Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 131 (2018) 691–698

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Congress of Information and Communication Technology (ICICT-2018)

# The comparison of Storage space of the two Methods used in SQL Server to Remove Duplicate records

Li Yue *

*Shandong Polytechnic, Jinan 250104, China*

**Abstract**

In relational database management SQL SERVER 2012, we find that there are some duplicate records when insert to table. Why? It's that we don't establish Primary Key or cluster-index for the table .This paper gives you two methods to remove duplicate records and compare the storage space of the two methods.



*Keywords:* SQL SERVER, Duplicate records, Primary Key, Identity Column, Temp Table, Cursor, Fix-length, Variable-length

## 1. The question that frequently comes up

To beginners， we first create a table and then insert records into table by using Transact SQL language. We always find that there are duplicate records in table. Oh, yes, we forget to make primary key or cluster-index. But, it's wrong when we establish primary key now because of the duplicate records. What shall we do? It's a good idea to remove the duplicate records. There are two methods below to deal with the question. But which one of the two methods using storage space is less? This paper will give the answer.

First let's create a sample table and insert 2 records. If we carelessly press down the button to execute these statements twice, there will be duplicate records. The table below named 'student' is established with six columns.

```
CREATE TABLE student
(   Stu_number CHAR(6) NOT  NULL,   --(6byte)
    Name CHAR(10)  NOT NULL,        --(10byte)
    Specialty CHAR(20) NOT NULL,    -- (20byte)
```

* Corresponding author. Tel.: +86-186-6375-6321
*E-mail address:jinantdxyyueli@163.com*


10.1016/j.procs.2018.04.313

```
    Gender BIT NOT NULL DEFAULT 1, -- (1bit)              /*NOTE: 1 male,0 female    */
    Birthday SMALLDATETIME NOT NULL,   --(4byte)
    Total_credits TINYINT NULL      --(4byte)
    )
GO
```

Below query statements should be execute twice or even more to make duplicate records.

```
INSERT INTO student
VALUES ('001101','Mary','Computer', 0,'2/10/1997 0:0:0', 50)
INSERT INTO student
VALUES ('001102','Kim',' Computer', 1,'6/21/1997 0:0:0', 50)
```

Now duplicate records appear, let's state the two methods.

**Nomenclature**

| | |
|---|---|
| A | Num_Rows |
| B | Num_Cols |
| C | Fixed_Data_Size |
| D | Variable_Data_Size |
| E | Num_Variable_Cols |
| F | Max_Var_Size |
| G | Null_Bitmap |
| H | Row_Size |
| I | Rows_Per_Page |
| J | Num_Pages |
| K | Heap size |

## 2. The First Method of Remove Duplicate Records: Local Temp Table

### *2.1. The First Method of Remove Duplicate Records: Local Temp Table*

There are two temporary tables in SQL Server, one is global temporary table, and the other is local temporary table. Their scope is different. The scope of the global temporary table is the entire process, but the scope of the local temporary table is the process where it is in. The two kinds of temporary tables are automatically deleted when they go out of scope. The table name of the global temporary table is prefixed with two' #', the table name of the local temporary table is prefixed with one' #'.Here we use local temp table, it is stored in system database tempDB.

We can use query statement with 'DISTINCT' keyword. The 'DISTINCT' keyword can eliminate duplicate rows from the results of the query statement. If no 'DISTINCT 'is specified, all rows are returned, including duplicate rows. We insert the result set into local temp table, then the duplicate records disappear. We empty the table and then insert record to table once more from local temp table. At last we drop temp table. Here is the query.

```
SELECT DISTINCT * INTO #tmp FROM student
TRUNCATE TABLE student
INSERT student SELECT * FROM #tmp
DROP TABLE #tmp
GO
```

### *2.2. But how many storage space of the first method is needed?*

The temp table's storage space is necessary. The method of estimating the size of a temporary table is the same as that of the heap size. Why? Because without primary key or cluster index the table is heap. The step of estimating the size of the heap is presented in 'Education and Teaching Forum' 2013 issue 5. The size of the heap is determined by the number of the records. The more records the more space.

