Databases II

Data Definition Language

HO
GENT

Data Definition Language

Introduction

HO
GENT

Data Definition Language

- DDL is used for
  - Defining databases
  - Defining tables
  - Determining data types in SQL Server
  - Defining constraints – data integrity
  - Defining indexes
  - Defining views (see previous chapter)

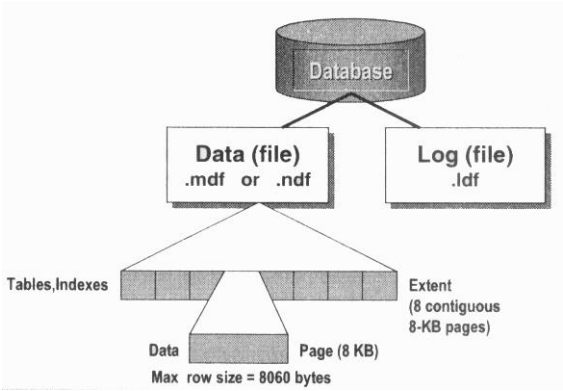
HO
GENT

DDL - Database

HO
GENT

Physical storage of data

- Data and log files



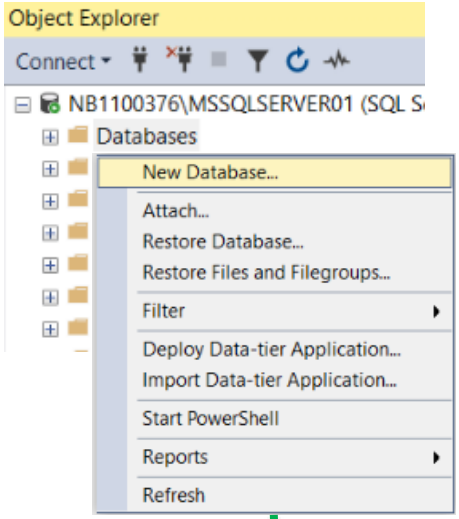
HO
GENT

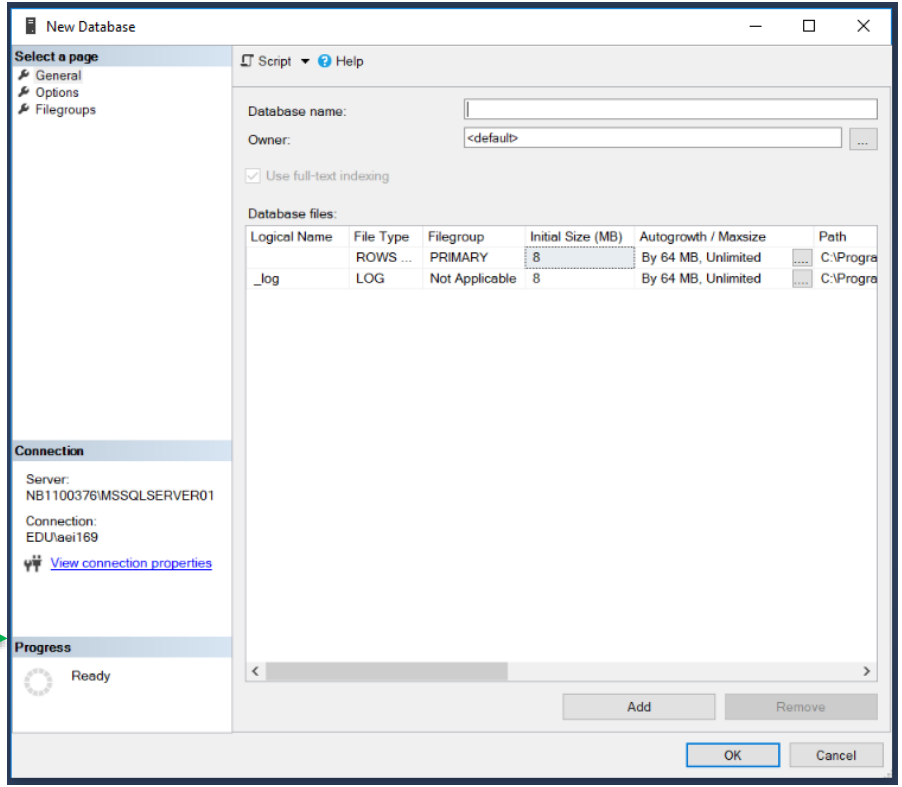
Creating a database

- Who?
  - sysadmin, use with "create database" permissions
- How?
  - A copy of a "model" database is made
    - *the sysdatabases table in the Master database contains information about each database*
  - Specify the physical storage characteristics when creating (or altering) a database:
    - *name of database/transaction log files*
    - *size of database/transaction log files*
    - *location of database(.mdf)/transaction log files(.ldf)*

HO GENT

Create a database with Management Studio





HO GENT

CREATE DATABASE command

```
CREATE DATABASE database_name
```

create database simple form

```
CREATE DATABASE database_name
[ON [<filespec> [ ,...n ]] [, <filegroup> [ ,...n ]]]
[LOG ON { < filespec > [ ,...n ] } ]
[COLLATE collation_name]
[FOR LOAD | FOR ATTACH]
<filespec> ::=
[PRIMARY]
([NAME = logical_file_name ,]
    FILENAME = 'os_file_name'
    [, SIZE = size ]
    [, MAXSIZE = { max_size | UNLIMITED } ]
    [, FILEGROWTH = growth_increment ] ) [ ,...n ]
<filegroup> ::= FILEGROUP filegroup_name <filespec> [ ,...n ]
```

