**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 because there is more than one value per column.**

The **"Courses"** column contains multiple values in a single field (e.g., "NN204, SI204, IT221").

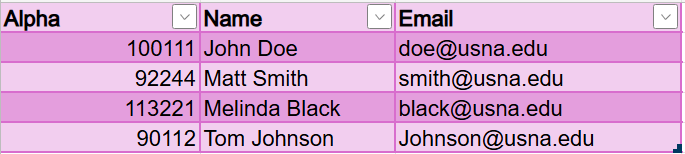
The **"GradePoints"** column also contains multiple values in a single field (e.g., "2,3,3").

To bring this table into 1NF, we would need to separate these multi-valued fields into individual rows

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.

**Students Table** >

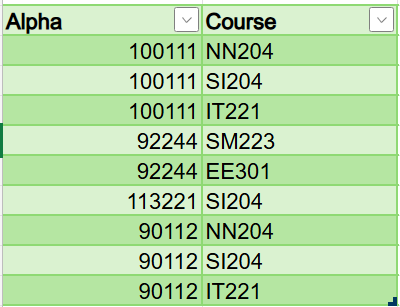
Primary Key: Alpha



**Student Courses Table** >

Primary Key > Composite Key: (Alpha, Course)

Foreign Key > Alpha references “Students.Alpha”



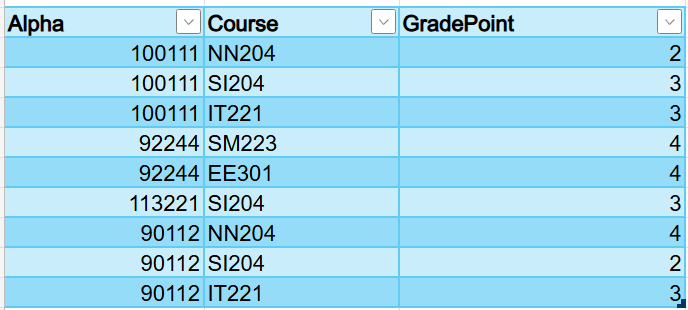
**Student Grades Table** >

Primary Key > Composite Key: (Alpha, Course)

Foreign Keys >

Alpha references “Students.Alpha”

Course references “StudentCourses.Course”



* Students Table contains the basic information about students.
* StudentCourses Table records which courses each student is taking.
* StudentGrades Table captures the grades for each course that the students are enrolled in

**Summary:**

* **Students Table**:
  + **PK:** Alpha
* **StudentCourses Table**:
  + **PK:** (Alpha, Course)
  + **FK:** Alpha references Students.Alpha
* **StudentGrades Table**:
  + **PK:** (Alpha, Course)
  + **FK:** Alpha references Students.Alpha
  + **FK:** Course references StudentCourses.Course

***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

*The dependency indicates that the CustomerID is uniquely determined by the RentalID.*

For the dependency RentalID → CustomerID to be true, each RentalID must uniquely identify one customer. This would mean that the RentalID is a unique identifier for a rental transaction that is associated with one and only one CustomerID. If a RentalID can be linked to multiple customers, then this functional dependency would not hold true.

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

RentalID --> CustomerID true?

*The functional dependency is NOT true.*

RentalID alone does not uniquely identify a single CustomerID. While in the given data, RentalID 1 and RentalID 2 are consistently linked to a specific CustomerID, there’s nothing preventing a different dataset from associating the same RentalID with a different CustomerID.

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

Director --> Title

The functional dependency Director → Title indicates that the Title of a movie is uniquely determined by the Director. This would mean that each director is associated with only one movie title in the context of the table.

The functional dependency Director → Title is true only if each director has directed exactly one movie in the dataset. If a director has directed multiple movies, the dependency would not hold.

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

Director --> Title true?

*The functional dependency Director → Title is* ***not*** *true.*

Since **John McTiernan** (and potentially other directors) is associated with more than one movie title, the dependency Director → Title is **not true** based on the sample data in the table.

