How to Delete Millions of Records?

Author: Prabin Baniya with No comments

Some companies have small databases while others have large databases. To accommodate large data sets, you will have tables with millions of records. I have experience working in an organization with big tables. The smallest table only had 21 millions records. I will let you define the "big table" for those organizations. Working with the bigger tables is not the same as working with tables that have few thousand records. **This article will disclose the important tips which you were not taught in your database class or in your job training.**

Inserting, Deleting, Updating, and building Index on bigger table requires extra steps, proper planning, and a full understanding of database engine & architecture. With this article, I will show you how to Delete or Insert millions of records to and from a giant table. Before we dig into how we do that, let's discus why big tables need proper planning or in other words, **why can't we just run the Delete or Insert on a table?**

To answer above questions, you will need to know the ACID properties of database. Which means either the transaction happened or did not happen, the transaction cannot be in a limbo state or 1/2 complete state. To make ACID properties work, all the transactions that are happening inside a database engine are recorded to the transaction log files. Each database has transaction log files but they may be given a different name. These log files help the transaction to rollback in case of failure or errors. Delete is considered a transaction in any DBMS. When delete statement is running, the transaction log files record everything with delete transaction. The same thing applies to Insert and Update as well. If someone killed your session before it is done, it will rollback your transaction to what it was before using transaction log files. Have you noticed a long running transaction which when killed by DBA will take almost same time to rollback? I bet you know now, what's going on here.

What happens when you delete millions of records in a single transaction? Deleting millions of records cause the transaction log file to grow quickly and fill up the disk space. When you run out of space, it might take your database down. To prevent this failure, we need to commit the transaction at a certain interval meaning we need to break this giant one transactions to many small transactions. Commits makes the change permanent and ends the transaction. I normally pick a commit point of on every 10 K for Deletes/Updates/Insert. This is something I would check with DBA because every database configurations is different and your DBA know the best.

ORACLE:

Insert Example

```
DECLARE
    CURSOR my_cursor IS
```

```
      SELECT id
      FROM   employee; -- replace this with your select statement
    commit_count NUMBER := 0;
BEGIN
    FOR cur_record IN my_cursor LOOP
        INSERT INTO employee_test
                    (id)
        VALUES      (cur_record.id );
        -- replace this with your insert/ statement.

        commit_count := commit_count + 1;

        IF MOD(commit_count, 10000) = 0 THEN
          COMMIT;
        END IF;
    END LOOP;
END;
```

In this example, we are Inserting 19 millions records with a commit interval on every 10,000th record.

Delete Example:

```
DECLARE
  v_count NUMBER;
BEGIN
  LOOP
    SELECT Count(*)
    INTO   v_count
    FROM   wareshour_admin.hardware_inventory
    WHERE  code = 'NEPUSA';

    IF v_count > 0 THEN
      DELETE
      FROM   wareshour_admin.hardware_inventory
      WHERE  code = 'NEPUSA'
      AND
      ROWNUM <= 10000;
      COMMIT;
    ELSE
      EXIT;
    END IF;
  END LOOP;
END;
```

In this example, we have 11 millions record to delete from a table of 62 millions records. Again, we have commit interval of 10 K.

SQL Server:

Delete Example:

```
SELECT 'Deleting records in batch of 10K'
WHILE @@ROWCOUNT > 0
  BEGIN
      DELETE TOP (10000) FROM [dba_test].[dbo].[employee]
      WHERE  salary > 80000
  END
```

In this SQL Server TSQL example, we have 13 millions records with salary above 80K. We have divided the transaction of 10K delete at a time.

Insert Example:

[This one is for you to try!!]

Big Table Tips:

Things to consider while working with big tables.

1. Use Truncate table, if you need to delete all
2. If you are deleting records more than 70% of data, I would create a temp table, copy only the records you need to this table and run truncate on the original table. This approach is much faster.
3. Breaking a big transaction into a small transactions applies to Insert and Update as well.
4. Where possible use table partitioning. This makes the maintenance easy. You can drop a partition if you need to delete it.
5. If you don't care about rollback, you can run your Insert and Delete with minimum logging.

In a nutshell, the key to success while dealing with big tables is to split the large transactions into many small transactions. This makes your DML statements run smooth without impacting the performance of your database.