Datenbank „Meine Firma“

Datenbank **erstellen**:
```
CREATE DATABASE Meine_Firma;
```
Bestimmte Datenbank **auswählen**:
```
USE Meine_Firma;
```
Datenbank **sichern**:
```
BACKUP DATABASE <Name>
   TO DISK = '<path>'
```
Datenbank wieder **herstellen**:
```
RESTORE DATABASE <Name>
   FROM DISK = '<path>'
```

Tabelle Ort

PLZ	Ort
3000	Bern
3036	Frieswil
3072	Ostermundigen
3110	Münsigen
3270	Aarberg
3280	Murten

Formale Tabellenschreibweise:
```
Ort (#PLZ, Ort)
```

Tabelle Kunde

kID	Name	Anrede	PLZ
1	Blum	Herrn	3000
2	Meier	Frau	3072
3	Haller	Frau	3036
4	Haller	Herrn	3072
5	Huber	Herrn	3280
6	Schmutz	Frau	

Formale Tabellenschreibweise:
```
Kunde (#kID, Name, Anrede,
       #PLZ)
```

CREATE TABLE Kunde

SQL-Statement:
```
CREATE TABLE Kunde (
   kID int IDENTITY(1,1)
                NOT NULL
   ,Name    nvarchar(20)
   ,Anrede  nvarchar(5)
   ,PLZ     int
);
```

TABLE Kunde löschen/leeren

Tabelle **löschen**:
```
DROP TABLE Kunde;
```
Tabelleninhalt **löschen**:
```
TRUNCATE TABLE Kunde;
```

```
Hinweis Datentyp (Auszug):
- char: Zeichen fester Länge
- varchar:   variable Länge
- nvarchar:  Unicode-Daten
- int, bigint:   Ganzzahl
- float, decimal: Kommazahl
```

INSERT

```
INSERT INTO Kunde
  (Name, Anrede, PLZ)
 VALUES
 ('Haller', 'Herrn', 3072),
 ('Huber','Herrn',
  (SELECT PLZ FROM Ort
    WHERE Ort = 'Murten'));
```
kID (PK) wird vom System ermittelt

UPDATE

```
UPDATE  Kunde
  SET Name = 'Meier',
      Anrede = 'Frau'
  WHERE kID = 2;

UPDATE  Kunde
 SET PLZ = (SELECT PLZ
  FROM Ort WHERE Ort = 'Bern')
  WHERE PLZ = 3012;
```

DELETE

Löscht Datensatz 3:
```
DELETE FROM Kunde
    WHERE kID = 3;
```
Löscht alle Datensätze mit Name M:
```
DELETE FROM Kunde
    WHERE Name LIKE M%;
```
Löscht alle Datensätze:
```
DELETE FROM Kunde;
(siehe TRUNCATE TABLE)
```

SELECT / DISTINCT

alle Attribute (*):
```
SELECT * FROM Kunde;
```
Attribute filtern:
```
SELECT Anrede, Name
  FROM Kunde;
```
Gibt Mehrfachdatensätze nur einmal aus (jeweils letzter Datensatz):
```
SELECT DISTINCT (Name)
  FROM Kunde;
```

Alias / Index

Alias: Benennung / Kürzel
Spalte: … `COUNT(*) AS “Anzahl“,…`
Tabelle: … `FROM Kunde AS k …`

Index erstellen:
```
CREATE INDEX IDX_<NAME>
  ON <Tabellenname>
      (<Spaltenname>)
```
Index löschen:
```
DROP INDEX IDX_<NAME>
  ON <Tabellenname>
```

WHERE Klausel

Datensätze filtern:
```
SELECT Name, PLZ FROM Kunde
   WHERE (PLZ = 3000
     OR PLZ = 3072)
     AND Anrede LIKE 'H%';
```
AND, OR: filtert Datensätze mit mehreren Bedingungen (AND vor OR)

GROUP BY-Klausel

Datensätze gruppieren und zählen:
```
SELECT COUNT(*) AS
“Anzahl“, Anrede FROM Kunde
   GROUP BY Anrede;
```

