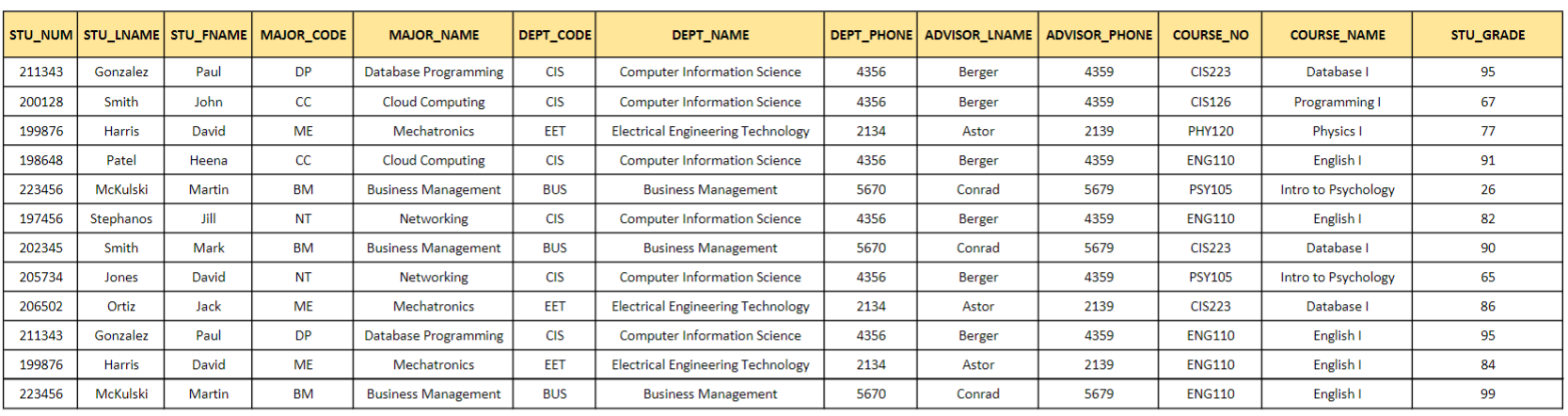
**JEREMY PETERSON DUTOYA**

**DR. MOHAMMAD AWWAL**

**PMBA-8317-RHB APPLIED DATA MANAGEMENT FOR BUSINESS USERS**

*Assignment - A3*

1. Write the relational schema and draw its dependency diagram. Identify all dependencies, including all transitive dependencies.

Here’s the Student un-normalized table structure. 

* Relational Schema is as follow:

For this relational schema, given that we didn’t have many details on the primary key of this unnormalized table, I assumed that the PK was a composite key including Stu\_Num, Dept\_Code and Course\_No. Given this assumption, I was able to further investigate the dependencies in the table.

The Partial Dependencies are represented by blue arrows, the transitive dependencies by orange.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STU\_ NUM** | STU\_ LNAME | STU\_ FNAME | MAJOR\_  CODE | MAJOR\_  NAME | **DEPT\_**  **CODE** | DEPT\_  NAME | DEPT\_  PHONE | ADV\_  LNAME | ADV\_  PHONE | **COURSE\_**  **NO** | COURSE\_  NAME | STU\_  GRADE |

***Partial***

***Dependency***

***Transitive***

***Dependency***

***Partial***

***Dependency***

***Partial***

***Dependency***

***Transitive***

***Dependency***

1. Write the relational schema and draw the dependency diagram to meet the 3NF requirements to the greatest practical extent possible.

* *If you believe that practical considerations dictate using a 2NF structure, explain why your decision to retain 2NF is appropriate.*
* *If necessary, add or modify attributes to create appropriate determinants and to adhere to the naming conventions.*

 If we want to meet the 3NF requirements, the Relational Schema would probably be close to the following:

**STUDENT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STU\_NUM** | STU\_LNAME | STU\_FNAME | MAJOR\_CODE | ADVISOR\_ID |

**DEPARTMENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| **DEPT\_CODE** | DEPT\_NAME | DEPT\_PHONE | ADVISOR\_ID |

**COURSE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE\_NO** | COURSE\_NAME | DEPT\_CODE | COURSE\_CREDIT |

**STUDENT\_ENROLLMENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE\_NO** | **STU\_NUM** | **DEPT\_CODE** | STU\_GRADE |

**MAJOR:**

|  |  |  |
| --- | --- | --- |
| **MAJOR\_CODE** | MAJOR\_NAME | DEPT\_CODE |

**ADVISOR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADVISOR\_ID** | ADVISOR\_LNAME | ADVISOR\_FNAME | ADVISOR\_PHONE |

Ultimately, as we are respecting the 3NF, there is no Partial or Transitive dependencies.

Moreover, I believe there is a way to represent this table using another Relational Schema that could be practical, with less entities and then with less JOINS. If we take the example of many universities: a Major is only taught in one Department of the university so we can change this to combine these two entities, we will have transitive dependencies so we will be in a 2NF. Thanks to this change we can remove the Advisor\_Id from the STUDENT Table as it is possible to retrieve this information from the MAJOR Table. :

**STUDENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STU\_NUM** | STU\_LNAME | STU\_FNAME | MAJOR\_CODE |

**MAJOR:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MAJOR\_CODE** | MAJOR\_NAME | DEPT\_CODE | DEPT\_NAME | DEPT\_PHONE | ADVISOR\_ID |

***Transitive***

***Dependency***

**COURSE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE\_NO** | COURSE\_NAME | DEPT\_CODE | COURSE\_CREDIT |

**STUDENT\_ENROLLMENT:**

|  |  |  |  |
| --- | --- | --- | --- |
| **COURSE\_NO** | **STU\_NUM** | **DEPT\_CODE** | STU\_GRADE |

**ADVISOR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADVISOR\_ID** | ADVISOR\_LNAME | ADVISOR\_FNAME | ADVISOR\_PHONE |

1. Perform logical design from 3NF dependency diagram- define entities, attributes, primary keys, foreign keys, and draw the ERD using Crow’s Foot notation.

The second part of my last answer doesn’t really count as we are solely focusing on representing the 3NF in a ERD using the Crow’s Foot notation:

**Parent/Child**

**Parent/Child**

**Junction Table**

|  |
| --- |
| **STUDENT** |
| **STU\_NUM**  **PK** |
| STU\_LNAME |
| STU\_FNAME |
| MAJOR\_CODE  **FK** |
| ADVISOR\_ID |

|  |
| --- |
| **STUDENT\_ENROLLMENT** |
| **COURSE\_NO**  **FK**  **PK** |
| **STU\_NUM**  **FK**  **PK** |
| **DEPT\_CODE**  **FK**  **PK** |
| STU\_GRADE |

|  |
| --- |
| **COURSE** |
| **COURSE\_NO**  **PK** |
| COURSE\_NAME |
| COURSE\_CREDIT |
| DEPT\_CODE |

Has

Has

**FK**

**FK**

Offers

Has

**Parent**

|  |
| --- |
| **DEPARTMENT** |
| **DEPT\_CODE**  **PK** |
| DEPT\_NAME |
| DEPT\_PHONE |
| ADVISOR\_ID  **FK** |

Advisor

**Child**

**Parent**

|  |
| --- |
| **MAJOR** |
| **MAJOR\_CODE**  **PK** |
| MAJOR\_NAME |
| DEPT\_CODE |

**FK**

Hires

Offers

|  |
| --- |
| **ADVISOR** |
| **ADVISOR\_ID** |
| ADVISOR\_LNAME  **PK** |
| ADVISOR\_PHONE |

1. Perform physical design- define data types, add/modify columns as appropriate, create DDL to create tables, primary and foreign keys. Add check constraints, not null constraints, default values as appropriate.

Use Oracle supported datatypes.

My choice of RDBMS is Oracle:

**STUDENT**

STU\_NUM INT (10) **NOT NULL UNIQUE (PK)**

STU\_LNAME varchar (30)

STU\_FNAME varchar (30)

MAJOR\_CODE char(3) FK references-> MAJOR (MAJOR\_CODE)

ADVISOR\_ID varchar(30) FK references -> ADVISOR (ADVISOR\_ID)

**DEPARTMENT**

DEPT\_CODE CHAR(3) **NOT NULL UNIQUE(PK)**

DEPT\_NAME VARCHAR (30)

DEPT\_PHONE INT (11) CHECK >0

ADVISOR\_LNAME VARCHAR (30) FK references -> ADVISOR (ADVISOR\_LNAME)

**COURSE**

COURSE\_NO CHAR(6) **NOT NULL UNIQUE(PK)**

COURSE\_NAME VARCHAR (30)

DEPT\_CODE CHAR (3) CHECK >0 FK references -> DEPARTMENT(DEPT\_CODE)

