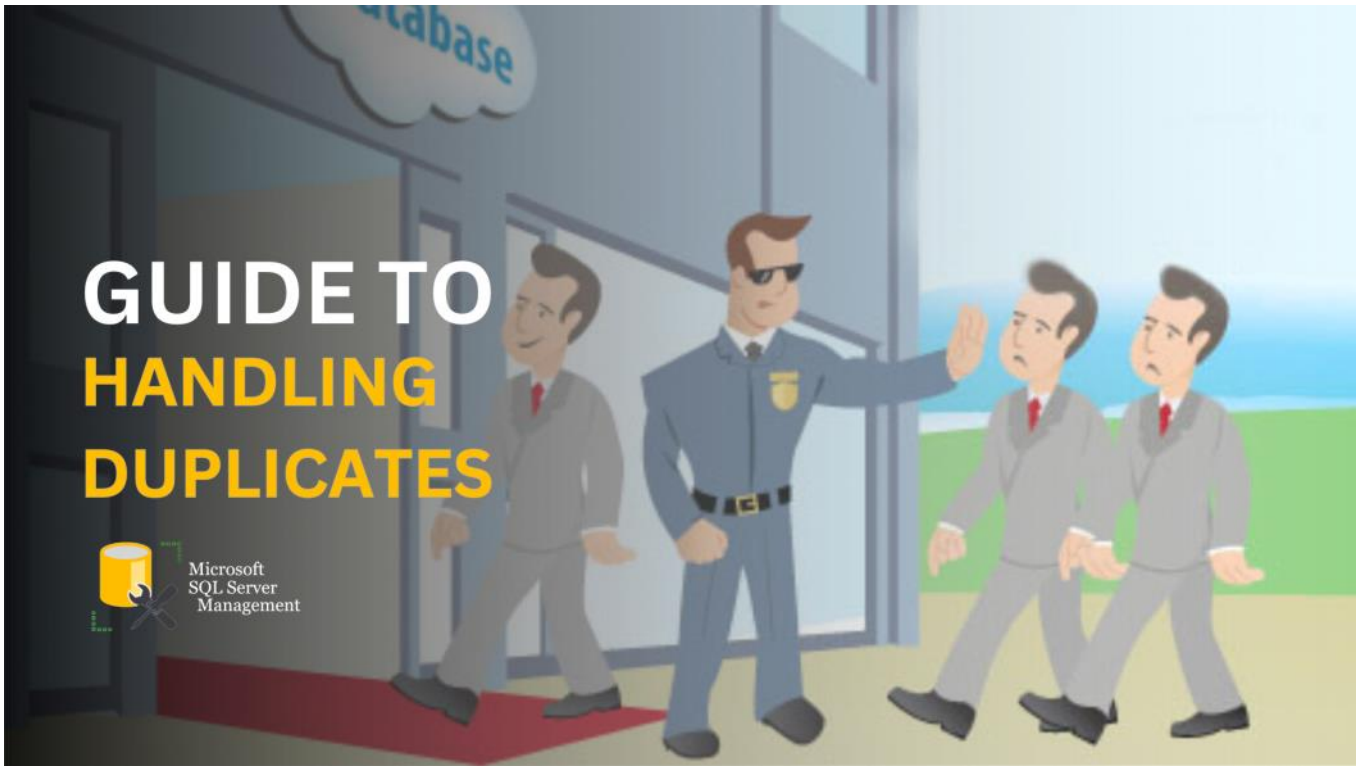
Handling Duplicates

Saturday, July 27, 2024 10:04 PM



INTRODUCTION

Tables or result sets in a database usually contain duplicate records. While duplicates are generally allowed, there are situations where it is necessary to prevent them. In such cases, it becomes essential to identify and remove duplicate records from a database table.

Importance of Handling SQL Duplicates

There are various reasons why handling duplicates in a database becomes necessary. One of the main reasons is that the existence of duplicates in an organizational database will lead to logical errors. In addition to it, we need to handle redundant data to prevent the following consequences

- Duplicate data occupies storage space, reducing the efficiency of database usage and increasing storage costs.
- Dealing with duplicate records consumes additional resources, driving up the overall cost of maintaining the database.
- Duplicates in a database can lead to logical errors in data, affecting the integrity and reliability of the information stored.

Preventing Duplicate Entries

You can use a **PRIMARY KEY** or a **UNIQUE** Index on a table with the appropriate fields to prevent duplicate record entries into a table.

Example:

```sql
/** Creating tables and inserting data*/
DROP TABLE IF EXISTS Employees2
CREATE TABLE Employees2 (
    EmployeeID INT PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Email NVARCHAR(100),
    DepartmentID INT,
    Salary DECIMAL(10, 2),
    HireDate DATE
)
```

```sql
DROP TABLE IF EXISTS Departments
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName NVARCHAR(50),
    ManagerID INT
)
```

The data:

	EmployeeID	FirstName	LastName	Email	DepartmentID	Salary	HireDate	DepartmentID	DepartmentName	ManagerID
1	1	John	Doe	john.doe@example.com	1	60000.00	2019-05-01	1	HR	1
2	2	Jane	Smith	jane.smith@example.com	2	75000.00	2020-01-15	2	Finance	NULL
3	3	Sam	Williams	sam.williams@example.com	NULL	50000.00	2018-03-10	3	IT	9
4	4	Emily	Brown		1	55000.00	2021-07-22	4	Sales	7
5	5	Michael	Johnson	michael.johnson@example.com	3	NULL	2018-11-30	5	NULL	NULL
6	6	Anna	Davis	anna.davis@example.com	3	48000.00	2022-03-10	6	Marketing	4
7	7	James	Miller	james.miller@example.com	4	62000.00	NULL	7	Operations	5
8	8	Laura	Lee	laura.lee@example.com	1	58000.00	2019-12-19	8	Support	8
9	9	David	Garcia	david.garcia@example.com	3	71000.00	2016-06-15	9	Legal	6
10	10	Sarah	Martinez		4	50000.00	2023-01-25	10	Research	3
11	11	John	Doe	john.doe@example.com	1	60000.00	2019-05-01	11	HR	1
12	12	Emily	Brown		1	55000.00	2021-07-22	12	Finance	NULL
13	13	Jane	Smith	jane.smith@example.com	2	75000.00	2020-01-15	13	IT	9
14	14	David	Garcia	david.garcia@example.com	3	71000.00	2016-06-15	14	Sales	7
15	15	Robert	Johnson	robert.johnson@example.com	NULL	64000.00	2021-04-12	15		2

Using DISTINC

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

Example from different source:

```sql
SELECT DISTINCT last_name, first_name
FROM CUSTOMERS
ORDER BY last_name;
```

Using COUNT() and GROUP BY

You can use the COUNT function and GROUP BY clause to count and identify duplicate records based on specific columns.

Example:

```sql
/** 1. Identify Duplicates:
Look for duplicate rows in the Departments table.*/
SELECT
    Empl.FirstName,
    Empl.LastName,
    COUNT(*) AS DuplicateRow
FROM
    [Advance Excercise]..Employees2 AS Empl
JOIN
    [Advance Excercise]..Departments AS Dep
ON Empl.EmployeeID = Dep.DepartmentID
GROUP BY
    Empl.FirstName,
    Empl.LastName
HAVING
    COUNT(*) > 1
```

130 %

Results | Messages

	FirstName	LastName	DuplicateRow
1	Emily	Brown	2
2	John	Doe	2
3	David	Garcia	2
4	Jane	Smith	2

This query will return a list of all the duplicate records in the CUSTOMERS table. In general, to identify sets of values that are duplicated, follow the steps given below.

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

Deleting Duplicate Rows

Is it advisable to remove or delete duplicate data? The answer from most profession is: When's it necessary? Duplicate data can skew prediction results. Thus, for columns that should contain unique values, it's important to search for and exclude any duplicate rows to achieve a more general and accurate prediction

Example of deleting procedure using SQL data:

```sql
WITH EmplDuplicateRow AS (
    SELECT
        EmployeeID,
        ROW_NUMBER() OVER(PARTITION BY FirstName
                    ORDER BY EmployeeID) AS DuplicateRows
    FROM
        [Advance Excercise]..Employees2
)
DELETE FROM [Advance Excercise]..Employees2
WHERE EmployeeID IN (
    SELECT EmployeeID
    FROM EmplDuplicateRow
    WHERE DuplicateRows > 1
)
```

```
(4 rows affected)

Completion time: 2024-07-28T19:54:29.6915119+08:00
```

```sql
WITH DepDuplicateRow AS (
    SELECT
        DepartmentID,
        ROW_NUMBER() OVER(PARTITION BY DepartmentName
                    ORDER BY DepartmentID) AS DuplicateRows
    FROM
        [Advance Excercise]..Departments
)
DELETE FROM [Advance Excercise]..Departments
WHERE DepartmentID IN (
    SELECT DepartmentID
    FROM DepDuplicateRow
    WHERE DuplicateRows > 1
)
```

```
(4 rows affected)

Completion time: 2024-07-28T19:55:34.6520700+08:00
```

This query consists of two parts, each handling the removal of duplicate rows from two different tables: Employees2 and Departments

- **Common Table Expressions (CTEs)**:
  - The query uses Common Table Expressions (CTEs) to identify duplicate rows in each table.
  - The CTE EmplDuplicateRow identifies duplicates in the Employees2 table.
  - The CTE DepDuplicateRow identifies duplicates in the Departments table.
- **ROW_NUMBER() Function**:
  - The ROW_NUMBER() function is used to assign a unique number to each row within a partition of rows that have the same FirstName in Employees2 and the same DepartmentName in Departments.
  - The PARTITION BY clause groups rows by the specified columns (FirstName and DepartmentName).
  - The ORDER BY clause determines the order of the rows within each partition, based on EmployeeID and DepartmentID, respectively.

