Key Topics: `ZEROFILL`, `UNSIGNED`, and `GENERATED ALWAYS AS`

In MySQL, these three keywords and attributes play distinct roles in handling data within tables, impacting how data is stored, displayed, and calculated. Let's break down each concept, explain its syntax and usage, and provide real-world examples for clarity.

---

1. `ZEROFILL`

What It Is

- `ZEROFILL` is an attribute used with numeric data types in MySQL (e.g., `INT`, `FLOAT`) to pad the value with zeros up to a specified width.
- When `ZEROFILL` is applied, it overrides the display of the number by padding with leading zeros to meet the field width specification.

Syntax

```
CREATE TABLE table_name (
    column_name INT(size) ZEROFILL
);
```

Explanation with Example

In the example provided:

```
CREATE TABLE emp2 (
    id INT(10) ZEROFILL,
    name VARCHAR(20),
    salary FLOAT
);
```

- Here, `id` has a `ZEROFILL` attribute with a width of 10. This means that if you insert an ID like `111`, it will be displayed as `0000000111` to fill the width of 10.
- Padding is purely visual—it doesn't change the actual integer value.

Real-World Use Case

- **Banking Systems:** Account numbers are often displayed with leading zeros (e.g., `0001234567`) to meet a certain format, making data visually consistent for users.

Output in the Example

When we insert records and display them:

```
+------------+--------+--------+
| id         | name   | salary |
+------------+--------+--------+
| 0000000111 | aman   |   8900 |
```

```
| 0000099999 | deepak |   7000 |
| 0000000001 | pankaj |   7200 |
| 0000000333 | saroj  |   5600 |
+------------+--------+--------+
```

Each `id` has leading zeros, ensuring all IDs are displayed in a 10-character wide field.

---

2. `UNSIGNED`

What It Is

- The `UNSIGNED` attribute specifies that a column can only hold non-negative values. By default, MySQL allows both positive and negative values for numeric data types.
- `UNSIGNED` doubles the range of positive values since it excludes negative numbers, which can be useful for storing only positive numbers with an extended range.

Syntax

```sql
CREATE TABLE table_name (
    column_name INT UNSIGNED
);
```

Explanation with Example

In the example provided:

```sql
CREATE TABLE emp2 (
    id INT(10) ZEROFILL,
    name VARCHAR(20),
    salary FLOAT UNSIGNED
);
```

- The `salary` column is `FLOAT UNSIGNED`, which means it can only store non-negative floating-point values.
- Attempting to insert a negative salary value will result in an error.

Real-World Use Case

- **Salary or Payment Systems:** Attributes like salary, payments, or product prices are usually non-negative, making `UNSIGNED` a practical attribute to avoid erroneous entries.

---

3. `GENERATED ALWAYS AS`

What It Is

- The `GENERATED ALWAYS AS` attribute defines a **computed column** that automatically calculates its value based on an expression, without manual input.
- These columns can be either `VIRTUAL` (calculated on the fly) or `STORED` (calculated and stored in the table).

Syntax

```
CREATE TABLE table_name (
    column_name1 INT,
    column_name2 FLOAT,
    generated_column FLOAT GENERATED ALWAYS AS (column_name1 *
column_name2) VIRTUAL
);
```

Explanation with Example

In the `orders` table example:

```
CREATE TABLE orders (
    pid INT,
    pname VARCHAR(20),
    qty INT,
    price FLOAT,
    Total FLOAT GENERATED ALWAYS AS (price * qty) VIRTUAL
);
```

- The `Total` column is a virtual column that calculates the total price by multiplying `price` and `qty` for each order.
- You don't need to insert a value for `Total` since MySQL calculates it automatically.

Real-World Use Case

- **E-commerce Systems:** Calculating the total price of items in an order by multiplying quantity (`qty`) and price per item (`price`) is a common requirement in e-commerce.

Output in the Example

When data is inserted:

```
INSERT INTO orders (pid, pname, qty, price) VALUES
(111, 'monitor', 20, 4500),
(222, 'keyboard', 120, 250),
(333, 'speaker', 20, 560);

SELECT * FROM orders;
```

The output is:

```
+------+----------+------+-------+-------+
| pid  | pname    | qty  | price | Total |
+------+----------+------+-------+-------+
|  111 | monitor  |   20 |  4500 | 90000 |
|  222 | keyboard |  120 |   250 | 30000 |
|  333 | speaker  |   20 |   560 | 11200 |
+------+----------+------+-------+-------+
```

The `Total` column correctly computes the values for each row based on `qty * price` .

Summary Table

Attribute	Description	Real-World Example
`ZEROFILL`	Pads numeric values with leading zeros up to the specified width.	Account numbers in banking systems
`UNSIGNED`	Restricts values to non-negative numbers, expanding the positive range for numeric data types.	Salary or payment systems, product prices
`GENERATED ALWAYS AS`	Creates a computed column that automatically calculates values based on other columns.	E-commerce order total calculations (e.g., price * quantity)

These attributes (`ZEROFILL` , `UNSIGNED` , and `GENERATED ALWAYS AS`) enhance data consistency, ensure data integrity, and automate calculations, making them essential for maintaining reliable and efficient database tables.

Benefits of Using These Attributes

- **Data Integrity**: `UNSIGNED` ensures only logical values are stored for inherently positive data like salary and age.
- **Consistency**: `ZEROFILL` ensures IDs or numeric codes appear in a uniform format, improving readability.
- **Efficiency**: `GENERATED ALWAYS AS` automates calculations, reducing redundant data entry and minimizing manual errors.

In [ ]: