SQL-1-CAS-2

Izmena strukture tabele (neophodno je dodeliti bazu pomoću naredbe USE)

ALTER TABLE

```
ALTER TABLE table
{ [ ALTER COLUMN column_name
{ new_data_type [ ( precision [ , scale ] ) ]
[ COLLATE < collation_name > ]
[ NULL | NOT NULL ]
| {ADD | DROP } ROWGUIDCOL }
]
| ADD
{ [ < column_definition > ]
| column_name AS computed_column_expression
} [ ,...n ]
| [ WITH CHECK | WITH NOCHECK ] ADD
{ < table_constraint > } [ ,...n ]
| DROP
{ [ CONSTRAINT ] constraint_name
| COLUMN column } [ ,...n ]
| { CHECK | NOCHECK } CONSTRAINT
{ ALL | constraint_name [ ,...n ] }
| { ENABLE | DISABLE } TRIGGER
{ ALL | trigger_name [ ,...n ] }
}
< column_definition > ::=
{ column_name data_type }
[ [ DEFAULT constant_expression ] [ WITH VALUES ]
| [ IDENTITY [ (seed , increment ) [ NOT FOR REPLICATION ] ] ]
]
[ ROWGUIDCOL ]
[ COLLATE < collation_name > ]
[ < column_constraint > ] [ ...n ]
< column_constraint > ::=
[ CONSTRAINT constraint_name ]
{ [ NULL | NOT NULL ]
| [ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[ WITH FILLFACTOR = fillfactor ]
[ ON { filegroup | DEFAULT } ]
]
| [ [ FOREIGN KEY ]
REFERENCES ref_table [ ( ref_column ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
]
| CHECK [ NOT FOR REPLICATION ]
( logical_expression )
}
< table_constraint > ::=
[ CONSTRAINT constraint_name ]
```

```
{ [ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
{ ( column [ ,...n ] ) }
[ WITH FILLFACTOR = fillfactor ]
[ ON {filegroup | DEFAULT } ]
]
| FOREIGN KEY
[ ( column [ ,...n ] ) ]
REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
| DEFAULT constant_expression
[ FOR column ] [ WITH VALUES ]
| CHECK [ NOT FOR REPLICATION ]
( search_conditions )
}
```

```
DEFAULT (getdate())
```

Use command:

USE { *database* }

In SQL, we have the following constraints:

NOT NULL - Indicates that a column cannot store NULL value
UNIQUE - Ensures that each row for a column must have a unique value
PRIMARY KEY - A combination of a NOT NULL and UNIQUE.
Ensures that a column (or combination of two or more columns) have an unique identity
which helps to find a particular record in a table more easily and quickly
FOREIGN KEY - Ensure the referential integrity of the data in one table
to match values in another table
CHECK - Ensures that the value in a column meets a specific condition
DEFAULT - Specifies a default value when specified none for this column

```
CREATE TABLE Persons
(
P_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

To allow naming of a UNIQUE constraint, and for defining a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons
(
```

```
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
)
```

To create a UNIQUE constraint on the "P_Id" column when the table is already created,
use the following SQL:

```
ALTER TABLE Persons
ADD UNIQUE (P_Id)
```

To allow naming of a UNIQUE constraint, and for defining a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
```

To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONSTRAINT uc_PersonID
```

DEFAULT:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)

CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
OrderDate date DEFAULT GETDATE()
)

ALTER TABLE Persons
ALTER COLUMN City SET DEFAULT 'SANDNES'

ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT
```

INDEXES:
Indexes allow the database application to find data fast; without reading the whole table.

```
CREATE INDEX index_name
ON table_name (column_name)

CREATE UNIQUE INDEX index_name
```

```
ON table_name (column_name)

DROP:

Indexes, tables, and databases can easily be deleted/removed with the DROP
statement.

DROP INDEX table_name.index_name

DROP TABLE table_name

DROP DATABASE database_name

What if we only want to delete the data inside the table, and not the table
itself?
Then, use the TRUNCATE TABLE statement:

TRUNCATE TABLE table_name


ALTER TABLE
Modifies a table definition by altering, adding, or dropping columns and
constraints,
 reassigning partitions, or disabling or enabling constraints and triggers.