Let us suppose: there are 1,000,000 duplicate rows in the table 'student'. Then the heap's storage space is necessary, let's calculate below:

(1) Let us presume there are 1,000,000 records in the table named 'student':
*Num_Rows* = 1,000,000

(2) Calculate the number and the storage space of the fixed-length columns, calculate the number and the storage space of the variable-length columns.
*Num_Cols* = 6 column
*Fixed_Data_Size* = 6+10+20+1+4+4=45(byte)
*Num_Variable_Cols* =0,
*Max_Var_Size* = 0.
Note: because there isn't no variable-length columns in table, so the value of *Max_Var_Size* is set to 0.

(3) Calculate the size of the null bitmap which is used to manage column's nullability.
*Null_Bitmap* = 2 + ((*6* + 7) / 8) =3
Reserve the integer part of the calculation.

(4) Calculate the size of variable-length column:
Because there is no variable-length column, so set *Variable_Data_Size* to 0.

(5) Calculate the storage space of the row:
*Row_Size* = *Fixed_Data_Size* + *Variable_Data_Size* + *Null_Bitmap* + 4 =45+0+3+4=52(byte)
The value 4 in the formula is the row header overhead of the data row.

(6) How many rows are there in one page? There are 8096 bytes per page, and one row need 52 byte, we provide two bytes of extra space for each row to identity the row. So we can use the below formula to calculate the number of the row per page.
*Rows_Per_Page* = 8096 / (*Row_Size* + 2) = 8096 / (*52* + 2) =149(row)
Because one row can't be store in two pages, the result of the above formula only keeps the integer.

(7) How many pages are needed to contain all the rows? We can use the total number of rows divide the number of rows per page, then we receive the number of pages needed.
*Num_Pages* = *Num_Rows* / *Rows_Per_Page*=1,000,000/149=6712(page)
The result of the above formula should be rounded up to the nearest integer.

(8) Estimate the storage space of the heap. Note: there are 8192 bytes per page. We can use the number of pages multiply the value of 8192 that is in on page.
Heap size = 8192 ×*Num_Pages*=8192×6712=54984704 (bytes) ≈53MB

In this method of removing duplicate records, the heap's storage space is about 53MB. That means when database management system executing this operation, it needs 53 MB space further.

## 3. The Second Method of Remove Duplicate Records: Identity Column

### *3.1. The Second Method of Remove Duplicate Records: Identity Column*

First let's learn how to define identity column. The key word 'IDENTITY' must be used in query CREATE TABLE or ALTER TABLE such as IDENTITY [(seed, increment)].

Here we can think the values of the identity column as an arithmetic progression. Seed is the first values of the arithmetic progression; the "increment" is the tolerance of the arithmetic progression. The values of the identity column are calculated by database management system automatically. If we don't specify the values of seed and increment, the database management system will take (1, 1) as default. Only one identity column can be created in one table.

If there are duplicate records in table, we can add an identity column to guarantee the row uniqueness. Then we remove duplicate record, last we can drop the identity column. When there is no duplicate row in table, we can use technologies of Primary Key and Unique index to guarantee the table's uniqueness of the values. The technologies of Primary Key and Unique index are Mandatory.

As above we suppose: there are 1,000,000 duplicate rows in the table 'student'. So the identity column should choose 'int' data type. It can count from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647).Notice:2,147,483,648<1,000,000<2,147,483,647.The query statement is below:

```
ALTER TABLE student
ADD Key_Col INT NOT NULL IDENTITY (1,1)
/*add an identity column*/
GO
/* removing duplicate rows is below */
DELETE FROM student
WHERE EXISTS
  (SELECT *
   FROM student AS T2
   WHERE T2.Stu_number=student. Stu_number
   AND T2.Key_Col<student.Key_Col)
/*the operator '<'can be replace by operator '>', it means to remove the previous record*/
GO
ALTER TABLE student
DROP COLUMN Key_Col
/* drop identity column */
    GO
```

### *3.2. But how many storage space of the second method is needed?*

The space of the column is necessary. We can't estimate the column's space, but we can calculate the space of the table containing the identity column, and then compare the space comprising identity column and not comprising identity column. Now let's calculate the space of the table containing identity column. Although we add a column to the table, the table is still a heap because there is no primary key or clustered index.

