(T14)討論Index  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T14)討論Index  
=======================================================================  
0. Summary

1. SSMS - Create/Delete Index

-----------

2. Query - Index

2.1. Create Sample Data

2.2. Clustered Index

2.3. Nonclustered Index

2.4. T014\_04\_(Non)UniqueIndex\_IGNORE\_DUP\_KEY

2.5. T014\_05\_GoodAndBadOfIndexes

2.5. T014\_06\_GoodAndBadOfIndexes  
=======================================================================

0. Summary

1.

Tables and Views can create Indexes to improve the performance of the query.

Indexes concept is similar to book index or table of content.

Firtly, look at index, then find out the data address,

and go to that address directly and find the data.

Without Indexes, you have to do Table Scan which means

search from first data row to last data rows.

Table Scan is no good for  the performance of the query.

2.

Types of Indexes in SQL server

2.1. Clustered

2.2. Nonclustered

2.3. Unique

2.4. Filtered

2.5. XML

2.6. Full Text

2.7. Spatial

2.8. Columnstore

2.9. Index with included columns

2.10. Index on computed columns

Here, we only discuss Clustered, Nonclustered, Unique Indexes.

3.

--DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

This will return Erro

You can not use Query to drop Clustered Index

But you can drop Clustered Index in SSMS.

In SSMS, delete the index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete

4.

Clustered Index V.S. NonClustered Index

4.1.

Clustered Index

4.1.1.

One table can only have ONE clustered index.

By default, SQL server will set Primary Key as the clustered index

if there is no clustered index yet at that time.

4.1.2.

A Clustered index is stored with table and

does not need additional disk space.

it determines the storage order of data physically in the table.

-------

4.2.

NonClustered Index

4.2.1.

One table can have many NonClustered Index.

4.2.2.

A Non-Clustered index is in one place and

refer to another place which stores data physically.

Because it need to refer back to the table,

Clustered index is slightly faster than a non clustered index.

-------

4.3.

A composite index is an index on two or more columns.

If you select ColmnA and ColumnB and

Both ColmnA and ColumnB are in the composite IndexA.

Then this is a covering query by the IndexA.

In this case, the data can simply be returned from the composite IndexA.

A Clustered Index always covers a query,

because it contains all data in a table.

5.

5.1.

Good at Index:

If we have Index in ColumnA,

Index is good for

WHERE, WHERE with DELETE/UPDATE, ORDER BY, GROUP BY in ColumnA

5.2.

Bad at Index:

5.2.1.

NonClustered Index need additional disk space.

5.2.2.

When there are a lof of data in the table,

then DELETE or UPDATE performace might be bad.

Because it need extra time to update Indexes.

6.

UNIQUE is a property which can be assigned to

both CLUSTERED and NON-CLUSTERED indexes.

UNIQUE property ensure there are no duplicate data.

E.g.

If the index has UNIQUE property which

contains 2 columns, LastName and FirstName.

That means UNIQUE property ensures

there are no two enties has the same LastName and FirstName.

By default of SQL server,

6.1.

When you create a PRIMARY KEY constraint

which automatically creates a unique clustered index.

6.2.

Add new UNIQUE CONSTRAINT will automatically add  UNIQUE NONCLUSTERED INDEX

Drop the  UNIQUE CONSTRAINT will automatically drop  UNIQUE NONCLUSTERED INDEX.

6.3.

if there are duplicate values in the Email column,

then you will have to do something

and ensure there is no duplicate values

before you set a UNIQUE constraint to Email Column.

7.

Discuss --WITH IGNORE\_DUP\_KEY;

--CREATE UNIQUE INDEX IX\_Gamer\_DepartmentID

--ON Gamer(DepartmentID)

--WITH IGNORE\_DUP\_KEY;

when ColumnA have a unique index or constraint,

then ColumnA ensures there is no duplicate data.

E.g.

If I try to insert 5 data rows,

but there are 2 data rows contain duplicates.

Then all 5 data rows will be rejected.

--WITH IGNORE\_DUP\_KEY;

In this case, it allow to ignore thoese 2 duplicate rows.

and only insert the rest 3 data rows.

8.

8.1.

Create Nonclustered Index Syntax1:

--CREATE INDEX IX\_TableName\_ColumnName

--ON TableName (ColumnName);

E.g.

--CREATE INDEX IX\_Gamer\_GameScore

--ON Gamer (GameScore ASC);

Create Nonclustered Index on GameScore

8.2.

Create Nonclustered Index Syntax2:

--CREATE NONCLUSTERED INDEX IX\_TableName\_ColumnName

--ON TableName (ColumnName);

E.g.

--CREATE NONCLUSTERED INDEX IX\_Gamer\_Email

--ON Gamer(Email);

Create Nonclustered Index on Email

9.

--sys.sp\_helpconstraint   V.S.   sys.sp\_helpindex

9.1.

--EXECUTE sys.sp\_helpconstraint @objname = N'TableName';

Get the constraint information of the Table

9.2.

--EXECUTE sys.sp\_helpindex @objname = N'TableName';

Or

--EXEC sp\_helpindex N'TableName'

Get the Index information of the Table

E.g.

