Learning Consolidation

Data Modeling Using RDBMS (DDL & DML)

Learning Objectives

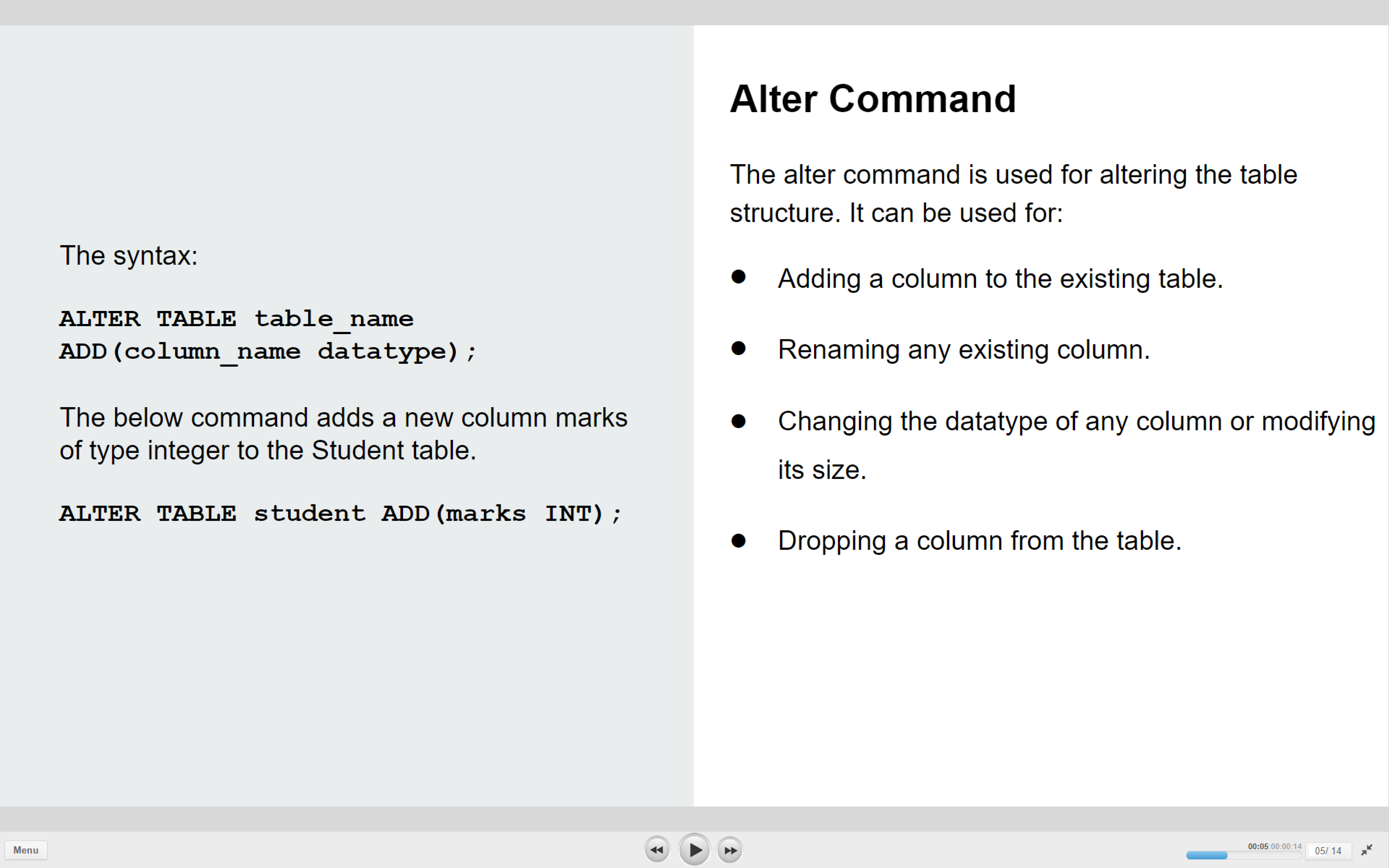
- What is RDBMS?
- Introduction of SQL
- Introduction to Alter commands
- Define Unique, NOT NULL Constraint

Definition of RDBMS

- A Relational Database Management System (RDBMS) is a Database Management System (DBMS) in which there is a relationship among database tables.

