Databashantering YH00541, Vecka 47

Introduktion, Praktiska frågor

Information på https://studentportal.nackademin.se/course/view.php?id=4158 uppdateras efter hand.

Obs!
På grund av vädret måndag 21 november har vi flyttat måndagens lektioner till tisdag, varpå tisdagens lektioner blir på onsdag.
Tisdag 22 november, onsdag 23 november och torsdag 24 november blir det undervisning på distans.

Vecka	**Dag**	**Område**
46	Mån 14 nov	Introduktion, Använda SQL
46	Tis 15 nov	Fortsättning SQL
46	Tors 17 nov	Modellera data, Skapa databaser
47	~~Mån 21 nov~~ Tis 22 nov	Installera SQL Server (eller motsvarande)
47	~~Tis 22 nov~~ Ons 23 nov	Hantera säkerhet, Optimera databaser
47	Tors 24 nov	Olika typer av databaser
48	Mån 28 nov	Koppla applikation till databas
48	Tis 29 nov	Lagra data, Olika driftmiljöer
48	Fre 2 dec	Inlämningsuppgift, Summering

Kl 09.00 – 16.00, med undantag för torsdag 24 nov endast kl 09.00 – 13.00.

Lunch 11.30 – 12.30 (istället för 12.00 – 13.00).

Kort repetition av förra veckan

Tabeller, kolumner, datatyper. Default-värden. Kontroll av giltiga värden med “check”, “not null”, “unique”.

Primära och främmande nycklar. Kopplingar mellan två eller flera tabeller.

Sql-frågor för att definiera schema, manipulera data, göra urval. Transaktioner.

Urvalsvillkor, sortera och gruppera data. Inbyggda funktioner för att bearbeta siffror, text mm.

Modeller med ER-diagram. Översätta logisk modell till fysisk modell.

Normalisering av databas.

En liten återblick på kunskaper och kompetenser enligt kursplan. Vilka delar har vi klarat av?

~~Måndag 21 nov~~ Tisdag 22 nov – Installera SQL Server

Mer exakt Microsoft SQL Server 2019 (15.0).

Transact-SQL (T-SQL), Microsofts “dialekt” av SQL.

https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15

Installation av SQL Server 2019, Developer Edition.

- Windows: https://learn.microsoft.com/en-us/sql/database-engine/install-windows/install-sql-server?view=sql-server-ver15
- Linux: https://learn.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-ver15
- MacOS mha Docker: https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15
  Vilket förutsätter att Docker for Mac är installerat, se https://docs.docker.com/desktop/install/mac-install/
  Användbara tips på bland annat https://www.twilio.com/blog/using-sql-server-on-macos-with-docker

Klientprogram:

Förslagsvis DBeaver eller Azure Data Studio.