create database extended form with several parameters

HO GENT

CREATE DATABASE examples

```
CREATE DATABASE exerciseDB
```

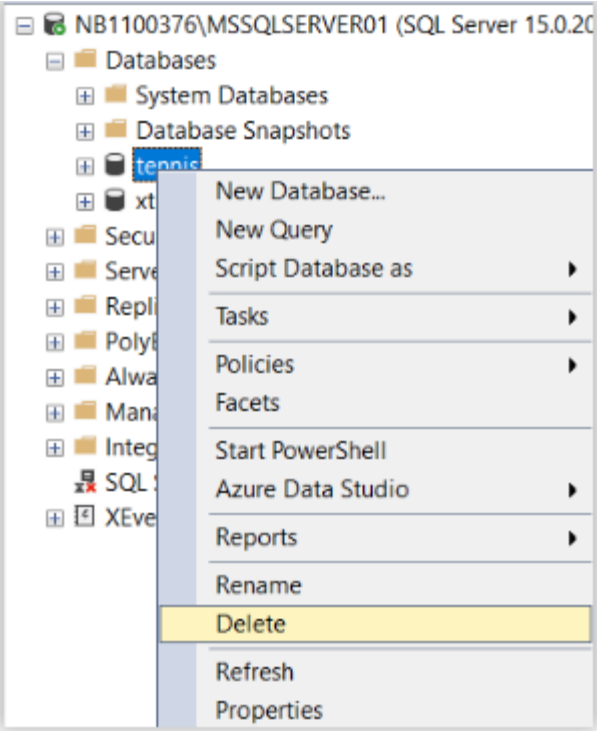
```
CREATE DATABASE exampleDB
ON
 PRIMARY(
  NAME = exDB_data,
  FILENAME = 'C:\Program Files\Microsoft SQL Server\
               MSSQL.1\MSSQL\Data\exDB.mdf',
  SIZE = 10MB, MAXSIZE = 15MB, FILEGROWTH = 20%)
 LOG ON (
  NAME = exDB_log,
  FILENAME = 'C:\Program Files\Microsoft SQL Server\
               MSSQL.1\MSSQL\Data\exDB.ldf',
  SIZE = 3MB, MAXSIZE = 5MB, FILEGROWTH = 1MB)
```

HO GENT

Deleting a database

With SSMS

(SQL Server Management Studio)



HO GENT

The DROP DATABASE command

```
DROP DATABASE database_name
```

You can never drop the system database

```
DROP DATABASE exDB
```

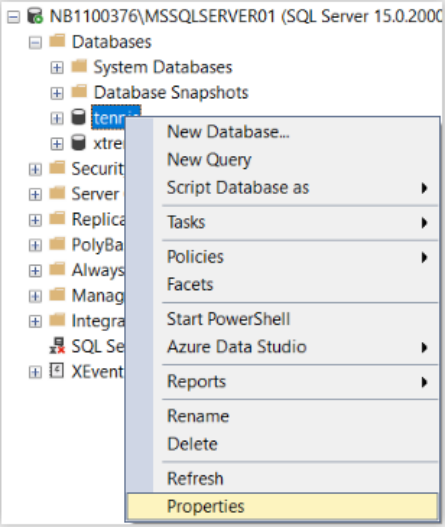
example: deleting the database exDB

HO GENT

Changing characteristics of a database

- What ?
  - Manage the growth of the data and the log file
  - Extend or reduce the size of the data and the log files
  - Add/remove secondary data files, log files

- Through SQL Server Enterprise manager or Management Studio



HO GENT

The ALTER DATABASE command

```
ALTER DATABASE database_name
{| MODIFY NAME = new_database_name
 | COLLATE collation_name
 | <file_and_filegroup_options>
 | <set_database_options> } [;]

<file_and_filegroup_options > ::= …
  <add_or_modify_files> ::= …
  <add_or_modify_filegroups >::= …
<set_database_options>::= …
  <optionspec>::= …
  <change_tracking_option> ::= …
  <database_mirroring_option> ::= …
  <db_encryption_option> ::= …
  <db_update_option> ::= …
  <external_access_option> ::= …
  <recovery_option> ::= …
  <snapshot_option> ::= …
  <termination> ::= …
  <filespec> ::= …
  <filegroup_updatability_option> ::= …
  <auto_option> ::= …
  <cursor_option> ::= …
  <date_correlation_optimization_option> ::= …
  <db_state_option> ::= …
  <db_user_access_option> ::= …
  <parameterization_option> ::= …
  <service_broker_option> ::= …
  <sql_option> ::= …
```

de alter database opdracht met zijn groot scala aan mogelijkheden …

HO GENT

The ALTER DATABASE command

```
ALTER DATABASE exampleDB
MODIFY FILE (name = 'exampleDB_log', size = 10MB)
```

example: change the size of the logfile

```
ALTER DATABASE exampleDB
ADD FILE (name = exampleDB2,
          filename = 'C:\Program Files\Microsoft SQL Server\
                 MSSQL.1\MSSQL\Data\exampleDB2.ndf',
          size = 10MB,
          maxsize = 15MB)
```

example: add a data file

HO GENT

DDL - Tables

HO
GENT

Definition of tables

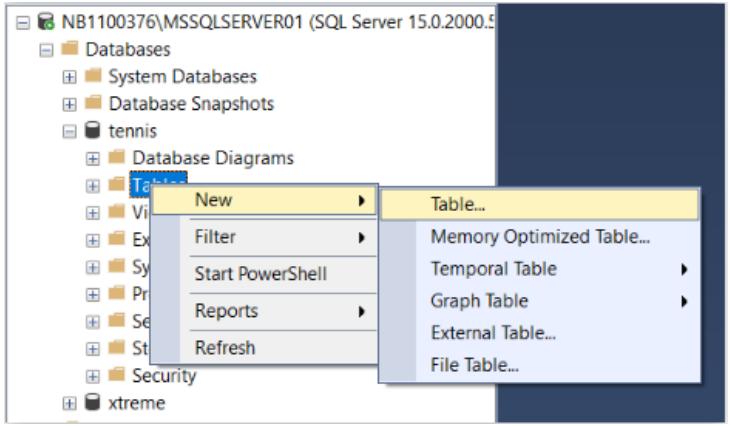
- Creating tables
- Changing table structures
- Deleting tables

