Hull University

Database Design Report

Maaz Selia

Database Techniques

3rd May, 2019

Introduction

This ACW aims to assess an understanding of Database design. This report will attempt to solve and answer the ACW's given exercises. The report will preserve the order of questions as in the ACW.

For the first exercise, it will present the prototype list with a justification to its choice of primary key. This will then be followed by an Entity relationship diagram which shows the relationships between the various entities and the foreign keys within.

This will then be followed by the 2nd exercise, which includes a diagram showing dependencies and candidate keys with an explanation to the chosen Primary key. This is then proceeded by a normalisation of the given data set. In a series of SQL statements complimented by explanation and a breakdown of tables at each step.

Finally, the report will analyse a given scenario to derive a list of entities, attributes, and relationships involved. This will then be consolidated into and Entity relationship diagram to represent the list identified previously.

Understanding an existing database structure

Cap (NAME, LANGU)

In this table, both columns contain repeating values. Therefore, the primary key must be a composition of both columns for each tuple to have a unique identifier.

Course (COURSE_NAME, COURSE_NUMBER, CREDIT_HOURS, OFFERING_DEPT)

In this table, COURSE_NUMBER was chosen as the primary key instead of COURSE_NAME because of the small chance that two different courses may have the same name and as COURSE_NUMBER is a natural key.

Department_to_major (Dcode, DNAME)

Dcode was chosen as it is a natural key.

Dependant (PNO, DNAME, RELATIONSHIP, SEX, AGE)

A composite key was chosen for this table, as one person can have multiple dependants which would make PNO unusable as a primary key. However, a composition of PNO and DNAME would make each dependant uniquely identifiable.

Grade_Report (STUDENT_NUMBER, SECTION_ID, GRADE)

A composite key of STUDENT_NUMBER and SECTION_ID was chosen as a student can have multiple grades and a section can have multiple students. However, a particular student can only have one grade for a particular section.

Prereq (COURSE_NUMBER, PREREQ)

COURSE_NUMBER was chosen as it is a natural key and PREREQ is functionally dependant on COURSE_NUMBER.

Room (BLDG, ROOM, CAPACITY, OHEAD)

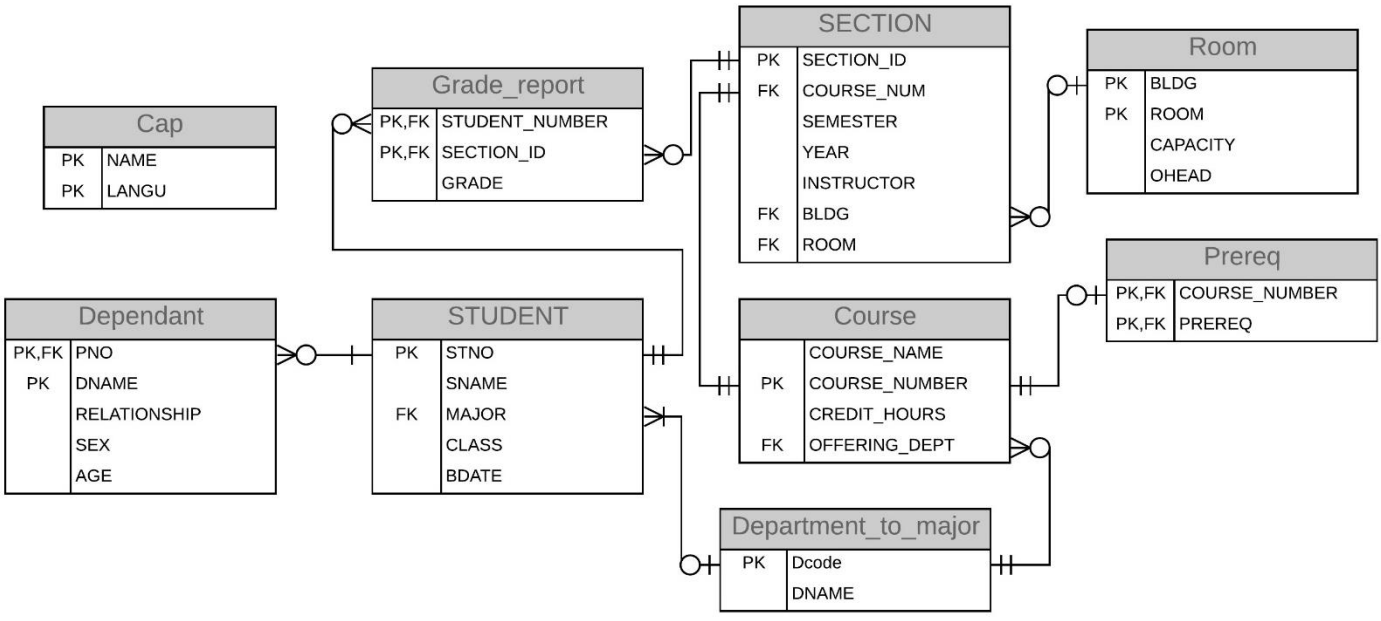
A composite key was chosen since a BLDG can have multiple rooms and the ROOM attribute contains repeating values. Hence, a composition of BLDG and ROOM.