EXEC sp\_helpindex N'Gamer';

==================================================

1. SSMS - Create/Delete Index

Create a new Index

Database Name --> Table Name --> Indexes

New Index --> **Non-Clustered Index**

Index Name

**IX\_Employee\_AnnualSalary**

-->

Add

--> Select the Column Name

--> OK

--> OK

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

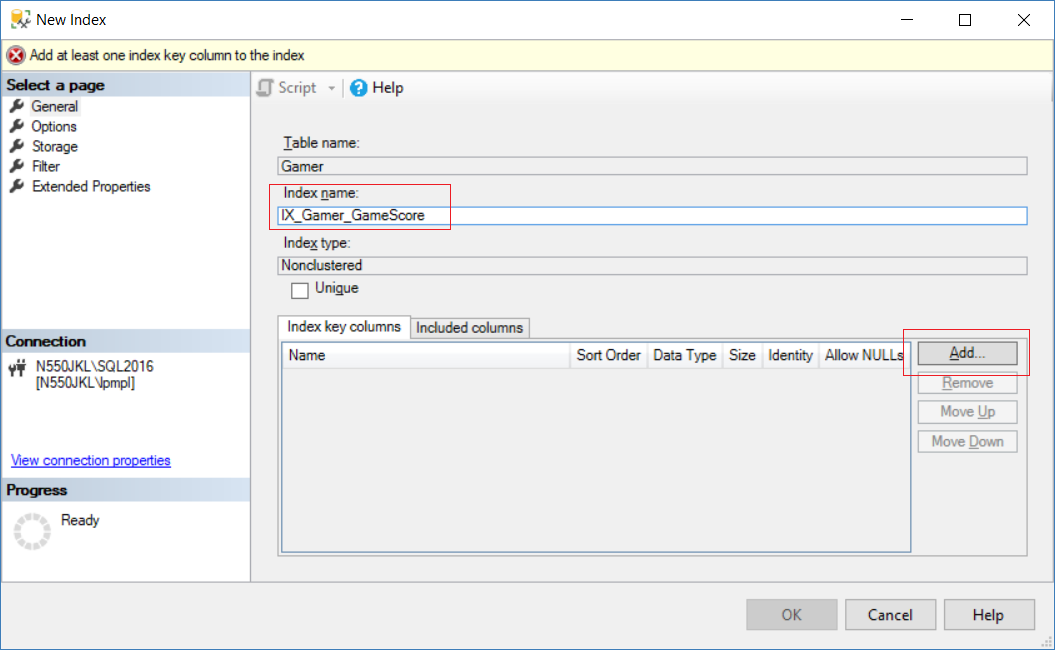
Delete/Disable an Index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete or Disable

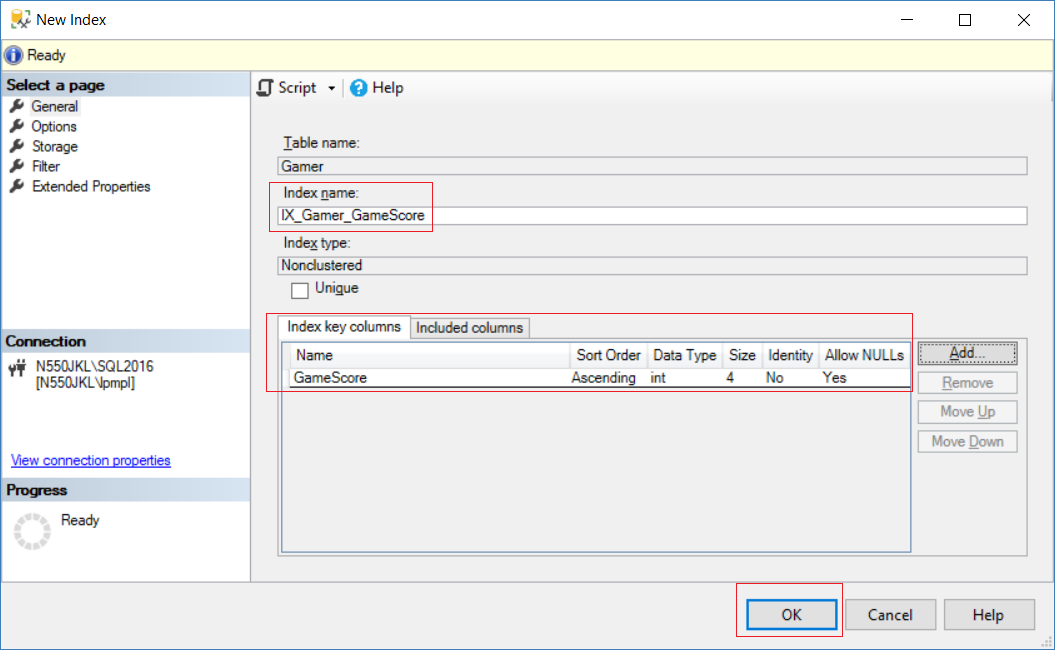
Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

2. Query - Index

2.1. Create Sample Data

--========================================================================

--T014\_01\_Create Sample Data

--========================================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer

(

  Id INT PRIMARY KEY ,

  FirstName NVARCHAR(100) ,

  LastName NVARCHAR(100) ,

  Email NVARCHAR(100) ,

  Gender NVARCHAR(10) ,

  GameScore INT

);

GO -- Run the previous command and begins new batch

INSERT  INTO Gamer

VALUES  ( 4, 'First4', 'Last4', '[4@4.com](mailto:4@4.com)', 'Male', 43000 );

