Datenbanken im WI-Unterricht mit

Inhaltsverzeichnis

Vorbemerkungen

Im Unterricht sollen die Grundlagen der Datenbankverarbeitung am Beispiel der relationalen Datenbank behandelt werden. Dazu zählen die Entwicklung eines Entity-Relationship-Modells, dessen Überführung in Tabellen einer Datenbank und die Bearbeitung dieser Datenbank mittels der Sprache SQL. Als Datenbankmanager wird SQLite[1] genutzt. Dieser ist nach eigenen Angaben das am meisten eingesetzte Datenbanksystem. Da es eine Datenbank in einer Datei verwaltet, wird es gerne in Web-Anwendungen und auch Apps eingesetzt. Auch ist eine spätere Einbindung in Java-Projekte im Unterricht recht einfach zu implementieren. Allerdings unterstützt SQLite nicht den vollständigen Befehlssatz von SQL 92, was im Unterricht allerdings zu keinerlei Einschränkungen führt.

1 ER-Modell - Entity Relationship Modell

Ein ER-Modell dient der graphischen Darstellung von Entitäten (= Objekten), deren Attribute (= Eigenschaften) und deren Beziehung untereinander.

[1] Homepage `https://sqlite.org`

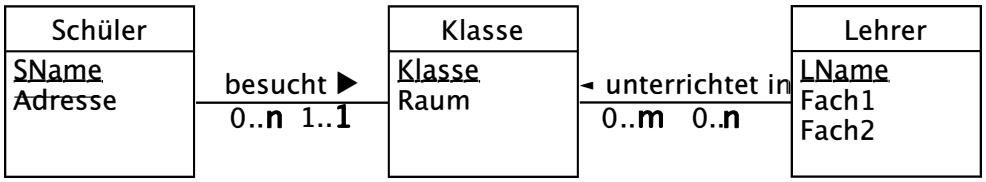
1.1 Entitäten

In diesem Diagramm[2] sind verschiedene Entitäten dargestellt. Im oberen teil steht der Name der Entitätsmenge (in OO der Klasse), darunter die Attribute (in OO sind dies die Instanzvariablen). Die unterstrichenen Attribute sind Schlüsselattribute. Diese Diagramme ähneln den Klassendiagrammen. Auffälligster Unterschied sind die fehlenden Methoden.

„Schüler“ beschreibt genauer eine **Entitätsmenge**, die einzelnen Schüler sind dann die **Entitäten** dieser Menge mit ihren **Attributen** wie SName = Schülername und Adresse.

1.2 Relationen

Vervollständigt wird das Diagramm durch die Beziehungen (= Relations).



Die Leseweise beginnt immer mit **einer** Entität. Hier die verbale Beschreibung des obigen Beispiels.

- **Ein** Schüler[3] besucht 1 bis **1** also genau eine Klasse.
- Umgekehrt sind in **einer** Klasse 0 bis **n** also keiner bis viele Schüler.
- In einer Klasse unterrichten 0 bis **n** also keiner bis viele Lehrer.
- **Ein** Lehrer unterrichtet in 0 bis **m** also keiner bis vielen Klasse(n).

2 Logisches Tabellen- oder Relationenmodell

Der nächste Schritt überführt das validierte ER-Modell in das logisches Tabellenmodell. In diesem wird beschrieben, wie die zu erstellenden Tabellen aussehen müssen.

2.1 Transformationsregeln

Die Transformation läuft unter folgenden Regeln ab:

1. Jede **Entität** wird zu einer Tabelle. Die Attribute sind die Spalten der Tabelle.
2. **Beziehungen**

   Bei den Kardinalitäten ist nur die zweite Kardinalitäts-Angabe „..**1**“ , „..**n**“ oder „..**m**“ von Bedeutung.

   1:n (hier „Klasse zu Schüler“) wird wie folgt umgesetzt:

   Der Schlüssel der Detailtabelle (1 bzw. Klasse) wird Fremdschlüssel und damit weiteres Attribut der n-Tabelle (Schüler). (umgekehrt macht keinen Sinn!)

   n:m (hier „Lehrer zu Klasse“) wird zu einer neuen Tabelle mit den Schlüsseln der beteiligten Tabellen als zusammengesetzter Primärschlüssel.

