Database Concepts: Assignment 2

Name: **Duncan Do**

Student Number: **s3718718**

1. SQL

```
--1.1
SELECT academic.acnum, academic.givename, academic.famname, COUNT(paper.panum)
FROM academic FULL JOIN author ON academic.acnum = author.acnum FULL JOIN paper ON author.panum = paper.panum
GROUP BY academic.acnum, academic.givename, academic.famname
--1.2
SELECT department.deptnum, department.deptname, department.instname
FROM department
WHERE department.deptnum IN (
SELECT academic.deptnum
FROM academic
WHERE academic.acnum NOT IN(
SELECT interest.acnum
FROM interest)
)
--1.3
SELECT field.fieldnum, field.title, COUNT(interest.acnum)
AS "NO.ACADEMICS INTERESTED"
FROM field, interest
WHERE interest.fieldnum = field.fieldnum
GROUP BY field.fieldnum, field.title
ORDER BY field.fieldnum ASC
--1.4
SELECT field.fieldnum, field.title, COUNT(interest.acnum)
FROM field, interest
WHERE interest.fieldnum = field.fieldnum
HAVING COUNT(interest.acnum) >= 10
```

```
GROUP BY field.fieldnum, field.title
--1.5 Assuming the max value is known
SELECT academic.acnum
FROM academic FULL JOIN author ON academic.acnum = author.acnum
GROUP BY academic.acnum
HAVING COUNT (author.panum) = 52
--1.6
SELECT count(academic.acnum)
FROM academic
WHERE academic.acnum NOT IN (
SELECT author.acnum
FROM author
WHERE academic.acnum = author.acnum
)
--1.7 Assuming we don’t need to find data with spaces on either end
SELECT field.fieldnum, field.title
FROM field
WHERE upper(field.title) LIKE '%DATA%' AND field.fieldnum EXISTS (
SELECT interest.fieldnum
FROM interest FULL JOIN academic ON interest.acnum = academic.acnum
WHERE interest.fieldnum = field.fieldnum academic.deptnum = 100
))
--1.8
SELECT DISTINCT a1.panum
FROM author a1, author a2, academic ac1, academic ac2
WHERE a1.acnum = ac1.acnum AND a2.acnum = ac2.acnum AND ac1.deptnum = ac2.deptnum AND ac2.acnum != ac1.acnum AND a1.panum = a2.panum
--1.9
SELECT DISTINCT interest.fieldnum
FROM interest
WHERE interest.fieldnum NOT IN (
```

SELECT interest.fieldnum

FROM interest, academic

WHERE academic.acnum = interest.acnum AND academic.deptnum = 126

)

--1.10

a) Finding academics who have an interest in a field that isn't logic

b) Finding academics who have an interest in a field that is logic as well as a field that isn't logic

2. The Relational Model

2.1 Give all likely FDs

***Assuming we include even redundant FDs since the question includes "all likely"**

customerNo → customerNo

customerNo → customerName, phone, streetAddr, city, state, postcode

orderNo → orderNo

orderNo, productNo → quantity

productNo → unitPrice

orderNo → customerNo

orderNo → salesRep

2.2 Give the candidate keys for the ABC relation. Explain your answer

- productNo

- orderNo

The only way to get the unit price of a product is to go through the ID for said product (ProductNo). Therefore, there are fields that are dependant on productNo, but on the other hand, no FDs exist X→Y where productNo is Y. Thus, it should be a key.

The only way to get to orderNo is itself (Which would be removed when normalising). Therefore it should be a key.

Any other fields are involved in FDs X → Y where they are Y. Regardless of if they exist as X; such as customerNo, since they are dependant on other fields they cannot be considered when selecting the key.

2.3 Give {CustomerNo}+ and {orderNo, salesRep}+ based on the FDs for Question 2.1

{customerNo}+ = {customerNo, customerName, phone, streetAddr, city, state, postcode}

{orderNo, salesRep}+ = {orderNo, salesRep, customerNo,}

2.4 Is the relation ABC in BCNF or 3NF? Explain your answer

Not BCNF. Because they key of the relation is {customerNo, productNo, orderNo} and FDs exist where the left-hand side is not the key. It is however 3NF, because all left-hand sides of FDs are candidate keys of the relation/are part of the key of the relation

3. Normalisation

3.1 Give the minimal basis for the given FDs

custNo → custName

custNo → address

custNo → redit-limit

custNo → discount

productNo → price

productNo → desc

orderNo → empID

orderNo → custNo

orderNo, productNo → quantity

3.2 The Transaction relation is not in BCNF or 3NF. Give the reason using the FDs on attributes

Not BCNF. FDs exist of X→Y where X is not the key full of the relation (E.g. orderNo → EmpID, lacks productNo on the left).

