(Optional) Hands-on Lab: Normalization, Keys and Constraints in Relational Databases

Estimated time needed: 25 minutes

In this lab, you will learn about normalization, keys, and constraints in IBM Db2 on Cloud using SQL. First, you will learn how to minimize data redundancy and inconsistency in a database by normalizing tables. Next, you will learn how to use keys to uniquely identify a record in a table, to establish a relationship between tables, and to identify the relation between them. Lastly, you will learn about different kinds of relational model constraints that help to maintain data integrity in a relational data model.

Software used in this lab

In this lab, you will use an IBM Db2 Database. Db2 is a relational database management system (RDBMS) from IBM, designed to efficiently store, analyze, and retrieve data.

Data set used in this lab

In this lab, you will use a **BookShop** data set.

Objectives

After completing this lab, you will be able to:

- Minimize data redundancy and inconsistency in a database by using normalization.
- Use keys to uniquely identify a record in a table, establish a relationship between tables, and identify the relation between them.
- Maintain data integrity in a relational data model using constraints.

Instructions

When you approach the exercises (specially from Exercise 1 Task B) in this lab, follow the instructions to run the queries on Db2:

- Go to the Resource List of IBM Cloud by logging in where you can find the Db2 service instance that you created in a previous lab under **Services** section.
- Click on the **Db2-xx service**.
- Next, open the Db2 Console by clicking **Open Console**.
- Click on the 3-bar menu icon in the top left corner and go to the **Run SQL** page. The Run SQL tool enables you to run SQL statements.

Exercise 1: Normalization

In this exercise, you will learn about first normal form (1NF) and implement second normal form (2NF).

Task A: First normal form (1NF)

In this task of normalization, you will be working with the **BookShop** table. The following image shows the **BookShop** table:

BOOK_ID	TITLE	AUTHOR_NA
B101	Introduction to Algorithms	Thomas H. C
B201	Structure and Interpretation of Computer Pro...	Harold Abels
B301	Deep Learning	Ian Goodfell
B401	Algorithms Unlocked	Thomas H. C
B501	Machine Learning: A Probabilistic Perspective	Kevin P. Murp

You will answer some questions to determine if the table above is in 1NF.

1. Does the above table have unique rows?

▶ Hint
▶ Answer

2. Does each cell of the above table have single/atomic value?

▶ Hint
▶ Answer

3. By definition, a table is in 1NF if every attribute in that relation contains single valued data and every tuple in that relation is unique. Does the above table fall in first normal form?

▶ Hint
▶ Answer

4. If your answer to question 3 is No, how can you normalize the table to ensure first normal form?

▶ Hint
▶ Answer

Task B: Second normal form (2NF)

1. Starting from this task of normalization, this lab requires you to have a version of **BookShop** table (shown below) different from the one shown in Exercise 1 Task A, at Db2 on cloud. Download the `BookShop-CREATE-INSERT.sql` script below, upload it to the Db2 console, and run it. The script will drop any previous **BookShop** table that exists, create the new **BookShop** table, and populate it with the sample data required for this lab.

BOOK_ID	TITLE	AUTHOR_
B101	Introduction to Algorithms	Thomas H
B201	Structure and Interpretation of Computer Pro...	Harold Ab
B301	Deep Learning	Ian Goodf
B401	Algorithms Unlocked	Thomas H
B501	Machine Learning: A Probabilistic Perspective	Kevin P. M

- BookShop-CREATE-INSERT.sql

▶ Click here to view the queries inside the script

```
**Tip**: If you are unsure how to upload and run the script, review the following lab to learn how to perform the tasks:
```

Hands-on Lab : Create Tables and Load Data in Db2

2. By definition, a relation is in second normal form if it is already in 1NF and does not contain any partial dependencies. If you look at the BookShop table, you will find every column in the table is single or atomic valued, but it has multiple books by the same author. This means that the AUTHOR_ID, AUTHOR_NAME and AUTHOR_BIO details for BOOK_ID B101 and B401 are the same. As the number of rows in the table increase, you will be needlessly storing more and more occurrences of these same pieces of information. And if an author updates their bio, you must update all of these occurrences.