HO
GENT

The CREATE TABLE command

- When creating a table we specify
  - The name of the table
  - The definition of its columns (name, datatype)
  - The definition of constraints

- With SQL Server Management Studio



HO GENT

The CREATE TABLE command

```
CREATE TABLE table_name(
  {<column_definition> |
   <computed_column_definition> |
   <column_set_definition> }
  [<table_constraint>] [ ,...n ])
```

Simplified syntax of the create table command

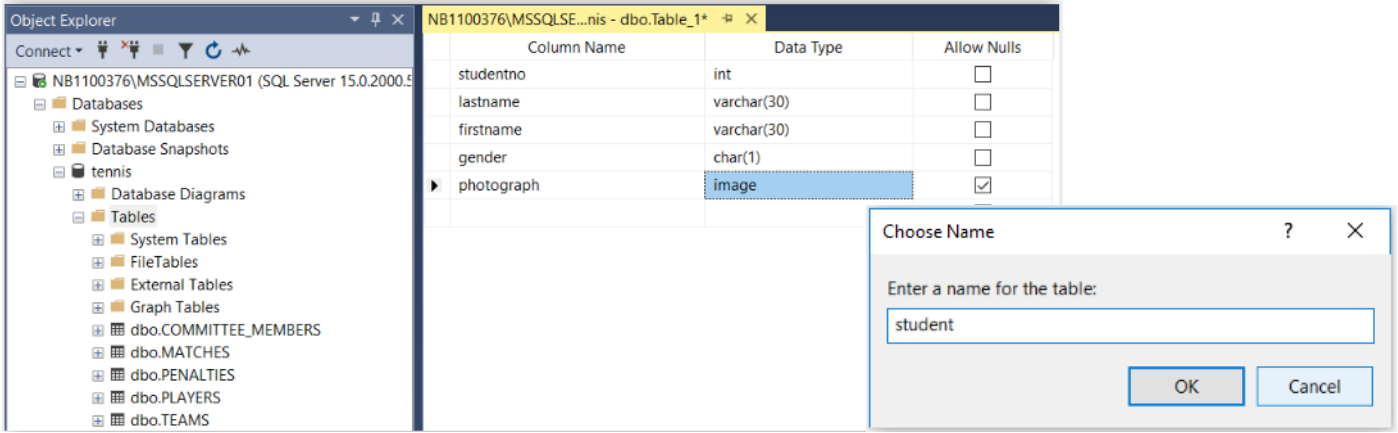
```
CREATE TABLE student(
  studentno int NOT NULL,
  lastname varchar(30) NOT NULL,
  firstname varchar(30) NOT NULL,
  gender char(1) NOT NULL,
  photograph image NULL)
```

example: add a table 'student'

HO GENT

The CREATE TABLE command

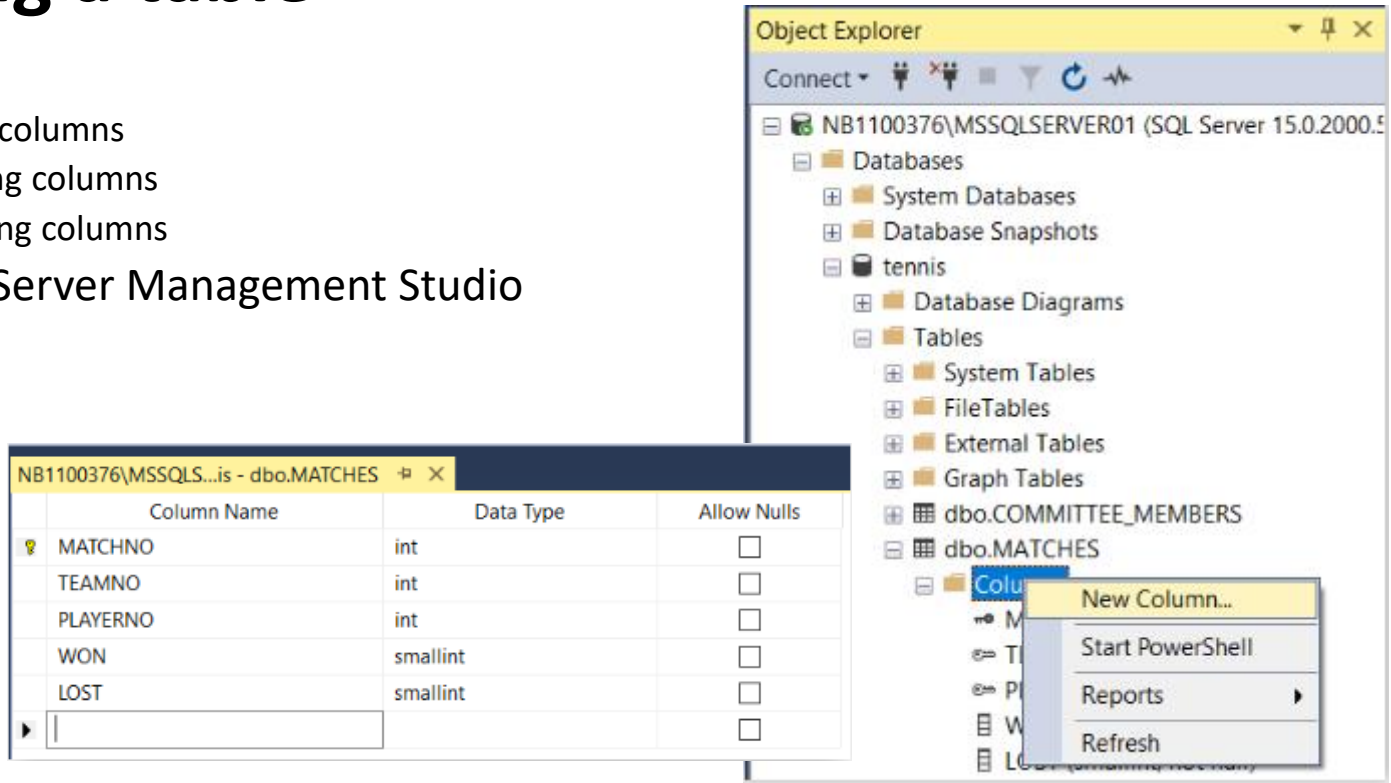
```
CREATE TABLE student(
  studentno int NOT NULL,
  lastname varchar(30) NOT NULL,
  firstname varchar(30) NOT NULL,
  gender char(1) NOT NULL,
  photograph image NULL)
```



