Geographic Information Technology Training Alliance (GITTA) presents:

Die relationale Anfragesprache SQL

Verantwortliche Personen:

Anca Dobre

(Content)

Susanne Bleisch

(Revision)

Adrian Moser

(Revision)

Michael Schrattner

(Revision)

Inhaltsverzeichnis

1. Die relationale Anfragesprache SQL

SQL (Structured Query Language) ist eine standardisierte Abfragesprache, die alle erforderlichen Sprachelemente enthält, um sämtliche Arbeiten, die beim Umgang mit einer relationalen Datenbank anfallen, auszuführen. Erste Versionen von SQL hiessen SEQUEL, SEQUEL2. Sie gingen aus SQUARE hervor, einer eher mathematisch orientierten Sprache. SQL ist ein ISO- und ANSI-Standard. Da sich SQL im Laufe ihrer Geschichte weiterentwickelt hat, kann man jedoch nicht von einem einzigen Standard sprechen. Der derzeit gültige Standard ist SQL-99. In den folgenden Units geben wir eine vereinfachende Übersicht von SQL.

Diese Lektion orientiert sich am Standard SQL-99 (GULUTZAN et al. 1999). Dieser wird jedoch nicht von allen Datenbanksystemen konsquent umgesetzt, z.B. sind gewisse Befehle anders benannt oder sind syntaktisch anders aufgebaut. Aus diesem Grund könnten SQL Befehle aus dieser Lektion in gewissen Systemen nicht funktionieren. In solchen Fällen muss anhand der jeweiligen Benutzeranleitungen festgestellt werden, ob der Befehl überhaupt unterstützt wird, anders benannt ist oder syntaktisch anders aufgebaut ist. Die meisten Datenbanksysteme gehen auch über den Standard hinaus. Bei diesen SQL Erweiterungen ist jedoch Vorsicht geboten, da sie sich von System zu System stark unterscheiden können.

Lernziele

- Sie verstehen die grundsätzlichen SQL-Konzepte und können den Einsatz von SQL in den Bereichen Datendefinition, Datenmanipulation und Datenkontrolle erläutern.
- Sie sind in der Lage, SQL für das Erstellen, Ändern und Löschen von Tabellen einzusetzen.
- Sie können mit Hilfe von SQL einfache und komplexe Anfragen formulieren und wissen, wie SQL-Anfragen in Schachtelungen oder im Verbund eingesetzt wird. Sie sind zudem in der Lage arithmetische Operatoren und Mengenoperatoren innerhalb von SQL-Anfragen korrekt anzuwenden.
- Sie beherrschen SQL beim Einfügen, Ändern und Löschen von Daten-Tupeln.

1.1. SQL-Konzepte

Die Sprache SQL (Structured Query Language) wird als einer der Hauptgründe für den kommerziellen Erfolg von relationalen Datenbanken in der Geschäftswelt angesehen. Die ANSI (American National Standards Institute) und die ISO (International Standards Organization) entwickelten eine Standard-Version von SQL (ANSI 1986) mit dem Namen SQL-86 oder SQL1. In den 90er Jahren wurde der erweiterte Standard SQL2 oder SQL-92 angewendet. SQL3 oder SQL-99 ist der im Moment gültige Standard. Durch den Einsatz von SQL in kommerziellen DBMS-Produkten wurde die Migration zwischen einzelnen DBMS-Produkten, die den Standard implementiert haben, für den Endbenutzer vereinfacht. Der Benutzer von relationalen Datenbanken muss sich im Idealfall keine Gedanken mehr machen, mit welchem DBMS-Produkt er momentan arbeitet, denn die Schnittstelle (SQL) zu den einzelnen Systemen bleibt für ihn immer die gleiche.

1.1.1. SQL-Grundlagen

Die Grundlage für die Datenmanipulation im Relationenmodell bildet die relationale Algebra. In ihrer unmittelbaren Form hat sie heute jedoch als Datenmanipulationsprache (DML) stark an Bedeutung verloren, da sie für Laien nur schwer zu durchschauen ist. SQL (Structured Query Language) hingegen hat sich seit Mitte der 80er Jahre als DML für relationale Systeme durchgesetzt. Es handelt sich dabei um eine deskriptive, mengenorientierte Sprache. Sie ist sowohl selbständig als auch eingebettet in eine Wirtssprache (C, C++, Java, PHP, ASP, u.v.m.) verfügbar.
Die Möglichkeiten von SQL beschränken sich nicht allein auf die Datenmanipulation (DML). SQL kann in drei konzeptionelle Einheiten aufgeteilt werden.