BOOK_ID	TITLE	AUTHOR_
B101	Introduction to Algorithms	Thomas H
B201	Structure and Interpretation of Computer Pro...	Harold Ab
B301	Deep Learning	Ian Goodf
B401	Algorithms Unlocked	Thomas H
B501	Machine Learning: A Probabilistic Perspective	Kevin P. M

3. In this scenario, to enforce 2NF you can take the author information such as AUTHOR_ID, AUTHOR_NAME and AUTHOR_BIO out of the **BookShop** table into another table, for example a table named **BookShop_AuthorDetails**. You then link each book in the BookShop table to the relevant row in the **BookShop_AuthorDetails** table, using a unique common column such as AUTHOR_ID to link the tables. To create the new **BookShop_AuthorDetails** table, copy the code below and paste it to the textbox of the **Run SQL** page after adding a new blank script. Click **Run all**.

```
CREATE TABLE BookShop_AuthorDetails
AS
(SELECT DISTINCT AUTHOR_ID, AUTHOR_NAME, AUTHOR_BIO FROM BookShop)
WITH DATA;
SELECT * FROM BookShop_AuthorDetails;
```

```
CREATE TABLE BookShop_AuthorDetails
AS
(SELECT DISTINCT AUTHOR_ID, AUTHOR_NAME, AUTHOR_BIO FROM
WITH DATA;

SELECT * FROM BookShop_AuthorDetails;
```

4. Now you can drop the redundant author information related columns from the **BookShop** table. Do not drop the AUTHOR_ID column though, because that will be used to maintain the relationship between the two tables. To drop the redundant author information related columns from the **BookShop** table, copy the code below and paste it to the current script textbox of the **Run SQL** page replacing the existing code with this new code. Click **Run all**.

```
ALTER TABLE BookShop
DROP COLUMN AUTHOR_NAME
DROP COLUMN AUTHOR_BIO;
SELECT * FROM BookShop;
```

RUN SQL

lab-10 * BookShop.sql * Untitled - 2

```
1  ALTER TABLE BookShop
2  DROP COLUMN AUTHOR_NAME
3  DROP COLUMN AUTHOR_BIO;
4
5  SELECT * FROM BookShop;
6
```

5. Now you are only storing the author information once per author and only have to update it in one place; reducing redundancy and increasing consistency of data. Thus 2NF is ensured.

```
SELECT * FROM BookShop
```

Result set 1

BOOK_ID	TITLE	AUTHOR_NAME	AUT
B101	Introduction to Algorithms	Thomas H. Cormen	Thor
B201	Structure and Interpretatio...	Harold Abelson	Harc
B301	Deep Learning	Ian Goodfellow	Ian J
B401	Algorithms Unlocked	Thomas H. Cormen	Thor
B501	Machine Learning: A Probab...	Kevin P. Murphy	

```
SELECT * FROM BookShop_AuthorDetails
```

Result set 1

AUTHOR_ID	AUTHOR_NAME
123	Thomas H. Cormen
157	Kevin P. Murphy
369	Ian Goodfellow
456	Harold Abelson

Exercise 2: Keys

In this exercise, you will see how to use a primary key to uniquely identify a record in a table, how to use a foreign key to establish a relationship between tables, and how to identify the relation between them.

Task A: Primary Key

1. By definition, a primary key is a column or group of columns that uniquely identify every row in a table. A table cannot have more than one primary key. The rules for defining a primary key are:
    - No two rows can have a duplicate primary key value.
    - Every row must have a primary key value.
    - No primary key field can be null.

2. You will create a primary key for the **BookShop** and **BookShop_AuthorDetails** tables to uniquely identify every row in each of the tables. You will set the BOOK_ID column of the **BookShop** table and AUTHOR_ID column of the **BookShop_AuthorDetails **table as a primary key for each of the tables. Both the columns were declared as NOT NULL when the tables were created (Check the sql script or table definition of the tables to verify the NOT NULL constraint. Because the **BookShop_AuthorDetails** table was created from the **BookShop** table, it inherits all the data types and column constraints like NOT NULL from the **BookShop** parent table).

