**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 (1:1).

FD2: HCID > HCNAME

The students home city ID defines home city name.

Therefore, HCNAME is defined by SID via transitive dependency.

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.

Therefore SID, JYEAR > TNAME via transitive dependency.

***TABLE: TOPIC***

Each TOPIC a student has their own advisor; the same student can have multiple advisors depending on their topics, therefore…

FD1: SID, TOPIC > ADVID

ADVID is provided by a combination of SID and TOPIC. An advisor may only advise one TOPIC.

FD2: ADVID > TOPIC

As an advisor may only advise one TOPIC this relationship also holds true.

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 TEXTBOOK.

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***: We have FD4 which 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 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 Team ID as a foreign key in normalisation. During creation of the database I would recommend not allowing NULL values to both attributes or creating an edge case catch for those without teams (and therefore no join year).

***In TABLE: TOPIC***: In topic we are told that an advisor can only cover one topic, so due to the database design, when inserting a new student with an advisor already present in the table, (see ADVID 1 (McReader)), we must duplicate information in this tables topic attribute. This means that if an Advisor were to change Topic, we would have to update many tuples.

***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. This is assuming that insertion would be possible within the context, given that each course has a set number of advisors.

In the same way, a new textbook for either Database or Advanced Database would require new tuples for each advisor 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

**3NF:**

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 or transitive dependencies on any non-prime attribute within all tables.

***TABLE: TOPIC***

TOPIC(SID, ADVNAME, ADVID, TOPIC)

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

3N?F:

Split Topic into two relations, Advisee and Advisor.

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.

ADVISOR(ADVID, ADVNAME)

ADVID uniquely identifies ADVNAME, which would be partially dependant on primary key before split (FD3).

**Task 1.4:**

Due to the nature of the relationship 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)

Therefore, I suggest a split of the tables as follows.

COURSE\_TEXTBOOK(COURSE, TEXTBOOK)

COURSE\_ADVISOR(ADVID, COURSE(FK))

The reasoning is that in this split, we create two tables with 2 attributes as the primary key.

Tables can be joined via courses, reducing the number of unnecessary tuples resulting from each individual textbook being recorded for each advisor within a course. This also allows us to deal with the issue arising when noting that the same textbook could be used in multiple courses, where COURSE\_TEXTBOOK(COURSE, TEXTBOOK) accounts for this. COURSE\_ADVISOR(ADVID, COURSE(FK)) accounts for the potential issues which may arise where the same advisor belongs to multiple courses.

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, however would not reduce generated tuples.

**Task 2: SQL Statements**

**Task 2.1: STUDENT**

***TABLE STUDENT:***

CREATE TABLE public."STUDENT"

(

"SID" integer NOT NULL,

"SNAME" character varying(20),

"HCID" character varying(2),

CONSTRAINT "PKSTUDENT" PRIMARY KEY ("SID"),

CONSTRAINT "FKSTUDENTHCID" FOREIGN KEY ("HCID")

REFERENCES public."HOMECITY" ("HCID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

***TABLE HOMECITY:***

CREATE TABLE public."HOMECITY"

(

"HCID" character varying(2) NOT NULL,

"HCNAME" character varying(30),

CONSTRAINT "PKHOMECITY" PRIMARY KEY ("HCID")

)

***TABLE STUDENT\_JOINYEAR:***

CREATE TABLE public."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 public."STUDENT" ("SID")

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT "FKTID" FOREIGN KEY ("TID")

REFERENCES public."TEAM" ("TID")

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

***TABLE TEAM:***

CREATE TABLE public."TEAM"

(

"TID" integer NOT NULL,

"TNAME" character varying(7),

CONSTRAINT "PKTEAM" PRIMARY KEY ("TID")

)

**TASK 2.1: TOPIC**

***TABLE ADVISEE\_TOPIC:***

CREATE TABLE public."ADVISEE\_TOPIC"

(

"SID" integer NOT NULL,

"TOPIC" character varying NOT NULL,

"ADVID" integer NOT NULL,

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

CONSTRAINT "FKADVID" FOREIGN KEY ("ADVID")

REFERENCES public."ADVISOR" ("ADVID")

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

***TABLE ADVISOR:***

CREATE TABLE public."ADVISOR"

(

"ADVID" integer NOT NULL,

"ADVNAME" character varying,

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 \*

FROM EMPLOYEE

GROUP BY EMPLOYEE.SSN

HAVING COUNT(\*) > 1

***3.2:*** SELECT COUNT(\*)

FROM EMPLOYEE

GROUP BY EMPLOYEE.SSN

HAVING COUNT(\*) > 1

***3.3:*** SELECT SALARY

FROM EMPLOYEE AS t

GROUP BY 1

HAVING COUNT(\*) <

(SELECT COUNT(z.SALARY)

FROM (SELECT DISTINCT SALARY

FROM EMPLOYEE) AS z

WHERE t.SALARY>=z.SALARY)

ORDER BY 1 desc