1. Based on your general knowledge of movies and rentals, is the functional dependency

Director --> Title true?

Given that most directors have directed multiple movies, the dependency Director → Title is generally not true in the real world. A more accurate functional dependency might be Title → Director, which will only be true if each movie has only one director.

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

**Movie Category → Price**

For example, if "Old" movies are always $4.25 and "New" movies are always $5.50, then knowing the MovieCategory would be sufficient to determine the Price of any movie in that category.

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.

Suppose that the movie **"Die Hard"** (RentalID = 1) is the last movie in the **"Old"** category that a customer has rented. The price for all movies in the "Old" category is $4.25.

If the **"Die Hard"** rental record is deleted from the Rentals table (e.g., because the customer returned the movie and the rental history is being cleared), then not only is the information about that specific rental lost, but also the **price associated with the "Old" category** is lost.

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).

Functional dependencies describe the relationships between different columns such that one or more columns (the determinant) uniquely determine another column (the dependent).

**Reasonable Functional Dependencies:**

RentalID -> Title, CustomerID, MailedOutDate, Director, MovieCategory, Price  
Each rental has a unique RentalID, which determines all the other attributes of that rental.

Title -> Director, MovieCategory, Price  
Each movie title determines its director, movie category, and price. This assumes that the price is fixed for each title regardless of the customer.

CustomerID, MailedOutDate -> RentalID  
A combination of CustomerID and MailedOutDate should uniquely determine the RentalID since a customer can't rent more than one DVD at the exact same time.

Analysis from the Table:

1. **RentalID determines CustomerID**  
   This means each rental ID is linked to a specific customer. This is generally true since RentalID is unique.
2. **CustomerID does not determine RentalID**One customer can have multiple rentals, so CustomerID alone does not uniquely determine RentalID.
3. **RentalID determines Title**  
   Each rental has a unique RentalID, so this also uniquely determines the Title.
4. **Title does not determine RentalID**  
   A title can be rented multiple times, so Title alone does not uniquely determine RentalID.
5. **RentalID determines MailedOutDate**  
   Each rental instance has a unique RentalID, which determines the date it was mailed out.
6. **MailedOutDate does not determine RentalID**  
   Multiple rentals can be mailed out on the same date, so MailedOutDate does not uniquely determine RentalID.

**Combining These Functional Dependencies >**

RentalID -> Title, CustomerID, MailedOutDate, Director, MovieCategory, Price

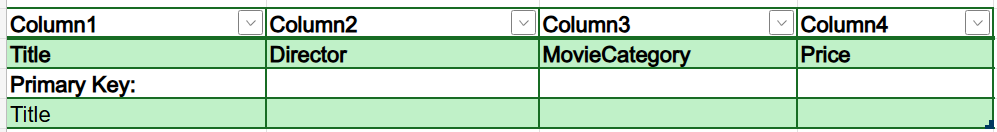
Title -> Director, MovieCategory, Price

CustomerID, MailedOutDate -> RentalID

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.

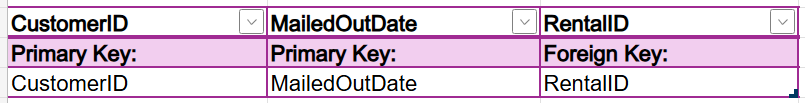
**Movies Table**

Stores information specific to each movie.



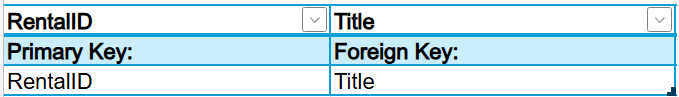
**Rentals Tables**

Links customers with the dates they rented movies, along with the RentalID.



**Rental Details Table**

Stores specific rental transactions, linking RentalID to movie titles.

****

***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

WHERE si.Value = (

SELECT MIN(Value)

FROM SHIPMENT\_ITEM

);

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

***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.

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.