Anzahl	Anrede
4	Herrn
2	Frau

HAVING-Klausel

Gruppierte Datensätze filtern:
```
SELECT COUNT(*) AS “Anzahl“,
       Anrede FROM Kunde
   GROUP BY Anrede
   HAVING Anzahl >= 3;
```

Anzahl	Anrede
4	Herrn

ORDER-Klausel

Aufsteigend sortiert (gleiche Namen und zusätzlich nach Anrede):
```
SELECT Anrede, Name
                FROM Kunde
  ORDER BY Name, Anrede ASC;
```
Absteigend sortiert:
```
  ORDER BY Name DESC;
```

TOP-Klausel

Gibt eine bestimmte Anzahl Datensätze aus:
```
… TOP [Anzahl Datensätze]
```
Ausgabe von 10 Datensätzen:
```
SELECT TOP 10 * FROM Kunde;
```

Vergleichsoperatoren

Operator	Erklärung
= < <= >= > <>	vergleicht ein Attributwert mit einem anderen
BETWEEN ... AND ...	vergleicht, Attributwert zwischen zwei Grenzen
(NOT) LIKE	Vergleich von Zeichenketten (Muster) %: Platzhalter für beliebige Zeichen (≥ 0 Zeichen) _: Platzhalter für ein Zeichen (=1 Zeichen (Zahl, Charakter))
IS (NOT) NULL	prüft, ob ein Attributwert (nicht) undefiniert ist

MIN / MAX

MIN: liefert den kleinsten Attributwert
```
SELECT MIN(Lohn)
   FROM Person;
```
MAX: liefert den grössten Attributwert
```
SELECT MAX(Lohn)
   FROM Person;
```

SUM / COUNT

SUM: liefert die Summe einer Spalte
```
SELECT SUM(Lohn)
    FROM Person;
```
COUNT: zählt die Anzahl vorhandenen (nicht {Null}) Einträge
```
SELECT COUNT(Name)
    FROM Kunde;
```

ROUND / AVG

ROUND: 2 Kommastellen runden

AVG: Mittelwert der Attributwerte
```
SELECT ROUND(AVG(Lohn), 2)
    FROM Person;
```

Tabelle Ort

PLZ	Ort
3000	Bern
3036	Frieswil
3072	Ostermundigen
3110	Münsigen
3270	Aarberg
3280	Murten

Formale Tabellenschreibweise:
`Ort (#PLZ, Ort)`

Tabelle Kunde

kID	Name	Vorname	PLZ
1	Blum	Hans	3000
2	Meier	Lisa	3072
3	Haller	Greti	3036
4	Haller	Anton	3072
5	Huber	Peter	3280
6	Schmutz	Ruedi	

Formale Tabellenschreibweise:
```
Kunde (#kID, Name, Vorname,
       #PLZ)
```

Tabellenname ändern

```
ALTER TABLE Ort
      RENAME Ortschaft;
```

Ändert den Namen der Tabelle "Ort" zu "Ortschaft".
ACHTUNG: Die Beziehungen werden NICHT automatisch mitgeändert!

Spaltenname ändern

```
ALTER TABLE Kunde
      CHANGE Name Nachname
                  VARCHAR(20);
```

Spalte löschen / einfügen

Spalte löschen:
```
ALTER TABLE Kunde
   DROP COLUMN Vorname;
```

Spalte einfügen:
```
ALTER TABLE Kunde
   ADD COLUMN Geschlecht
       CHAR(1);
```

(INNER) JOIN

```
SELECT o.Ort, k.Name
       FROM Ort as o
    JOIN Kunde As k
       ON o.PLZ = k.PLZ;
```
Liefert alle Orte mit den Kunden (d.h. nur grau markierte Datensätze in Tabelle 1)

Mehrfach-JOIN:
```
... JOIN x ON x.a = o.a
    JOIN y ON y.b = k.b ...
```

LEFT JOIN

```
SELECT o.Ort, k.Name
       FROM Ort as o
    LEFT JOIN Kunde As k
       ON o.PLZ = k.PLZ;
```
Liefert alle Orte auch ohne Kunden (d.h. liefert alle Datensätze der Tabelle 1)

Ausgabe JOIN & LEFT JOIN

Tabelle 1

Ort	Name
Bern	Blum
Frieswil	Haller
Ostermundigen	Meier
Ostermundigen	Haller
Münsigen	{null}
Aarberg	{null}
Murten	Huber

RIGHT JOIN

```
SELECT o.Ort, k.Name
       FROM Ort as o
   RIGHT JOIN Kunde As k
       ON o.PLZ = k.PLZ;
```
Liefert alle Kunden auch jene Kunden ohne Ort
(d.h. liefert die Datensätze der Tabelle 2)

Ausgabe RIGHT-JOIN

Tabelle 2

Ort	Name
Bern	Blum
Ostermundigen	Meier
Frieswil	Haller
Ostermundigen	Haller
Murten	Huber
{null}	Schmutz

CONSTRAINT (Integritätsregeln)

Primärschlüssel Attribut festlegen:
```
… CONSTRAINT PK_Kunde
     PRIMARY KEY (kID)
```
Fremdschlüssel Attribut festlegen:
```
… CONSTRAINT FK_Kunde_Ort
     FOREIGN KEY(PLZ)
     REFERENCES Ort (PLZ)
```

Bedingung nachträglich festlegen:

Primärschlüssel:
```
ALTER TABLE Kunde
   ADD CONSTRAINT PK_Kunde
      PRIMARY KEY (kID);
```
Fremdschlüssel
```
ALTER TABLE Kunde
   ADD CONSTRAINT FK_Kunde_Ort
      FOREIGN KEY(PLZ)
      REFERENCES Ort (PLZ);
```
(FK-Wert -> PK-Wert)

Referentielle Integritätsbedingungen:
- … ON DELETE <Option>
- … ON UPDATE <Option>
- Option CASCADE
  führt zum Löschen bzw. Ändern der FK
- Option RESTRICT (Defaultverhalten)
  nur möglich wenn kein FK besteht
- Option SET NULL
  FK-Werte werden NULL
- Option SET DEFAULT
  FK-Werte werden zum DEFAULT-Wert

Berechtigungen

Berechtigung zuteilen:
```
GRANT <Rechte> ON <TabName>
      TO <Username/Role>
```
Berechtigung entfernen:
```
REVOKE <Rechte> ON <TabName>
       FROM <Username/Role>
```
Alle Rechte: `ALL`

Einzelne Rechte: `SELECT, UPDATE INSERT …`

Rolle

Rolle **erstellen**:
```
CREATE ROLE <Name>
```
Rolle **löschen**
```
DROP ROLE <Name>
```
Rolle **zuordnen/entfernen**
```
ALTER ROLE <Role>
   ADD/DROP MEMBER <Username>
```
MS SQL-Server:
```
EXEC sp_addrolemember
   '<Role>', '<Username>';

EXEC sp_droprolemember
   '<Role>', '<Username>';
```

FETCH (Mengenverarbeitung)

Variable deklarieren:
```
DECLARE @<VarName> int;
```
Cursor deklarieren:
```
DECLARE <cursorname>
   CURSOR FOR <SELECT-
               Anweisung>
```

Cursor **öffnen**:
```
OPEN <Cursorname>;
```
Cursor **lesen**:
```
Fetch NEXT FROM <Cursorname>
   INTO @<VarName>;
```

@@FETCH_Status **prüfen**:
```
WHILE @@FETCH_STATUS = 0
BEGIN
 … mit Datensatz etwas tun …
 FETCH NEXT FROM <cursorname>
   INTO @<VarName>
END
```

Cursor **schliessen**:
```
CLOSE <Cursorname>;
```
Ressourcen **freigeben**:
```
DEALLOCATE <Cursorname>;
```

Login / User

Login erstellen:
```
CREATE Login <LoginName>
 WITH PASSWORD='<Passwort>';
```
Benutzer zu Login erstellen:
```
CREATE USER <UserName>
  FOR LOGIN <LoginName>
  WITH DEFAULT_SCHEMA = dbo;
```