Table Definition

BOOKSHOP

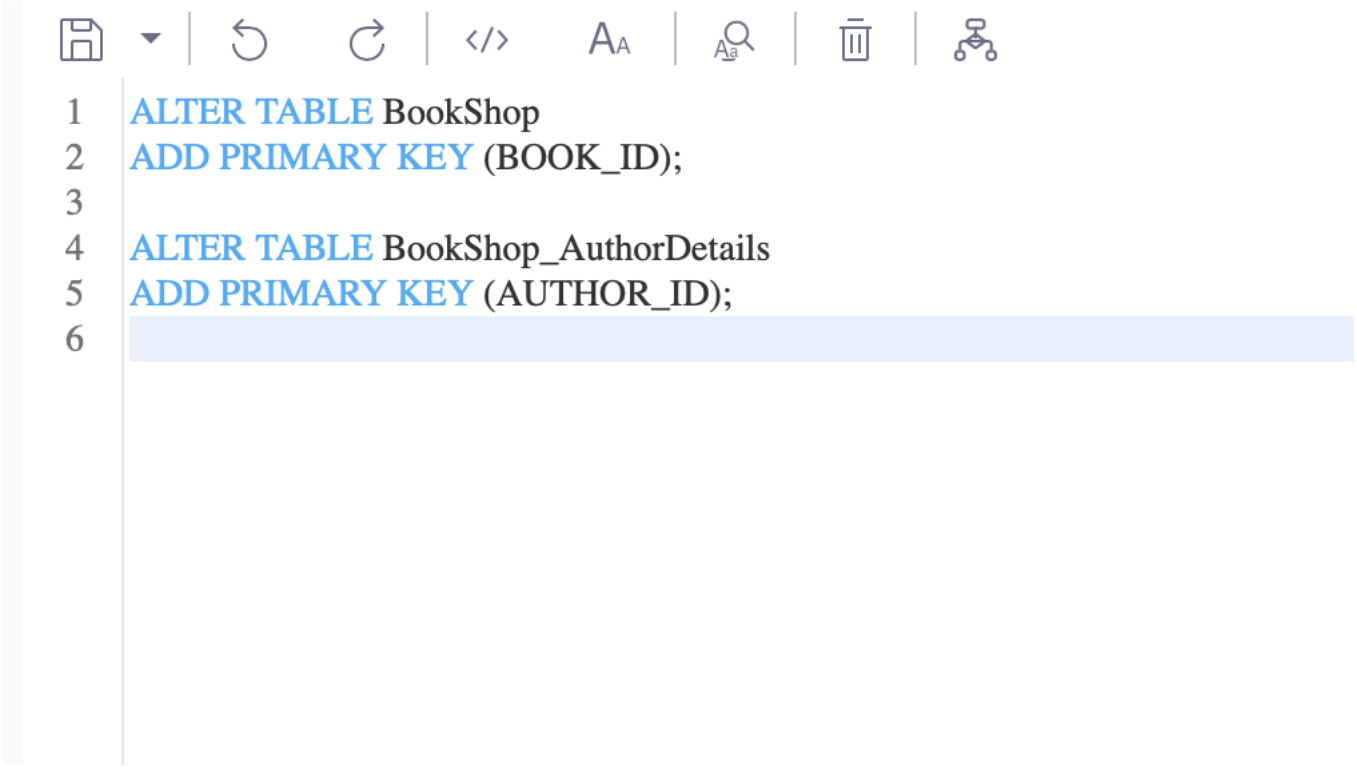
COLUMN NAME	DATA TYPE	NULLABLE	L
BOOK_ID	VARCHAR	N	
TITLE	VARCHAR	N	
AUTHOR_NAME	VARCHAR	N	
AUTHOR_BIO	VARCHAR	Y	
AUTHOR_ID	INTEGER	N	
PUBLICATION_DATE	DATE	N	
PRICE_USD	DECIMAL	N	

3. The ALTER TABLE statement you ran in the previous task puts the table into what's called a 'reorg pending' state. This occurs after any ALTER TABLE statement that contains a REORG-recommended operation is run. Therefore, to put the table back into an 'organized' state, you need to call a **Sysproc.admin** command that reorganizes the table before you can continue. Copy the code below and paste it to the current script textbox of the **RUN SQL** page, replacing the existing code with this new code. Then click **Run all**.

```
Call Sysproc.admin_cmd ('reorg Table BookShop');
```