HO GENT

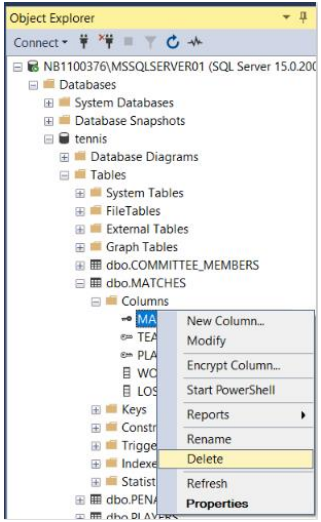
Changing a table

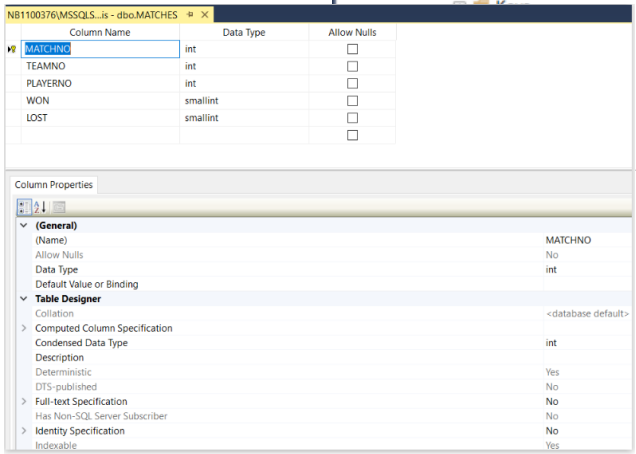
- Example
  - Adding columns
  - Changing columns
  - Removing columns
- With SQL Server Management Studio

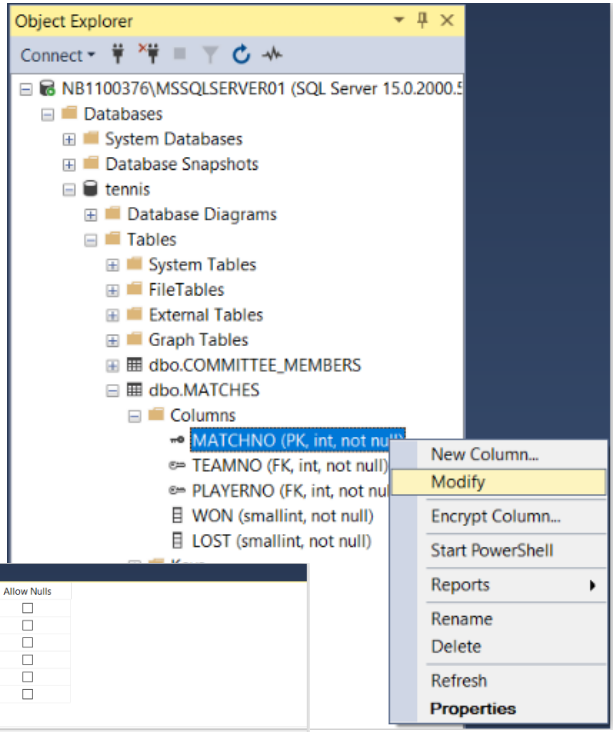


Changing a table

- Example
  - Adding columns
  - Changing columns
  - Removing columns
- With SQL Server Management Studio







HO GENT

The ALTER TABLE command

```
ALTER TABLE table_name {
   ALTER COLUMN column_name {type_name [({ precision[, scale] |
                max})]}|

   ADD  {<column_definition> |
   <computed_column_definition> | <table_constraint> |
   <column_set_definition> } [ ,...n ] |

   DROP  {[CONSTRAINT] constraint_name  |
   COLUMN column_name } [ ,...n ]
```

Simplified syntax of the alter table command

HO GENT

The ALTER TABLE command

- Example
  - Adding a column

```
ALTER TABLE student
ADD address varchar(40) NULL
```

Add the column address

  - Changing a column

```
ALTER TABLE student
ALTER COLUMN address varchar(50)
NULL
```

Extend the number of positions to 50

  - Removing a column

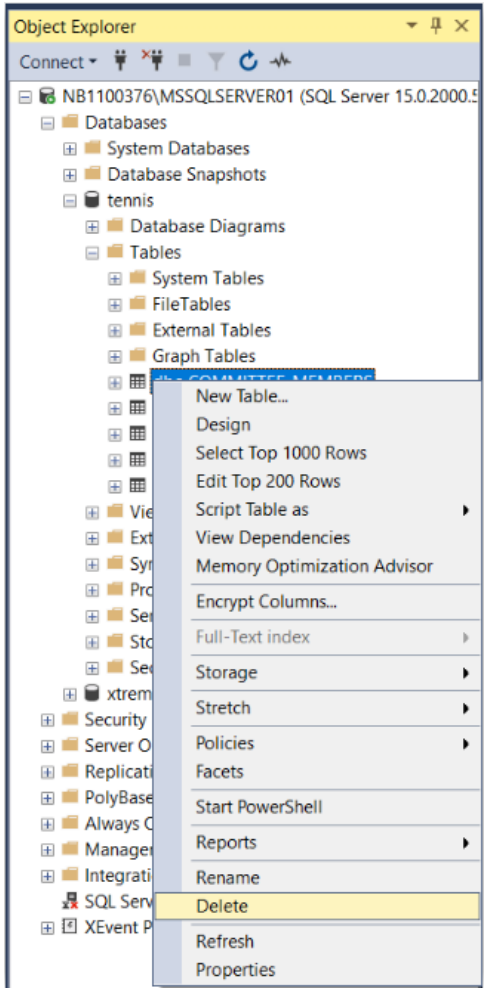
```
ALTER TABLE student
DROP COLUMN address
```

Remove the column address

HO GENT

Deleting a table

- Dependencies are taken into account
  - ex. foreign key constraints, see below
- With SQL Server Management Studio



HO GENT

The DROP TABLE command

```
DROP TABLE table_name
```

Simplified syntax of the drop table command

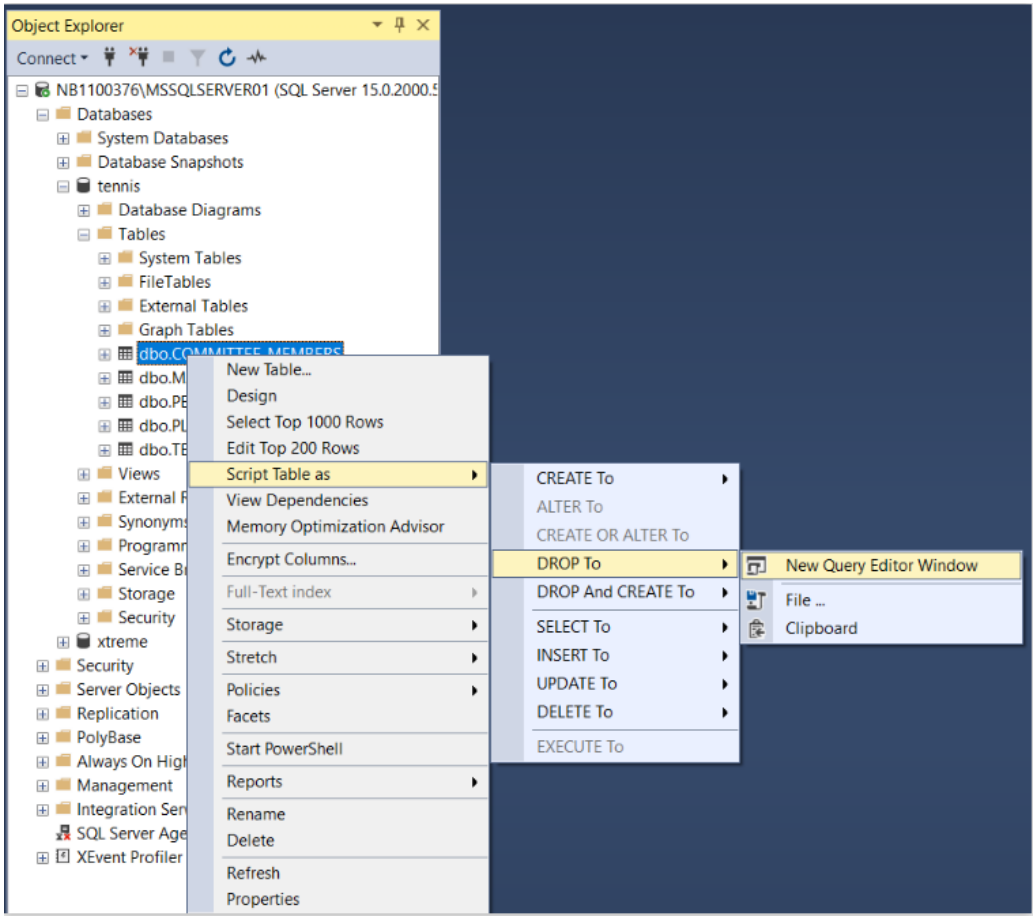
```
DROP TABLE student
```

voorbeeld: verwijder de tabel student

HO GENT

Scripts

- Used for
  - batch processing
  - creation of test and production environment
- With SQL Server Management Studio



Scripts

- Example of resulting script

```
USE [exDB]
GO
IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[class_fk]') AND parent_object_id = OBJECT_ID(N'[dbo].[student]'))
ALTER TABLE [dbo].[student] DROP CONSTRAINT [class_fk]
GO
IF  EXISTS (SELECT * FROM sys.check_constraints WHERE object_id =
OBJECT_ID(N'[dbo].[CK__student__gender__44FF419A]') AND parent_object_id =
OBJECT_ID(N'[dbo].[student]'))
ALTER TABLE [dbo].[student] DROP CONSTRAINT [CK__student__gender__44FF419A]
GO
USE [exDB]
GO
/****** Object:  Table [dbo].[student]    Script Date: 03/25/2009 00:07:08 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[student]')
AND type in (N'U'))
DROP TABLE [dbo].[student]
```

HO GENT

SQL Datatypes

HO
GENT

Datatypes

- Categories of data types

Exact numerics	**Unicode character strings**
Approximate numerics	**Binary strings**
Date and time	**Other data types**
Character strings	

HO GENT

Exact numeric values

datatype	domain	storage
bigint	-2^63 (-9,223,372,036,854,775,808) tot 2^63-1 (9,223,372,036,854,775,807)	8 Bytes
int	-2^31 (-2,147,483,648) tot 2^31-1 (2,147,483,647)	4 Bytes
smallint	-2^15 (-32,768) tot 2^15-1 (32,767)	2 Bytes
tinyint	0 tot 255	1 Byte
bit	0 tot 1	1 Byte (column optimised)
decimal/numeric	- 10^38 +1 tot 10^38 - 1 with maximum precision	5 tot 7 Bytes (~precisie)
money (deprecated)	-922,337,203,685,477.5808 tot 922,337,203,685,477.5807	8 Bytes
smallmoney (deprecated)	- 214,748.3648 tot 214,748.3647	4 Bytes

Remarks

- With **decimal/numeric** you specify the precision (total number of digits) and the scale (number of decimals)
  ex: decimal(5, 2) <-> 123.45
- **bit** can also be threated as **boolean** (1 is TRUE, 0 is FALSE)

HO GENT

Approximative numeric values

datatype	domain	storage
float	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depending on n 4 to 8 Bytes
real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	4 Bytes

Remarks

- Speficy the precision whith **float**: number of bits for the mantissa (1 - 53)
- float(24) is equivalent to SQL 92 **real**
- float(53) is equivalent to SQL 92 **double precision**

HO GENT

Date and time values

datatype	range	storage
datetime	January 1, 1753, through December 31, 9999	2 x 4 Bytes
smalldatetime	January 1, 1900, through June 6, 2079	2 x 2 Bytes
date	Date only	
time	Time only	

Remarks

- The precision of datetime: 3.33 milliseconds
- The precision of **smalldatetime:** 1 minute

HO GENT

Character strings

datatype	domain	storage
char[(n)]	strings with max. n characters	n Bytes
varchar[(n | *max*)]	strings with max. n characters	1 Byte per karakter + 2 Bytes for length

Remarks

- **char** and **varchar** contain **non-unicode** characters (2^8 combinations = ASCII)
- use **char** if column data has a consistent length (e.g. national identification number)
- use **varchar** when the data in this column has a varying length
- **n** is between 1 and 8000
- use **varchar(max)** if very long strings can be expected (> 8000 Bytes)
- ***text*** *data type is no longer supported; use* ***varchar(max)***

HO GENT

Character strings

datatype	domain	storage
nchar[(n)]	strings with max. n characters	2 x n Bytes
nvarchar[(n | *max*)]	strings with max. n characters	2 Bytes per karakter + 2 Bytes for length

Remarks

- **nchar** and **nvarchar** contain **unicode** characters (2^{16} combinations)
- For use see char and varchar
- **n** is between 1 and 4000, use max for longer strings
- ***ntext*** *data type is no longer supported; use n****varchar(max)***

HO GENT

Binary data

datatype	domain	storage
binary[(n)]	data size up to n Bytes	n Bytes
varbinary[(n | *max*)]	data size up to n Bytes	1 Byte per Byte + 2 Bytes voor lengte

Remarks

- ***ntext*** *data type is no longer supported; use* ***varbinary(max)***
- **Write photo to database (**column photo = type varbinary(max)**):**

```
update players
set photo=( SELECT *
            FROM OPENROWSET(BULK N'C:\temp\beer.jpg', SINGLE_BLOB) as img)
where playerno=2;
```

HO GENT

Type conversion

- Implicit
  - Automatically for some conversions
  - Example: UnitPrice * 0,5
    UnitPrice (integer) is automatically converted to decimal
- Explicit
  - **CAST** (<value expression> AS <data type>)
  - Example: PRINT CAST(-25.25 AS INTEGER) -> -25
  - **CONVERT** (<data type, <expression> [, <style>])
  - Example: CONVERT(VARCHAR, getdate(), 106) -> 20 jan 2004
  - FORMAT:

```
select orderid, format(orderdate,'dd-MM-yyyy')
from orders;
```

HO GENT

CONSTRAINTS

HO
GENT

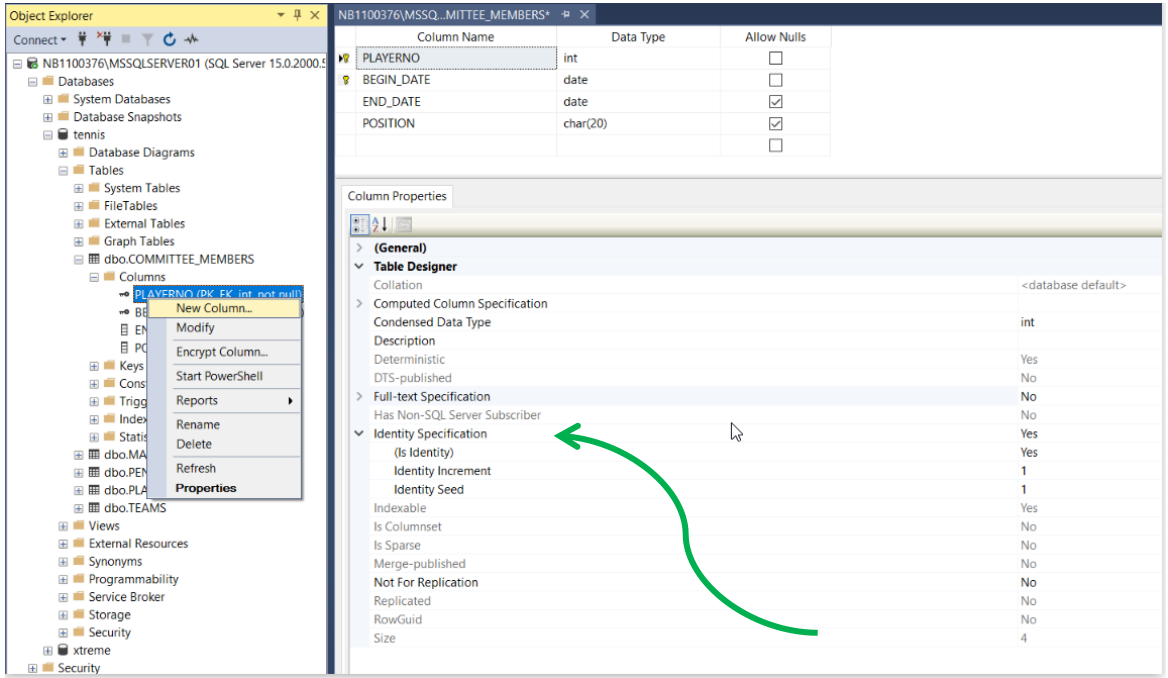
IDENTITY values

- an identity colunm contains
  - A **unique** value for each row
  - **System generated** sequential values
- Only 1 identity column per table is allowed
- Uses integer datatypes (or decimal with scale 0)
- An identity column can't contain null values
- An identity column is not updatable
- De function @@IDENTITY (see stored procedures) returns the last generated identity value