INSERT  INTO Gamer

VALUES  ( 2, 'First2', 'Last2', '[2@2.com](mailto:2@2.com)', 'Female', 44000 );

INSERT  INTO Gamer

VALUES  ( 1, 'First1', 'Last1', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

INSERT  INTO Gamer

VALUES  ( 5, 'First5', 'Last5', '[5@5.com](mailto:5@5.com)', 'Male', 42000 );

INSERT  INTO Gamer

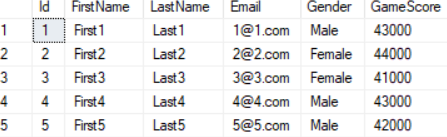
VALUES  ( 3, 'First3', 'Last3', '[3@3.com](mailto:3@3.com)', 'Female', 41000 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    Gamer;

GO -- Run the previous command and begins new batch



2.2. Clustered Index

--========================================================================

--T014\_02\_Clustered Index

--========================================================================

/\*

1.

Tables and Views can create Indexes to improve the performance of the query.

Indexes concept is similar to book index or table of content.

Firtly, look at index, then find out the data address,

and go to that address directly and find the data.

Without Indexes, you have to do Table Scan which means

search from first data row to last data rows.

Table Scan is no good for  the performance of the query.

2.

Types of Indexes in SQL server

2.1. Clustered

2.2. Nonclustered

2.3. Unique

2.4. Filtered

2.5. XML

2.6. Full Text

2.7. Spatial

2.8. Columnstore

2.9. Index with included columns

2.10. Index on computed columns

Here, we only discuss Clustered, Nonclustered, Unique Indexes.

3.

In SSMS, delete the index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete

You are not allowed to delete index by query,

--DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

but you may delete the index in SSMS.

4.

Clustered Index V.S. NonClustered Index

4.1.

Clustered Index

4.1.1.

One table can only have ONE clustered index.

By default, SQL server will set primary key Column as the clustered index

if there is no clustered index yet at that time.

4.1.2.

A Clustered index is stored with table and

does not need additional disk space.

it determines the storage order of data physically in the table.

-------

4.2.

NonClustered Index

4.2.1.

One table can have many NonClustered Index.

4.2.2.

A Non-Clustered index is in one place and

refer to another place which stores data physically.

Because it need to refer back to the table,

Clustered index is slightly faster than a non clustered index.

-------

4.3.

A composite index is an index on two or more columns.

If you select ColmnA and ColumnB and

Both ColmnA and ColumnB are in the composite IndexA.

Then this is a covering query by the IndexA.

In this case, the data can simply be returned from the composite IndexA.

A Clustered Index always covers a query,

because it contains all data in a table.

\*/

--========================================================================

--T014\_02\_01

--By default, SQL server will set Primary Key as the clustered index

SELECT  \*

FROM    Gamer;

/\*

1.

Clustered Index

1.1.

One table can only have ONE clustered index.

By default, SQL server will set Primary Key as the clustered index

if there is no clustered index yet at that time.

1.2.

A Clustered index is stored with table and

does not need additional disk space.

it determines the storage order of data physically in the table.

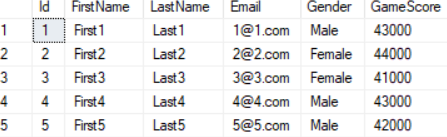
1.3.

--SELECT  \*

--FROM    Gamer;

It will order by Clustered index

\*/



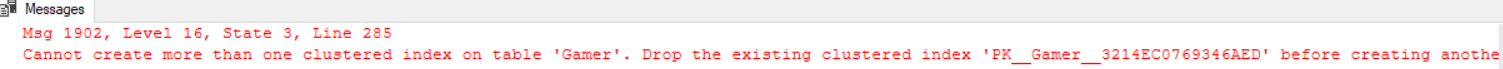
--========================================================================

--T014\_02\_02

--Create Clustered Index

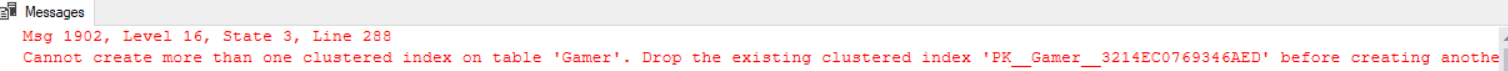
CREATE CLUSTERED INDEX IX\_Gamer\_Name

ON Gamer(FirstName);



CREATE CLUSTERED INDEX IX\_Gamer\_Name\_Gender

ON Gamer(FirstName DESC, Gender ASC);



GO -- Run the prvious command and begins new batch

/\*

1.

--CREATE CLUSTERED INDEX IX\_Gamer\_Name

--ON Gamer(FirstName);

--CREATE CLUSTERED INDEX IX\_Gamer\_Name\_Gender

--ON Gamer(FirstName DESC, Gender ASC);

Both Query will return Error

2.

One table can only have ONE clustered index.

By default, SQL server will set primary key Column as the clustered index

if there is no clustered index yet at that time.

\*/

--========================================================================

--T014\_02\_03

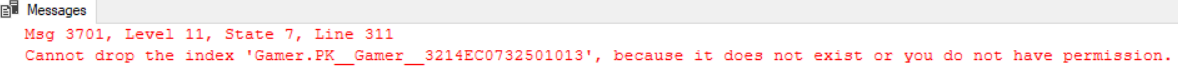
--sp\_helpindex and Drop Clustered Index

EXECUTE sp\_helpindex Gamer;



DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

GO -- Run the prvious command and begins new batch



/\*

1.