https://dbeaver.io/download/

https://learn.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15

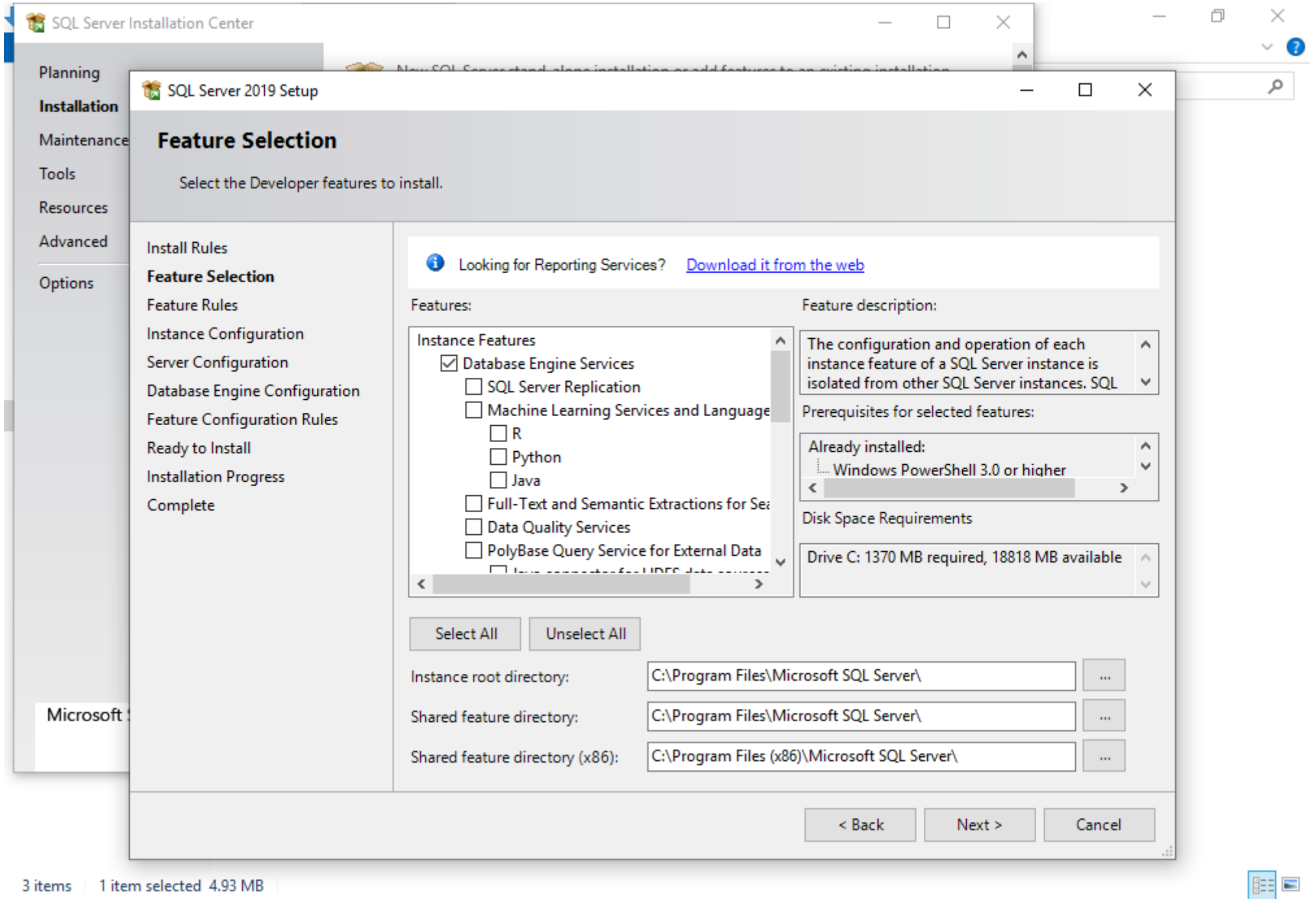
Installation i Windows

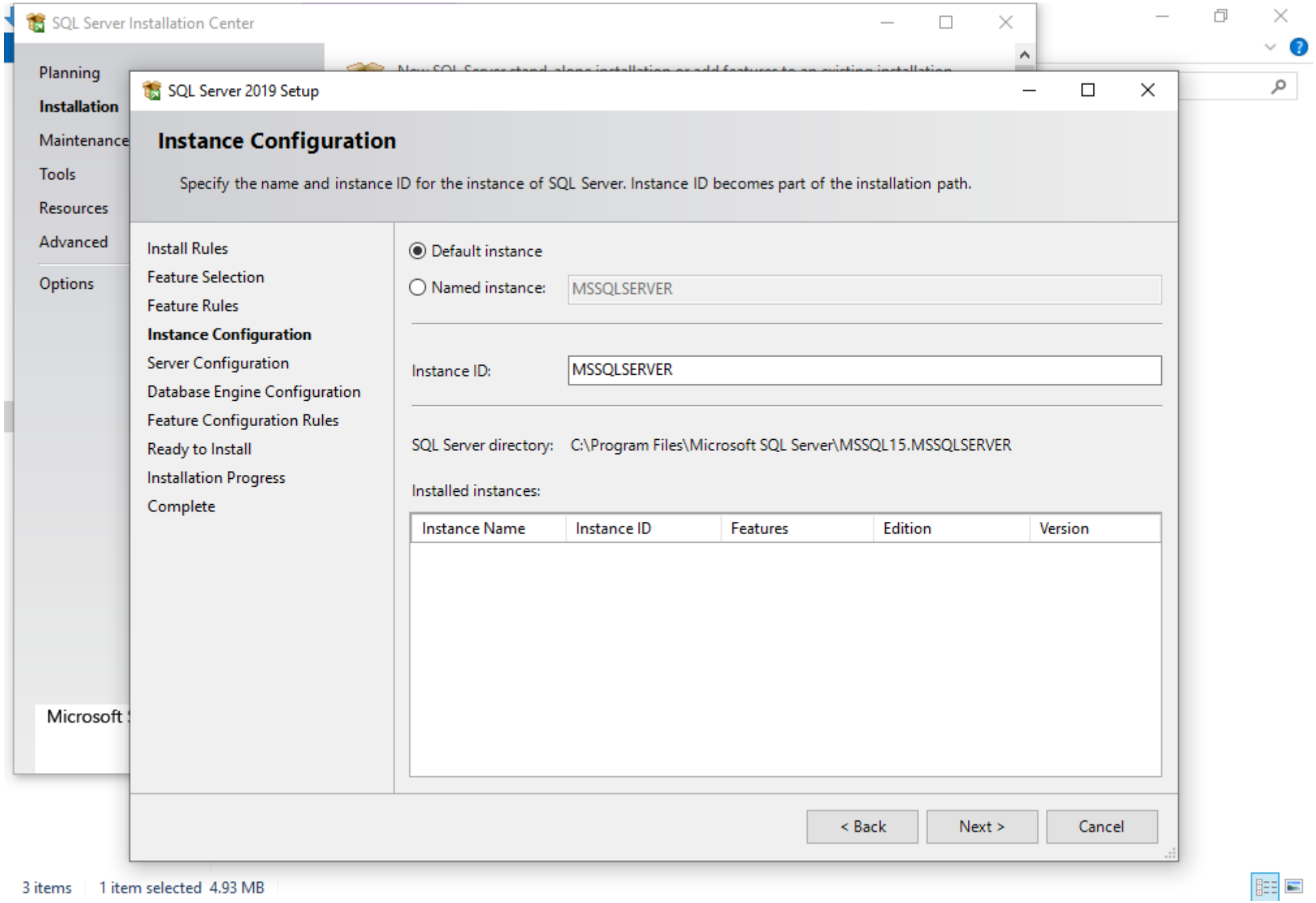
Listan nedan är på intet sätt fullständig, utan visar endast på de viktigaste inställningarna.