4. To set the BOOK_ID column of the **BookShop** table and AUTHOR_ID column of the **BookShop_AuthorDetails** table as a primary key for each of the tables, copy the code below and paste it to the current script textbox of the **RUN SQL** page replacing the existing code with this new code. Click **Run all**.

```
ALTER TABLE BookShop
ADD PRIMARY KEY (BOOK_ID);
ALTER TABLE BookShop_AuthorDetails
ADD PRIMARY KEY (AUTHOR_ID);
```



5. Now you can use the BOOK_ID column to uniquely identify every row in the **BookShop** table and the AUTHOR_ID column to uniquely identify every row in the **BookShop_AuthorDetails** table.

Task B: Foreign Key

1. By definition, a foreign key is a column that establishes a relationship between two tables. It acts as a cross-reference between two tables because it points to the primary key of another table. A table can have multiple foreign keys referencing primary keys of other tables. Rules for defining a foreign key:

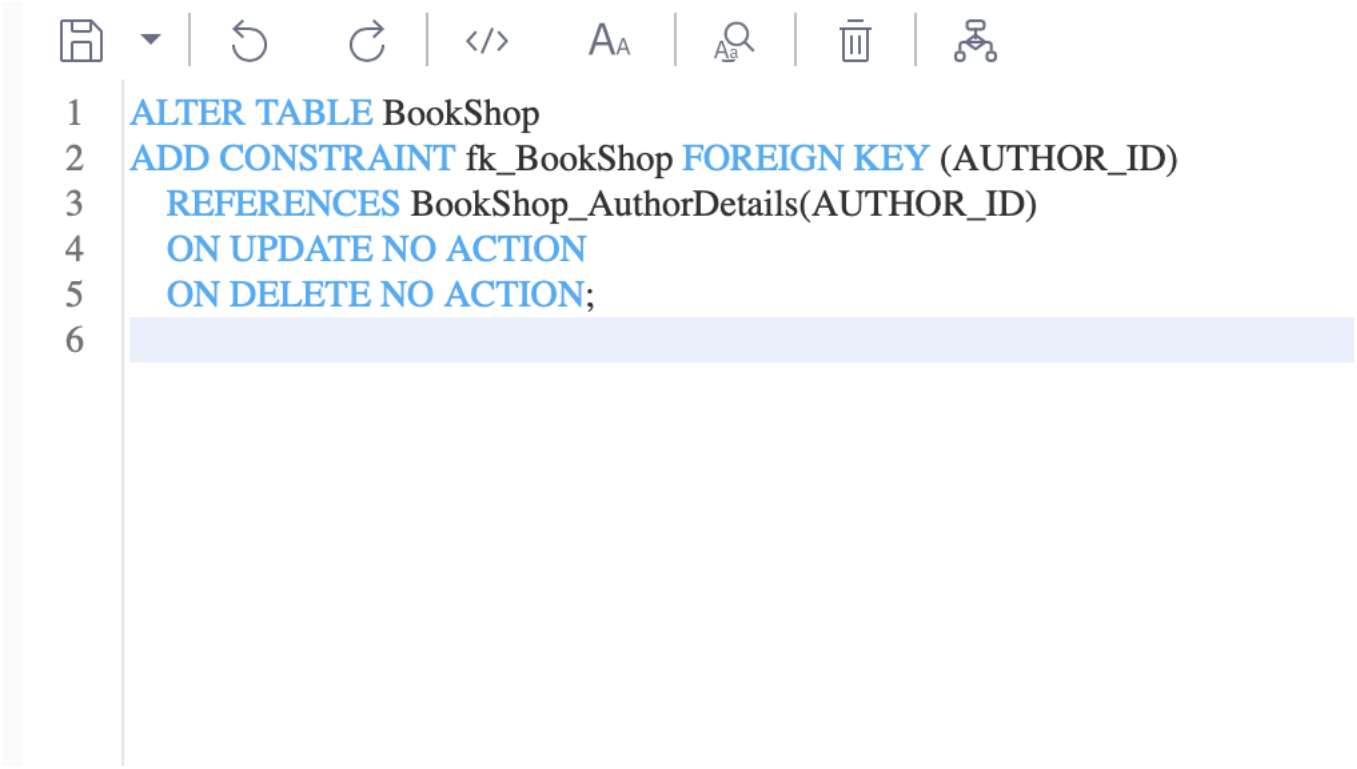
    - A foreign key in the referencing table must match the structure and data type of the existing primary key in the referenced table.
    - A foreign key can only have values present in the referenced primary key.
    - Foreign keys do not need to be unique. Most often, they aren't.
    - Foreign keys can be null.