1.1.

--EXEC sp\_helpindex N'TableName'

Get the Index of the Table

E.g.

EXEC sp\_helpindex N'Gamer';

1.2.

--EXECUTE sp\_helpindex Gamer;

You will this data row.

--index name :  PK\_\_Gamer\_\_3214EC071E222ACE

--index description  :  clustered, unique, primary key located on PRIMARY

--index\_keys :  Id

2.

2.1.

--DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

will return Error

2.2.

You can not use Query to drop Clustered Index

But you can drop Clustered Index in SSMS

In SSMS, delete the index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete

Then we can delete the index in SSMS.

\*/

===============================================================

2.3. Nonclustered Index

--========================================================================

--T014\_03\_Nonclustered Index

--========================================================================

--========================================================================

--T014\_03\_01

--Create Nonclustered Index

CREATE INDEX IX\_Gamer\_GameScore

ON Gamer (GameScore ASC);

CREATE NONCLUSTERED INDEX IX\_Gamer\_Email

ON Gamer(Email);

GO -- Run the prvious command and begins new batch

Graphical user interface, text, application

Description automatically generated

/\*

1.

1.1.

Create Nonclustered Index Syntax1:

--CREATE INDEX IX\_TableName\_ColumnName

--ON TableName (ColumnName);

E.g.

--CREATE INDEX IX\_Gamer\_GameScore

--ON Gamer (GameScore ASC);

Create Nonclustered Index on GameScore

1.2.

Create Nonclustered Index Syntax2:

--CREATE NONCLUSTERED INDEX IX\_TableName\_ColumnName

--ON TableName (ColumnName);

E.g.

--CREATE NONCLUSTERED INDEX IX\_Gamer\_Email

--ON Gamer(Email);

Create Nonclustered Index on Email

\*/

--========================================================================

--T014\_03\_02

--sp\_helpindex, Drop Index

EXEC sp\_helpindex N'Gamer';

A picture containing table

Description automatically generated

DROP INDEX Gamer.IX\_Gamer\_GameScore;

DROP INDEX Gamer.IX\_Gamer\_Email;

EXEC sp\_helpindex N'Gamer';

GO -- Run the prvious command and begins new batch



/\*

1.

--EXEC sp\_helpindex N'TableName'

Get the Index of the Table

E.g.

EXEC sp\_helpindex N'Gamer';

2.

SSMS - Create/Delete Index

2.1.

Create a new Index

Database Name --> Table Name --> Indexes

New Index --> Non-Clustered Index

Index Name

IX\_Gamer\_GameScore

-->

Add

--> Select the Column Name

--> OK

--> OK

2.2.

Delete/Disable an Index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete or Disable

\*/

2.4. T014\_04\_(Non)UniqueIndex\_IGNORE\_DUP\_KEY

--========================================================================

--T014\_04\_(Non)UniqueIndex\_IGNORE\_DUP\_KEY

--========================================================================

/\*

1.

--DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

will return Error

You can not use Query to drop Clustered Index

But you can drop Clustered Index in SSMS

In SSMS, delete the index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete

Then we can delete the index in SSMS.

2.

UNIQUE is a property which can be assigned to

both CLUSTERED and NON-CLUSTERED indexes.

UNIQUE property ensure there are no duplicate data.

E.g.

If the index has UNIQUE property which

contains 2 columns, LastName and FirstName.

That means UNIQUE property ensures

there are no two enties has the same LastName and FirstName.

3.

By default of SQL server,

3.1.

When you create a PRIMARY KEY constraint

which automatically creates a unique clustered index.

3.2.

Add new UNIQUE CONSTRAINT will automatically add  UNIQUE NONCLUSTERED INDEX

Drop the  UNIQUE CONSTRAINT will automatically drop  UNIQUE NONCLUSTERED INDEX.

4.

if there are duplicate values in the Email column,

then you will have to do something

and ensure there is no duplicate values

before you set a UNIQUE constraint to Email Column.

5.

--CREATE UNIQUE INDEX IX\_Gamer\_DepartmentID

--ON Gamer(DepartmentID)

--WITH IGNORE\_DUP\_KEY;

when ColumnA have a unique index or constraint,

then ColumnA ensures there is no duplicate data.

E.g.

If I try to insert 5 data rows,

but there are 2 data rows contain duplicates.

Then all 5 data rows will be rejected.

--WITH IGNORE\_DUP\_KEY;

In this case, it allow to ignore thoese 2 duplicate rows.

and only insert the rest 3 data rows.

\*/

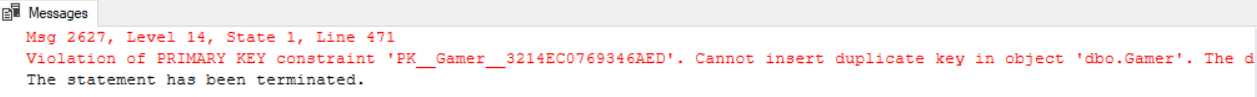
--========================================================================

--T014\_04\_01

--When you create Pimary Key, it automatically creates UNIQUE Index.

INSERT  INTO Gamer

VALUES  ( 1, 'First1A', 'Last1A', '[1A@1.com](mailto:1A@1.com)', 'Male', 43000 );



/\*

1.