**STUDENT\_ENROLLMENT**

COURSE\_NO CHAR(6) **NOT NULL (PK)** FK references -> COURSE(COURSE\_NO)

STU\_NUM INT (10) **NOT NULL (PK)** FK references -> STUDENT(STU\_NUM)

DEPT\_CODE CHAR (3) **NOT NULL (PK)** FK references -> DEPARTMENT(DEPT\_CODE)

STU\_GRADE INT(3) CHECK >0

**MAJOR**

MAJOR\_CODE CHAR(3) **NOT NULL UNIQUE (PK**)

MAJOR\_NAME VARCHAR(30) NOT NULL

DEPT\_CODE CHAR(3) NOT NULL FK references -> DEPARTMENT (DEPT\_CODE)

**ADVISOR**

ADVISOR\_ID INT (10) **NOT NULL UNIQUE (PK)**

ADVISOR\_LNAME VARCHAR (30)

ADVISOR\_PHONE INT (11) ) CHECK >0

**SQL Statements to create tables**

CREATE TABLESTUDENT (

STU\_NUM INT(10) NOT NULL,

STU\_LNAME VARCHAR(30),

STU\_FNAME VARCHAR(30),

MAJOR\_CODE CHAR(3) ,

ADVISOR\_ID INT(10),

CONSTRAINT STUDENT\_PK PRIMARY KEY(STU\_NUM),

CONSTRAINT STUDENT\_FK1 FOREIGN KEY (MAJOR\_CODE) REFERENCES MAJOR(MAJOR\_CODE),

CONSTRAINT STUDENT\_FK2 FOREIGN KEY (ADVISOR\_ID) REFERENCES ADVISOR(ADVISOR\_ID)

);

CREATE TABLE DEPARTMENT (

DEPT\_CODE CHAR(3) NOT NULL,

DEPT\_NAME VARCHAR (30),

DEPT\_PHONE INT (11),

ADVISOR\_LNAME VARCHAR (30),

CONSTRAINT DEPT\_PK PRIMARY KEY (DEPT\_CODE),

CONSTRAINT DEPT\_CK CHECK (DEPT\_PHONE>0),

CONSTRAINT DEPT\_FK (ADVISOR\_ID) REFERENCES ADVISOR(ADVISOR\_ID)

);

CREATE TABLECOURSE (

COURSE\_NO CHAR(6) NOT NULL,

COURSE\_NAME VARCHAR(30),

DEPT\_CODE CHAR(3)

CONSTRAINT COURSE\_PK PRIMARY KEY(COURSE\_NUM),

CONSTRAINT COURSE\_FK FOREIGN KEY(DEPT\_CODE) REFERENCES DEPARTMENT(DEPT\_CODE)

);

CREATE TABLEMAJOR (

MAJOR\_CODE CHAR(3) NOT NULL,

MAJOR\_NAME VARCHAR(30),

DEPT\_CODE CHAR (3)

CONSTRAINT MAJOR\_PK PRIMARY KEY(MAJOR\_CODE),

CONSTRAINT MAJOR\_FK FOREIGN KEY(DEPT\_CODE) REFERENCES DEPARTMENT (DEPT\_CODE)

);

CREATE TABLEADVISOR (

ADVISOR\_ID INT(10) NOT NULL,

ADVISOR\_LNAME VARCHAR(30),

ADVISOR\_FNAME VARCHAR(30)

ADVISOR\_PHONE INT(11)

CONSTRAINT ADVISOR\_PK PRIMARY KEY(ADVISOR\_ID),

CONSTRAINT ADVISOR\_CK CHECK (ADVISOR\_PHONE>0),

);

CREATE TABLESTUDENT\_ENROLLMENT (

COURSE\_NO CHAR(6) NOT NULL,

STU\_NUM INT (10) NOT NULL,

DEPT\_CODE CHAR(3),

STU\_GRADE INT(3),

CONSTRAINT SE\_PK PRIMARY KEY(COURSE\_NO,STU\_NUM,DEPT\_CODE),

CONSTRAINT SE\_FK1 FOREIGN KEY (COURSE\_NO) REFERENCES COURSE (COURSE\_NO),

CONSTRAINT SE\_FK2 FOREIGN KEY (STU\_NUM) REFERENCES STUDENT (STU\_NUM)

CONSTRAINT SE\_FK1 FOREIGN KEY (DEPT\_CODE) REFERENCES DEPARTMENT (DEPT\_CODE),

CONSTRAINT SE\_CK CHECK (STU\_GRADE>0),

);