Let's create a sample PostgreSQL database schema that demonstrates various types of constraints (such as primary keys, foreign keys, unique constraints, check constraints, and not null constraints) using a business case scenario.

**Business Case Scenario**

**Scenario**: We are managing a simple library system. The system has two main tables: books and authors. Each book is written by an author, and we want to ensure data integrity in our database through various constraints.

**Step 1: Create the authors Table**

This table will store information about authors, including a unique identifier and name.

sql

Copy code

CREATE TABLE authors (

author\_id SERIAL PRIMARY KEY, -- Primary key with auto-increment

name VARCHAR(100) NOT NULL UNIQUE, -- Unique constraint to ensure no duplicate names

birthdate DATE CHECK (birthdate <= CURRENT\_DATE) -- Check constraint to ensure birthdate is not in the future

);

**Explanation of Constraints**

* **Primary Key**: author\_id is defined as the primary key, which uniquely identifies each author.
* **Not Null Constraint**: The name column cannot be null, meaning every author must have a name.
* **Unique Constraint**: The name column has a unique constraint, preventing duplicate author names.
* **Check Constraint**: The birthdate must be a valid date and cannot be in the future.

**Step 2: Create the books Table**

This table will store information about books, including a foreign key that references the authors.

sql

Copy code

CREATE TABLE books (

book\_id SERIAL PRIMARY KEY, -- Primary key for the books table

title VARCHAR(200) NOT NULL, -- Title of the book, must not be null

author\_id INT, -- Foreign key to link to authors

publication\_date DATE NOT NULL, -- Publication date of the book, must not be null

isbn VARCHAR(13) UNIQUE, -- Unique constraint for the ISBN number

pages INT CHECK (pages > 0), -- Check constraint to ensure pages are greater than 0

FOREIGN KEY (author\_id) REFERENCES authors(author\_id) -- Foreign key constraint

);

**Explanation of Constraints**

* **Primary Key**: book\_id serves as the primary key, uniquely identifying each book.
* **Not Null Constraint**: The title and publication\_date columns cannot be null, ensuring that every book has a title and a publication date.
* **Foreign Key Constraint**: The author\_id column in the books table references the author\_id in the authors table. This creates a relationship between books and their authors, ensuring that a book can only reference an existing author.
* **Unique Constraint**: The isbn column is defined as unique, ensuring no two books can have the same ISBN.
* **Check Constraint**: The pages column must be greater than 0.

**Step 3: Insert Sample Data**

Now, let's insert some sample data into the authors and books tables.

sql

Copy code

-- Inserting authors

INSERT INTO authors (name, birthdate) VALUES

('George Orwell', '1903-06-25'),

('J.K. Rowling', '1965-07-31'),

('Harper Lee', '1926-04-28');

-- Inserting books

INSERT INTO books (title, author\_id, publication\_date, isbn, pages) VALUES

('1984', 1, '1949-06-08', '978-0451524935', 328),

('Harry Potter and the Philosopher\'s Stone', 2, '1997-06-26', '978-0747532699', 223),

('To Kill a Mockingbird', 3, '1960-07-11', '978-0061120084', 281);

**Step 4: Constraints in Action**

Now, let’s see how the constraints work by trying to perform various operations:

1. **Try to Insert a Duplicate Author Name**:

sql

Copy code

INSERT INTO authors (name, birthdate) VALUES ('George Orwell', '1903-06-25');

This will fail due to the unique constraint on the name column.

1. **Try to Insert a Book with a Non-existent Author**:

sql

Copy code

INSERT INTO books (title, author\_id, publication\_date, isbn, pages) VALUES

('Some Book', 99, '2023-01-01', '978-1234567890', 150);

This will fail due to the foreign key constraint, as there is no author with author\_id = 99.

1. **Try to Insert a Book with an Invalid Page Count**:

sql

Copy code

INSERT INTO books (title, author\_id, publication\_date, isbn, pages) VALUES

('Another Book', 1, '2023-01-01', '978-1234567891', -50);

This will fail due to the check constraint on the pages column, which requires it to be greater than 0.

**Summary**

In this example:

* We created two tables: authors and books.
* We applied various constraints to ensure data integrity:
  + **Primary Key**: Uniquely identifies records.
  + **Not Null**: Ensures essential data is provided.
  + **Unique**: Prevents duplicate entries.
  + **Check**: Validates data according to specified rules.
  + **Foreign Key**: Maintains referential integrity between tables.

**Final DDL Summary**

Here is the complete DDL for reference:

sql

Copy code

CREATE TABLE authors (

author\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL UNIQUE,

birthdate DATE CHECK (birthdate <= CURRENT\_DATE)

);

CREATE TABLE books (

book\_id SERIAL PRIMARY KEY,

title VARCHAR(200) NOT NULL,

author\_id INT,

publication\_date DATE NOT NULL,

isbn VARCHAR(13) UNIQUE,

pages INT CHECK (pages > 0),

FOREIGN KEY (author\_id) REFERENCES authors(author\_id)

);

**Business Scenario Question**

**Question**: How can we ensure that each book has a valid author and that no two authors can have the same name? What happens if we attempt to add a book with a non-existent author?

This question can lead to a discussion about the importance of constraints in maintaining data integrity and the role of foreign keys in relationships between tables.