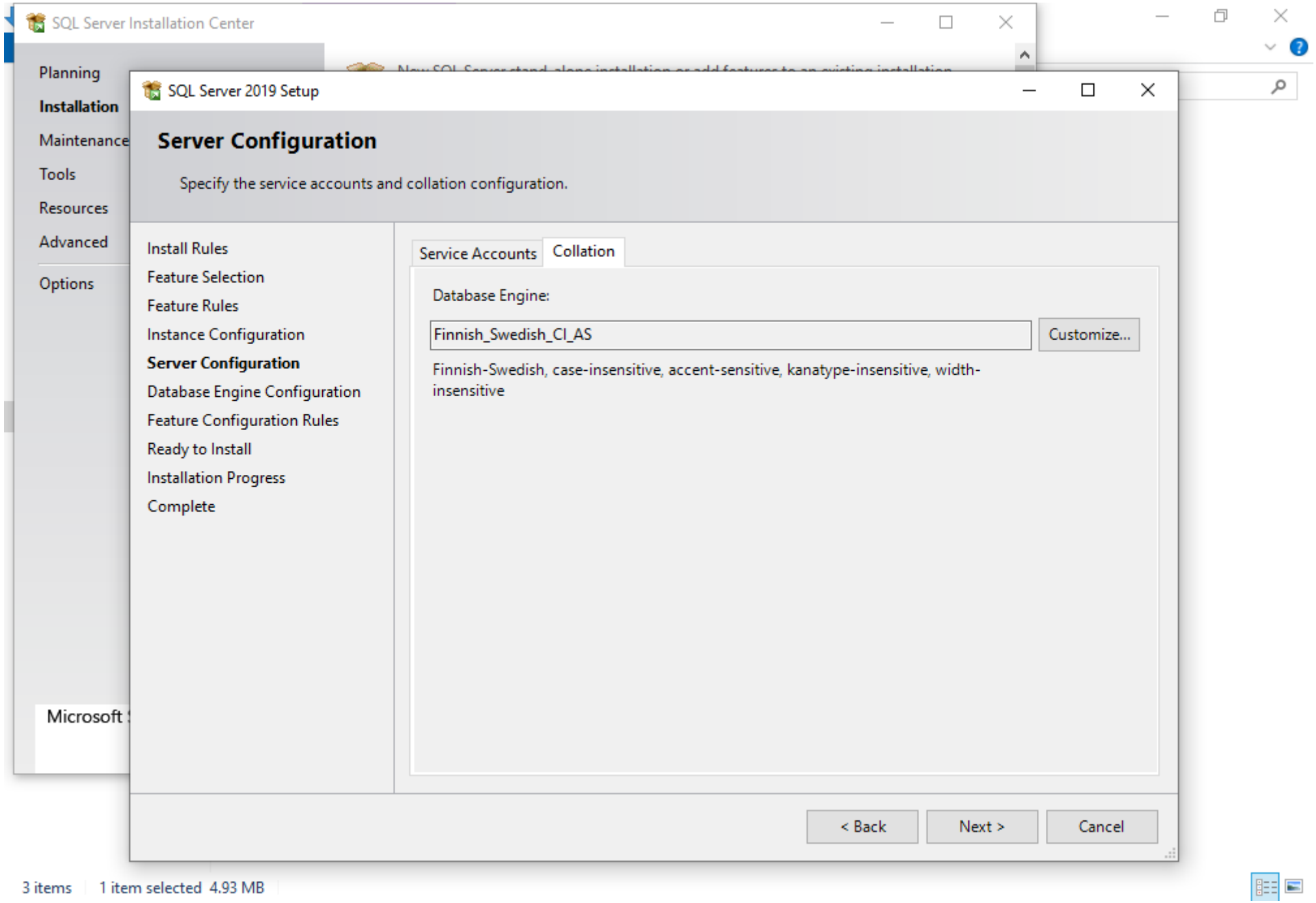
1. Välj “Custom installation type”.
2. Ange var installationsmedia finns.
3. Under “Installation”, välj “New SQL Server stand-alone installation”.
4. Under “Product Key”, välj “Developer” och acceptera villkor.
5. Vänta ut att “Install Rules” söker efter eventuella problem.
6. Under “Feature Selection”, kryssa för …
    1. Database Engine Services
    2. Client Tools Connectivity
    3. Integration Services
    4. Client Tools Backward Compatibility
7. Under “Instance Configuration”, låt ID=MSSQLSERVER stå kvar.
8. Under “Server Configuration”, ange Collation=Finnish_Swedish_CI_AS (case insensitive, accent sensitive).
9. Under “Database Engine Configuration” …
    1. Välj Authentication=Mixed mode.
    2. Ange lösenord för kontot “sa”.
    3. Lägg till Current User som SQL Server admin.
10. Ready to install!

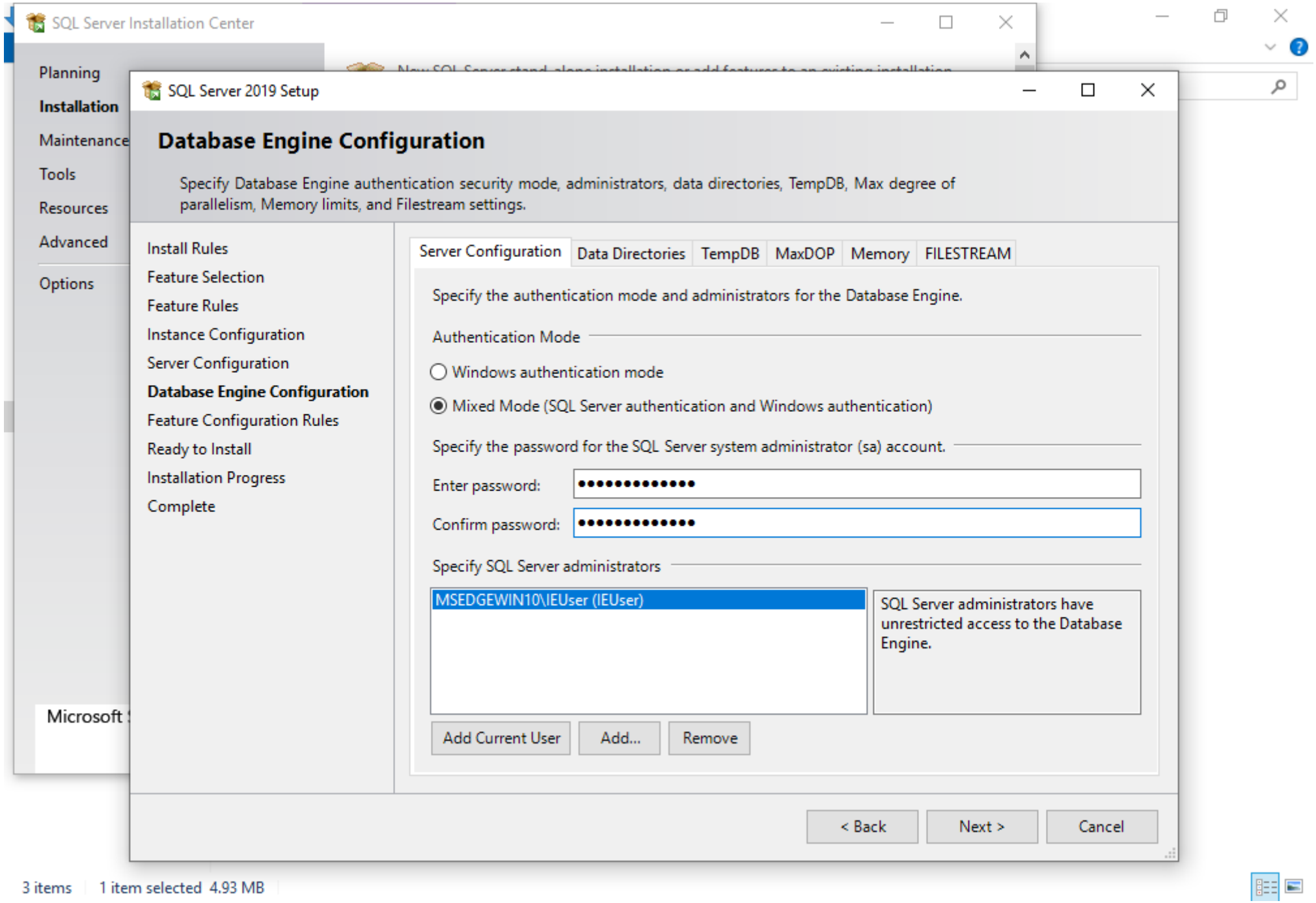
Testa att ansluta till databasen.