Not 3NF. FDs exist of X→Y where X is not a candidate key/part of the full key of the relation (E.g. custNo → discount, custNo is not a candidate key).

3.3 Follow the 3NF decomposition algorithm to decompose the Transaction relation into relations in BCNF or 3NF

1. Minimal Basis for FDs:

custNo → custName

custNo → address

custNo → redit-limit

custNo → discount

productNo → price

productNo → desc

orderNo → empID

orderNo → custNo

orderNo, productNo → quantity

2. Key of the Class

{productNo, orderNo}

3. Constructing relations:

R1(custNo, custName)

R2(custNo, address)

R3(custNo, redit-limit)

R4(custNo, discount)

R5(productNo, price)

R6(productNo, desc)

R7(orderNo, empID)

R8(orderNo, custNo)

R9(orderNo, productNo, quantity)

Combining relations:

Class1(<u>custNo</u>, custName, address, redit-limit, discount)

Class2(<u>productNo</u>, price, desc)

Class3(<u>orderNo</u>, empID, custNo*)

Class4(<u>orderNo*, productNo*</u>, quantity)

4. Relation for the key of the class

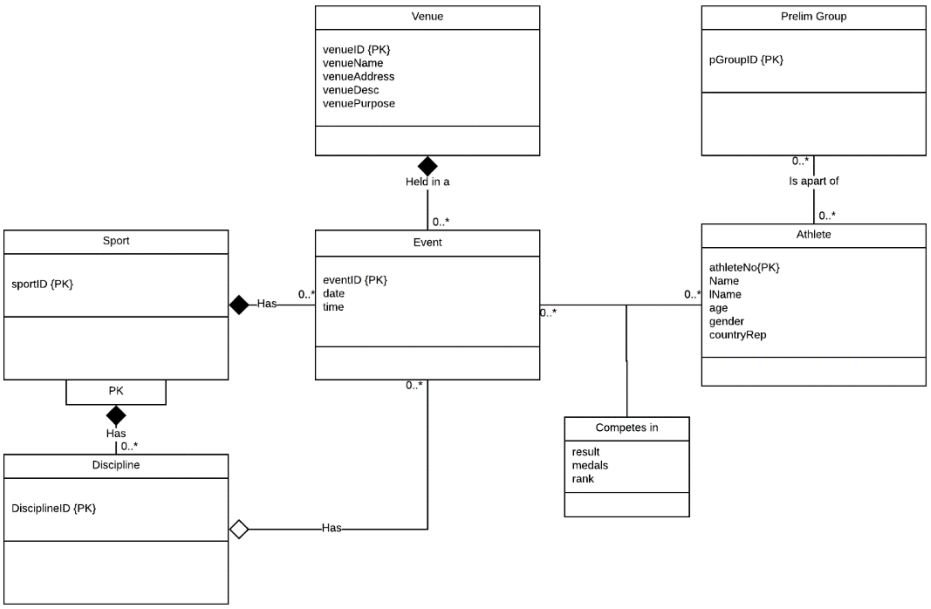
Not Needed, the key of the class exists in relation 'Class4'

Now for all FDs X → Y, X is the key of (one of) the relation

4. <u>ER Model</u>

Assumptions:

- Each event can only be hosted once, thus only needing 1 venue per event
- A sport can have 0 to many events or have 1 or more disciplines that have 0n to many events
- Foreign keys need not be represented
- Even if an event is of a discipline, it must be a part of a sport which the discipline is of. On the other hand, an event needs a sport but doesn't need a disciple if the sport itself doesn't have one
- A disciple is identified by the sport it is of
- An event must be held at some sort of venue
- An athlete keeps the preliminary group he/she's in even when they progress to individual competition
- An athlete is apart of a preliminary group but still competes as an individual



5. ER to relational schema mapping

Driver(driverEmpID, givename, surname)

TicketInspector(tickInspEmpID, givename, surname)

StationMaster(sMasterEmpID, givename, surname, sName*)

DriveFor(driverEmpID*, number*)

Inspect(tickInspEmpID*, name*)

Has(lName*, sName*, stopNo)

Run(number, lname*, time, direction, express)

Line(lName, length)

Station(sName, lName*, premium)

Terminate(sName*)