Section (SECTION_ID, COURSE_ID, SEMESTER, YEAR, INSTRUCTOR, BLDG, ROOM)

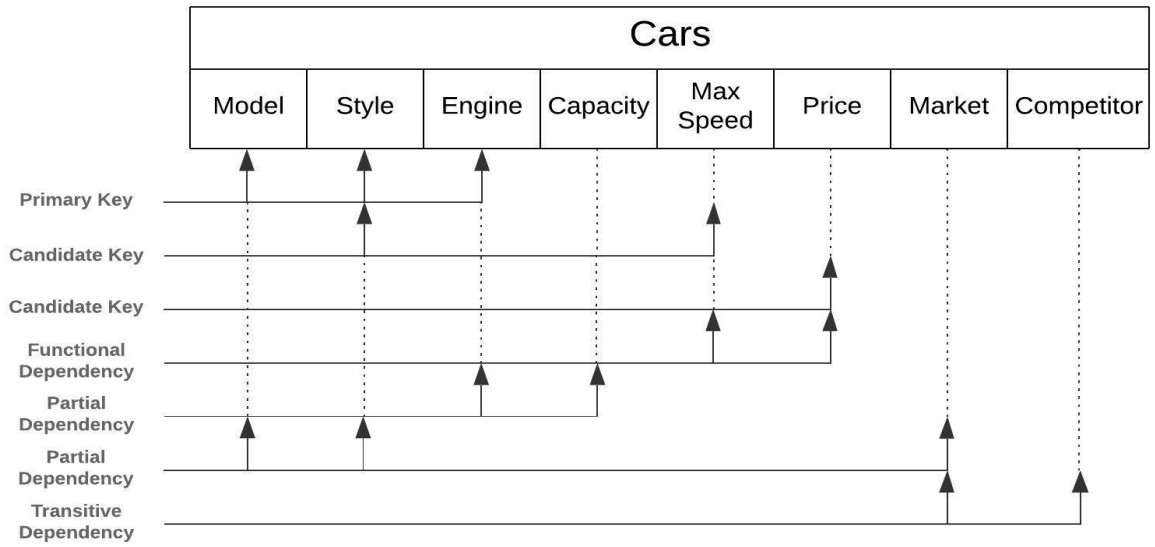
SECION_ID was chosen as it is a natural key. Although COURSE_ID is also a candidate key, it was not chosen since the relation concerns "Section".

Student (STNO, SNAME, MAJOR, CLASS, BDATE)

STNO was chosen as it is a natural key.



Normalising a bulk data set



The primary key for this was the composition of Model, Style, and Engine. This was chosen of the three candidate keys as it was the most meaningful. Engine + Max Speed, though unique, would not hold much meaning when searching through the table. This is also the case with using Price as a primary key.

Normalisation

To normalise the data, we need to start with normalising to 1NF. The form, however, is already in a state of 1NF. As such, we can move onto normalising to 2NF. 2NF requires that the form should be in 1NF and no partial dependencies present.

To achieve this the Market and Capacity columns must be removed from the table and a separate table created for them. As they are partially dependant on the primary key. Also, to note, the column Competitor is dependent on the Market column. And since the Market column is to be removed, we will also need to create another table to store the Competitor column.

Removing Engine-Capacity partial dependency

```
SELECT DISTINCT Cars.ENGINE, Cars.CAPACITY INTO EngineCapacity FROM Cars
GO
ALTER TABLE EngineCapacity ALTER COLUMN ENGINE NVARCHAR(20) NOT NULL
GO
ALTER table EngineCapacity add primary key (ENGINE)
```

Removing Model+Style-Market partial dependency

```
SELECT DISTINCT Cars.MODEL, Cars.STYLE, Cars.MARKET INTO ModelStyleMarket FROM Cars
GO
ALTER TABLE ModelStyleMarket ALTER COLUMN MODEL NVARCHAR(20) NOT NULL
ALTER TABLE ModelStyleMarket ALTER COLUMN STYLE NVARCHAR(20) NOT NULL
GO
```

```
ALTER table ModelStyleMarket add primary key (MODEL,STYLE)
```

Accounting for removal of Market column from main table

```
SELECT DISTINCT Cars.Market, Cars.COMPETITOR INTO MarketCompetitor FROM Cars
GO
ALTER TABLE MarketCompetitor ALTER COLUMN Market NVARCHAR(20) NOT NULL
GO
ALTER table MarketCompetitor add primary key (MARKET)
```

Following these queries, the database is now in 2NF. As all partial dependencies have been removed.