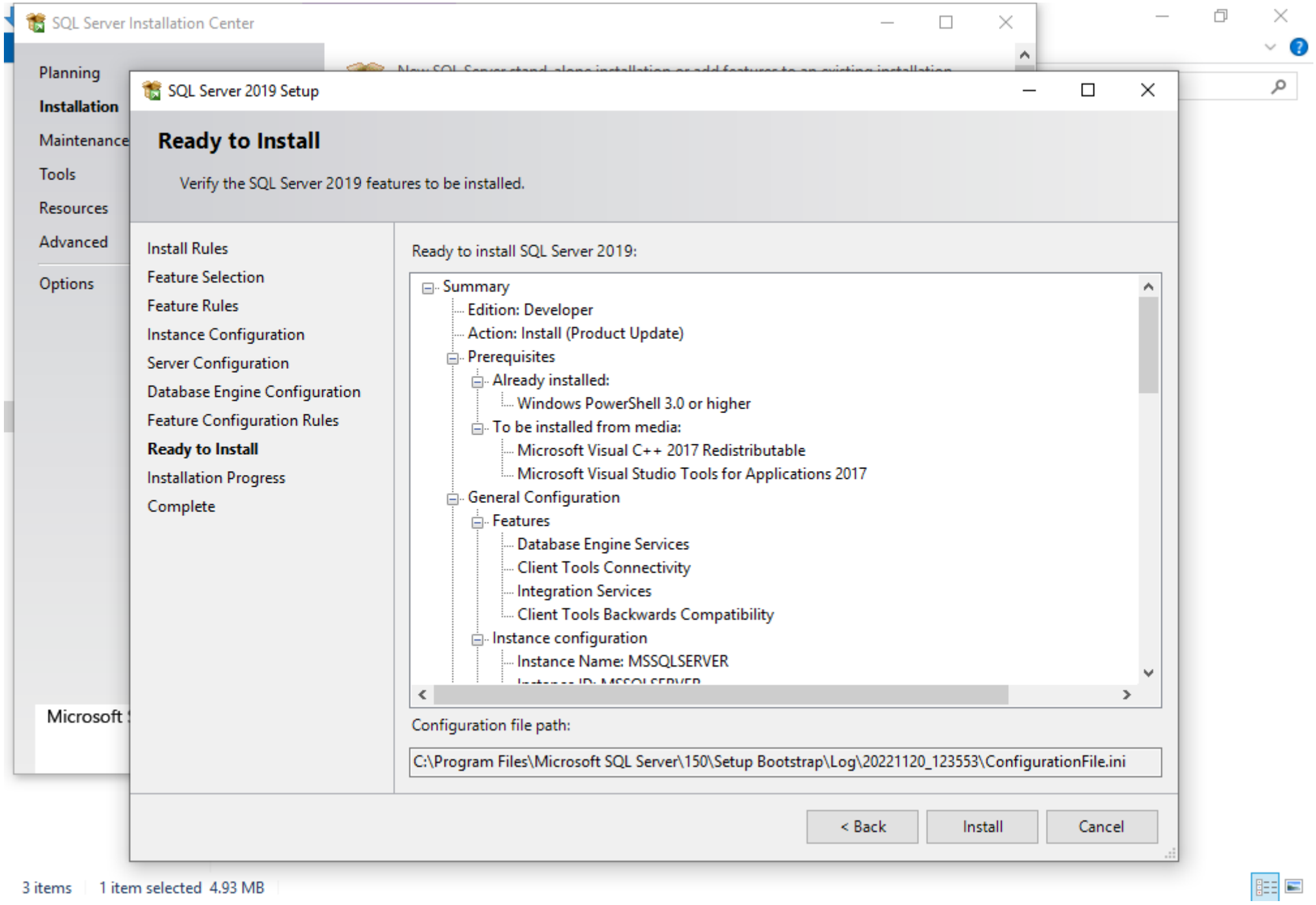
Vid problem, kolla att TCP/IP=Enabled under SQL Server Network Configuration.