Adding a new column
ALTER TABLE dbo.doc_exa ADD column_b VARCHAR(20) NULL ;

Dropping a column
ALTER TABLE dbo.doc_exb DROP COLUMN column_b ;

Changing the data type of a column
The following example changes a column of a table from INT to DECIMAL.
CREATE TABLE dbo.doc_exy (column_a INT ) ;
GO
INSERT INTO dbo.doc_exy (column_a) VALUES (10) ;
GO
ALTER TABLE dbo.doc_exy ALTER COLUMN column_a DECIMAL (5, 2) ;
GO

Adding a DEFAULT constraint to an existing column
CREATE TABLE dbo.doc_exz ( column_a INT, column_b INT) ;
GO
INSERT INTO dbo.doc_exz (column_a)VALUES ( 7 ) ;
GO
ALTER TABLE dbo.doc_exz
ADD CONSTRAINT col_b_def
DEFAULT 50 FOR column_b ;
GO

Adding several columns with constraints
CREATE TABLE dbo.doc_exe ( column_a INT CONSTRAINT column_a_un UNIQUE) ;
GO
ALTER TABLE dbo.doc_exe ADD

-- Add a PRIMARY KEY identity column.
column_b INT IDENTITY
```

```
CONSTRAINT column_b_pk PRIMARY KEY,

-- Add a column that references another column in the same table.
column_c INT NULL
CONSTRAINT column_c_fk
REFERENCES doc_exe(column_a),


Disabling and re-enabling a constraint
CREATE TABLE dbo.cnst_example
(id INT NOT NULL,
 name VARCHAR(10) NOT NULL,
 salary MONEY NOT NULL
    CONSTRAINT salary_cap CHECK (salary < 100000)
);

-- Valid inserts
INSERT INTO dbo.cnst_example VALUES (1,'Joe Brown',65000);
INSERT INTO dbo.cnst_example VALUES (2,'Mary Smith',75000);

-- This insert violates the constraint.
INSERT INTO dbo.cnst_example VALUES (3,'Pat Jones',105000);

-- Disable the constraint and try again.
ALTER TABLE dbo.cnst_example NOCHECK CONSTRAINT salary_cap;
INSERT INTO dbo.cnst_example VALUES (3,'Pat Jones',105000);

-- Re-enable the constraint and try another insert; this will fail.
ALTER TABLE dbo.cnst_example CHECK CONSTRAINT salary_cap;
INSERT INTO dbo.cnst_example VALUES (4,'Eric James',110000) ;


Alter table command:
```

Modifies a table definition by altering, adding, or dropping columns and constraints, reassigning partitions, or disabling or enabling constraints and triggers.

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name . ] table_name
{
    ALTER COLUMN column_name
    {
        [ type_schema_name. ] type_name [ ( { precision [ , scale ]
            | max | xml_schema_collection } ) ]
        [ COLLATE collation_name ]
        [ NULL | NOT NULL ] [ SPARSE ]
    | {ADD | DROP }
        { ROWGUIDCOL | PERSISTED | NOT FOR REPLICATION | SPARSE }
    }
        | [ WITH { CHECK | NOCHECK } ]

    | ADD
    {
        <column_definition>
      | <computed_column_definition>
      | <table_constraint>
      | <column_set_definition>
```

```
    } [ ,...n ]

    | DROP
    {
        [ CONSTRAINT ] constraint_name
        [ WITH ( <drop_clustered_constraint_option> [ ,...n ] ) ]
        | COLUMN column_name
    } [ ,...n ]

    | [ WITH { CHECK | NOCHECK } ] { CHECK | NOCHECK } CONSTRAINT
        { ALL | constraint_name [ ,...n ] }

    | { ENABLE | DISABLE } TRIGGER
        { ALL | trigger_name [ ,...n ] }

    | { ENABLE | DISABLE } CHANGE_TRACKING
        [ WITH ( TRACK_COLUMNS_UPDATED = { ON | OFF } ) ]

    | SWITCH [ PARTITION source_partition_number_expression ]
        TO target_table
        [ PARTITION target_partition_number_expression ]

    | SET ( FILESTREAM_ON = { partition_scheme_name | filegroup |
                "default" | "NULL" } )

    | REBUILD
      [ [PARTITION = ALL]
        [ WITH ( <rebuild_option> [ ,...n ] ) ]
      | [ PARTITION = partition_number
           [ WITH ( <single_partition_rebuild_option> [ ,...n ] )]
        ]
      ]

    | (<table_option>)
}
[ ; ]
<column_set_definition> ::=
    column_set_name XML COLUMN_SET FOR ALL_SPARSE_COLUMNS

<drop_clustered_constraint_option> ::=
    {
        MAXDOP = max_degree_of_parallelism

      | ONLINE = {ON | OFF }
      | MOVE TO { partition_scheme_name ( column_name ) | filegroup
          | "default" }
    }
<table_option> ::=
    {
        SET ( LOCK_ESCALATION = { AUTO | TABLE | DISABLE } )
    }

<single_partition_rebuild__option> ::=
{
      SORT_IN_TEMPDB = { ON | OFF }
    | MAXDOP = max_degree_of_parallelism
    | DATA_COMPRESSION = { NONE | ROW | PAGE} }
```

```
}
```