[2] Die Diagramme sind mit UMLet (`http://www.umlet.com`) erstellt worden.

[3] Im gesamten Text wird immer die männliche Grundform der leichteren Lesbarkeit wegen verwendet. Es sind aber immer alle Geschlechter damit gemeint.

2.2 Am Beispiel

Diese Transformation wird nun an obigem Beispiel durchgeführt werden.

Regel 1:

- **Schüler**(SName[4], Adresse)
- **Klasse**(Klasse, Raum)
- **Lehrer**(LName, Fach1, Fach2)

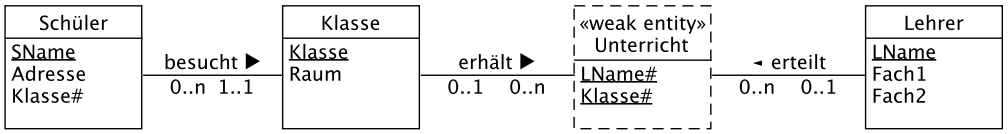
Regel 2.1: (1:n-Beziehungen):

- Schüler(SName, Adresse, **Klasse#** [5])
- Klasse(Klasse, Raum)
- Lehrer(LName, Fach1, Fach2)

Regel 2.2: n:m-Beziehungen):

- Schüler(SName, Adresse, Klasse#)
- Klasse(Klasse, Raum)
- Lehrer(LName, Fach1, Fach2)
- **LehrerKlasse**(LName#, Klasse#)[6] oder mit anderen Namen **Unterricht**(LName#, Klasse#)

Auch dieses kann wieder in einem (ER-ähnlichen!) Diagramm dargestellt werden.



Wichtig ist hier, dass nun keine m:n-Beziehungen aufgelöst werden müssen und für alle 1:n-Beziehungen die Fremdschlüssel vorhanden sind. Ohne diese Fremdschlüssel sind die Tabellen nicht untereinander logisch verbunden. Diese Diagramme sieht man auch im Beziehungsfenster von MS Access.

3 Implementierung in einer Relationalen Datenbank

Mit Hilfe der SQL können diese Tabellen nun auch in einer Datenbank erzeugt werden. Dazu sollten alle Namen nur mit ANSI-Zeichen (also insbesondere ohne Umlaute) und in „Kamelhöcker'-Schreibweise dargestellt werden und - falls nicht vorgegeben - mit sinnvollen Datentypen ergänzt werden.

Dies könnte beispielsweise so aussehen. Die genaue Syntax kann dem folgenden Kapitel entnommen werden.

```
CREATE TABLE "Schueler" (
        "SName" TEXT,
        "Adresse"       TEXT,
        "Klasse"        TEXT,
        PRIMARY KEY("SName")
);

CREATE TABLE "Klasse" (
        "Klasse"        TEXT,
        "Field2"        INTEGER,
        PRIMARY KEY("Klasse")
);
```

[4] Schlüsselattribute werden unterstrichen dargestellt
[5] Fremd-Schlüsselattribute werden durch „#"gekennzeichnet
[6] Diese Tabelle (oder Entität) existiert nur, wenn ein Lehrer in einer Klasse unterrichtet. Daher nennt man dies Art von Entität auch „schwache Entität" oder „weak entity".

```
CREATE TABLE "Unterricht" (
        "LName" TEXT,
        "Klasse"        TEXT,
        PRIMARY KEY("LName","Klasse")
);

CREATE TABLE "Lehrer" (
        "LName" TEXT,
        "Fach1" TEXT,
        "Fach2" TEXT,
        PRIMARY KEY("LName")
);
```

Die Fremdschlüssel-Beziehungen werden später noch präziser definiert (Stichwort „Referentielle Integrität“).

4 SQL - Structured Query Language

Die Syntax der einzelnen Befehle ist hier für den Unterricht reduziert angegeben. SCHLÜSSELWORTE sind in Großbuchstaben und *Variable* oder *Werte* in kursiver Schrift dargestellt. Optionale Angaben sind in eckigen Klammern „[*optional*]“ aufgeführt, Alternativen durch einen senkrechten Strich „|“ getrennt.

Alle Befehle sind detaillierter unter `https://www.sqlitetutorial.net/` zu finden.

4.1 Datenbankstruktur bearbeiten

Diese Sprachelemente dienen zum Bearbeiten der **Struktur** der Datenbank, d. h. zum Anlegen, Ändern und Löschen von Datenbanken und Tabellen sowie weiteren Elementen. Die Befehle gehören zur Untergruppe **Data Definition Language - DDL**.

4.1.1 Create - Erstellen

```
CREATE TABLE [IF NOT EXITS] Tabellenname (
        Attributname     Typ   [NOT NULL],
        ...,
        [PRIMARY KEY(Attributname, ....) ,]
        [FOREIGN KEY (Attributname) REFERENCES
                                Tabellenname (Attributname)]
           [ON UPDATE Aktion]
           [ON DELETE Aktion]
);
```

Als **Typ** stehen TEXT, BLOB, INTEGER, REAL, NUMERIC zur Verfügung.
Als **Aktion** stehen SET NULL, SET DEFAULT, RESTRICT, CASCADE und NO ACTION zur Verfügung.

4.1.2 Alter - Ändern

```
ALTER TABLE   Tabellenname
        [RENAME TO NeuerTabellenName]
oder
        [ADD COLUMN Attributname Typ]
;
```

4.1.3 Drop - Löschen

```
DROP TABLE [IF EXISTS]   Tabellenname;
```

4.2 Daten berarbeiten

Diese Sprachelemente dienen zum Bearbeiten des **Inhalts** von Tabellen, d. h. zum Einfügen, Ändern und löschen von Zeilen in einer Tabelle. Diese Befehle gehören zur Untergruppe **Data Manipulation Language - DML**.

4.2.1 Insert - Einfügen

```
INSERT INTO Tabellenname
        Attributname, ...)
        VALUES (NeuerWertVonAttribut, ....)
[WHERE Bedingung] ;
```

4.2.2 Update - Ändern

```
UPDATE Tabellenname
        SET Attributname = NeuerWert
 [WHERE Bedingung] ;
```

4.2.3 Delete - Löschen

```
DELETE FROM Tabellenname
 [WHERE Bedingung] ;
```

4.3 Daten abfragen

Der SELECT- Befehl wird auch schon einmal einer eigenen Kategorie DQL (Data Query Language) meistens jedoch auch der DML zugeordnet.

Dieser Befehl wird wohl am häufigsten verwendet.

```
SELECT [ALL | DISTINCT]
        * | Attributname oder Formel [AS Alias] | Aggregatfunktion(...) , ...
 FROM TABLE Tabellenname [Alias] , ...
 [WHERE Bedingung]
 [GROUP BY Attributname oder Alias, ... [HAVING Bedingung]]
 [ORDER BY Attributname oder Alias, ... [ASC | DESC] ] ;
```

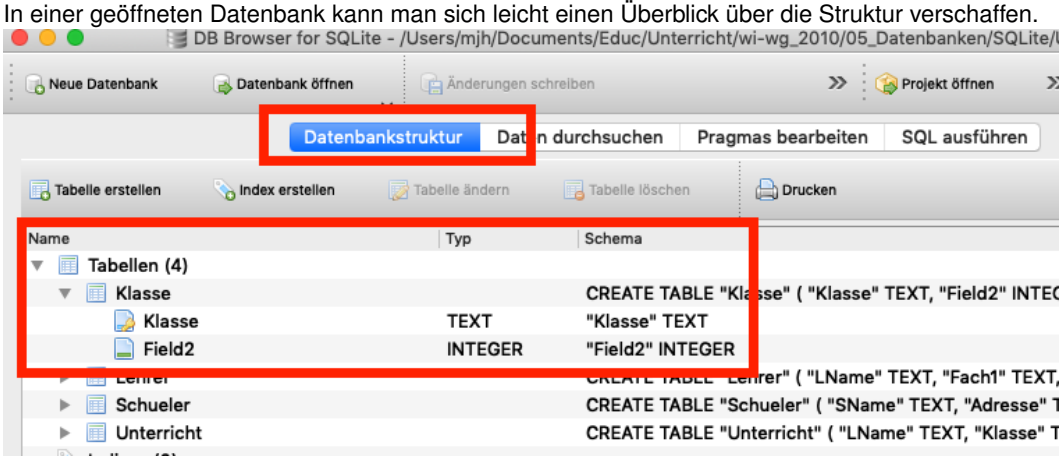
Aggregatfunktionen sind COUNT(...), SUM(...), AVG(...), MAX(..) oder MIN(...).

5 Die wichtigsten Funktionen des DB Browser for SQLite

Das im Unterricht zur Bearbeitung von SQLite-Datenbanken verwendete Programm kann unter `https://sqlitebrowser.org/` heruntergeladen werden.

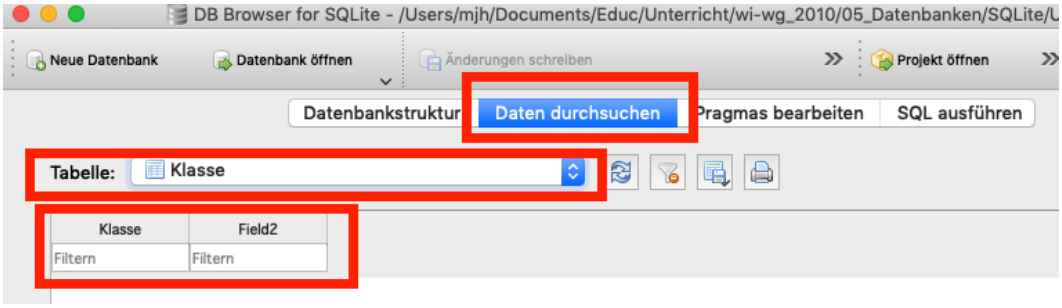
Die Datenbank in diesem System besteht immer nur aus einer Datei gewöhnlich mit der Endung „db“. Man kann auch ganze Projekte inklusive Abfragen als „sqbpro‘-Datei sichern. Die Datenbank wird aber nur als Link in der Projektdatei gespeichert.

In einer geöffneten Datenbank kann man sich leicht einen Überblick über die Struktur verschaffen.

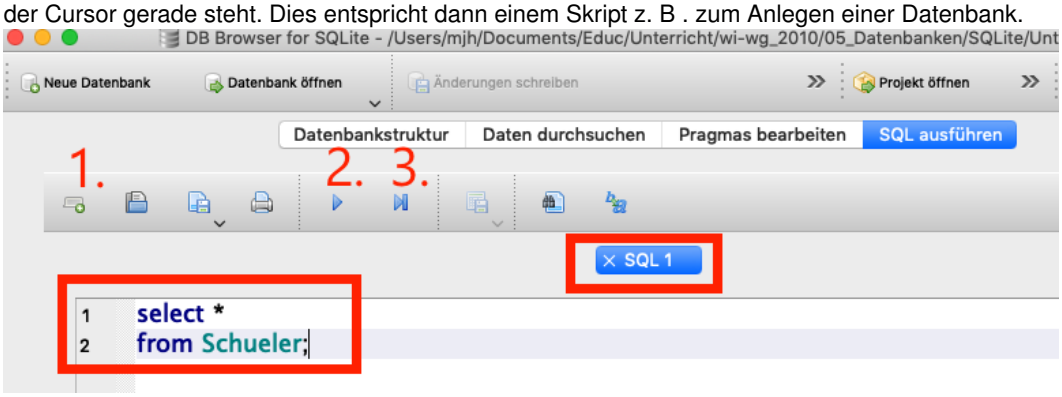


Hier können auch Tabellen angelegt, verändert bzw. gelöscht werden, ohne SQL einzusetzen.

Über den Reiter „Daten durchsuchen“verschafft man sich einen Überblick über den Inhalt der einzelnen Tabellen. Dies entspricht der Ausgabe des Befehls *SELECT * from Tabellenname;*.



Im SQL-Fenster stehen alle SQL-Befehle zur Verfügung. Mit **1.** erzeugt man ein neues SQL-Eingabefenster, mit **2.** werden alle Befehle im Fenster hintereinander ausgeführt und mit **3.** wird der Befehl ausgeführt, in dem der Cursor gerade steht. Dies entspricht dann einem Skript z. B . zum Anlegen einer Datenbank.



Weitere Möglichkeiten finden Sie in der Hilfefunktion des Programms. Und viele SQL-Beispiele sind unter `https://www.sqlitetutorial.net/` zu finden.