2. You will create a foreign key for the **BookShop** table by setting its AUTHOR_ID column as a foreign key, to establish a relationship between the **BookShop** and **BookShop_AuthorDetails** tables. Copy the code below and paste it to the current script textbox of the **RUN SQL** page replacing the existing code with this new code. Click **Run all**.

```
ALTER TABLE BookShop
ADD CONSTRAINT fk_BookShop FOREIGN KEY (AUTHOR_ID)
    REFERENCES BookShop_AuthorDetails(AUTHOR_ID)
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;
```

Note: ON UPDATE NO ACTION means if any existing row is updated in the foreign key column of the referencing table (the table containing the foreign key), the update will only be allowed if the new value of the foreign key column exists in the referenced primary key column of the referenced table (the table containing the primary key). However, any update on a row of the referenced primary key column of the referenced table is always rejected if there is the existence of a corresponding row in the referencing foreign key column of the referencing table.

ON DELETE NO ACTION means if any row in the referenced table (the table containing the primary key) is deleted, that row in the referenced table and the corresponding row in the referencing table (the table containing the foreign key) are not deleted.



```
1  ALTER TABLE BookShop
2  ADD CONSTRAINT fk_BookShop FOREIGN KEY (AUTHOR_ID)
3     REFERENCES BookShop_AuthorDetails(AUTHOR_ID)
4     ON UPDATE NO ACTION
5     ON DELETE NO ACTION;
6
```

3. Now that you have created the relationship, each book in the BookShop table is linked to the relevant row in the BookShop_AuthorDetails table through AUTHOR_ID.

Exercise 3: Constraints

In this exercise, you will review different kinds of relational model constraints that help to maintain data integrity in a relational data model.

1. **Entity Integrity Constraint**: Entity integrity ensures that no duplicate records exist within a table and that the column identifying each record within the table is not a duplicate and not null. The existence of a primary key in both the **BookShop** and **BookShop_AuthorDetails** tables satisfies this integrity constraint because a primary key mandates NOT NULL constraint as well as ensuring that every row in the table has a value that uniquely denotes the row.

2. **Referential Integrity Constraint**: Referential integrity ensures the existence of a referenced value if a value of one column of a table references a value of another column. The existence of the foreign Key (AUTHOR_ID) in the **BookShop** table satisfies this integrity constraint because a cross-reference relationship between the **BookShop** and **BookShop_AuthorDetails** tables exists. As a result of this relationship, each book in the **BookShop** table will be linked to the relevant row in the **BookShop_AuthorDetails** table through the AUTHOR_ID columns.

3. **Domain Integrity Constraint**: Domain integrity ensures that the purpose of a column is clear and the values of a column are consistent as well as valid. The existence of data types, length, date format, check, and null constraints in the CREATE statement of the **BookShop** table makes sure this integrity constraint is satisfied.

▼ Click here to view queries inside the script

```
-- Drop the table in case it exists

DROP TABLE BookShop;

-- Create the table

CREATE TABLE BookShop (
    BOOK_ID VARCHAR(4) NOT NULL,
    TITLE VARCHAR(100) NOT NULL,
    AUTHOR_NAME VARCHAR(30) NOT NULL,
    AUTHOR_BIO VARCHAR(250),
    AUTHOR_ID INTEGER NOT NULL,
    PUBLICATION_DATE DATE NOT NULL,
    PRICE_USD DECIMAL(6,2) CHECK(Price_USD>0) NOT NULL
    );
```

Conclusion

Congratulations! You have completed this lab, and you have learned to use normalization, use keys to uniquely identify a record in a table, establish a relationship between tables, and identify the relation between them. In addition, you know how to maintain data integrity in a relational data model using constraints.

Author: Sandip Saha Joy

Skills Network