--INSERT  INTO Gamer

--VALUES  ( 1, 'First1A', 'Last1A', '[1A@1.com](mailto:1A@1.com)', 'Male', 43000 );

will return Error

When you create Pimary Key, it automatically creates UNIQUE Index.

Thus, you may not enter duplicate data in Primary Key.

2.

UNIQUE is a property which can be assigned to

both CLUSTERED and NON-CLUSTERED indexes.

UNIQUE property ensure there are no duplicate data.

E.g.

If the index has UNIQUE property which

contains 2 columns, LastName and FirstName.

That means UNIQUE property ensures

there are no two enties has the same LastName and FirstName.

3.

By default of SQL server,

3.1.

When you create a PRIMARY KEY constraint

which automatically creates a unique clustered index.

3.2.

Add new UNIQUE CONSTRAINT will automatically add  UNIQUE NONCLUSTERED INDEX

Drop the  UNIQUE CONSTRAINT will automatically drop  UNIQUE NONCLUSTERED INDEX.

\*/

--========================================================================

--T014\_04\_02

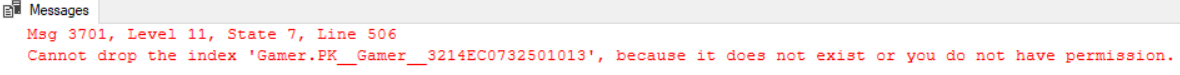
--sp\_helpindex and Drop Clustered Index

EXECUTE sp\_helpindex Gamer;



DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

GO -- Run the prvious command and begins new batch



/\*

1.

1.1.

--EXEC sp\_helpindex N'TableName'

Get the Index of the Table

E.g.

EXEC sp\_helpindex N'Gamer';

1.2.

--EXECUTE sp\_helpindex Gamer;

You will this data row.

--index name :  PK\_\_Gamer\_\_3214EC071E222ACE

--index description  :  clustered, unique, primary key located on PRIMARY

--index\_keys :  Id

2.

2.1.

--DROP INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013;

will return Error

2.2.

You can not use Query to drop Clustered Index

But you can drop Clustered Index in SSMS

In SSMS, delete the index

Database Name --> Table Name --> Indexes

--> Right Click --> Delete

Then we can delete the index in SSMS.

\*/

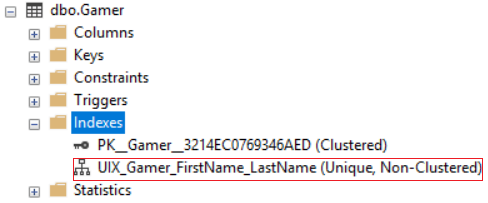
--========================================================================

--T014\_04\_03

--Add new UNIQUE NONCLUSTERED INDEX

CREATE UNIQUE NONCLUSTERED INDEX UIX\_Gamer\_FirstName\_LastName

ON Gamer(FirstName, LastName);



INSERT  INTO Gamer

VALUES  ( 6, 'First1', 'Last1', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

--Return Error



INSERT  INTO Gamer

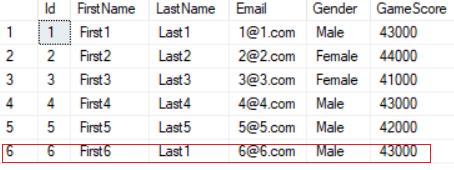
VALUES  ( 6, 'First6', 'Last1', '[6@6.com](mailto:6@6.com)', 'Male', 43000 );

--Insert Successfully

SELECT  \*

FROM    Gamer;

GO -- Run the prvious command and begins new batch



/\*

1.

--CREATE UNIQUE NONCLUSTERED INDEX UIX\_Gamer\_FirstName\_LastName

--ON Gamer(FirstName, LastName);

Create an UNIQUE NONCLUSTERED INDEX for FirstName and LastName columns.

--INSERT  INTO Gamer

--VALUES  ( 6, 'First1', 'Last1', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

'First1', 'Last1' is already existed, thus insert will fail.

--INSERT  INTO Gamer

--VALUES  ( 6, 'First6', 'Last1', '[6@6.com](mailto:6@6.com)', 'Male', 43000 );

'First6', 'Last1' is Not existed, thus insert will be success.

2.

UNIQUE is a property which can be assigned to

both CLUSTERED and NON-CLUSTERED indexes.

UNIQUE property ensure there are no duplicate data.

E.g.

If the index has UNIQUE property which

contains 2 columns, LastName and FirstName.

That means UNIQUE property ensures

there are no two enties has the same LastName and FirstName.