Structured Query Language (SQL)

- SQL is a structured query language used to write structured queries that an RDBMS can understand.
- It manages data in a relational database management system (RDBMS).
- SQL is the first commercial language introduced for E.F. Codd's Relational Model of the database.
- It is used to perform all types of data operations on an RDBMS.
- Multiple vendors provide RDBMS like Oracle, MySQL, etc.
- SQL queries must be standardized for all vendors.
- A programmer's query must function for both Oracle and MySQL.
- ANSI SQL provides this standardization.
- All SQL queries must follow the protocols that ANSI SQL provides.



Alter Command

The alter command is used for altering the table structure. It can be used for:

- Adding a column to the existing table.
- Renaming any existing column.
- Changing the datatype of any column or modifying its size.
- Dropping a column from the table.

The syntax:

```
ALTER TABLE table_name
ADD(column_name datatype);
```

The below command adds a new column marks of type integer to the Student table.

```
ALTER TABLE student ADD(marks INT);
```

Alter Command - Drop a Column

The ALTER command can be used to drop or remove columns.

The syntax:

```
ALTER TABLE table_name
DROP(column_name);
```

The below command will drop the age column from the table student.

```
ALTER TABLE student DROP(age);
```

Alter Command - Modify an Existing Column

The ALTER command can be used to modify the data type of any existing column.

The syntax:

```
ALTER TABLE table_name
modify(column_name datatype);
```

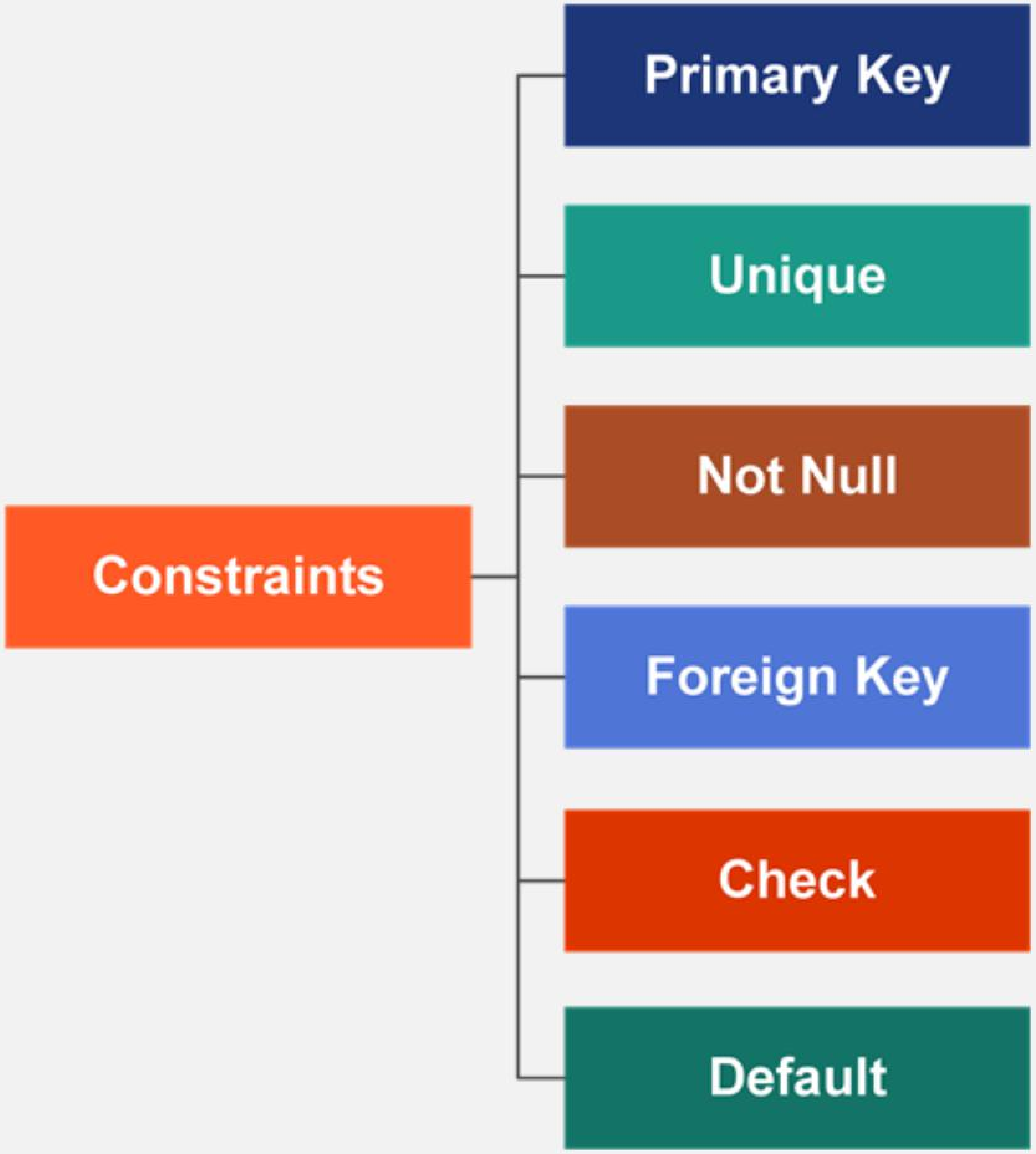
The below command will modify the name column of the student table, to now hold up to 300 characters.

```
ALTER TABLE student
MODIFY(name varchar(300));
```