HO GENT

IDENTITY values

SQL Server Management Studio



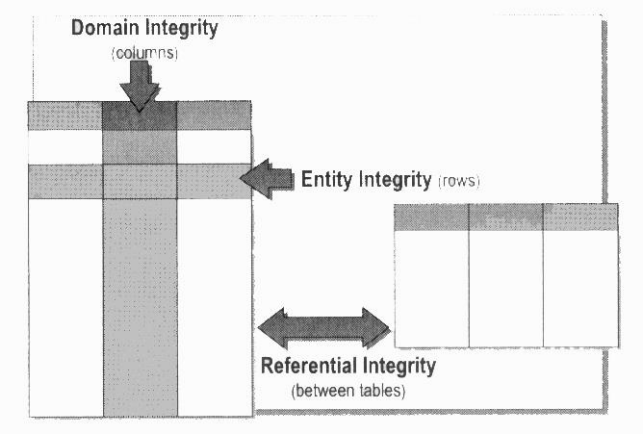
HO GENT

IDENTITY values

```
CREATE TABLE student (
  studentno int identity(100, 5) NOT NULL,
  lastname varchar(30) NOT NULL,
  firstname varchar(30) NOT NULL,
  gender char(1) NOT NULL,
  photograph image NULL)
```

HO GENT

Data integrity



- Types
  - Ddomain integrity
  - Entity integrity
  - Referential integrity
- Ensure through
  - Database definitions: constraints, defaults and rules
  - Procedures: client (programs) or server (triggers and stored procedures)
- Always prefer constraints through data definitions if possible
  - More secure than applications: constraints can’t be bypassed
  - Generally better performance

HO
GENT

Data integrity

Constraints

- Restrictions to data
- When adding, updating or deleting rows constraints are checked
- Different types

Type of integrity	constraint type
domain	DEFAULT
	CHECK
entity	PRIMARY KEY
	UNIQUE
referential	FOREIGN KEY
	CHECK

- Constraints can be defined on one or more columns
  - COLUMN-LEVEL constraint : 1 column
  - TABLE-LEVEL constraint : several columns

HO GENT

Definition of constraints

SQL Server Management Studio

- Use the design-mode for table

or

- Use *create table* or *alter table* command
  - As part of column definition

```
alter table student
add ssno int not null unique
```

example: add constraint
(remark: this only works on an empty table

  - As a separate line

```
CONSTRAINT constraint_name type_name expression
```

```
constraint ssno_U unique(ssno)
```

  - NULL and DEFAULT can only be specified at column definition, not through a separate command

HO GENT

Constraints example

```
create table class (
  classID smallint identity not null primary key,
  className varChar(30))
```

Example 1: create table class with pk constraint

```
create table student(
  studentno  int   identity(1,1)  not null primary key,
  lastname  varchar(30) not null,
  firstname varchar(30) not null,
  gender  char(1) default 'M' check(gender in ('M','F')) not null,
  ssno int not null,
  class smallint null,
  photograph varbinary(max) null,
  constraint ssno_u unique(ssno),
  constraint class_fk foreign key(class) references class(classID))
```

Example 2: create table student with some constraints

```
alter table student
add constraint lastname_c check (lastname NOT LIKE '%[0-9]%')
```

Example 3: change table student and add constraint

HO GENT

The DEFAULT constraint

Specifies the **default value** for a column

- Used when no value is given with INSERT
- You can use only one default constraint per column

```
create table student(
  studentno  int   identity(1,1)  not null primary key,
  lastname  varchar(30) not null,
  firstname varchar(30) not null,
  gender  char(1) default 'M' check(gender in ('M','F')) not null,
  ssno int not null,
  class smallint null,
  photograph varbinary(max) null,
  constraint ssno_u unique(ssno),
  constraint class_fk foreign key(class)
             references class(classID))
```

HO GENT

The CHECK constraint

Specify **allowed values** in a column

- Checked with INSERT and UPDATE
- Specification through a value WHERE clause *(no subqueries)*
  - List of values

```
gender  char(1) default 'M' check(gender in ('M','F')) not null
```

Example: check constraint in column defnition

  - like operator (e.g. check constraint as a separate line)

```
constraint lastname_c check (lastname NOT LIKE '%[0-9]%')
```

Example: check constraint as separate line in table definition

  - minimum and maximum values

```
… CHECK (age >= 18)
… CHECK (BirthDay > '01-01-1985' AND BirthDay <= getdate())
```

  - ranges :

```
… CHECK(price BETWEEN 12 AND 45)
```

HO GENT

The UNIQUE constraint

Specifies that no two rows can have the same value for a certain column

- Is defined for 1 or a combination of columns
- >1 unique constraints per table allowed
- DBMS creates a unique index on those column(s)
  (per default a nonclustered index, see further)
- Typical usage: to impose a unique value to non primary key columns

```
ssno int unique
```

example: unique constraint as part of column definition

```
constraint ssno_U unique(ssno )
```

example : unique constraint as separate line in table definition

HO GENT

The PRIMARY KEY constraint

Specification of the primary key

- 1 primary key constraint per table
- Can be defined on 1 or a combination of columns (= composed key)
- Value (or combination of values) has to be unique
- NULL values not allowed
- DBMS creates a unique index on the columns *(by default a clustered index is created unless specified otherwise, see further)*

```
studentno int primary key
```

example: definition of primary key as part of column definition

```
constraint studentno_PK primary key(studentno)
```

example: definition of primary key as separate line in table definition

HO GENT

The FOREIGN KEY constraint

Used to link two tables together

- 0, 1, n foreign keys per table
- NULL values might be allowed (depends on NULL setting)
- This constraint guarantees **referential integrity:**
  - FK's refer to a *primary key* or *unique constraint* in (another or the same) table

```
constraint class_fk foreign key(class) references class(classID)
```

  - The value of a NOT NULL foreign key has to exist in the referenced table
  - Also defines cascading referential integrity actions
    - ON DELETE
    - ON UPDATE