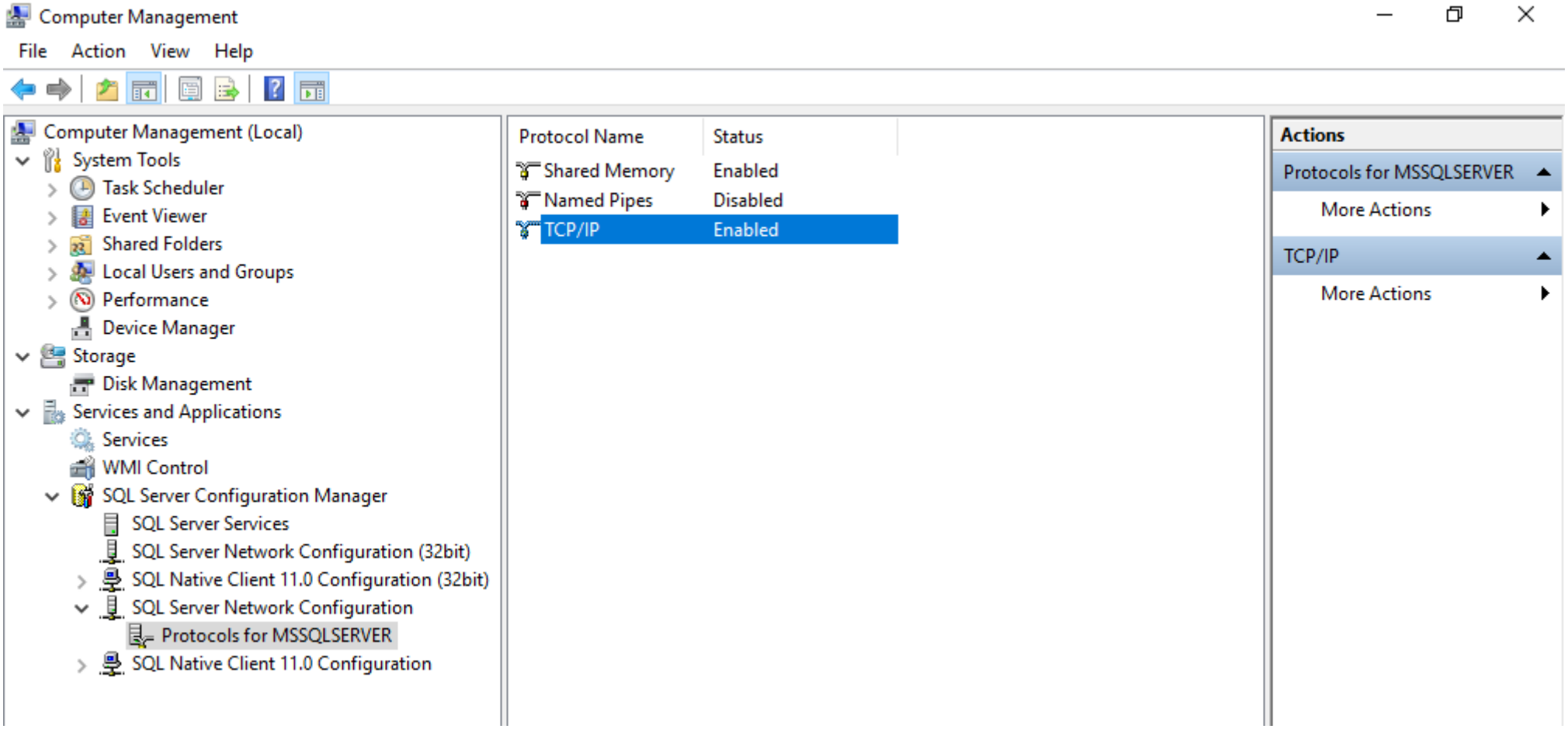

SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Advanced
Options
SQL Server 2019 Setup
Feature Selection
Select the Developer features to install.
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete
Looking for Reporting Services?
Download it from the web
Features:
Instance Features
Database Engine Services
SQL Server Replication
Machine Learning Services and Language
R
Python
Java
Full-Text and Semantic Extractions for Sea
Data Quality Services
PolyBase Query Service for External Data
Feature description:
The configuration and operation of each instance feature of a SQL Server instance is isolated from other SQL Server instances. SQL
Prerequisites for selected features:
Already installed:
Windows PowerShell 3.0 or higher
Disk Space Requirements
Drive C: 1370 MB required, 18818 MB available
Select All
Unselect All
Instance root directory:
C:\Program Files\Microsoft SQL Server\
Shared feature directory:
C:\Program Files\Microsoft SQL Server\
Shared feature directory (x86):
C:\Program Files (x86)\Microsoft SQL Server\
< Back
Next >
Cancel
3 items
1 item selected 4.93 MB


SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Advanced
Options
SQL Server 2019 Setup
Instance Configuration
Specify the name and instance ID for the instance of SQL Server. Instance ID becomes part of the installation path.
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete
Default instance
Named instance:
MSSQLSERVER
Instance ID:
MSSQLSERVER
SQL Server directory:
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER
Installed instances:
Instance Name
Instance ID
Features
Edition
Version
< Back
Next >
Cancel
3 items
1 item selected 4.93 MB


SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Advanced
Options
SQL Server 2019 Setup
Server Configuration
Specify the service accounts and collation configuration.
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete
Service Accounts
Collation
Database Engine:
Finnish_Swedish_CI_AS
Customize...
Finnish-Swedish, case-insensitive, accent-sensitive, kanatype-insensitive, width-insensitive
< Back
Next >
Cancel
3 items
1 item selected 4.93 MB


SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Advanced
Options
SQL Server 2019 Setup
Database Engine Configuration
Specify Database Engine authentication security mode, administrators, data directories, TempDB, Max degree of parallelism, Memory limits, and Filestream settings.
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete
Server Configuration
Data Directories
TempDB
MaxDOP
Memory
FILESTREAM
Specify the authentication mode and administrators for the Database Engine.
Authentication Mode
Windows authentication mode
Mixed Mode (SQL Server authentication and Windows authentication)
Specify the password for the SQL Server system administrator (sa) account.
Enter password:
Confirm password:
Specify SQL Server administrators
MSEDGEWIN10\IEUser (IEUser)
SQL Server administrators have unrestricted access to the Database Engine.
Add Current User
Add...
Remove
< Back
Next >
Cancel
3 items
1 item selected 4.93 MB


SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Advanced
Options
SQL Server 2019 Setup
Ready to Install
Verify the SQL Server 2019 features to be installed.
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete
Ready to install SQL Server 2019:
Summary
Edition: Developer
Action: Install (Product Update)
Prerequisites
Already installed:
Windows PowerShell 3.0 or higher
To be installed from media:
Microsoft Visual C++ 2017 Redistributable
Microsoft Visual Studio Tools for Applications 2017
General Configuration
Features
Database Engine Services
Client Tools Connectivity
Integration Services
Client Tools Backwards Compatibility
Instance configuration
Instance Name: MSSQLSERVER
Configuration file path:
C:\Program Files\Microsoft SQL Server\150\Setup Bootstrap\Log\20221120_123553\ConfigurationFile.ini
< Back
Install
Cancel
3 items
1 item selected 4.93 MB


Computer Management
File Action View Help
Computer Management (Local)
System Tools
Task Scheduler
Event Viewer
Shared Folders
Local Users and Groups
Performance
Device Manager
Storage
Disk Management
Services and Applications
Services
WMI Control
SQL Server Configuration Manager
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
SQL Server Network Configuration
Protocols for MSSQLSERVER
SQL Native Client 11.0 Configuration
Protocol Name
Status
Shared Memory
Enabled
Named Pipes
Disabled
TCP/IP
Enabled
Actions
Protocols for MSSQLSERVER
More Actions
TCP/IP
More Actions

Installation i Linux (Ubuntu 20.04)

Kopierat från https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver15

1. Import the public repository GPG keys:
   wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
2. Register the SQL Server Ubuntu repository:
   sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/20.04/mssql-server-2019.list)"
3. Run the following commands to install SQL Server:
   sudo apt-get update
   sudo apt-get install -y mssql-server
4. After the package installation finishes, run mssql-conf setup and follow the prompts to set the SA password and choose your edition.
   sudo /opt/mssql/bin/mssql-conf setup
5. Once the configuration is done, verify that the service is running:
   systemctl status mssql-server --no-pager
6. If you plan to connect remotely, you might also need to open the SQL Server TCP port (default 1433) on your firewall.

Se även ytterligare steg på learn.microsoft.com för att installera mssql-tools. (Inte nödvändigt om du använder andra verktyg för att ansluta.)

För att avinstallera:

sudo apt-get remove mssql-server

För att även radera alla databasfiler:

sudo rm -rf /var/opt/mssql

Användare och behörigheter

Att använda “sa” för alla databaskopplingar är mindre bra.

Bättre skapa användare med minsta nödvändiga behörighet för olika ändamål.

Gruppera användare genom olika roller. (Mer om detta senare.)

Hur är själva databasen lagrad?

Datafiler med namn .mdf

- Uppdelade i logiska “pages”.

Loggfiler med namn .ldf

- Ändringar i schema.
- Varje insert, update, delete.
- Info om transaktioner.

Säkerhetskopior med namn .bak

Övningar

Skissa på en lösning för att hantera Studenter och Kurser, med bland annat följande förutsättningar.

- En student har som minimum förnamn, efternamn och födelsedatum.
- En kurs har som minimum kursnamn, kurskod, antal poäng och en lärare, samt hör till ett visst övergripande ämnesområde.
- En kurs kan också ha start- och slutdatum.
- Samma student kan vara registrerad på flera olika kurser, men bara en gång per kurs.
- Man får inte radera en kurs som har studenter, eller en student som är registrerad på en kurs.
- Efter avslutad kurs får en student ett betyg (IG, G eller VG).

Rita först ER-diagram.

Översätt därefter till fysisk datamodell, och formulera lämpliga SQL-kommandon (för MS SQL Server 2019).

Befolka databasen med information om studenter, kurser mm.

Formulera SQL-frågor för till exempel ...

- Urval som visar vilka kurser de olika studenterna är registrerade på.
- Urval som visar hur många studenter som är registrerade på respektive kurs.
- Urval som visar betygsfördelningen på avslutade kurser.

Extrauppgifter:

Testa med olika användare med olika nivåer av behörighet, till exempel …

- En användare med rätt att ändra schema för databasen.
- En användare med rätt att lägga till, ändra och radera data.
- En användare med (endast) rätt att göra urval.

Fundera över vilka olika index som skulle kunna vara användbara för databasen i exemplet ovan.

~~Tisdag 22 november~~ Onsdag 23 november – Hantera säkerhet, Optimera databaser

Även denna dag undervisning på distans.

Före lunch mestadels teoretisk genomgång. Efter lunch mestadels egna övningar.

Vid kl 14.30 samling för summering av dagen och möjlighet att ställa frågor.

Mer om användare, behörigheter, mm

Lite repetition från gårdagens installation, “Windows authentication mode” vs “Mixed mode”.

Begreppen “login” vs “user”.
På servernivå “logins” och “server roles”.
På databasnivå “users” och “database roles”.

Vem är jag? Testa med “select current_user”. Login “sa” är alltså mappat mot user “dbo”.

Skapa ett nytt login: create login my_login with password='Testing!';
Kolla vilka login som finns: select * from sys.sql_logins;

Skapa en ny användare: create user my_user for login my_login;
Kolla vilka användare som finns: select * from sys.database_principals;

Men kan min användare göra något i databasen? Nedanstående bör resultera i “select permission was denied”.
execute as user='my_user'; select * from Students; revert;

Först måste vi dela ut några rättigheter: grant select on database::Test to my_user;

Förkortat skrivsätt: grant control … ger rätt att göra insert+update+delete.

Vad har jag för rättigheter? Testa med “select * from fn_my_permissions('Students', 'OBJECT');”.

En snyggare lösning hade naturligtvis varit att skapa en roll, ge rollen rättigheter, och lägga till en eller flera användare som medlemmar av rollen.
Utforska gärna på egen hand. Sök efter “create role” och “alter role” + “add member”.

Skapa index, begränsningar (constraints), mm

Introduktion till index.
Kommer någon ihåg bibliotekens gamla kortregister med alfabetisk ordning och systematisk ordning?

Index går att skapa för enstaka kolumner eller för kombinationer av två eller flera kolumner.
Poängen är att förenkla och snabba upp sökning i tabeller.
Nyttan av index är i högsta grad beroende av vad för slags frågor vi ställer till databasen.
Ska index vara i stigande eller fallande ordning?

Ett smart drag är att indexera kolumner som används vid join-operationer.
Primära nycklar blir alltid också index, men inte samma sak för främmande nycklar.

Om “select” går fortare kommer dock “insert”, “update” och “delete” att gå långsammare.
När vi ändrar i en tabell måste ju även index uppdateras.

Via till exempel “Estimated plan” i Azure Data Studio går att skapa sig bilder av hur olika SQL-frågor utförs.
Det går även att använda sig av SQL-kommandot “set showplan_xml on”.

En specialvariant av index är dem vi skapar med “unique”.