(1) Let us presume there are 1,000,000 records in the table named 'student':
*Num_Rows* = 1,000,000

(2) Calculate the number and the storage space of the fixed-length columns, calculate the number and the storage space of the variable-length columns.
*Num_Cols* = 7 column
*Fixed_Data_Size* = 6+10+20+1+4+4+4=49(byte)
*Num_Variable_Cols* =0,
*Max_Var_Size* = 0.
Note:because there isn't no variable-length columns in table, so the value of *Max_Var_Size* is set to 0.

(3) Calculate the size of the null bitmap which is used to manage column's nullability.
*Null_Bitmap* = 2 + ((Num_Cols+7)/8)
=2 + ((*7* +7) / 8) =3
Reserve the integer part of the calculation.

(4) Calculate the variable-length data size:
Because there are no variable-length columns, so set *Variable_Data_Size* to 0.

(5) Calculate the storage space of the row:
*Row_Size* = *Fixed_Data_Size* + *Variable_Data_Size* + *Null_Bitmap* + 4
=49+0+3+4=56(byte)
The value 4 in the formula is the row header overhead of the data row.

(6) How many rows are there in one page? There are 8096 bytes per page, and one row need 56 byte, we provide two bytes of extra space for each row to identity the row. So we can use the below formula to calculate the number of the row per page.
*Rows_Per_Page* = 8096 / (*Row_Size* + 2) = 8096 / (*56* + 2) =139(row)
Because one row can't be store in two pages, the result of the above formula only keeps the integer.

(7) How many pages are needed to contain all the rows? We can use the total number of rows divide the number of rows per page, then we receive the number of pages needed.
*Num_Pages* = *Num_Rows* / *Rows_Per_Page*=1,000,000/139=7195(page)
The result of the above formula should be rounded up to the nearest integer.

(8) Estimate the storage space of the heap. Note: there are 8192 bytes per page. We can use the number of pages multiply the value of 8192 that is in on page.
Heap size = 8192 ×*Num_Pages*=8192×7195=58941440 (bytes) ≈57MB
Increment size=57-53=4MB

That means adding an identity column the database system needs 4MB space additional.

## 4. Compare of Storage space of the two Methods

From above steps, let's compare the storage space that is used in the two methods. In the first method Database system needs the storage space as much as the table. Database system usually create a local temp table in system Database 'tempDB'.When we commit the transaction the storage space of the local temp table is freed.

In the second method Database system needs the storage space as much as the identity column. Database system create an identity column in the table. When we commit the transaction of remove duplicate records the storage space of the identity column is freed.

So, the storage space that is needed to remove duplicate records is temp, but the space that is used in second method is less than that in the first methods. The identity column is a special technology in SQL Server Database system. It can be uniquely identify each row in the table.

But the most important thing is to make primary key or unique index, because they can guarantee that there is no duplicate records in the table. The primary key and the unique index key always contain one column or combination of columns, the value of the key is unique in one table. If there are the same values in the key, database system will stop and warn the user. If we don't build a primary key, the duplicate record appears. So the primary key (or unique index) is important.

## 5. Discussion

This part we discuss the other method removing duplicate records. In this method the technology cursor is used. Then let's talk about the fixed-length data type and variable-length data type.

### *5.1. Use cursor to Remove Duplicate Records*

In SQL Server management system the query statements always produce a result set. But more application program especially T-SQL embedded in the main language(such as C、VB、PowerBuilder and other development tool) can't deal with the result set as a unit, these applications requires a mechanism to ensure that one or a few records in the result set was deal with every time—this is cursor.