\*/

--========================================================================

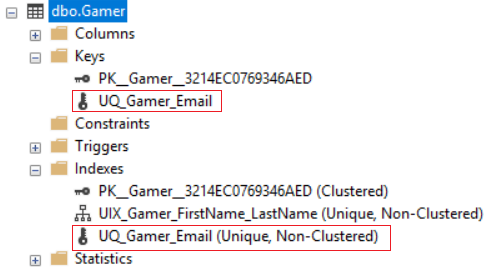
--T014\_04\_04

--Add new UNIQUE CONSTRAINT, automatically add  UNIQUE NONCLUSTERED INDEX

ALTER TABLE Gamer

ADD CONSTRAINT UQ\_Gamer\_Email

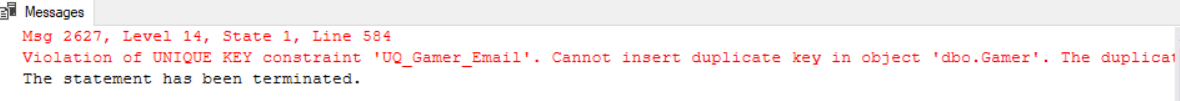
UNIQUE NONCLUSTERED (Email);



INSERT  INTO Gamer

VALUES  ( 7, 'First7', 'Last7', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

--Return Error



INSERT  INTO Gamer

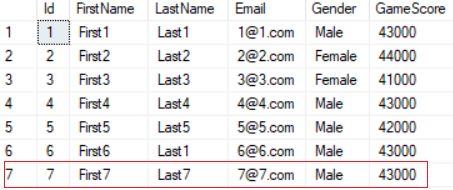
VALUES  ( 7, 'First7', 'Last7', '[7@7.com](mailto:7@7.com)', 'Male', 43000 );

--Insert Successfully

SELECT  \*

FROM    Gamer;

GO -- Run the prvious command and begins new batch



/\*

1.

--ALTER TABLE Gamer

--ADD CONSTRAINT UQ\_Gamer\_Email

--UNIQUE NONCLUSTERED (Email);

Add new UNIQUE CONSTRAINT, automatically add  UNIQUE NONCLUSTERED INDEX

--INSERT  INTO Gamer

--VALUES  ( 7, 'First7', 'Last7', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

'[1@1.com](mailto:1@1.com)' is already existed, thus insert will fail.

--INSERT  INTO Gamer

--VALUES  ( 7, 'First7', 'Last7', '[7@7.com](mailto:7@7.com)', 'Male', 43000 );

'[7@7.com](mailto:7@7.com)' is Not existed, thus insert will be success.

2.

Add new UNIQUE CONSTRAINT will automatically add  UNIQUE NONCLUSTERED INDEX

Drop the  UNIQUE CONSTRAINT will automatically drop  UNIQUE NONCLUSTERED INDEX.

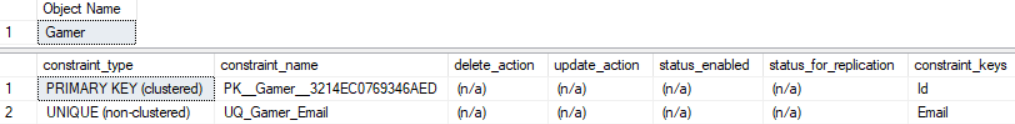
\*/

--========================================================================

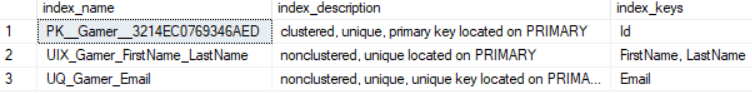
--T014\_04\_05

--sys.sp\_helpconstraint   V.S.   sys.sp\_helpindex

EXECUTE sys.sp\_helpconstraint @objname = N'Gamer';



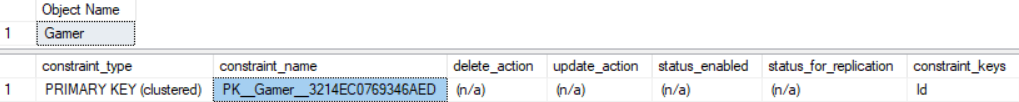
EXECUTE sys.sp\_helpindex @objname = N'Gamer';



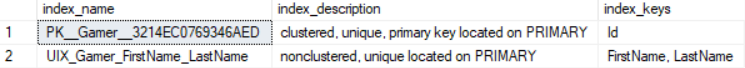
ALTER TABLE Gamer

DROP CONSTRAINT UQ\_Gamer\_Email;

EXECUTE sys.sp\_helpconstraint @objname = N'Gamer';



EXECUTE sys.sp\_helpindex @objname = N'Gamer';



GO -- Run the prvious command and begins new batch

/\*

1.

--sys.sp\_helpconstraint   V.S.   sys.sp\_helpindex

1.1.

--EXECUTE sys.sp\_helpconstraint @objname = N'TableName';

Get the constraint information of the Table

1.2.

--EXECUTE sys.sp\_helpindex @objname = N'TableName';

Or

--EXEC sp\_helpindex N'TableName'

Get the Index information of the Table

E.g.

EXEC sp\_helpindex N'Gamer';

2.

Add new UNIQUE CONSTRAINT will automatically add  UNIQUE NONCLUSTERED INDEX

Drop the  UNIQUE CONSTRAINT will automatically drop  UNIQUE NONCLUSTERED INDEX.

\*/

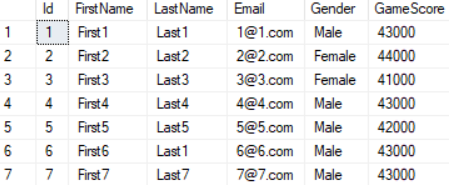
--========================================================================

--T014\_04\_06

--WITH IGNORE\_DUP\_KEY;

SELECT  \*

FROM    Gamer;



--Create UIX\_Gamer\_Email WITH IGNORE\_DUP\_KEY;

CREATE UNIQUE INDEX UIX\_Gamer\_Email

ON Gamer(Email)

WITH IGNORE\_DUP\_KEY;

--Insert 2 rows with Duplicate Email, these 2 rows insert will fail.