Andra slags begränsningar, eller “constraints”, vi har stött på tidigare:

- Default-värden (“default”).
- Giltiga värden (“check”).
- Obligatoriska fält (“not null”).

Använda transaktioner

Nyttan av att kunna använda transaktioner, sammanfattad med ett enkelt exempel:

```
begin transaction;
update Students set name='Nisse' where name='Johan';
select * from Students;
rollback transaction;
select * from Students;
```

Hade vi velat spara ändringarna skulle vi istället för “rollback” ha gjort “commit”.

“Allt eller inget.”

I princip ett måste att ha transaktionshantering när flera olika användare nyttjar samma databas.

Inne i en transaktion ser jag min uppdaterade eller ändrade “bubbla”. Övriga användare ser ännu inte ändringarna.

Men vad händer om två transaktioner låser varandra?

Lite överkurs: Med “save transaction” går också att sätta en “savepoint” i en transaktion.

Övningar

Om man inte är klar med gårdagens övningar går att spara dem till senare.

Testa att skapa egna login och users till din databas, och dela ut olika slags rättigheter till dem.
Pröva att skapa roller för att gruppera databasanvändare, och ge rollerna rättigheter, istället för att ge enskilda användare rättigheter.

Utforska olika sätt att granska “query execution plans”, och testa vilka slags index som verkar förbättra (eller försämra?) prestandan.

Ta reda på hur man kan använda funktioner i en check-constraint (till exempel för att jämföra med innevarande års årtal).

Bekanta dig med transaktionshantering.
Använd två olika klientprogram och undersök hur poster blir låsta för en användare om annan användare är inne i en transaktion.

Presentation av inlämningsuppgift

Inlämning senast fredag 2 december kl 16. I praktiken måndag 5 december kl 09.

Inlämning i pdf-format, ett samlat dokument, inga separata bilagor. Antalet sidor bör inte överstiga 20.

”Please, do not consider the page limit as a target!
It is in your interest to keep your text as concise as possible,
since experts rarely view unnecessarily long proposals in a positive light.”

Tänk på att hålla nere filstorleken. Om du infogar bilder, se till att komprimera dem först. Undvik tunga skärmdumpar.

Uppgiften ska redovisas individuellt. Under arbetets gång får du dock gärna ta hjälp av andra för att diskutera designval, problem mm.

Uppgiften och förutsättningarna är dessa:

Skapa en databas åt en hyresvärd i en stad som vill hålla reda på lägenheter uthyrda till personer.

En lägenhet har ett antal rum och en area i kvadratmeter.

En lägenhet är belägen på ett våningsplan och har ett ordningsnummer inom detta våningsplan (plan + nummer är unikt för varje adress).

En lägenhet har en gatuadress. På samma adress kan det finnas en eller flera lägenheter.

En gatuadress hör till ett hus. Ett hus kan ha en eller flera adresser (olika portar).

Ett hus ligger i ett kvarter, och har ett ordningsnummer inom det kvarter där det ligger. Ett kvarter kan ha ett eller flera hus.

Ett kvarter har en fastighetsbeteckning sammansattt av kommunnamn, traktnamn, blocknummer och enhetsnummer (tex “Stockholm Norrmalm 5:3”).

En person har personnummer, för- och efternamn, telefonnummer och epostadress. Telefonnummer och epostadress behöver inte vara unika.

En person kan endast hyra en lägenhet i taget. En lägenhet kan vara uthyrd till flera personer som delar på avtalet.

Ett hyresavtal gäller från ett startdatum och har en årshyra i kronor.
Om hyran ändras skapas ett nytt avtal med nytt startdatum, och det gamla avtalet förses med ett slutdatum som är dagen före det nya avtalets startdatum.

Se även uppgifterna på nästa sida för ledtrådar till vad som ska vara möjligt att få fram ur databasen.

Kom ihåg: Databasen kommer omöjligen kunna motsvara verkligheten fullständigt, men du ska åstadkomma bästa möjliga likhet med verkliga förhållanden.

Fortsättning på nästa sida ...

Inlämningsuppgift, fortsättning

Uppgiften ska lösas genom att …

1. Rita ER-diagram. Identifiera entiteter och relationer, samt kardinaliteter. Använd helst “crow’s foot notation”.
   Beskriv kortfattat hur du har tolkat kraven, och motivera de viktigaste designvalen du har gjort.
   Redovisa med bild av diagram och korta förklarande texter.
2. Skapa tabeller, med ER-diagrammet som underlag. Koppla relaterade tabeller med primära och främmande nycklar.
   Lägg till de regler för giltiga värden, unika index mm du anser är de viktigaste, och motivera kortfattat varför.
   Redovisa med SQL-kommandon för att skapa tabeller mm, i textform, med korta förklarande texter.
3. Lägga till exempeldata, antingen genom att infoga en post i taget eller genom att importera från textfiler,
   med som minimum två kvarter med vardera minst två hus, med i sin tur minst två lägenheter per hus, alla uthyrda till olika personer.
   Redovisa med SQL-kommandon för att lägga till data.
4. Göra följande urval från databasen.
   1. Lista aktiva hyresgäster i bokstavsordning, med uppgifter om adresser och övriga egenskaper för de lägenheter de hyr.
   2. Lista kvartersnamn i bokstavsordning, med uppgifter per kvarter om antal hus, antal lägenheter, och totala antalet kvadratmeter.
   3. Lista uppgifter om antal kvarter, antal hus, antal lägenheter, och totala antalet kvadratmeter, grupperat per traktnamn (del av fastighetsbeteckning).
   4. Lista vad hyrorna kommer att bli nästa kalendermånad för alla icke avslutade hyresavtal.

   Redovisa med SQL-kommandon för urval, med vid behov korta förklarande texter.

För betyget G ska uppgifterna 1-3 och 4.1-4.2 vara utförda på ett i stora drag godtagbart sätt.

För betyget VG ska uppgifterna 1-2 dessutom vara utförda med välgrundade resonemang kring designen, samt även uppgifterna 4.3-4.4 vara utförda.

Torsdag 24 november – Olika typer av databaser

Även denna dag undervisning på distans.

