


UNIT 3: ADVANCED CONCEPTS OF DATA AND DATABASES

INSERT, UPDATE & DELETE

LAGUARDIA COMMUNITY COLLEGE

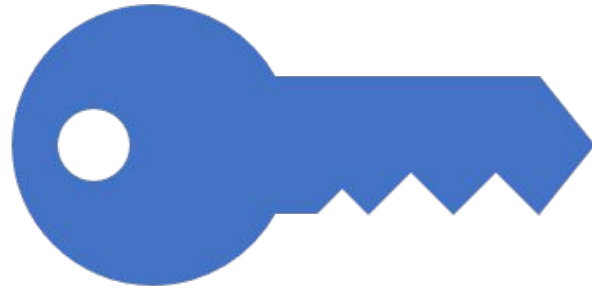
A decorative graphic on the left side of the slide, consisting of two parallel, wavy, light blue lines that create a sense of movement and depth against the dark blue background.

Fun Fact: The concept of the computer database was introduced way before the digital age we live in today. One of the earliest known forms of a computerized database was developed in 1964 for the Apollo program to catalog the vast amounts of information necessary for the moon landing. This endeavor required a novel approach to data storage and retrieval, marking a significant milestone in the evolution of data management technologies. The success of managing such a complex dataset paved the way for the development of relational databases and SQL, revolutionizing how data is stored, accessed, and manipulated across myriad applications worldwide.

PRIMARY AND FOREIGN KEYS

PRIMARY AND FOREIGN KEYS

- A **primary key**, also called a **primary** keyword, is a **key** in a relational database that is unique for each record.
- A primary key is a column or a group of columns that is used to identify a row uniquely in a table
- The Database admin can define primary keys.



PRIMARY KEYS

Each table in a database can only have **one** primary key.

Every table should have **its own** primary key.

PRIMARY KEYS

- A primary key, also called a primary keyword, is a key in a relational database that is unique for each record. It is a unique identifier, such as a driver license number, telephone number (including area code), or vehicle identification number (VIN).

FOREIGN KEYS

- The **foreign key** is defined in a second table, but it refers to the primary **key** or a unique **key** in the first table.
- A **foreign key** is a field or group of fields in another table that uniquely identifies a row in another table

PRIMARY KEYS – CREATING A TABLE

- Example of the syntax required to create a PRIMARY KEY within a new table of ``movie_reviews`` we are creating

```
CREATE TABLE movie_reviews (  
    review_id SERIAL PRIMARY KEY,  
    film_id INTEGER,  
    review_text TEXT,  
    rating INTEGER CHECK (rating BETWEEN 1 AND 5),  
    review_date DATE  
);
```


PRIMARY KEYS – CREATING A TABLE

- Example of the syntax required to create a PRIMARY KEY within a new table of ``movie_reviews`` we are creating

review_id as the primary key, integer type.

film_id as an integer to link to the film table (assuming it exists).

review_text as text to hold the review content.

rating as an integer to represent a movie rating out of 5.

review_date as a date to record when the review was posted.

INSERTING DATA

- Example of the syntax required to insert data into the `movie_reviews` table we created previously.

INSERT INTO *movie_reviews* (*film_id*, *review_text*, *rating*, *review_date*)

VALUES

(1, 'Great movie, highly recommended!', 5, '2023-01-10'),

(1, 'Good film, but a bit lengthy.', 4, '2023-01-12'),

(1, 'Not what I expected, but decent.', 3, '2023-01-15');

Create a Table

```
CREATE TABLE students (  
  student_id SERIAL PRIMARY KEY,  
  name VARCHAR(100)  
);
```

```
CREATE TABLE enrollments (  
  enrollment_id SERIAL PRIMARY KEY,  
  student_id INTEGER REFERENCES students(student_id),  
  course_name VARCHAR(100)  
);
```

```
INSERT INTO students (name) VALUES ('John Doe');
```

```
INSERT INTO enrollments (student_id, course_name) VALUES (1, 'Introduction to SQL');
```

Update and Delete

```
UPDATE students SET name = 'Jane Doe' WHERE student_id = 1;
```

‘Introduction to SQL’

```
DELETE FROM enrollments WHERE enrollment_id = 1;
```

Dropping and Updating Constraints

```
ALTER TABLE enrollments  
DROP CONSTRAINT enrollments_pkey;
```

```
ALTER TABLE enrollments  
ADD PRIMARY KEY (student_id);
```

INSERTING DATA

- Example of the syntax required to insert data into the `movie_reviews` table we created previously.

	review_id [PK] integer 	film_id integer 	review_text text 	rating integer 	review_date date 
1	1	1	Great movie, highly recommen...	5	2023-01-10
2	2	1	Good film, but a bit lengthy.	4	2023-01-12
3	3	1	Not what I expected, but decent.	3	2023-01-15

UPDATING DATA

- Example of the syntax to update rating from review_id 2, to rating to 3 and replace the original review text into " Good film, but could be shorter. Definitely worth a watch though.'

UPDATE movie_reviews

SET rating = 3, review_text = 'Good film, but could be shorter. Definitely worth a watch though.'

WHERE review_id = 2;

UPDATING DATA

- Example of the syntax to update data from review_id 2, to rating to 3 and replace the original review text into " Good film, but could be shorter. Definitely worth a watch though.'

	review_id [PK] integer	film_id integer	review_text text	rating integer	review_date date
1	1	1	Great movie, highly recommended!	5	2023-01-10
2	2	1	Good film, but could be shorter. Definitely worth a watch thou...	3	2023-01-12
3	3	1	Not what I expected, but decent.	3	2023-01-15


```
CREATE TABLE students (  
    student_id SERIAL PRIMARY KEY,  
    name VARCHAR(100),  
    birth_year INTEGER,  
    favorite_color VARCHAR(20),  
);
```

```
CREATE TABLE enrollments (  
    enrollment_id SERIAL PRIMARY KEY,  
    student_id INTEGER REFERENCES students(student_id),  
    Course_name VARCHAR(100)  
);
```

```
INSERT INTO students (name) VALUES ('John Doe');
```

```
INSERT INTO enrollments (student_id, course_name) VALUES (1, 'Introduction to SQL');
```

```
UPDATE students SET name = 'Jane Doe' WHERE student_id = 1;
```

```
DELETE FROM enrollments WHERE enrollment_id = 1;
```

DELETING DATA

- Example of the syntax required to delete the lowest rating from the `movie_reviews` table.

DELETE FROM movie_reviews

WHERE rating = (SELECT MIN(rating) FROM movie_reviews);

DROP THE ENTIRE TABLE

- Example of the syntax required to delete the entire `movie_reviews` table.

```
DROP TABLE movie_reviews;
```

FOREIGN KEYS

Tables can have multiple foreign keys

The table that contains the foreign key is called a referencing or child table.

The table the foreign key references is called the referenced or parent table.

SQL NOT NULL

- By Default, the columns are able to hold NULL values. A SQL NOT NULL constraint is used to prevent inserting NULL values into the specified column, considering it as a not accepted value for that column.

UNIQUE

- The UNIQUE constraint is used to ensure that no duplicate values will be inserted into a specific column

PRIMARY KEY

- The PRIMARY KEY constraint consists of one column or multiple columns with values that uniquely identify each row in the table.

FOREIGN KEY

- A **foreign key** is a field or group of fields in another table that uniquely identifies a row in another table

TABLE CONSTRAINTS

Project Group Work (7:20PM to 9PM)