Insert command:

Adds one or more new rows to a table or a view in SQL Server 2008 R2. For examples, see INSERT Examples (Transact-SQL).

```
-- Standard INSERT syntax
[ WITH <common_table_expression> [ ,...n ] ]
INSERT
{
        [ TOP ( expression ) [ PERCENT ] ]
        [ INTO ]
        { <object> | rowset_function_limited
          [ WITH ( <Table_Hint_Limited> [ ...n ] ) ] }
    {
        [ ( column_list ) ]
        [ <OUTPUT Clause> ]
        { VALUES ( { DEFAULT | NULL | expression } [ ,...n ] ) [ ,...n ]
          | derived_table
          | execute_statement
          | <dml_table_source>
          | DEFAULT VALUES
        }
    }
}
[; ]
```

Update command:

```
[ WITH <common_table_expression> [...n] ]
UPDATE
    [ TOP ( expression ) [ PERCENT ] ]
    { { table_alias | <object> | rowset_function_limited
         [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]
      }
      | @table_variable
    }
    SET
        { column_name = { expression | DEFAULT | NULL }
          | { udt_column_name.{ { property_name = expression
                                | field_name = expression }
                                | method_name ( argument [ ,...n ] )
                              }
            }
          | column_name { .WRITE ( expression , @Offset , @Length ) }
          | @variable = expression
```

```
          | @variable = column = expression
          | column_name { += | -= | *= | /= | %= | &= | ^= | |= } expression
          | @variable { += | -= | *= | /= | %= | &= | ^= | |= } expression
          | @variable = column { += | -= | *= | /= | %= | &= | ^= | |= }
expression
        } [ ,...n ]

    [ <OUTPUT Clause> ]
    [ FROM { <table_source> } [ ,...n ] ]
    [ WHERE { <search_condition>
            | { [ CURRENT OF
                  { { [ GLOBAL ] cursor_name }
                      | cursor_variable_name
                  }
                ]
              }
            }
    ]
    [ OPTION ( <query_hint> [ ,...n ] ) ]
[ ; ]

<object> ::=
{
    [ server_name . database_name . schema_name .
    | database_name .[ schema_name ] .
    | schema_name .
    ]
    table_or_view_name}
```

Delete command:

```
[ WITH <common_table_expression> [ ,...n ] ]
DELETE
    [ TOP ( expression ) [ PERCENT ] ]
    [ FROM ]
    { <object> | rowset_function_limited
      [ WITH ( <table_hint_limited> [ ...n ] ) ]
    }
    [ <OUTPUT Clause> ]
    [ FROM <table_source> [ ,...n ] ]
    [ WHERE { <search_condition>
            | { [ CURRENT OF
                  { { [ GLOBAL ] cursor_name }
                      | cursor variable name
                  }
                ]
              }
            }
    ]
    [ OPTION ( <Query Hint> [ ,...n ] ) ]
[; ]

<object> ::=
{
    [ server_name.database_name.schema_name.
```

```
      | database_name. [ schema_name ] .
      | schema_name.
    ]
    table_or_view_name
}
```

Delete command:

Removes all rows from a table without logging the individual row deletions. TRUNCATE TABLE is similar to the DELETE statement with no WHERE clause; however, TRUNCATE TABLE is faster and uses fewer system and transaction log resources.

```
TRUNCATE TABLE
    [ { database_name .[ schema_name ] . | schema_name . } ]
    table_name
[ ; ]
```