Obs! Idag endast kl 09.00 – 13.00.

Vid kl 12.00 samling för summering av dagen och möjlighet att ställa frågor.

En återblick på kursens övergripande innehåll

- Relationsdatabaser och hur olika servrar och databashanterare fungerar.
  => Tänk kvalitativt, kunna förstå på ett principiellt plan. Även bra att skaffa sig kännedom om skillnader mellan olika implementationer.
- Lagra data, i såväl fysisk som virtualiserad miljö.
  => Tänk även här på ett principiellt plan, att vara orienterad om begreppen. Inga krav på att förstå allt ner till filnivå.
- Administrera databaser, hantera användare, säkerhetskopiera mm.
  => Mer av "hands on" inom detta område. Dock inte ner på detaljnivå för att fintrimma.
- Modellera data och utforma databaser
  => Förhållandevis mycket ändå inom detta område, för att det är ett sätt att lära sig tolka mellan verksamhet och teknisk lösning.
- Normalisera och optimera databaser.
  => Även här kanske mer på ett principiellt plan, att ha nosat på de första stegen och vara orienterad om begreppen.
- Manipulera databaser och data, genom såväl direkta anrop som lagrade procedurer.
  => Ganska omfattande, men inte för att alla ska bli experter på SQL, utan snarare för att det är ett bra sätt att "lära sig kartan".

Skillnader i SQL mellan olika lösningar

Alla dessa olika databashanterare som går att välja mellan ...

Vi har redan jämfört mellan Sqlite och MS SQL Server. Lite som David vs Goliat.

För- och nackdelar med snabba flexibla lösningar jämfört med mer “stabila” implementationer.

Delvis olika syntax. Stora skillnader i funktionalitet.

ACID = Atomicity, Consistency, Isolation, Durability.

BASE = Basically Available, Soft state, Eventually consistent.

En pessimistisk approach vs en optimistisk?

Jämförelser mellan MySQL, PostgreSQL mm

Punkterna nedan redovisar endast några exempel på databashanterare och deras egenskaper. Fler går att räkna upp.

- MySQL / MariaDB
  - Går att välja mellan två olika “motorer”, InnoDB eller MyISAM (ACID respektive icke ACID).
  - Stöd för JSON.
  - Funktionsbaserade index.
  - Många användare av kombinationen Linux, Apache, MySQL, PHP.
  - MySQL Workbench, ett populärt verktyg för både modellering och administration.
- PostgreSQL[1]
  - Stöd för objektdatabaser (ORDBMS).
  - JSON-data.
  - Geografiska datatyper.
  - Rikt på funktioner.
  - Avancerad transaktionshantering.
- MS Access
  - Filbaserad databas (format .mdb).
  - Många användare på grund av att den följer med MS Office.
  - Bra för enkla lokala ändamål. Usel för applikationer med flera olika användare.
- Paradox
  - Filbaserad databas, med en fil per tabell, en fil per index, osv. (Udda lösning, som dock har sina fördelar.)
  - Starkt kopplat till applikationer i Delphi / Pascal.
  - Förekommer sällan i nya lösningar, men kan leva kvar i gamla system.

1 Lite nostalgi, en artikel om att köra PostgreSQL på Windows för 20 år sedan: https://www.sitepoint.com/use-postgresql-php-windows/

NoSQL

Betyder det No som i “None”, eller som i “Not only”? En bättre benämning kunde vara NoRel, Not Relational.

MongoDB är ett exempel, som inte har tabeller och rader på det sätt vi vant oss vid, utan är en databas bestående av dokument.
BSON = Binary encoded Javascript Object Notation (JSON).

Måste vi nu glömma allt vi lärt oss om att definiera schema?

Annorlunda sätt att göra urval.

- .find(…) istället för kommandot select
- “field: value” istället för villkoret “field=value”
- , (komma) istället för operatorn “and”
- $or: [… , …] istället för operatorn “or”

Fördelar:

- Flexibla strukturer.
- Inga join-operationer, eftersom hierarkiska data redan är ordnade i dokument (ibland inbäddade i andra dokument).
- Snabbare, och effektivare minneshantering.
- Skalbart. Bra för att processa och analysera stora datamängder. (Därav namnet “humongous”?)

Nackdelar:

- Kan vara svårt (för den som är van vid relationsdatabaser) att sätta sig in i annorlunda tänk kring “schema”.
- Inte samma strikthet kring datatyper.
- Inte lika utvecklad felhantering.
- Begränsat frågespråk.

Kort övning med MongoDB

Installationsanvisningar på https://www.mongodb.com/docs/manual/tutorial/ .

Skapa en ny databas “Nackademin”: use Nackademin

(Finns databasen nu? Testa med: show dbs Nej, den visas inte förrän vi faktiskt har lagt till data.)

Skapa en ny collection: db.createCollection("Students")

Lägg till data:

```
db.Students.insertMany( [
{ name: "Knatte", birth_year: 2010, courses: ["Databashantering", "Programmering"] },
{ name: "Fnatte", birth_year: 2010, courses: ["Databashantering"] },
{ name: "Tjatte", birth_year: 2010, courses: ["Databashantering"] }
] )
```

Att vi får lägga till en lista med inte bara en utan två eller flera kurser på en student verkar bryta mot alla regler om normalisering?

Gör urval, motsvarigheten till “select * from Students”: db.Students.find()

Lägg till sortering efter namn: db.Students.find().sort("name")

För att hitta en specifik student: db.Students.find({name: "Knatte"})

För att lägga till en kurs till på en student: db.Students.updateOne({ name: "Tjatte" }, { $push: { courses: "Järnvägsbyggande" } })

För att lägga till en ny egenskap på en student: db.Students.updateOne({ name: "Fnatte" }, { $set: { email: "fnatte@ankeborg.se" } })

(Får man verkligen göra så? Lägga till en “kolumn” som inte fanns tidigare, på bara en “rad”? Ja, faktiskt. För det är NoSQL.)

Läs mer på till exempel https://www.mongodb.com/docs/mongodb-shell/run-commands/ eller https://www.w3schools.com/mongodb/index.php .