- **Filtering Duplicates**:
  - In each CTE, rows that have a ROW_NUMBER() value greater than 1 (DuplicateRows > 1) are considered duplicates.
  - The DuplicateRows column identifies which rows are duplicates within their respective partitions.
- **Deleting Duplicates**:
  - The main DELETE statements remove rows from the original tables (Employees2 and Departments).
  - The DELETE operations use WHERE clauses that match EmployeeID and DepartmentID values identified as duplicates in the CTEs.

Breaking Down the Delete Query

Using CTE to break down code and find duplicate rows using ROW_NUMBER() Function.

```
WITH EmplDuplicateRow AS (
   SELECT
      EmployeeID,
      ROW_NUMBER() OVER(PARTITION BY FirstName ORDER BY EmployeeID) AS DuplicateRows
   FROM
      [Advance Excercise]..Employees2
)
```

- Assigns a row number to each row partitioned by FirstName and ordered by EmployeeID.

Delete query

```
DELETE FROM [Advance Excercise]..Employees2
WHERE EmployeeID IN (
   SELECT EmployeeID
   FROM EmplDuplicateRow
   WHERE DuplicateRows > 1
)
```

- Deletes rows from Employees2 where DuplicateRows is greater than 1.

Using CTE to break down code and find duplicate rows using ROW_NUMBER() Function.

```
WITH DepDuplicateRow AS (
   SELECT
      DepartmentID,
      ROW_NUMBER() OVER(PARTITION BY DepartmentName ORDER BY DepartmentID) AS DuplicateRows
   FROM
      [Advance Excercise]..Departments
)
```

- Assigns a row number to each row partitioned by DepartmentName and ordered by DepartmentID.

Delete query

```
DELETE FROM [Advance Excercise]..Departments
WHERE DepartmentID IN (
   SELECT DepartmentID
   FROM DepDuplicateRow
   WHERE DuplicateRows > 1
)
```

- Deletes rows from Departments where DuplicateRows is greater than 1.

Understanding ROW_NUMBER(), OVER(), PARTITION BY And ORDER BY to Find Duplicate Rows in SQL Window Function

ROW_NUMBER():
- The ROW_NUMBER() function assigns a unique number to each row within each partition.
- The first occurrence of each duplicate combination will have ROW_NUMBER() = 1, the second occurrence will have ROW_NUMBER() = 2, and so on.

OVER()
The OVER() clause is used in SQL with window functions like ROW_NUMBER(), RANK(), DENSE_RANK(), and others. It defines the set of rows (the "window") over which the function operates.

PARTITION BY
PARTITION BY within the OVER() clause is used to divide the result set into partitions (groups) based on the values in one or more columns. The window function is then applied separately to each partition.

ORDER BY
ORDER BY within the OVER() clause specifies the order in which the rows are processed within each partition. This ordering determines how the window function assigns values to the rows.

Bonus: Calculating the Total Number of Employees per Department Using COUNT() and ROW_NUMBER()

```sql
-- calculating the total number of employees per department
WITH TotalNumEmployeePerDep AS (
    SELECT --DISTINCT
        Dep.DepartmentName [Department Name],
        COUNT(Empl.EmployeeID) OVER(PARTITION BY Dep.DepartmentName)
                                    [Total Num. Employees Per Dept]
    FROM
        dbo.Employees2 Empl
    JOIN
        dbo.Departments Dep
    ON Empl.EmployeeID = Dep.DepartmentID
),

FinalTotalEmployee AS (
    SELECT
        [Department Name],
        [Total Num. Employees Per Dept],
        ROW_NUMBER() OVER(PARTITION BY [Department Name] -- using row_number()
                    ORDER BY [Total Num. Employees Per Dept]) AS RowNum
    FROM TotalNumEmployeePerDep
)

SELECT [Department Name],
    [Total Num. Employees Per Dept]
FROM FinalTotalEmployee
WHERE RowNum = 1
```

Results | Messages

	Department Name	Total Num. Employees Per Dept
1	Finance	1
2	HR	1
3	IT	1
4	Legal	1
5	Marketing	1
6	Operations	1
7	Research	1
8	Sales	1
9	Support	1
10	Unassigned	2

References:

- https://www.tutorialspoint.com/mysql/mysql-handling-duplicates.htm
- https://chatgpt.com