**DTA (M) Technical Report 2019** 2460681S > Wesley Scott

**Task 1: Relational Schema Modelling**

***Initial tables:***

STUDENT (SID, SNAME, HCID, HCNAME, TID, TNAME, JYEAR)

TOPIC(SID, ADVNAME, ADVID, TOPIC)

TEXTBOOK(COURSE, ADVID, TEXTBOOK)

**Task 1.1:**

***TABLE: STUDENT***

FD1: SID > SNAME, HCID

The unique student ID entry defines the student’s name, and as a student may only have one static home city, also defines their HCID.

FD2: HCID > HCNAME

The students home city ID defines home city name.

Therefore, HCNAME is defined by SID via transitive dependency such:

SID > HCID > HCNAME

FD3: TID > TNAME

The unique team ID (TID) defines the team name.

FD4: SID, JYEAR > TID

As it is possible for a student to join 0 or 1 teams per year, we can also say that a combination of SID and JYEAR can uniquely identify a Student’s Team ID (TID). As a student may join the same team again, we cannot with certainty say that {SID, TID} > JYEAR will always hold true. Also, as SID, JYEAR > TID therefore SID, JYEAR > TNAME via transitive dependency.

Since SID, JYEAR is the chosen Primary Key, it can be said that SNAME and HCID become partial dependencies, only requiring SID. This will be considered during normalisation.

***TABLE: TOPIC***

For each TOPIC a student has their own advisor; the same student can have multiple advisors, depending on which topic is referenced, therefore FDs are as follows:

FD1: SID, TOPIC > ADVID

ADVID is provided by a combination of SID and TOPIC. An advisor may only advise one TOPIC. A student will have multiple advisors, one for each of their topics.

FD2: ADVID > TOPIC

However, it is also true that an advisor may only advise one TOPIC.

FD3: ADVID > ADVNAME

Each unique advisor has their own name.

Therefore SID, TOPIC > ADVNAME via transitive dependency.

***TABLE: TEXTBOOK***

Each course has a set number of TEXTBOOKs, these textbook(s) must ALL be accounted for every time an ADVIOR for a COURSE is referenced in the relation.

To uniquely identify a tuple in this table currently we need all three attributes to act as a primary key.

COURSE, ADVID, TEXTBOOK > COURSE, ADVID, TEXTBOOK

**Task 1.2:**

***In TABLE: STUDENT***: (FD4: SID, JYEAR > TID) states that a student’s team ID is defined by both the student ID and the join year. As a student can join zero teams in a year, it is then possible they have joined no team, and thereby an inserted tuple could potentially have NULL values in both Team ID (TID), as well as JYEAR. This would affect the use of JYEAR as a primary key, which cannot be null, and also creates issues using Team ID as a foreign key. During creation of the database it would be recommend NOT allowing NULL values for both attributes or, creating an edge case catch for those without teams (and therefore no join year).

***In TABLE: TOPIC***: With TOPIC, we are told that an advisor can only cover one topic, so due to this model, when inserting a new student with an advisor already present in the table, (see ADVID 1 (McReader)), we must duplicate information in the topic and advisor name attributes. This means that if an Advisor were to change Topic, we would have to update a large number of tuples. This can be improved upon when normalising.

***In TABLE: TEXTBOOK***: Due to the nature of this table, where each Advisor will use all of the necessary textbooks within a course, every time we insert a new ADVID against a course, we will require multiple tuples for each of the textbooks applicable to that course with the new Advisor. In the same way, a new textbook for either Database or Advanced Database would require new tuples for each of the advisors assigned to that course.

**Task 1.3:**

***TABLE: STUDENT***

STUDENT (SID, SNAME, HCID, HCNAME, TID, TNAME, JYEAR)

The table STUDENT is currently in 1NF, due to partial dependencies on the Primary Key.

**In BCNF:**

STUDENT(SID, SNAME, HCID(FK))

SID and JYEAR are the Primary Key, HCID becomes the foreign key, referencing HOMECITY(HCID).

STUDENT\_JOINYEAR(SID(FK), JYEAR, TID(FK))

By moving SID and JYEAR to a new relationship we remove transitive dependencies from STUDENT. TID is the FK referencing TEAM(TID).

HOMECITY(HCID, HCNAME)

HCID becomes the primary key of a new relation HOMECITY.

TEAM(TID, TNAME)

Team ID becomes the primary key of a new relation TEAM.