HO GENT

The FOREIGN KEY constraint

ON **DELETE** actions: options

- CASCADE
  - Cascaded delete
- NO ACTION
  - Delete only if no referring values, otherwise error
  - This is the default action
- SET NULL
  - Referring values are set to NULL
  - Remark: *only possible if no NOT NULL constraint on FK columns*
- SET DEFAULT
  - Referring values are set to their defaults
  - Remark: *if no default constraint exists, NULL is used*

HO GENT

The FOREIGN KEY constraint

ON **UPDATE** actions: options

- CASCADE
  - Cascaded update
- NO ACTION
  - Update only if no referring values, otherwise error
  - This is the default action
- SET NULL
  - Referring values are set to NULL
  - Remark: *only possible if no NOT NULL constraint on FK columns*
- SET DEFAULT
  - Referring values are set to their defaults
  - Remark: *if no default constraint exists, NULL is used*

HO GENT

The FOREIGN KEY constraint

```
create table class (
  classID smallint identity not null primary key,
  className varChar(30))
```

```
create table student(
  studentno  int   identity(1,1)  not null primary key,
  lastname  varchar(30) not null,
  firstname varchar(30) not null,
  gender  char(1) default 'M' check(gender in ('M','F')) not null,
  ssno int not null,
  class smallint null,
  photograph varbinary(max) null,
  constraint ssno_u unique(ssno),
  constraint class_fk foreign key(class) references class(classID)
              on delete no action
              on update cascade)
```

example: on delete and on update constraints for foreign key

HO GENT

The FOREIGN KEY constraint

```
create table class (
  classID smallint identity not null primary key,
  className varChar(30))
```

```
create table student(
  studentno  int   identity(1,1)  not null primary key,
  lastname  varchar(30) not null,
  firstname varchar(30) not null,
  gender  char(1) default 'M' check(gender in ('M','F')) not null,
  ssno int not null,
  class smallint null,
  photograph varbinary(max) null)
```

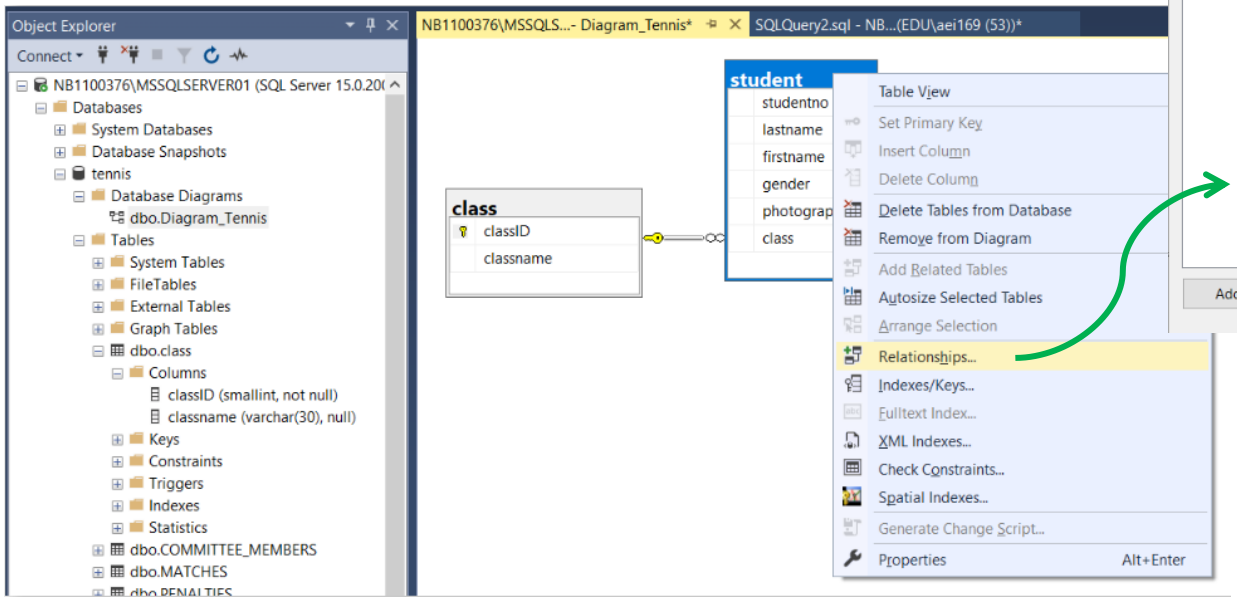
```
alter table student
add constraint ssno_u unique(ssno)
```

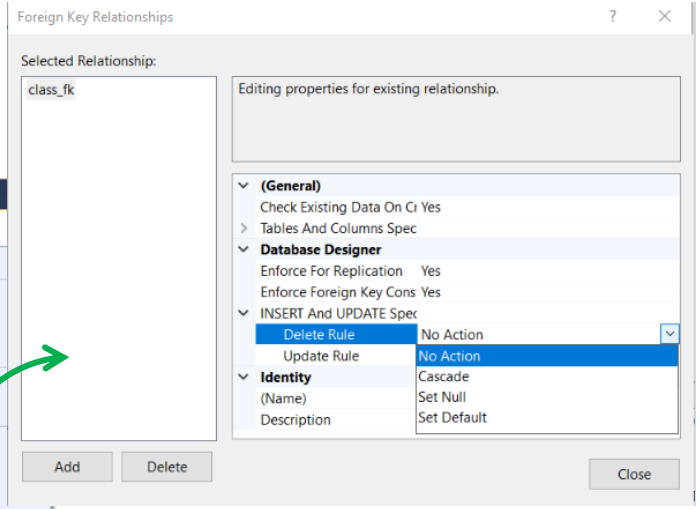
```
alter table student
add constraint class_fk foreign key(class) references class(classID)
              on delete no action
              on update cascade
```

HO GENT

The FOREIGN KEY constraint

SQL Server Management Studio





HO GENT

The FOREIGN KEY constraint

SQL Server Management Studio:

- Create and manage foreign key constraint directly from ERD diagram