkChannel * TrunkRoutes)
)

[6 marks]
1 mark for recognizing 416, 905 as valid routing area codes
1 mark for recognizing 000 as valid routing area code
1 mark for selecting only idle channels
1 mark for correct logic (AND and OR etc
1 mark for correct Join (i.e. TrunkChannels with TrunkRoutes
1 mark for projecting the correct information