There are now no partial dependencies on any non-prime attribute within all tables, as well as no transitive dependencies, satisfying Boycee-Codd- Normal Form.

***TABLE: TOPIC***

TOPIC(SID, ADVNAME, ADVID, TOPIC)

ADVNAME is given by ADVID, and is therefore transitively dependent on {SID, TOPIC}.

**3NF:**

Consider the alternative, split Topic into two relations:

ADVISEE\_TOPIC(SID, TOPIC, ADVID(FK))

A student can have multiple Advisors, but only one for each topic, therefor SID and TOPIC are required to uniquely identify a tuple. Due to the FD ADVID > TOPIC, this table cannot be normalised to BCNF.

ADVISOR(ADVID, ADVNAME)

ADVID uniquely identifies ADVNAME, which would be transitively dependant on primary key (FD4: SID, TOPIC > ADVID > ADVNAME) before split using FD3(ADVID > ADVNAME).

**Task 1.4:**

Due to the nature of the relationships in this table, where each Advisor will use all of the necessary textbooks within a course, every time we insert a new ADVID against a course, we will require multiple tuples for each of the textbook(s) applicable to that course.

**TEXTBOOK**  
  
TEXTBOOK(COURSE, ADVID, TEXTBOOK)

This table causes some issues regarding splitting, due to the fact that no unique foreign key can be determined in any split state.

Based on this finding, creating on an additional intermediary table with just the splitting attribute would be required to use this current model effectively, and succeed in improving upon the anomalies.

Consider:

COURSE\_TEXTBOOK(COURSE, TEXTBOOK)

ADVISOR\_COURSE(ADVID, COURSE)

In this ideal split, we create two tables with 2 attributes as the primary key.

It is desired to join these tables via courses. This reduces the number of unnecessary tuples from each individual textbook being recorded individually for each advisor within a course. This also allows us to deal with the issue arising when the same textbook could be used in multiple courses, where COURSE\_TEXTBOOK(COURSE, TEXTBOOK) accounts and uniquely identifies these occurrences. ADVISOR\_COURSE(ADVID, COURSE) accounts for the potential issues which may arise where the same advisor belongs to multiple courses.

Therefore, the proposed solution is as follows:

COURSE\_TEXTBOOK(COURSE(FK), TEXTBOOK) – references INTERMEDIARY\_COURSE

INTERMEDIARY\_COURSE(COURSE) – unique when defined within its own table

ADVISOR\_COURSE (ADVID, COURSE(FK)) – references INTERMEDIARY\_COURSE

An aside may be that if it were possible, addition of a TEXTBOOKID attribute may avoid confusion between textbooks with the same name, where it is possible, they are not exactly the same textbook, or are different editions of the same textbook.

**Task 2: SQL Statements**

**FAO of marker: SQL statements have been verified using pgAdmin, and as such had included some syntax (e.g. “public.”TABLE”.”ATTRIBUTE”) in proven solutions. This has been edited out where possible, but remained in others to avoid introducing errors.**

**Task 2.1: STUDENT**

***TABLE STUDENT:***

CREATE TABLE STUDENT

(

SID integer NOT NULL,

SNAME character varying(20),

HCID character varying(2),

CONSTRAINT "PKSTUDENT" PRIMARY KEY (SID),

CONSTRAINT "FKSTUDENTHCID" FOREIGN KEY (HCID)

REFERENCES HOMECITY (HCID)

ON UPDATE CASCADE

ON DELETE CASCADE

)

If a HOMECITY is deleted or updated, we want those changes to be reflected in the child of the relationship.

***TABLE HOMECITY:***

CREATE TABLE HOMECITY

(

HCID character varying(2) NOT NULL,

HCNAME character varying(20),

CONSTRAINT "PKHOMECITY" PRIMARY KEY (HCID)

)

***TABLE STUDENT\_JOINYEAR:***

CREATE TABLE STUDENT\_JOINYEAR

(

SID integer NOT NULL,

JYEAR integer NOT NULL,

TID integer NOT NULL,

CONSTRAINT "PKJOINYEAR" PRIMARY KEY (SID, JYEAR),

CONSTRAINT "FKSID" FOREIGN KEY (SID)

REFERENCES STUDENT (SID)

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "FKTID" FOREIGN KEY (TID)

REFERENCES TEAM (TID)

ON UPDATE CASCADE

ON DELETE CASCADE

)

Where a STUDENTID or TEAMID is deleted or updated, it is desirable to update affected tuples with the child.

***TABLE TEAM:***

CREATE TABLE TEAM

(

TID integer NOT NULL,

TNAME character varying(7),

CONSTRAINT "PKTEAM" PRIMARY KEY (TID)

)

**TASK 2.1: TOPIC**

***TABLE ADVISEE\_TOPIC:***

CREATE TABLE ADVISEE\_TOPIC

(

SID integer NOT NULL,

TOPIC character varying(20) NOT NULL,

ADVID integer NOT NULL,

CONSTRAINT "PKADVISEE\_TOPIC" PRIMARY KEY (SID, TOPIC),

CONSTRAINT "FKADVID" FOREIGN KEY (ADVID)

REFERENCES ADVISOR (ADVID)

ON UPDATE CASCADE

ON DELETE CASCADE

)

As the referencing of ADVID from ADVISOR provides part of the Primary Key in this table, updating and deleting via the parent is required for non-erroneous tuples in ADVISEE\_TOPIC.

***TABLE ADVISOR:***

CREATE TABLE ADVISOR

(

ADVID integer NOT NULL,

ADVNAME character varying(20),

CONSTRAINT "PKADVISOR" PRIMARY KEY (ADVID)

)

**Task 2.2:**

**SQL1:** SELECT COUNT(public."STUDENT"."SID"), public."HOMECITY"."HCID"

FROM public."STUDENT", public."HOMECITY"

WHERE public."HOMECITY"."HCID" = public."STUDENT"."HCID"

GROUP BY public."HOMECITY"."HCID"

**SQL2:** SELECT COUNT(public."STUDENT"."SID"), public."TEAM"."TNAME"

FROM public."STUDENT", public."STUDENT\_JOINYEAR", public."TEAM"

WHERE public."STUDENT"."SID" = public."STUDENT\_JOINYEAR"."SID"

AND public."TEAM"."TID" = public."STUDENT\_JOINYEAR"."TID"

AND public."STUDENT\_JOINYEAR"."JYEAR" = 2001

GROUP BY public."TEAM"."TNAME"

**SQL3:** SELECT public."ADVISOR"."ADVNAME"

FROM public."ADVISOR", public."ADVISEE\_TOPIC"

WHERE public."ADVISOR"."ADVID" = public."ADVISEE\_TOPIC"."ADVID"

GROUP BY public."ADVISOR"."ADVNAME"

HAVING COUNT(DISTINCT public."ADVISEE\_TOPIC"."SID") > 10

**SQL4:** SELECT

public."ADVISEE\_TOPIC"."TOPIC", COUNT(DISTINCT public."ADVISOR"."ADVID")

FROM public."ADVISOR", public."ADVISEE\_TOPIC"

WHERE public."ADVISOR"."ADVID" = public."ADVISEE\_TOPIC"."ADVID"

GROUP BY public."ADVISEE\_TOPIC"."TOPIC"

LIMIT 1

**SQL5:** SELECT public."STUDENT"."SNAME", COUNT(\*)

FROM public."STUDENT", public."STUDENT\_JOINYEAR", public."TEAM"

WHERE public."STUDENT"."SID" = public."STUDENT\_JOINYEAR"."SID"

AND public."TEAM"."TID" = public."STUDENT\_JOINYEAR"."TID"

AND public."STUDENT\_JOINYEAR"."JYEAR" >= 2001

GROUP BY public."STUDENT"."SNAME"

HAVING COUNT (\*) =

(SELECT MAX(TEAMCOUNT) FROM

(SELECT COUNT(DISTINCT public."STUDENT\_JOINYEAR"."TID") AS TEAMCOUNT FROM "STUDENT\_JOINYEAR") AS SJY)

***TASK 3:***

***3.1:***

SELECT "EMPLOYEE"."FName"

FROM "EMPLOYEE"

GROUP BY "EMPLOYEE"."FName"

HAVING COUNT("EMPLOYEE"."SSN") > 1

***3.2:***

SELECT COUNT("EMPLOYEE"."FName")

FROM "EMPLOYEE"

GROUP BY "EMPLOYEE"."FName"

HAVING COUNT("EMPLOYEE"."SSN") > 1

***3.3:***

SELECT Salary

FROM (

SELECT Salary,

row\_number() over (order by Salary desc) as sorted

FROM EMPLOYEE

) t

WHERE sorted <= 3