SQL | SEQUENCES

Sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demand.

A sequence is a user defined schema bound object that generates a sequence of numeric values.

Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.

The sequence of numeric values is generated in an ascending or descending order at defined intervals and can be configured to restart when exceeds max\_value.

Syntax:

CREATE SEQUENCE sequence\_name

START WITH initial\_value

INCREMENT BY increment\_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

sequence\_name: Name of the sequence.

initial\_value: starting value from where the sequence starts.

Initial\_value should be greater than or equal

to minimum value and less than equal to maximum value.

increment\_value: Value by which sequence will increment itself.

Increment\_value can be positive or negative.

minimum\_value: Minimum value of the sequence.

maximum\_value: Maximum value of the sequence.

cycle: When sequence reaches its set\_limit

it starts from beginning.

nocycle: An exception will be thrown

if sequence exceeds its max\_value.

Example

Following is the sequence query creating sequence in ascending order.

Example 1:

CREATE SEQUENCE sequence\_1

start with 1

increment by 1

minvalue 0

maxvalue 100

cycle;

Above query will create a sequence named sequence\_1.Sequence will start from 1 and will be incremented by 1 having maximum value 100. Sequence will repeat itself from start value after exceeding 100.

Example 2:

Following is the sequence query creating sequence in descending order.

CREATE SEQUENCE sequence\_2

start with 100

increment by -1

minvalue 1

maxvalue 100

cycle;

Above query will create a sequence named sequence\_2.Sequence will start from 100 and should be less than or equal to maximum value and will be incremented by -1 having minimum value 1.

Example to use sequence : create a table named students with columns as id and name.

CREATE TABLE students

(

ID number(10),

NAME char(20)

);

Now insert values into table

INSERT into students VALUES(sequence\_1.nextval,'Ramesh');

INSERT into students VALUES(sequence\_1.nextval,'Suresh');

where sequence\_1.nextval will insert id’s in id column in a sequence as defined in sequence\_1.

Output:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| ID | NAME |

------------------------

| 1 | Ramesh |

| 2 | Suresh |

----------------------

This article is contributed by ARSHPREET SINGH. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

A sequence is a set of integers 1, 2, 3, ... that are generated in order on demand. Sequences are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

This chapter describes how to use sequences in MySQL.

Using AUTO\_INCREMENT column

The simplest way in MySQL to use sequences is to define a column as AUTO\_INCREMENT and leave the rest to MySQL to take care.

Example

Try out the following example. This will create a table and after that it will insert a few rows in this table where it is not required to give a record ID because its auto-incremented by MySQL.

mysql> CREATE TABLE INSECT

-> (

-> id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

-> PRIMARY KEY (id),

-> name VARCHAR(30) NOT NULL, # type of insect

-> date DATE NOT NULL, # date collected

-> origin VARCHAR(30) NOT NULL # where collected

);

Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO INSECT (id,name,date,origin) VALUES

-> (NULL,'housefly','2001-09-10','kitchen'),

-> (NULL,'millipede','2001-09-10','driveway'),

-> (NULL,'grasshopper','2001-09-10','front yard');

Query OK, 3 rows affected (0.02 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT \* FROM INSECT ORDER BY id;

+----+-------------+------------+------------+

| id | name | date | origin |

+----+-------------+------------+------------+

| 1 | housefly | 2001-09-10 | kitchen |

| 2 | millipede | 2001-09-10 | driveway |

| 3 | grasshopper | 2001-09-10 | front yard |

+----+-------------+------------+------------+

3 rows in set (0.00 sec)

Obtain AUTO\_INCREMENT Values

The LAST\_INSERT\_ID( ) is an SQL function, so you can use it from within any client that understands how to issue SQL statements. Otherwise PERL and PHP scripts provide exclusive functions to retrieve auto-incremented value of last record.

PERL Example

Use the mysql\_insertid attribute to obtain the AUTO\_INCREMENT value generated by a query. This attribute is accessed through either a database handle or a statement handle, depending on how you issue the query. The following example references it through the database handle.

$dbh->do ("INSERT INTO INSECT (name,date,origin)

VALUES('moth','2001-09-14','windowsill')");

my $seq = $dbh->{mysql\_insertid};

PHP Example

After issuing a query that generates an AUTO\_INCREMENT value, retrieve the value by calling the mysql\_insert\_id( ) function.

mysql\_query ("INSERT INTO INSECT (name,date,origin)

VALUES('moth','2001-09-14','windowsill')", $conn\_id);

$seq = mysql\_insert\_id ($conn\_id);

Renumbering an Existing Sequence

There may be a case when you have deleted many records from a table and you want to re-sequence all the records. This can be done by using a simple trick, but you should be very careful to do this and check if your table is having a join with another table or not.

If you determine that resequencing an AUTO\_INCREMENT column is unavoidable, the way to do it is to drop the column from the table, then add it again.

The following example shows how to renumber the id values in the insect table using this technique.

mysql> ALTER TABLE INSECT DROP id;

mysql> ALTER TABLE insect

-> ADD id INT UNSIGNED NOT NULL AUTO\_INCREMENT FIRST,

-> ADD PRIMARY KEY (id);

Starting a Sequence at a Particular Value

By default, MySQL will start the sequence from 1, but you can specify any other number as well at the time of table creation.

The following code block has an example where MySQL will start sequence from 100.

mysql> CREATE TABLE INSECT

-> (

-> id INT UNSIGNED NOT NULL AUTO\_INCREMENT = 100,

-> PRIMARY KEY (id),

-> name VARCHAR(30) NOT NULL, # type of insect

-> date DATE NOT NULL, # date collected

-> origin VARCHAR(30) NOT NULL # where collected

);

Alternatively, you can create the table and then set the initial sequence value with ALTER TABLE.

mysql> ALTER TABLE t AUTO\_INCREMENT = 100;