Below is the SQL statement using cursor to remove duplicate records.

```
DECALRE @stu_number  as char(6)
DECLARE @CountNumber as int
DECLARE my_cursor cursor for
   SELECT Stu_number,count(*) AS COUNT
   FROM student
   GROUP BY Stu_number
   HAVING count (*)>1
```

```
OPEN my_cursor
FETCH next FROM my_cursor into @stu_number, @CountNumber
  WHILE @@fetch_status=0
/*--- If the extract data from the cursor operation is correct, then continue the loop execution ---*/
  BEGIN
    SET @CountNumber=@CountNumber-1
/*---The total of duplicate records minus 1---*/
    SET rowcount @CountNumber
/*---the number of times of loop statement is the duplicate records minus 1 ---*/
    DELETE FROM XS WHERE Stu_number =@stu_number
/*--- Remove duplicate records---*/
    FETCH next FROM my_cursor INTO @stu_number, @CountNumber
/*---Fetch the next record from cursor ---*/
  END
CLOSE mycursor
DEALLOCATE mycursor
```

Due to limited space here we only give the SQL statement, if you want to know how many storage space is used in this method, you can try it by yourself as a discussion.

### *5.2. Fixed-length column*

The length of the column is fixed or variable is determined by the data type of the column. Some system data types, the length of the columns is fixed, for example：

*5.2.1 Exact numeric.* Exact numeric data type is presented below in the table1, the storage space is also display as a column in the table.

Table 1. The storage space of exact numeric data type.

| Datatype | Storage space |
|---|---|
| Bigint | 8byte |
| Int | 4byte |
| Small int | 2byte |
| Tinyint | 1byte |
| Money | 8byte |
| Smallmoney | 4byte |
| Bit | 8bit=1byte |
| Decimal | Depending on precision |
| Numeric | Depending on precision |

Data type decimal [(p [, s])] and numeric [(p [, s])] is marked, the ‘p’ is the precision. It represents the total number of decimal digits that can be stored. We can estimate the storage space according to the precision value p such as table 2.

Table 2. The storage space of different precision ‘p’.

| Precision | Storage space |
|---|---|
| 1≤p≤9 | 5byte |
| 10≤p≤19 | 9 byte |
| 20≤p≤28 | 13 byte |
| 29≤p≤38 | 17 byte |

*5.2.2 Approximate numeric.* Approximate numeric data type is presented below in the table3, the storage space is also display as a column in the table.

Table 3. The storage space of approximate numeric data type.

| Data type | Storage space |
|---|---|
| Float[(n)] | Determined by value 'n' |
| Real | 4 byte |

The value 'n' is used to store the mantissa of the float. If we specify the value 'n' we can know the storage space such as table 4.

Table 4. The storage space of different 'n'.

| n | precision | Storage space |
|---|---|---|
| 1-24 | 7 digits | 4byte |
| 25-53 | 15 digits | 8 byte |

*5.2.3Date and time.* Date and time data type is presented below in the table5, the storage space is also display as a column in the table.

Table 5. The storage space of date and time data type.

| Datatype | Storage space |
|---|---|
| Date | 3byte |
| Time | 5byte |
| Datetime | 8byte |
| Smalldatetime | 8byte |
| Datetime2 | 8byte（max） |
| Datetimeoffset | 10byte |

*5.2.4Character.*The character data type- char [(n)] is the fixed length. The value 'n' represents the length of the string. Its storage space is n byte.
*5.2.5Unicode character.* The Unicode character data type-nchar [(n)] is the fixed length. Double the value 'n' represents the length of the string. Its storage space is 2n byte.
*5.2.6 Binary strings.* The binary strings data type –binary [(n)] is the fixed length. The value 'n' represents the length of the string. Its storage space is n byte.

*5.3. Variable-length column*

The system data type such as varchar, nvarchar, varbinary is variable-length.
Note: the storage space of the variable-length column is the actual length of the string plus 2 byte.

**Acknowledgements**

This paper is sponsored by Project of Shandong Province Higher Educational Science and Technology Program (The Study of Relational Database Architecture Refine Based on Storage space).The project number is J16LN71.

**References**

1. SQL SERVER Books Online.
2. Li Yue. "Estimate the Size of a Heap in SQL SERVER" .Education and Technology Forum, 2013, 5 (22), 146-147.

3. Li Yue. 2014. ICMRA Relational Database Architecture Refine Based on the Storage Space Estimate. In Proceeding of the 2014 2nd International Conference on Mechatronics, Robotics and Automation.647-652.
4. Li Yue. 2014. ICEEP The Key Factors of Mathematical Formula Affecting the Size of a Clustered Index. In Proceeding of the 2014 3rd International Conference on Energy and Environmental Protection.2877-2880.