**Lab 1**

Some of the most difficult decisions that you face as a database developer are what tables to create and what columns to place in each table, as well as how to relate the tables that you create. Normalization is the process of applying a series of rules to ensure that your database achieves optimal structure. Normal forms are a progression of these rules. Each successive normal form achieves a better database design than the previous form did. Although we discussed several levels of normal forms, this lab focuses on 1st Normal Form (1NF), and Boyce-Codd Normal Form (BCNF). If you do not understand functional dependencies, then review the discussion on functional dependencies on the slides and your notes.

***Exercise 1: 1st Normal Form (1NF)***

Consider the Students table, with the primary key underlined, and the following data:

Students:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alpha** | **Name** | **Email** | **Courses** | **GradePoints** |
| 100111 | John Doe | doe@usna.edu | NN204, SI204, IT221 | 2,3,3 |
| 092244 | Matt Smith | smith@usna.edu | SM223, EE301 | 4,4 |
| 113221 | Melinda Black | black@usna.edu | SI204 | 3 |
| 090112 | Tom Johnson | Johnson@usna.edu | NN204, SI204, IT221 | 4,2,3 |

1. Is the Students table in 1NF? Why?

No, in the table "Courses" and "GradePoints" columns contain multiple values ​​separated by commas, which means they are not atomic values.

1. If the Students table is not in 1NF, redesign the tables such that all the information currently in the Students table is found in the resulting tables, and the resulting tables are in 1NF. For each of the resulting tables, give the table name, column names, primary keys, and foreign keys.

#### **Table 1: Students**

Students

|  |  |  |
| --- | --- | --- |
| **Alpha** | **Name** | **Email** |
| 100111 | John Doe | doe@usna.edu |
| 092244 | Matt Smith | smith@usna.edu |
| 113221 | Melinda Black | black@usna.edu |
| 090112 | Tom Johnson | Johnson@usna.edu |

**PK:** Alpha

#### **Table 2: Curses**

#### Curses

|  |  |  |
| --- | --- | --- |
| **Alpha** | **Course** | **GradePoint** |
| 100111 | NN204 | 2 |
| 100111 | SI204 | 3 |
| 100111 | IT221 | 3 |
| 092244 | SM223 | 4 |
| 092244 | EE301 | 4 |
| 113221 | SI204 | 3 |
| 090112 | NN204 | 4 |
| 090112 | SI204 | 2 |
| 090112 | IT221 | 3 |

**PK:** Course

**FK:** Alpha (references to Students table)

***Exercise 2: Boyce-Codd Normal Form (BCNF)***

For a table to be in Boyce-Codd normal form, the table must be in 1NF and the determinants of all the functional dependencies in that table must be candidate keys (either primary key or alternate key). Below is the Rentals table created for the DVD-by-mail division of Neatflix.

Rentals:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RentalID** | **Title** | **CustomerID** | **MailedOutDate** | **Director** | **MovieCategory** | **Price** |
| 1 | Die Hard | 1001 | 3/3/2010 | John McTiernan | Old | $4.25 |
| 1 | The last man standing | 1001 | 3/3/2010 | Walter Hill | Old | $4.25 |
| 1 | Wedding Crashers | 1001 | 3/3/2010 | David Dobkin | New | $5.50 |
| 2 | Dodgeball | 1002 | 3/4/2010 | Rawson Marshall Thurber | New | $5.50 |
| 2 | Die Hard | 1002 | 3/4/2010 | John McTiernan | Old | $4.25 |
| 3 | As good as it gets | 1003 | 1/7/2011 | James Brooks | Old | $4.25 |
| 4 | Forest Gump | 1001 | 1/7/2011 | Robert Zemeckis | Old | $4.25 |

The primary key of the Rentals table is the composite key (**RentalID, Title)**.

1. Explain the conditions under which the following functional dependency is true:

RentalID -->CustomerID

For the RentalID --> CustomerID functional dependency to be true, each unique RentalID value must be associated with a single CustomerID value. This means that each RentalID uniquely identifies a CustomerID.

1. Based on the sample data on the table, is the functional dependency

RentalID --> CustomerID true?

Yes, based on the example data in the table, the RentalID --> CustomerID functional dependency is true. Each RentalID has only one CustomerID associated with it:

* RentalID 1 is associated with CustomerID 1001.
* RentalID 2 is associated with CustomerID 1002.
* RentalID 3 is associated with CustomerID 1003.
* RentalID 4 is associated with CustomerID 1001.

1. Explain the conditions under which the following functional dependency is true:

Director --> Title

For the Director-> Title functional dependency to be true, each director must have directed only one film title. This means that each Director uniquely identifies a Title.

1. Based on the sample data on the table, is the functional dependency

Director --> Title true?

No, based on the example data in the table, the Director-> Title functional dependency is not true. The same director may have directed multiple titles:

* John McTiernan has directed "Die Hard".
* Walter Hill has directed "The Last Man Standing."
* David Dobkin has directed "Wedding Crashers".
* Rawson Marshall Thurber has directed "Dodgeball."
* James Brooks has directed "As good as it gets".
* Robert Zemeckis has directed "Forest Gump".

1. Based on your general knowledge of movies and rentals, is the functional dependency Director --> Title true?

No, based on general knowledge of movies and rentals, the functional dependence Director--> Title is not true. A director may have directed multiple movie titles. For example, John McTiernan has also directed "Predator" in addition to "Die Hard."

1. Write a functional dependency that expresses the fact that the cost of all movies in a given category is the same.

This means that for each movie category, there is a fixed price associated with it.