Continuing, to make the database into 3NF. The database should be in 2NF and there should be no transitive dependency present. Since the only transitive dependency, (Competitor column), has already been removed. We have already reached 3NF.

Finally, to reach BCNF. It is required that the database be in 3NF and for A -> B, A must be a superkey. The only remaining non-prime columns in the main table are Max Speed and Price. These are determined by all three values of the primary key. Therefore, the main table satisfies BCNF. And of the remaining three tables. Two contain only two attributes, thus, satisfying BCNF. The remaining table, ModelStyleMarket, contains a composite key of Model and Style with a non-prime attribute Market. In this case, neither of the two prime attributes can be derived from the non-prime attribute. Therefore, this table also satisfies BCNF.

The database has been normalised to BCNF standard.

Constructing BCNF standard main table

```
SELECT Cars.MODEL, Cars.STYLE, Cars.ENGINE, Cars.MAX_SPEED, Cars.PRICE INTO NewCars
FROM Cars
GO
ALTER TABLE NewCars ALTER COLUMN MODEL NVARCHAR(20) NOT NULL
ALTER TABLE NewCars ALTER COLUMN STYLE NVARCHAR(20) NOT NULL
ALTER TABLE NewCars ALTER COLUMN ENGINE NVARCHAR(20) NOT NULL
GO
ALTER table NewCars add primary key (MODEL,STYLE,ENGINE)
```

Developing a potential entity model

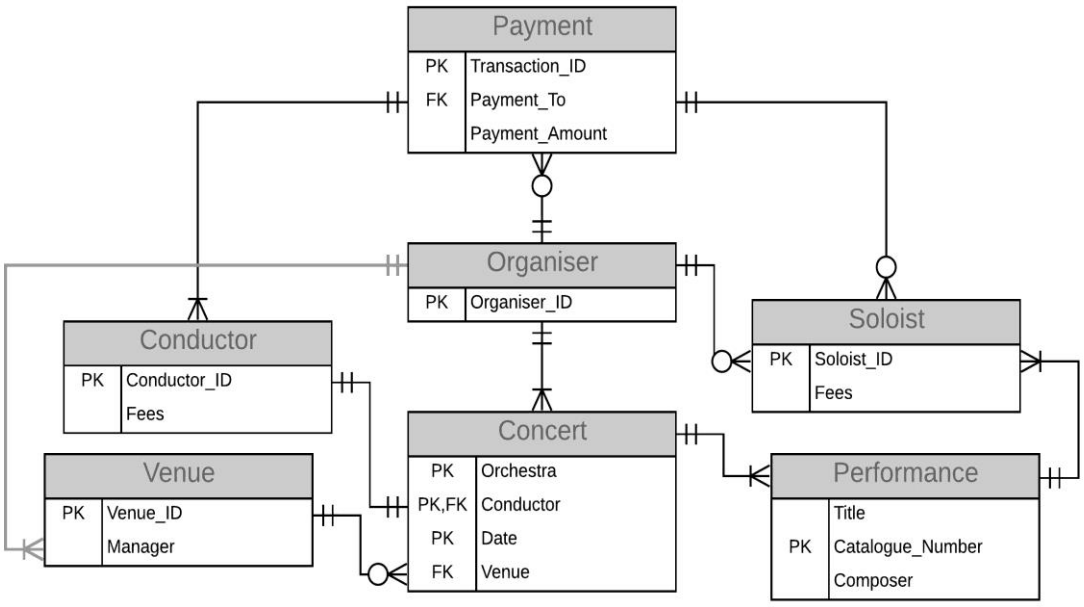
See Appendix for evidence of derivation of Entities, attributes, and relationships.

Derived Entities, attributes, and relationships

Entities	Attributes
Organiser	
Concert	Orchestra, Conductor, Date, Venue
Performance	Title, Catalogue number, Composer
Venue	Manager
Soloist	Fees
Conductor	Fees

Relationships

Organiser	Schedules	Many Concerts
Concert	Given by	1 Orchestra, 1 Conductor
Concert	Has	Many Performances
Performance	Requires	1 or more Soloist



End of Report.

Appendix

Process of derivation of Entities, attributes, and relationships.

Entity Attribute Relationship Cardinality

"A concert season organiser schedules a number of musical performances, on various dates at various venues. Each concert is given by a particular orchestra and conductor, but not always the same ones throughout the season. A number of different works may be performed in the programme at one concert, each having a specific title, catalogue number and composer, and possibly requiring one or more soloists to play. The organiser has to book each venue through its manager, arrange the services of any soloists needed, and pay performance fees to the conductor and any soloists involved."