MYSQL HANDLING DUPLICATES

Tables or result sets sometimes contain duplicate records. Sometimes, it is allowed but sometimes it is required to stop duplicate records. Sometimes, it is required to identify duplicate records and remove them from the table. This chapter will describe how to prevent duplicate records occurring in a table and how to remove already existing duplicate records.

Preventing Duplicates from Occurring in a Table:

You can use a **PRIMARY KEY** or **UNIQUE** Index on a table with appropriate fields to stop duplicate records. Let's take one example: The following table contains no such index or primary key, so it would allow duplicate records for first_name and last_name.

```
CREATE TABLE person_tbl
(
    first_name CHAR(20),
    last_name CHAR(20),
    sex CHAR(10)
);
```

To prevent multiple records with the same first and last name values from being created in this table, add a PRIMARY KEY to its definition. When you do this, it's also necessary to declare the indexed columns to be NOT NULL, because a PRIMARY KEY does not allow NULL values:

```
CREATE TABLE person_tbl
(
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10)
   PRIMARY KEY (last_name, first_name)
);
```

The presence of a unique index in a table normally causes an error to occur if you insert a record into the table that duplicates an existing record in the column or columns that define the index.

Use **INSERT IGNORE** rather than **INSERT**. If a record doesn't duplicate an existing record, MySQL inserts it as usual. If the record is a duplicate, the IGNORE keyword tells MySQL to discard it silently without generating an error.

Following example does not error out and same time it will not insert duplicate records.

```
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 0 rows affected (0.00 sec)
```

Use **REPLACE** rather than INSERT. If the record is new, it's inserted just as with INSERT. If it's a duplicate, the new record replaces the old one:

```
mysql> REPLACE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Ajay', 'Kumar');
Query OK, 1 row affected (0.00 sec)
mysql> REPLACE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Ajay', 'Kumar');
Query OK, 2 rows affected (0.00 sec)
```

INSERT IGNORE and REPLACE should be chosen according to the duplicate-handling behavior you want to effect. INSERT IGNORE keeps the first of a set of duplicated records and discards the rest. REPLACE keeps the last of a set of duplicates and erase out any earlier ones.

Another way to enforce uniqueness is to add a UNIQUE index rather than a PRIMARY KEY to a table.

```
CREATE TABLE person_tbl
(
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10)
   UNIQUE (last_name, first_name)
);
```

Counting and Identifying Duplicates:

Following is the query to count duplicate records with first_name and last_name in a table.

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name
    -> FROM person_tbl
    -> GROUP BY last_name, first_name
    -> HAVING repetitions > 1;
```

This query will return a list of all the duplicate records in person_tbl table. In general, to identify sets of values that are duplicated, do the following:

- Determine which columns contain the values that may be duplicated.
- List those columns in the column selection list, along with COUNT(*).
- List the columns in the GROUP BY clause as well.
- Add a HAVING clause that eliminates unique values by requiring group counts to be greater than one.

Eliminating Duplicates from a Query Result:

You can use **DISTINCT** along with SELECT statement to find out unique records available in a table.

```
mysql> SELECT DISTINCT last_name, first_name
    -> FROM person_tbl
    -> ORDER BY last_name;
```

An alternative to DISTINCT is to add a GROUP BY clause that names the columns you're selecting. This has the effect of removing duplicates and selecting only the unique combinations of values in the specified columns:

```
mysql> SELECT last_name, first_name
    -> FROM person_tbl
    -> GROUP BY (last_name, first_name);
```

Removing Duplicates Using Table Replacement:

If you have duplicate records in a table and you want to remove all the duplicate records from that table, then here is the procedure:

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex
    ->                  FROM person_tbl;
    ->                  GROUP BY (last_name, first_name);
mysql> DROP TABLE person_tbl;
mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

An easy way of removing duplicate records from a table is to add an INDEX or PRIMAY KEY to that table. Even if this table is already available, you can use this technique to remove duplicate records and you will be safe in future as well.

```
mysql> ALTER IGNORE TABLE person_tbl
    -> ADD PRIMARY KEY (last_name, first_name);
```