1. We discussed ***insertion anomalies,*** ***deletion anomalies and update anomalies*** as examples of problems that can appear in tables that are not normalized. The following is an example of an insertion anomaly in the Rentals table: if we want to create a new category of movies, “Must See”, there is no way to store the price of this type of movie in the database, until someone rents a movie in this category, and the rental information is recorded into the Rentals table. Give one example of a deletion anomaly in the Rentals table.

An example of a deletion anomaly in the Rentals table is that if we remove all rows containing the movie "Die Hard", we also remove all information about its director and price from the "Old" category. This could result in the loss of valuable information.

1. State what you believe are reasonable functional dependencies for the Rentals table for a DVD-by-mail business (include the functional dependencies from points a) to f) that you believe are/should be true).

* RentalID --> CustomerID
* MovieCategory --> Price

1. Given your answer above, decompose the Rentals table such that the resulting tables are in BCNF. For each of the resulting tables, give the table name, column names, primary keys, and foreign keys.

#### **Table 1: Alquileres**

Alquileres

|  |  |  |
| --- | --- | --- |
| **RentalID** | **CustomerID** | **MailedOutDate** |
| 1 | 1001 | 3/3/2010 |
| 2 | 1002 | 3/4/2010 |
| 3 | 1003 | 1/7/2011 |
| 4 | 1001 | 1/7/2011 |

**Primary Key:** RentalID

#### **Table 2: Películas**

Películas

|  |  |  |
| --- | --- | --- |
| **Title** | **Director** | **MovieCategory** |
| Die Hard | John McTiernan | Old |
| The last man standing | Walter Hill | Old |
| Wedding Crashers | David Dobkin | New |
| Dodgeball | Rawson Marshall Thurber | New |
| As good as it gets | James Brooks | Old |
| Forest Gump | Robert Zemeckis | Old |

**Primary Key:** Title

#### **Table 3: AlquilerDetalles**

AlquilerDetalles

|  |  |  |
| --- | --- | --- |
| **RentalID** | **Title** | **Price** |
| 1 | Die Hard | $4.25 |
| 1 | The last man standing | $4.25 |
| 1 | Wedding Crashers | $5.50 |
| 2 | Dodgeball | $5.50 |
| 2 | Die Hard | $4.25 |
| 3 | As good as it gets | $4.25 |
| 4 | Forest Gump | $4.25 |

**Primary Key:** (RentalID, Title)

**Foraign Key:** RentalID (references to Alquileres table), Title (references to Películas table)

***Exercise 3: SQL***

Given the following tables:

ITEM(ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmt, ExchangeRate)

SHIPMENT\_ITEM(ShipmentID, ShipmentItemNb, *ItemID*, Value)

Write the SQL query to find the ItemID and Description for the item with the lowest shipped Value.

SELECT I.ItemID, I.Description

FROM ITEM I

JOIN SHIPMENT\_ITEM SI ON I.ItemID = SI.ItemID

ORDER BY SI.Value ASC

LIMIT 1;

***Exercise 4 (Extra credit): 4th Normal Form (4NF)***

For information on multivalued dependencies and 4NF, review the slides.

Suppose we have the following Courses table with columns CourseID, Instructor, Book that stores the courses, the instructor teaching the course, and the recommended books for the course. The book(s) recommended for a course does not depend on the teacher teaching the course, just on the course. Here is an example of instantiation for this table:

Courses:

|  |  |  |
| --- | --- | --- |
| **CourseID** | **Instructor** | **Book** |
| IT360 | Crainiceanu | Kroenke |
| IT360 | Crainiceanu | Welling |
| IT360 | DeLooze | Kroenke |
| IT360 | DeLooze | Welling |
| SI440 | Crainiceanu | Kroenke |
| SI440 | Crainiceanu | Ramakrishnan |
| SI440 | Crainiceanu | Stonebraker |

a) Give an example of a multivalued dependency in the Courses table.

A multivalued dependency (MVD) occurs when one attribute in a table uniquely determines another attribute, independently of other attributes. In the Courses table, the Book recommended for a CourseID does not depend on the Instructor teaching the course. This means there is a multivalued dependency between CourseID and Book.

Multivalued Dependency:

* CourseID →→ Book

This means for a given CourseID, the Book values are independent of the Instructor.

b) Is the Courses table in 4NF? If answer to yes, say why. If not, decompose the table such that the resulting tables are in 4th normal form. For each of the resulting tables, give the table name, column names, primary keys, and foreign keys.

The Courses table is **not** in 4NF. Fourth Normal Form (4NF) is violated when a table has a multivalued dependency and non-trivial multivalued dependencies (other than a candidate key) exist in the table.

To bring the Courses table into 4NF, we need to decompose it into separate tables to eliminate the multivalued dependency.

#### **Decomposition**

**CourseInstructor Table**:

* **Columns**: CourseID, Instructor
* **Primary Key**: (CourseID, Instructor)

**CourseBook Table:**

* Columns: CourseID, Book
* Primary Key: (CourseID, Book)

By decomposing the original Courses table into CourseInstructor and CourseBook, we achieve 4NF because each table now has no non-trivial multivalued dependencies other than a candidate key.

#### **Example Data in Decomposed Tables**

**CourseInstructor:**

|  |  |
| --- | --- |
| **CourseID** | **Instructor** |
| IT360 | Crainiceanu |
| IT360 | DeLooze |
| SI440 | Crainiceanu |

**CourseBook:**

|  |  |
| --- | --- |
| **CourseID** | **Book** |
| IT360 | Kroenke |
| IT360 | Welling |
| SI440 | Kroenke |
| SI440 | Ramakrishnan |
| SI440 | Stonebraker |

In this structure:

* CourseInstructor captures the relationship between courses and instructors.
* CourseBook captures the relationship between courses and recommended books.

This decomposition eliminates the multivalued dependency and ensures that the tables are in 4NF.