- Datendefinition (DDL – Data Definition Language) – zuständig für die Erstellung und Veränderung der Struktur der Datenbank
- Datenmanipulation (DML – Data Manipulation Language) – zuständig für den (Daten-) Inhalt der Datenbank (Daten hinzufügen, ändern, löschen, abfragen,...)
- Datenkontrolle / -steuerung (DCL – Data Controlling Language) – zuständig für die Sicherheit der Datenbank

Zusätzlich wurden in den meisten Implementierungen von SQL Konstrukte aus Programmiersprachen hinzugefügt (Fallabfragen, Iterationen etc.), so dass Spracherweiterungen wie PL/SQL (Oracle) schon fast als eigenständige Programmiersprachen angesehen werden können.

1.1.2. Datendefinition (DDL)

In SQL werden die Begriffe TABLE, ROW und COLUMN synonym für Relation, Tupel und Attribute gebraucht. Mittels des CREATE TABLE-Befehls wird ein Relationenschema in der Datenbank definiert. Das Relationenschema muss genau spezifiziert werden, oder in anderen Worten, die zur Relation gehörenden Attribute sowie deren Domänen müssen angegeben werden. Zusätzlich sind noch eine Reihe weiterer Deklarationen möglich wie z. B. Wertebeschränkungen (CHECK-Klausel), Standardwerte oder Primär- und Fremdschlüsseldeklarationen.
Beispiele für Domänen in SQL sind CHAR, NUMBER, LONG und DATE. Durch die Deklaration NOT NULL wird festgelegt, dass für das jeweilige Attribut keine NULL-Werte zulässig sind. Folglich muss beim Einfügen eines Tupels grundsätzlich ein Wert für dieses Attribut angegeben werden (es sei denn, ein Wert wird vorgeschrieben oder automatisch generiert). Primärschlüssel werden durch eine sogenannte Relationenbedingung (TABLE CONSTRAINT) mit Hilfe der PRIMARY KEY-Klausel deklariert.

```
create table Abonnement (
      Name char (30) not null,
      KundNr number (5) not null,
      AboArt char (15) default 'jährlich' check (AboArt in ('jährlich', 'halbjährlich')),
      AboBeginn date,
            constraint pk_abo primary key (Name, KundNr),
            constraint fk_name foreign key (Name) references Zeitung (Name),
            constraint fk_kundnr foreign key (KundNr) references Kunde (KundNr));
```

Code-Beispiel: CREATE TABLE

Dies ist ein Beispiel, wie eine Datenbanktabelle mit Hilfe von SQL definiert werden kann. Teile der Datendefinition (DDL) werden in der Unit Erstellen und Ändern von Tabellen [Seite 8] vertieft behandelt.
Für den Moment müssen Sie dieses Beispiel-SQL-Statement nicht im Detail verstehen.

1.1.3. Datenmanipulation (DML)

Die Möglichkeiten der Datenmanipulation mit SQL können in zwei Kategorien eingeteilt werden. Die eine Gruppe bilden die Datenbankanfragen, welche Inhalte abfragen, aber keine Änderungen an den Daten vornehmen. Die zweite Gruppe sind die Datenmanipulationen, die die Datenbank verändern, indem sie zum Beispiel Daten einfügen, löschen oder abändern.
Die Datenbankanfragen werden in der Unit Datenbankanfragen [Seite 13] detaillierter behandelt. Die Unit Einfügen, Löschen und Ändern [Seite 28] befasst sich mit den Datenbankänderungen.

1.1.4. Datenkontrolle / -steuerung (DCL)

SQL-Befehle der Datenkontroll-Sprache (DCL) kontrollieren die Sicherheit und die Zugriffsrechte für Objekte oder Teile eines Datenbanksystems. Diese Befehle liegen näher bei der Sprache des DBMS und können in verschiedenen Implementierungen der SQL stark varieren.
Typische Befehle sind:

- `GRANT` – vergibt Zugriffsrechte
- `DENY` – verweigert Zugriffsrechte
- `REVOKE` – löscht vorher vergebene oder verweigerte Zugriffsrechte

Die Befehle zur Datenkontrolle oder -steuerung werden im Rahmen dieses Moduls nicht detailliert behandelt.

1.2. Erstellen und Ändern von Tabellen

In dieser Unit wird gezeigt, wie mittels SQL-Befehlen Tabellen erstellt, verändert und gelöscht werden können.

1.2.1. Tabellen erstellen

Mit `CREATE TABLE` kann in einer Datenbank eine neue Tabelle erstellt werden. Der Befehl hat folgende Grundstruktur:

```
CREATE TABLE <Tabellenname> (<Attributdefinitionen und Beschränkungen>);
```

Der Tabellenname muss innerhalb der aktuellen Datendank oder des aktuellen Schemas eindeutig sein.

Attributdefinition

Attribute werden mit einem Namen und einem Datentyp definiert, wobei der Namen innerhalb der Tabelle eindeutig sein muss. Diese Angaben sind für alle Attribute zwingend notwendig.
Die Reihenfolge der Attribute bei der Definition entspricht der Reihenfolge der Spalten in der erstellten Tabelle. Wird eine bestimmte Ordnung angestrebt ist diese bereits bei der Erstellung der Tabelle zu definieren. Danach bleibt sie, sofern keine Änderungen an der Struktur der Tabelle vorgenommen werden (siehe Tabellenstruktur ändern [Seite 11]) bestehen.

Beschränkungen

Es gibt zwei Arten von Beschränkungen: Tabellenbeschränkungen und Attributbeschränkungen. Der Unterschied liegt darin, dass Attributbeschränkungen sich auf nur ein Attribut beziehen und Tabellenbeschränkungen sich auf meherere Attribute beziehen können, dies aber nicht müssen. Mit diesen Beschränkungen kann der Wertebereich der Attribute eingeschränkt werden oder es wird verhindert, dass Werte eingegeben werden, die nicht erlaubt sind. Ein Datensatz kann nicht erfasst werden, wenn er eine Beschränkung verletzt.
Es gibt vier Arten von Beschränkungen, welche in der Folge kurz beschrieben werden:

- `UNIQUE` - das Attribut oder die Attributkombination muss innerhalb der Tabelle eindeutig sein
- `PRIMARY KEY` - das Attribut oder die Attributkombination ist Primärschlüssel der Tabelle
- `FOREIGN KEY` - das Attribut ist ein Fremdschlüssel
- `CHECK` - Bedingung die für ein Attribut oder eine Attributkombination erfüllt sein muss

Die Beschränkugen können benannt werden. Dies ist jedoch nicht notwendig.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

In diesem Beispiel wird einer Datenbank eine Tabelle hinzugefügt, weche mit einer bereits bestehenden Tabelle verknüpft ist. Im SQL Befehl kommt der Unterschied zwischen Attribut- und Tabellenbeschränkung zum Ausdruck. projekt_ID und leiter_ID haben eine Attributbeschränkung (Beschränkung wird direkt hinter die Attributdefiniton geschrieben). projekt_ID hat die Beschränkung PRIMARY KEY, ist also Primärschlüssel dieser Tabelle, d.h. das Attribut muss eindeutig und nicht NULL sein. leiter_ID hat die Beschränkung NOT NULL (Spezialfall einer CHECK Beschränkung), muss also immer einen Wert enthalten.
Der Verknüpfung mit der bestehenden Tabelle ist als Tabellenbeschränkung definiert (FOREIGN KEY) und hat einen Namen (projektleiter). Diese Beschränkung könnte jedoch auch als Attributbeschränkung definiert werden, da sie nur ein Attribut beinhaltet.

Das Beispiel zeigt, dass grundsätzlich kein Unterschied zwischen Attribut- und Tabellenbeschränkungen besteht, sofern nur ein Attribut betroffen ist. Es handelt sich dabei um zwei verschiedene Möglichkeiten Beschränkungen zu erfassen.