Types of SQL Constraints

- There are six different types of SQL constraints.

Note – We will only discuss about Primary and Foreign Key Constraints only.



Constraints

Primary Key

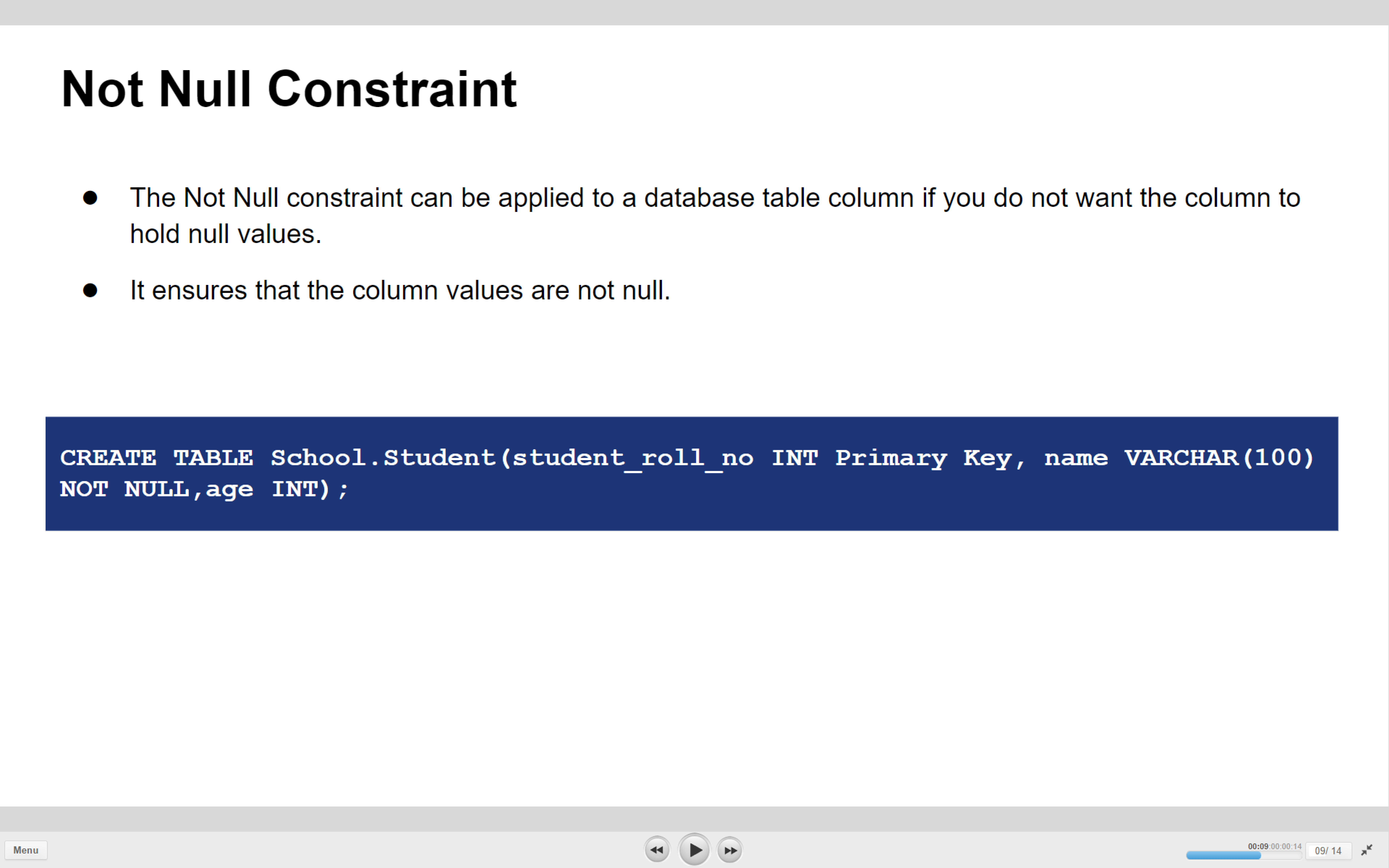
Unique

Not Null

Foreign Key

Check

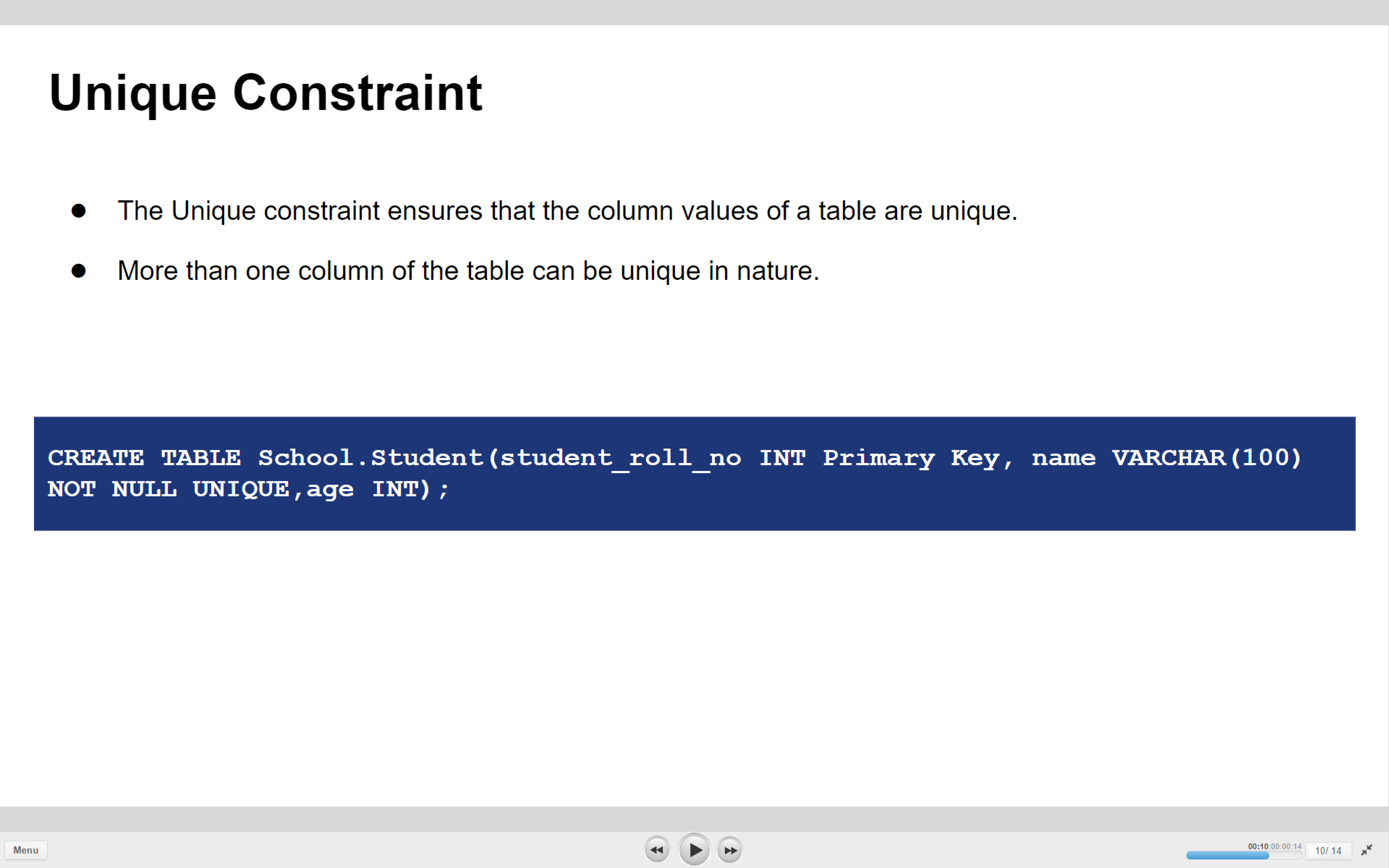
Default



Not Null Constraint

- The Not Null constraint can be applied to a database table column if you do not want the column to hold null values.
- It ensures that the column values are not null.

```
CREATE TABLE School.Student(student_roll_no INT Primary Key, name VARCHAR(100)
NOT NULL,age INT);
```



Unique Constraint

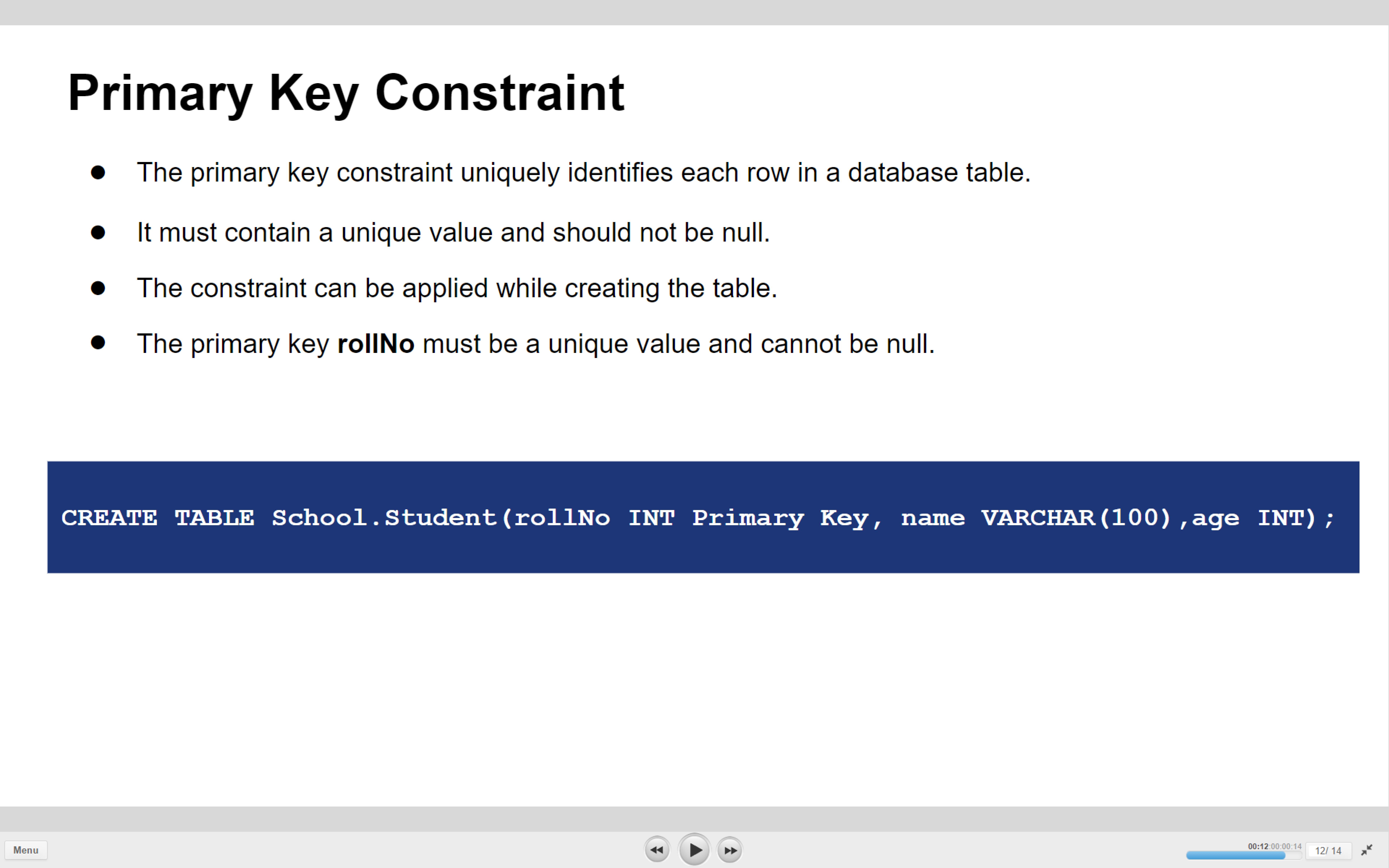
- The Unique constraint ensures that the column values of a table are unique.
- More than one column of the table can be unique in nature.

```
CREATE TABLE School.Student(student_roll_no INT Primary Key, name VARCHAR(100)
NOT NULL UNIQUE,age INT);
```

Default Constraint

The default constraint is used to assign a default value to a column if the values are not specified when inserting data into the table.

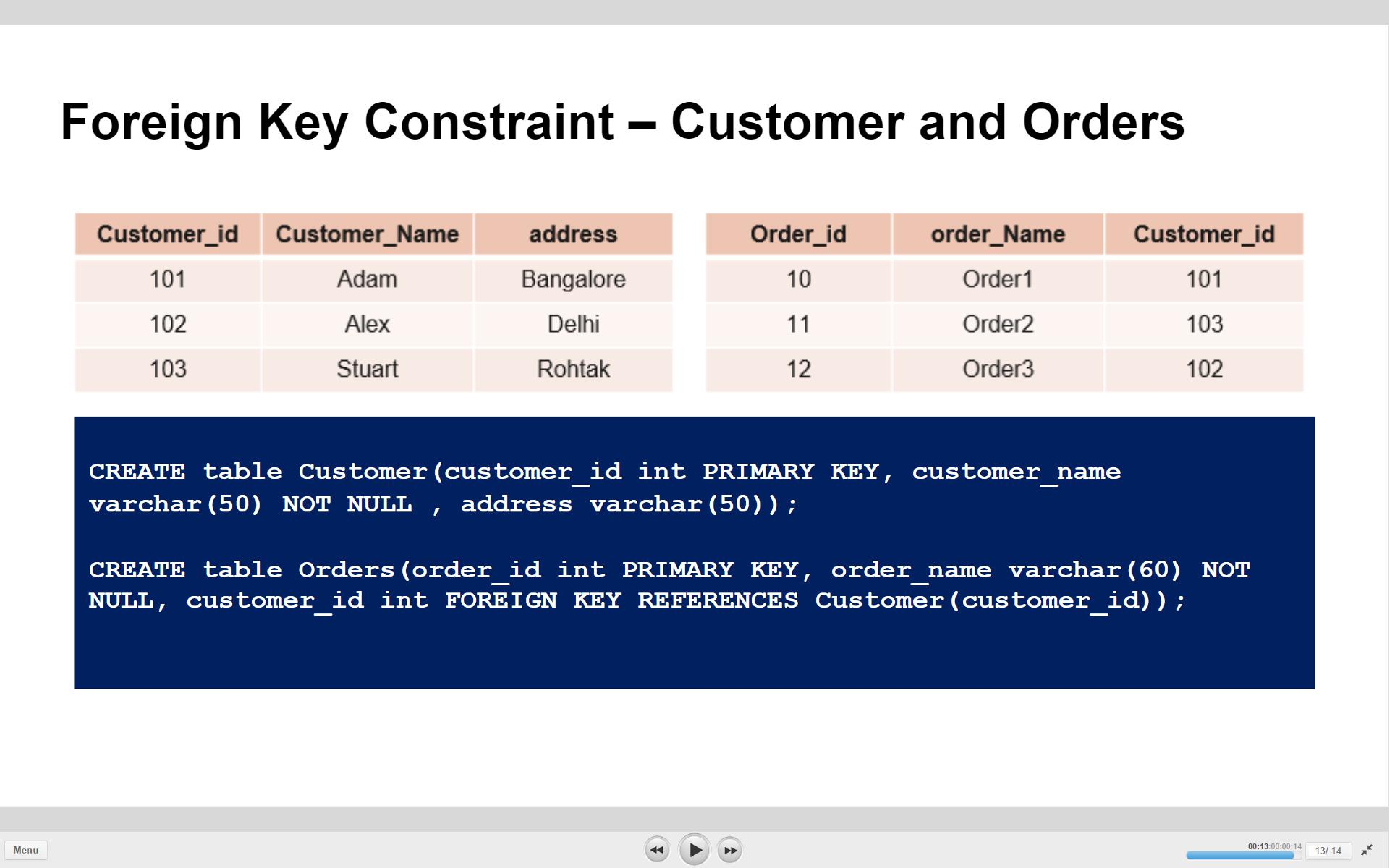
```
CREATE TABLE School.Student(student_roll_no INT Primary Key, name VARCHAR(100)
NOT NULL,age INT CHECK(age > 0), city VARCHAR(50) DEFAULT 'Mumbai';
```



Primary Key Constraint

- The primary key constraint uniquely identifies each row in a database table.
- It must contain a unique value and should not be null.
- The constraint can be applied while creating the table.
- The primary key **rollNo** must be a unique value and cannot be null.

```
CREATE TABLE School.Student(rollNo INT Primary Key, name VARCHAR(100),age INT);
```



Foreign Key Constraint – Customer and Orders

Customer_id	Customer_Name	address
101	Adam	Bangalore
102	Alex	Delhi
103	Stuart	Rohtak

Order_id	order_Name	Customer_id
10	Order1	101
11	Order2	103
12	Order3	102

```
CREATE table Customer(customer_id int PRIMARY KEY, customer_name
varchar(50) NOT NULL , address varchar(50));

CREATE table Orders(order_id int PRIMARY KEY, order_name varchar(60) NOT
NULL, customer_id int FOREIGN KEY REFERENCES Customer(customer_id));
```