INSERT  INTO Gamer

VALUES  ( 8, 'First8', 'Last8', '[1@1.com](mailto:1@1.com)', 'Female', 44000 );

INSERT  INTO Gamer

VALUES  ( 9, 'First9', 'Last9', '[2@2.com](mailto:2@2.com)', 'Male', 43000 );

--Insert 3 rows WITHOUT Duplicate Email, these 2 rows insert will success.

INSERT  INTO Gamer

VALUES  ( 10, 'First10', 'Last10', '[10@10.com](mailto:10@10.com)', 'Female', 44000 );

INSERT  INTO Gamer

VALUES  ( 11, 'First11', 'Last11', '[11@11.com](mailto:11@11.com)', 'Male', 43000 );

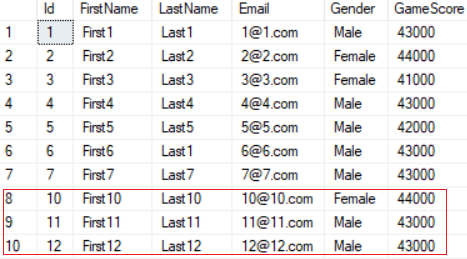
INSERT  INTO Gamer

VALUES  ( 12, 'First12', 'Last12', '[12@12.com](mailto:12@12.com)', 'Male', 43000 );

SELECT  \*

FROM    dbo.Gamer;

GO -- Run the prvious command and begins new batch



/\*

1.

Output as following

--Duplicate key was ignored.

--(0 rows affected)

--Duplicate key was ignored.

--(0 rows affected)

--(1 row affected)

--(1 row affected)

--(1 row affected)

2.

--CREATE UNIQUE INDEX UIX\_Gamer\_Email

--ON Gamer(Email)

--WITH IGNORE\_DUP\_KEY;

when ColumnA have a unique index or constraint,

then ColumnA ensures there is no duplicate data.

E.g.

If I try to insert 5 data rows,

but there are 2 data rows contain duplicates.

Then all 5 data rows will be rejected.

--WITH IGNORE\_DUP\_KEY;

In this case, it allow to ignore thoese 2 duplicate rows.

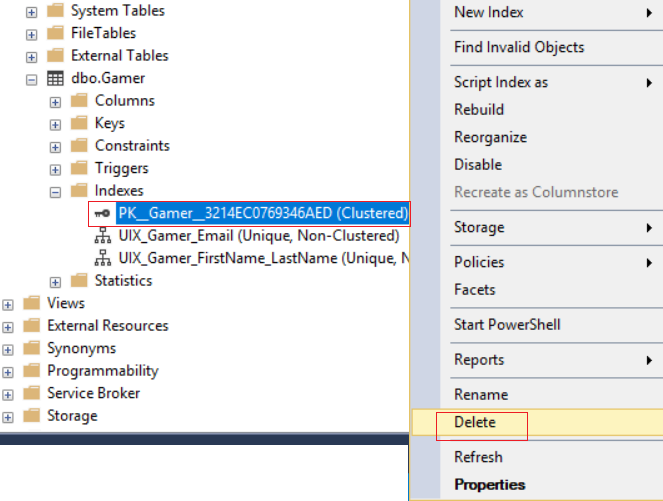
and only insert the rest 3 data rows.

\*/

--========================================================================

--T014\_04\_07

--Delete INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013 in SSMS and insert duplicate data.

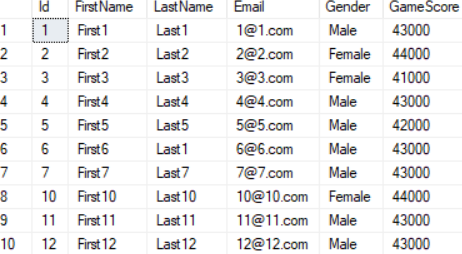


Graphical user interface, text, application

Description automatically generated

SELECT  \*

FROM    dbo.Gamer;



INSERT  INTO Gamer

VALUES  ( 1, 'First1A', 'Last1A', '[1A@1.com](mailto:1A@1.com)', 'Male', 43000 );

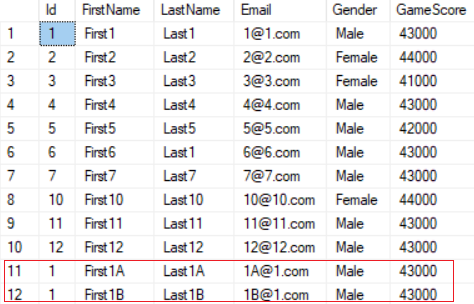
INSERT  INTO Gamer

VALUES  ( 1, 'First1B', 'Last1B', '[1B@1.com](mailto:1B@1.com)', 'Male', 43000 );

SELECT  \*

FROM    dbo.Gamer;

GO -- Run the prvious command and begins new batch



/\*

1.

--INSERT  INTO Gamer

--VALUES  ( 1, 'First1A', 'Last1A', '[1A@1.com](mailto:1A@1.com)', 'Male', 43000 );

will return Error

When you create Pimary Key, it automatically creates UNIQUE Index.

Thus, you may not enter duplicate data in Primary Key.

2.

Now,

Delete INDEX Gamer.PK\_\_Gamer\_\_3214EC0732501013 in SSMS and insert duplicate data.

and insert duplicated Id again.