1.2.2. Tabellenstruktur verändern

Mit `ALTER TABLE` kann die Struktur einer Tabelle geändert werden. Es können somit die mit CREATE TABLE erzeugten Attribute und Beschränkungen geändert, neue hinzugefügt oder vorhandene gelöscht werden. Der Befehl hat folgende Syntax:

```
ALTER TABLE <Tabellenname> <Änderung>;
```

wobei `<Änderung>` verschieden Befehle beinhalten kann:

- `ADD [COLUMN] <Attributdefintion>`
  Attribut hinzufügen (Attributdefiniton wie bei CREATE)
- `ALTER [COLUMN] <Attributname> SET DEFAULT <Standardwert>`
  neuer Standardwert festlegen
- `ALTER [COLUMN] <Attributname> DROP DEFAULT`
  aktuellen Standardwert löschen
- `DROP [COLUMN] <Attributname> {RESTRICT | CASCADE}`
  löschen eines Attributes
- `ADD <Tabellenbeschränkung>`
  neue Tabellenbeschränkung hinzufügen (Tabellenbeschränkung wie bei CREATE)
- `DROP CONSTRAINT <Tabellenbeschränkung>`
  löschen einer Tabellenbeschränkung

Mit den oben genannten Befehlen können Attribute und Beschränkungen hinzugefügt bzw. gelöscht werden. Zudem können die Standardwerte für die Attribute gesetzt oder gelöscht werden. SQL sieht noch andere Befehle vor die hier nicht erwähnt sind.
SQL beinhaltet im Standard keine Befehle um Attribute zu ändern oder umzubnennen. Dies würde auch zu Problemen führen, wenn bereits Daten vorhanden sind. Denoch sind diese Befehle in einigen Datenbanken vorhanden (z.B. MODIFY oder RENAME). Die Syntax unterscheidet sich jedoch von System zu System. Wenn keine Daten vorhanden sind, kann das Attribut, das geändert werden soll, gelöscht und neu hinzugefügt werden.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

In diesem Beispiel wird einer Tabelle ein neues Attribut hinzugefügt. Der angezeigte Datensatz enthält danach NULL für dieses Attribut, weil noch kein Wert zugewiesen wurde. Anschliessend wird dieses Attribut wieder aus der Tabelle gelöscht. Das Schlüsselwort `RESTRICT` bewirkt, dass nur Attribute gelöscht werden können, die nicht mit anderen Tabellen verbunden sind (Fremdschüssel). Alternativ kann das Schlüsselwort `CASCADE` verwendet werden. Dabei wird nicht nur die gewünschte Spalte, sondern auch die verbundene Spalte in der anderen Tabelle gelöscht.

1.2.3. Tabellen löschen

Mit DROP TABLE kann eine bestehende Tabelle gelöscht werden. Dabei werden die Daten und die Struktur der Tabelle gelöscht.
Der Befehl hat folgende Syntax:

```
DROP TABLE <Tabellenname>;
```

> **Achtung:** Mit `DROP TABLE` wird die Tabellenstruktur samt den Daten gelöscht. Dies kann in den meisten Fällen nicht mehr rückgängig gemacht werden.

1.3. Datenbankanfragen

In dieser Unit wird besprochen, wie man mittels SQL Anfragen an die Datenbank stellt, um auf gegebene Fragestellungen die relevanten Daten aus der Datenbank herausziehen zu können.

1.3.1. SELECT-FROM-WHERE Struktur von SQL Anfragen

Mit Hilfe der SELECT-FROM-WHERE Struktur können Anfragen an eine Dabenbank gestellt werden. Anfragen bestehen aus einer Auswahl der gewünschten Spalten (SELECT) und einer Liste mit einer oder mehreren Relationen (FROM). In dieser einfachen Form werden immer alle Datensätze zurückgeben. Zusätzlich kann ein Suchkriterium übergeben werden (WHERE). Damit können gezielt Datensätze selektiert werden. Eine SQL Anfrage gibt als Resultat wiederum eine Relationen zurück.
Die Standardform einer Datenbankanfrage mittels SQL ist wie folgt aufgebaut:

```
SELECT <Attributliste>
FROM <Relationenliste>
WHERE <Bedingungen>;
```

Wobei:

- <Attributliste> besteht aus den Namen der Attribute, deren Werte man durch die Anfrage erhalten möchte.
- <Relationenliste> ist die Aufführung der Namen der Relationen, die für die Anfrage gebraucht werden.
- <Bedingungen> die jene Tupel identifizieren, die durch die Anfrage zurückgegeben werden sollen.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Damit haben wir die Grundstruktur von SQL als Anfragesprache bereits gezeigt. Es gibt nun eine Reihe von möglichen Erweiterungen des obigen Grundschemas, die die Flexibilität von Anfragen massiv erhöhen. Wir unterscheiden folgende Fälle:

- mehrere Bedingungen (konjunktiv oder disjunktiv verknüpft)
- komplexere Bedingungen (Unteranfragen, Bereichsanfragen, Verbundoperatoren)
- spezielle Selektionen (Verknüpfung von Attributen)
- nicht-relationale Konstrukte (Sortieren, Gruppierungen, Aggregatfunktionen)
- Mengenoperationen (Vereinigung, Durchschnitt, Differenz)

1.3.2. Verkettung von Bedingungen

Bedingungen im WHERE Teil einer Anfrage können beliebig durch die Anwendung der logischen Operatoren AND und OR miteinander verkettet werden. Einzelne Bedingungen können mit dem NOT-Operator negiert werden. Der Datensatz wird selektiert, wenn die gesamte Bedingung TRUE ergibt.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Weiter Beispiele für verkettete Bedingungen:

- `PLZ = 8000 AND NOT Nachname = 'Schmidt'`
- `PLZ = 8000 OR PLZ = 8006`
- `(Nachname = 'Müller' OR Nachname = 'Meier') AND Ortsname = 'Zürich'`

In SQL Anfragen (z.B. in Bedingungen) muss Text in Hochkommas geschrieben werden. Numerische Werte werden ohne Hochkommas geschrieben. Dies ist in den Beispielen ersichtlich.

Statt einer langen Verkettung mit AND gibt es auch die Möglichkeit mehrere Attribute gleichzeitig miteinander zu vergleichen:

```
Vorname = 'Ursula' AND Nachname = 'Müller'
```

kann auch folgendermassen geschrieben werden:

```
(Vorname, Nachname) = ('Ursula', 'Müller')
```

1.3.3. Vergleichs- und Mengenoperationen

SQL unterstützt eine Reihe von Vergleichs- und Mengenoperatoren. Diese können dazu verwendet werden, Attribute mit Konstanten oder mit anderen Attributen zu vergleichen. Die Attribute können dabei aus derselben oder verschiedenen Relationen Stammen.
Folgende Vergleichsoperatoren für nummerische Attribute werden unterstützt:

- `=` gleich
- `>` grösser
- `<` kleiner
- `>=` grösser gleich
- `<=` kleiner gleich
- `<>` ungleich
- `BETWEEN` zwischen

Bedingungen haben grundsätzlich die Syntax `<Attribut> <Operator> <Wert>`. Der Operator `BETWEEN` hat eine etwas andere Syntax (siehe dazu das Beispiel).
Weitere Vergleichsoperatoren werden in den nächsten Abschnitten besprochen.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Im SELECT-Teil kann anstelle einer Attributliste auch ein * geschrieben werden. Der Effekt besteht darin, dass alle Attribute der im FROM Teil der Anfrage angegebenen Relationen ausgegeben werden.

1.3.4. Zeichenkettenvergleiche und arithmetische Operatoren

Vergleichsoperator LIKE
Durch den Vergleichsoperator LIKE kann eine Zeichenkette mit einem Ausschnitt aus einer Zeichenkette verglichen werden. Der Ausschnitt wird mit zwei reservierten Zeichen dargestellt:

- „%“ ersetzt eine unbestimmte Anzahl von Zeichen (auch kein Zeichen)
- „_“ ersetzt ein Zeichen

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

`NOT LIKE` untersucht ob das gegebene Ausschnitt nicht in der Zeichenkette vorhanden ist.

Arithmetische Operatoren
Die arithmetischen Standard-Operatoren für Addition (+), Subtraktion (-), Multiplikation (*) und Division (/) können für alle numerischen Werte oder Attribute mit numerischen Domänen eingesetzt werden. Damit können Attribute in einer Anfrage miteinander verrechnet werden.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Das obige Beispiel liefert eine Ergebnisrelation mit drei Attributen, deren drittes mit dem Namen „Wucher“ versehen wurde (da „Preis / Grösse“ kein sonderlich geeignete Bezeichnung ist) und den Preis je Grösseneinheit angibt. Als Wucher wird ein Verhältnis grösser als 15 angenommen.
Arithmetische Operatoren können im SELECT Teil einer Anfrage oder in Bedingungen verwendet werden.

Im SELECT oder im FROM Teil einer Anfrage kann mit dem Befehl `AS` Attributen oder Relationen ein anderer Namen gegeben werden: `<Attribut/Relation> AS <neuer Name>` (Das Schlüsselwort `AS` kann auch weggelassen werden). Dies kann dazu verwendet werden, um wie im Beispiel einem berechneten Wert einen sinnvollen Namen zu geben oder um SQL Anfragen lesbarer zu machen.

1.3.5. Geschachtelte Anfragen

Bedingungen haben prinzipiell die Struktur <Attribut> <Operator> <Wert> (z.B. Name=„John“) wobei die Werte wiederum Attribute, Konstanten oder aber das Resultat von Unteranfragen sein können, d. h. man muss die Werte von Bedingungen nicht unbedingt fix definieren, sondern kann sie mittels einer verschachtelten Anfrage generieren. Anfragen können so beliebig tief verschachtelt werden.
Es gibt drei Typen von Unteranfragen, welche sich durch ihr Resultat unterscheiden:

- Unteranfragen die einen Wert zurückgeben (eine Spalte und eine Zeile)
- Unteranfragen die eine Zeile zurückgeben
- Unteranfragen die meherere Zeilen zurückgeben

Wird nur ein Wert oder eine Zeile zurückgegeben, können die normalen Vergleichsoperatoren verwendet werden.
Werden mehrere Zeilen zurückgeben, kommen spezielle Operatoren zum Einsatz:

- `IN` sucht ob der Wert in der Unteranfrage vorkommt
- `<Vergleichsoperator> ALL` die Bedingung muss für alle Zeilen in der Unterabfrage TRUE ergeben
- `<Vergleichsoperator> ANY (SOME)` die Bedingung muss für mindestens eine Zeile in der Unterabfrage TRUE ergeben

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Unteranfragen können auch im FROM Teil einer Anfrage vorkommen. So kann eine Relation speziell für eine Anfrage zusammengestellt werden. Der Unteranfrage muss mit AS ein Namen zugewiesen werden: `(<Unterabfrage>) AS <Name>`

1.3.6. Verbund

Oft kommt es vor, dass in einer Anfrage Daten aus mehreren Relationen benötigt werden. Hierfür müssen die Relationen miteinander verknüpft werden. Dazu werden die identischen Attribute (Fremdschlüssel) in den beiden Relationen miteinander verbunden.
Es gibt zwei Möglichkeiten Relationen in Anfragen miteinander zu verküpfen:

im WHERE Teil
Bei dieser Möglichkeit werden die identischen Attribute mit dem Vergleichsoperator verknüpft und als "Bedingung" (keine eingentliche Bedingung) im WHERE Teil der Anfrage eingefügt.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Grundsätzlich kann auch z.B. ein `<` für den Verbund verwendet werden. Dies ist geschieht jedoch meist in Kombination mit einer Verbundsbedingung mit `=`, weil ein solcher Verbund alleine keinen Sinn macht.

im FROM Teil
Eine andere Möglichkeit besteht darin, den Verbund im FROM Teil einer Anfrage vorzunehmen. Dies ist sinnvoller, da so die eigentlichen Suchkriterien (Bedingungen im WHERE-Teil) von den Tabellenverknüpfungen getrennt sind.
Dazu stehen folgende Befehle zur Verfügung:

- `<Relation> JOIN <Relation> USING (<Attribut>)`
  die Relationen werden über ein in beiden Tabellen gleich benanntes Attribut verknüpft
- `<Relation> NATURAL JOIN <Relation>`
  verknüpft automatisch alle Attribute die in beiden Relationen gleich benannt sind.
- `<Relation> JOIN <Relation> ON <Attribut> <Vergleichsoperator> <Attribut>`
  bei dieser Variante kann angegeben werden, über welche Attribute beider Relationen verknüpft werden soll und welcher Operator verwendet werden soll (es können auch mehrere Verknüpfungen angegeben werden)

Das obige Beispiel würde mit diesen Befehlen folgendermassen aussehen:

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde JOIN Abonnement USING (KundNr);
```

oder

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde JOIN Abonnement ON Kunde.KundNr = Abonnement.KundNr;
```

oder

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde NATURAL JOIN Abonnement;
```

Die oben beschriebenen Befehle geben nur Datensätze aus, die in beiden Relationenen vorkommen. Sollen **alle** Datensätze der einen Relation mit den zugehörigen Datensätzen der zweiten Relation ausgegeben werden, kommen folgende Befehle zum Einsatz:

- `<Tabelle> RIGHT OUTER JOIN <Tabelle> USING (<Attribut>)`
  alle Datensätze der rechten Relation und die zugehörigen Datensätze der linken Relation
- `<Tabelle> LEFT OUTER JOIN <Tabelle> USING (<Attribut>)`
  alle Datensätze der linken Relation und die zugehörigen Datensätze der rechten Relation

Falls bei `RIGHT OUTER JOIN` in der linken Relation kein Datensatz verknüpft werden kann, wird für die Attribute dieser Relation NULL ausgegeben. Dies gilt ungekehrt auch für `LEFT OUTER JOIN`.

1.3.7. Nicht relationale Konstrukte

„ORDER BY“-Klausel

SQL unterstützt auch Konstrukte, die mit der ursprünglichen relationalen Algebra wenig bis gar nichts zu tun haben. So besitzt eine Menge per Definition keine Ordnung. Mittels des Konstrukts „ORDER BY“ kann man jedoch eine Ordnung auf der Ergebnisrelation definieren. Die Schlüsselwörter „ASC“ und „DESC“ bezeichnen aufsteigende- resp. absteigende Anordung der Werte im Resultat. Wird keines dieser Schlüsselwörter angegeben, wird standardmässig aufsteigend sortiert.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Alle Kunden werden sortiert nach ihrem Namen (aufsteigend, wobei ASC als Standardeinstellung auch entfallen könnte) und ihrem Vornamen (absteigend, DESC) ausgegeben.

„GROUP BY“-Klausel

Gruppierungen („GROUP BY“-Klausel) dienen dazu, die Menge aller Tupel einer Relation nach bestimmten Kriterien in Teilmengen zu unterteilen, um für diese Teilmengen bestimmte Statistiken zu berechnen. Als Gruppierungskriterium dienen die Werte eines bestimmten Attributs. Alle Tupel, die für dieses Attribut den gleichen Wert besitzen, werden zu einer Gruppe zusammengefasst. Diese Gruppen können dann weiter bearbeitet werden (im Spezialfall auch durch eine weitere Gruppierung, um Gruppen von Gruppen zu bearbeiten etc.). Dazu dienen sogenannte Aggregatfunktionen, die nur auf numerische Attribute angewendet werden können. Folgende Aggregatfunktionen stehen üblicherweise zur Verfügung:

- `min` zum Bestimmen des minimalen Werts eines Attributs
- `max` zum Bestimmen des maximalen Werts eines Attributs
- `sum` zum Berechnen der Summe der selektierten Werte eines Attributs
- `count` zeigt die Anzahl der selektierten Werte eines Attributs (nicht der verschiedenen Werte!)
