Abschlussprojekt

Dozent: Lev A. Brodski
Modul: SQL

Aufgabe:
Eine relationale Datenbank entwerfen und implementieren

- dabei **Redundanzen** und **Inkonsistenzen** vermeiden
- DB soll **dynamische Daten** verwalten (nicht nur statische Tabellen)
- **Business-Logik implementieren** (s. unten - Prozeduren und Trigger)
- *für die Note „gut“* - mit 7 bis 9 Tabellen
- *für die Note „sehr gut“* - mit mind. 10 Tabellen
- **Primärindizes** und gegebenenfalls **Sekundärindizes** (`NONCLUSTERED INDEX`) erstellen
- **Beziehungen** zwischen Tabellen erstellen
- **Mindestens eine m:n Beziehung**
- **Standartwerte und Einschränkungen** erstellen
- **Diagramm** erstellen
- **CREATE - Skripte** speichern:
  - `CREATE DATABASE`
  - **Je ein Skript:**
    - `CREATE TABLE`
    - `CREATE NONCLUSTERED INDEX`
    - `ADD CONSTRAINT FOREIGN KEY`
    - `ADD CONSTRAINT CHECK`

Sichten (VIEW)
- ein SELECT - Sicht mit WHERE und ORDER BY über eine Tabelle erstellen
- ein SELECT - Sicht über mehreren Tabellen erstellen, die Tabellen mit INNER JOIN verknüpft
- ein SELECT - Sicht über mehreren Tabellen mit GROUP BY, SUM oder COUNT und HAVING erstellen
- *für die Note „sehr gut“* - Sicht über mehreren Tabellen erstellen, die Tabellen mit LEFT/RIGHT/FULL OUTER JOIN oder SELF JOIN verknüpfen
- **alle Skripte speichern**

Gespeicherte Funktionen
- eine **Skalarwertfunktion** erstellen
- eine **Tabellenwertfunktion** erstellen
- **Testskripte** erstellen
- **alle Skripte speichern**

Gespeicherte Prozeduren

- eine **gespeicherte Prozedur** mit Parameter erstellen
- die Prozedur soll **Business-Logik** implementieren
- die Prozedur soll **Eingangsparameter prüfen** und **Daten** in Tabellen **ändern**
- *für die Note „sehr gut “*: in der Prozedur eigene gespeicherte Funktionen verwenden
- *für die Note „sehr gut “*: in der Prozedur OUTPUT-Parameter für die Fehlermeldungen etc. verwenden
- Skripte für Ausführen erstellen
- **alle Skripte speichern**

Trigger

- ein Trigger (INSERT, UPDATE oder DELETE)
- *für die Note „sehr gut “*: Trigger soll Business-Prozesse automatisieren
- Testskript erstellen
- **alle Skripte speichern**

Anmeldungen, Benutzer, Rechte

- DB-Benutzer mit nur Leserechte anlegen
- DB-Benutzer mit Schreib- und/oder EXEC-Rechte anlegen

BACKUP

- DB sichern, .bak-Datei speichern
- Ein Skript für gesamte DB und alle DB-Objekte vom SSMS generieren lassen:
  - alle Tabellen, Sichten, Prozeduren etc.
  - alle Logins, User, Rollen
  - alle Daten
- Die zwei DB-Datei, .mdf und .ldf speichern (nicht obligatorisch, nur wenn nicht mehr zu tun ist:)

für die Note „sehr gut “:

- **CURSOR-Skript** erstellen
- oder **Daten importieren** (z.B. aus EXCEL)
- oder **mehrere** Funktionen, Prozeduren und/oder Trigger schreiben
- oder Access verbinden und **Front-End** entwickeln (1 Formular und 1 Bericht)

Projektdokumentation:

- Eine kurze Projektbeschreibung (s. Muster) als .pdf
- Datenbankdiagramm hinzufügen

Für alle Aufgaben gilt:

- Als Muster für die Skripte können die Übung-Skripte genutzt werden
- Benutzen Sie hierbei eigene Namen (D/EN, in camelCase oder PascalCase) für alle Ihre Funktionen, Variablen, Feldnamen etc

- Alle Skripte / Funktionen müssen ausreichend kommentiert sein (lieber auf Deutsch, aber Englisch ist auch OK)

Sollten Sie noch Zeit haben, erweitern Sie ihr Projekt nach Ihrem Geschmack und Kenntnisstand (mehr Tabellen, Funktionen, Prozeduren etc.).