--INSERT  INTO Gamer

--VALUES  ( 1, 'First1A', 'Last1A', '[1A@1.com](mailto:1A@1.com)', 'Male', 43000 );

--INSERT  INTO Gamer

--VALUES  ( 1, 'First1B', 'Last1B', '[1B@1.com](mailto:1B@1.com)', 'Male', 43000 );

Now, both rows will be inserted successfully.

\*/

2.5. T014\_05\_GoodAndBadOfIndexes

--========================================================================

--T014\_05\_GoodAndBadOfIndexes

--========================================================================

/\*

5.

5.1.

Good at Index:

If we have Index in ColumnA,

Index is good for

WHERE, WHERE with DELETE/UPDATE, ORDER BY, GROUP BY in ColumnA

5.2.

Bad at Index:

5.2.1.

NonClustered Index need additional disk space.

5.2.2.

When there are a lof of data in the table,

then DELETE or UPDATE performace might be bad.

Because it need extra time to update Indexes.

\*/

--========================================================================

--T014\_05\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer

(

  Id INT PRIMARY KEY ,

  FirstName NVARCHAR(100) ,

  LastName NVARCHAR(100) ,

  Email NVARCHAR(100) ,

  Gender NVARCHAR(10) ,

  GameScore INT

);

GO -- Run the previous command and begins new batch

INSERT  INTO Gamer

VALUES  ( 4, 'First4', 'Last4', '[4@4.com](mailto:4@4.com)', 'Male', 43000 );

INSERT  INTO Gamer

VALUES  ( 2, 'First2', 'Last2', '[2@2.com](mailto:2@2.com)', 'Female', 44000 );

INSERT  INTO Gamer

VALUES  ( 1, 'First1', 'Last1', '[1@1.com](mailto:1@1.com)', 'Male', 43000 );

INSERT  INTO Gamer

VALUES  ( 5, 'First5', 'Last5', '[5@5.com](mailto:5@5.com)', 'Male', 42000 );

INSERT  INTO Gamer

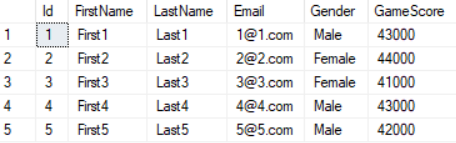
VALUES  ( 3, 'First3', 'Last3', '[3@3.com](mailto:3@3.com)', 'Female', 41000 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    Gamer;

GO -- Run the previous command and begins new batch



--========================================================================

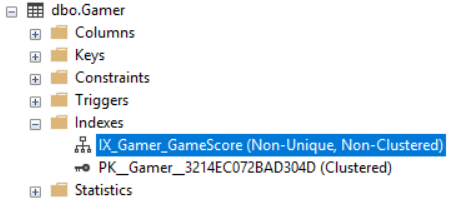
--T014\_05\_02

--Create a Non-Clustered Index on GameScore Column

CREATE NONCLUSTERED INDEX IX\_Gamer\_GameScore

ON Gamer(GameScore ASC);

GO -- Run the previous command and begins new batch



--========================================================================

--T014\_05\_03

--Index is good for WHERE

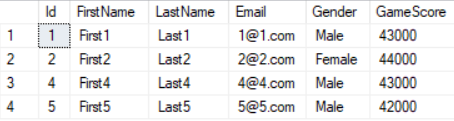
SELECT  \*

FROM    dbo.Gamer

WHERE   GameScore > 41000

        AND GameScore < 48000;

GO -- Run the previous command and begins new batch



/\*

Because GameScore Column has a Non-Clustered Index.

Thus, SQL server doesn't have to search from first row to last row.

SQL server will just look at index and

find out the exact address of the data.

\*/

--========================================================================

--T014\_05\_04

--Index is good for WHERE with Delete or Update

SELECT  \*

FROM    dbo.Gamer

DELETE  FROM Gamer

WHERE   GameScore = 44000;

UPDATE  dbo.Gamer

SET     GameScore = 49000

WHERE   GameScore = 41000;

SELECT  \*

FROM    dbo.Gamer

GO -- Run the previous command and begins new batch

/\*

Because GameScore Column has a Non-Clustered Index.

Thus, SQL server doesn't have to search from first row to last row.

SQL server will just look at index and

find out the exact address of the data.

Then Update or Delete it

\*/

--========================================================================

--T014\_05\_05

--Index is good for ORDER BY

SELECT  \*

FROM    dbo.Gamer

ORDER BY GameScore;

SELECT  \*

FROM    Gamer

ORDER BY GameScore DESC;

GO -- Run the previous command and begins new batch

/\*

Because GameScore Column has a Non-Clustered Index.

Thus, SQL server doesn't have to search from first row to last row.

SQL server will just look at index, then order the rows by using index.

\*/

--========================================================================

--T014\_05\_06

--Index is good for GROUP BY

SELECT  GameScore ,

        COUNT(GameScore) AS TotalGameScore

FROM    dbo.Gamer

GROUP BY GameScore;

/\*

Because GameScore Column has a Non-Clustered Index.

Thus, SQL server doesn't have to search from first row to last row.

SQL server will just look at index, then group the rows by using index.

\*/

2.5. T014\_06\_GoodAndBadOfIndexes

--========================================================================

--T014\_06\_GoodAndBadOfIndexes

--========================================================================

--Clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch