



MySQL Integrity Constraints

Soma Soleiman Zadeh
Database Systems II (IT 226)
Spring 2025 - 2026
Week 2
February 9, 2026

Outline



- What are **Constraints** in MySQL?
- Common **MySQL Constraints**
- Applying Constraints in **CREATE TABLE** Statement
- Applying Constraints in **ALTER TABLE** Statement



MySQL Integrity Constraints

- **Integrity Constraints** are rules to ensure that the data in the database is accurate, consistent and reliable.
- If the data action in the database violated any integrity constraint, the action is avoided.
- For example, PRIMARY KEY is a constraint in a table. According to this constraint, the values of the primary key column must be unique and not null. If we try to enter a repeated value or null value in the primary key column, we get error.



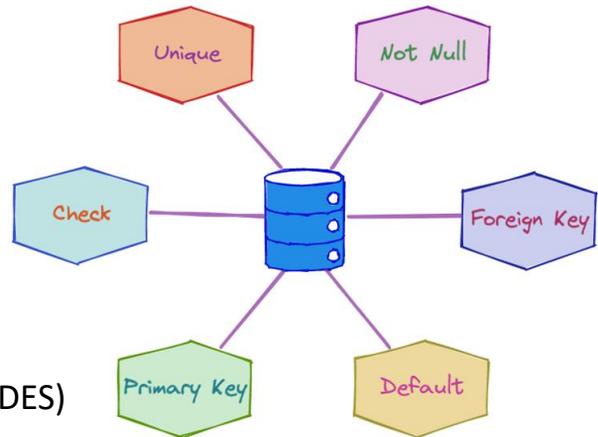
Defining MySQL Constraints in a Table

- Constraints can be specified in two ways:
 - **1st way:** while creating a table, using the **CREATE TABLE** statement.
 - **2nd way:** after a table is created, using the **ALTER TABLE** statement.



Most Common MySQL Constraints

- NOT NULL
- AUTO_INCREMENT
- UNIQUE
- DEFAULT
- CHECK
- PRIMARY KEY
- FOREIGN KEY (including CASCADES)



NOT NULL Constraint

- By default, a column can hold NULL values (empty cell).
- The **NOT NULL** constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to that field.

NOT NULL Constraint

Specifying NOT NULL Constraint in CREATE TABLE	Specifying NOT NULL Constraint in ALTER TABLE
<pre>CREATE TABLE table_name (Column1 datatype1 NOT NULL, Column2 datatype2, Column3 datatype3);</pre> 	<pre>ALTER TABLE table_name MODIFY Column1 datatype1 NOT NULL;</pre> 
Example	Example
<pre>CREATE TABLE Student (ID int, fullName varchar(100) NOT NULL, age int);</pre>	<pre>ALTER TABLE Student MODIFY fullName varchar(100) NOT NULL;</pre>



Removing NOT NULL Constraint from a Column

Syntax:

```
ALTER TABLE table_name
  MODIFY Column1 datatype1;
```

Example:

```
ALTER TABLE Student
  MODIFY fullName varchar(100);
```

`fullName` column already had NOT NULL constraint. After executing this statement, the NOT NULL constraint is removed from the `fullName` column and `fullName` can be null.



AUTO_INCREMENT Constraint

- **Auto-increment** allows a unique number to be generated automatically when a new record is inserted into a table.
- Often it is used with **Primary Key** fields.
- By default, the starting value for **AUTO-INCREMENT** is 1, and it will be incremented by 1 for each new record (e.g.: 1, 2, 3, 4, etc.).

AUTO_INCREMENT Constraint

Specifying **AUTO_INCREMENT** Constraint in **CREATE TABLE**

```
CREATE TABLE table_name  
( Column1 datatype1 AUTO_INCREMENT ,  
  Column2 datatype2 ,  
  Column3 datatype3);
```

Example

```
CREATE TABLE Student  
( ID int AUTO_INCREMENT ,  
  fullName varchar(100),  
  username varchar(100) );
```

Specifying **AUTO_INCREMENT** Constraint in **ALTER TABLE**

```
ALTER TABLE table_name  
MODIFY column1 datatype1 AUTO_INCREMENT ;
```

Example

```
ALTER TABLE Student  
MODIFY ID int AUTO_INCREMENT ;
```



Removing AUTO_INCREMENT Constraint from a column

- Syntax:

```
ALTER TABLE table_name  
MODIFY column1 datatype1;
```

- Example:

```
ALTER TABLE Student  
MODIFY ID int;
```

ID column already had AUTO_INCREMENT constraint. After executing this statement, the AUTO_INCREMENT constraint is removed from the ID column.



UNIQUE Constraint

- The **UNIQUE** constraint ensures that all values in a column are different.
- Both the **UNIQUE** and **PRIMARY KEY** constraints provide a guarantee for uniqueness for a column or set of columns.
- There can have many **UNIQUE** constraints per table, but only one **PRIMARY KEY** constraint per table.
- Do we need to add a **UNIQUE** constraint with **PRIMARY KEY**?
 - No. Because the **primary key** column has **UNIQUE** constraint by default.

UNIQUE Constraint in CREATE TABLE

Specifying UNIQUE Constraint in CREATE TABLE	Example
There are two ways in specifying UNIQUE constraint in CREATE TABLE:	
<pre>CREATE TABLE table_name (Column1 datatype1 UNIQUE, Column2 datatype2);</pre>	<pre>CREATE TABLE Student (ID int, fullName varchar(100), username varchar(100) UNIQUE);</pre>
<pre>CREATE TABLE table_name (Column1 datatype1, Column2 datatype2, CONSTRAINT constraint_name UNIQUE (column_name));</pre>	<pre>CREATE TABLE Student (ID int, fullName varchar(100), username varchar(100), CONSTRAINT Unique_Username UNIQUE (username));</pre>

UNIQUE Constraint in ALTER TABLE

Specifying UNIQUE Constraint in ALTER TABLE	Example
There are two ways in specifying UNIQUE constraint in ALTER TABLE:	
<pre>ALTER TABLE table_name ADD UNIQUE (Column_name);</pre>	<pre>ALTER TABLE Student ADD UNIQUE (username);</pre>
<pre>ALTER TABLE table_name ADD CONSTRAINT constraint_name UNIQUE (Column1);</pre>	<pre>ALTER TABLE Student ADD CONSTRAINT uniqueUsername UNIQUE (username);</pre>



Removing UNIQUE Constraint from a Column

◦ Syntax: **ALTER TABLE** table_name
DROP INDEX column_name ;

◦ Example:

```
ALTER TABLE Student  
DROP INDEX username;
```



CHECK Constraint

- The **CHECK** constraint is used to limit the value range that can be inserted into a column.
- If you define a **CHECK** constraint on a single column it allows only certain values for this column.

CHECK Constraint in CREATE TABLE

Specifying CHECK Constraint in CREATE TABLE

Example

There are two ways in specifying CHECK constraint in CREATE TABLE:

CREATE TABLE table_name
(Column1 datatype1 **CHECK** (rule));

1

CREATE TABLE Student
(ID int,
Gender varchar(10),
Age int **CHECK** (Age > 18));

CREATE TABLE table_name
(Column1 datatype1,
Column2 datatype2,
CONSTRAINT constraint_name **CHECK** (rule));

2

CREATE TABLE Student
(ID int,
Gender varchar(10),
Age int,
CONSTRAINT CHK_Age **CHECK** (Age>18));

CHECK Constraint in ALTER TABLE

Specifying CHECK Constraint in ALTER TABLE

Example

There are two ways in specifying CHECK constraint in ALTER TABLE:

ALTER TABLE table_name
ADD CHECK (rule);

1

ALTER TABLE Student
ADD CHECK (Age>18);

ALTER TABLE table_name
ADD CONSTRAINT constraint_name **CHECK** (rule);

2

ALTER TABLE Student
ADD CONSTRAINT CHK_Age **CHECK** (Age>18);



CHECK Constraint

- In CHECK constraint, it is possible to check more than one rule on more than one column in a table.
- For example, in the following statement, we add a constraint to check **age>18** and **level=2**. So, if we enter a new record to **student** table, we can enter only values greater than 18 for **Age** column and value 2 for **Level** column.

```
CREATE TABLE Student
( ID int,
  Gender varchar(10),
  Level int,
  Age int,
  CONSTRAINT CHK_Age_Gender CHECK (Age>18 AND Level=2) );
```



Removing CHECK Constraint from a column

- Syntax: **ALTER TABLE** table_name
DROP CHECK constraint_name;
- Example:

```
ALTER TABLE Student
DROP CHECK CHK_Age;
```



DEFAULT Constraint

- The **DEFAULT** constraint is used to provide a default value to a column.
- The default value will be added to new records if no any value is added.

DEFAULT Constraint

Specifying **DEFAULT** Constraint in
CREATE TABLE

```
CREATE TABLE table_name  
( Column1 datatype1,  
  Column2 datatype2 DEFAULT 'default_value' );
```



Example

```
CREATE TABLE Student  
( ID int,  
  Gender enum('male', 'female') DEFAULT 'male');
```

Specifying **DEFAULT** Constraint in
ALTER TABLE

```
ALTER TABLE table_name  
ALTER column1 SET DEFAULT 'default_value' ;
```



Example

```
ALTER TABLE Student  
ALTER Gender SET DEFAULT 'male';
```



Removing DEFAULT Constraint from a column

- Syntax: `ALTER TABLE table_name
ALTER column1 DROP DEFAULT;`

- Example:

```
ALTER TABLE Student  
ALTER gender DROP DEFAULT;
```