- `avg` zum Berechnen des Mittelwertes aller selektierten Werte eines Attributs, wobei NULL-Werte nicht in die Berechnung einfliessen.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Mittels dieser Anfrage werden die Nummern aller Kunden bestimmt, die mit Kleinanzeigen (Preis des einzelnen Inserats weniger als 300 Franken) schon Inserate für mehr als 200 Franken in Auftrag gegeben haben. Bei der Bearbeitung dieser Anfrage wird erst die Restriktion berücksichtigt (A004 wird aussortiert), dann werden die verbliebenen Tupel gruppiert, und zum Schluss wird geprüft, welche Tupel, die Gruppen repräsentieren, tatsächlich ausgegeben werden sollen (die Gruppen der Kunden 002 und 005 werden aussortiert, da sie noch nicht das Limit erreicht haben).

Anfragen, die eine GROUP BY-Klausel enthalten, werden folgendermassen abgearbeitet. Zuerst wird die Bedingung im WHERE Teil ausgeführt, falls eine solche vorhanden ist. Danach wird nach den angegeben Spalten gruppiert. Mit der Bedingung im HAVING Teil der Anfrage kann anschliessend, falls nötig, noch eine Bedingung für die gruppierten Attributwerte angegeben werden. Auf Grund dieser Reihenfolge ist ersichtlich, dass die Bedingung im WHERE Teil der Anfrage keine Aggregatfunktionen enthalten kann, weil zu diesem Zeitpunkt noch gar nicht gruppiert wurde. Alle Attribute, die in der Anfrage vorkommen müssen aber trotzdem entweder in der GROUP BY-Klausel vorkommen oder in einer Aggregatfunktion stehen.

1.3.8. Mengenoperationen

Mengenoperationen werden verwendet, um Tupel in einer Ergebnisrelation zusammenzufassen, die aus im Grunde verschiedenen Anfragen stammen. Voraussetzung dafür ist natürlich, dass die Anzahl der selektierten Attribute jeder beteiligten Anfrage identisch ist. Ausserdem müssen die jeweiligen Domänen kompatibel sein (und zwar jeweils in der gleichen Reihenfolge). Für die Verknüpfung von Tupelmengen stehen die üblichen Operatoren zur Verfügung, wie man sie von der Mengenlehre her kennt:

- `UNION` für die Vereinigung von Tupelmengen
- `INTERSECT` für den Durchschnitt von Tupelmengen
- `EXCEPT` für die Differenz von Tupelmengen

Standardmässig werden bei einer Anfrage mit diesen Mengenoperationen Duplikate nicht ausgegeben. Wenn die Duplikate ausgegeben werden sollen, muss hinter `UNION`, `INTERSECT` oder `EXCEPT` das Schlüsselwort `ALL` angegeben werden.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Durch diese Anfrage werden die Vor- und Nachnamen aller Kunden bestimmt, die sowohl ein Abonnement besitzen als auch schon einmal ein Inserat aufgegeben haben.

1.3.9. Zusammenfassung

In dieser Unit wurde in einzelen Schritten gezeigt wie SQL-Anfragen formuliert werden. Um all diese Teile im Gesamtkontext aufzuzeigen ist nachstehend die Syntax für eine gesamte SQL-Anfrage aufgeführt.
Komplette Syntax einer SQL Anfrage:

```
SELECT [DISTINCT | ALL]
<Attribut> [AS <Name>] [, ...] | *
FROM <Relation> [, <Relation>]
[WHERE <Bedingungen>]
[GROUP BY <Spalten> [HAVING <Bedingung>]
[ORDER BY <Spalte> [ASC | DESC], [, ...]];
```

Ein grosser Teil dieser Syntax ist optional und muss deshalb nur in gewissen Fällen angegeben werden. Die einfachste syntaktisch korrekte Anfrage, besitzt nur einen SELECT- und einen FROM-Teil. Nur bei komplexeren Fragestellungen werden alle Teile in der gleichen Anfrage verwendet.

> Es kann vorkommen, dass im Resultat einer Anfrage gewisse Datensätze mehrmals vorkommen. Das Schlüsselwörter `DISTINCT` bewirkt, dass solche Duplikate aus dem Resultat gelöscht werden. Das Schlüsselwort `ALL` bewirkt, dass die Duplikate nicht gelöscht werden. Da dies standardmässig der Fall ist, muss es nicht unbedingt angegeben werden.

1.3.10. Übung „Datenbankanfragen“

Diese Übung soll Ihnen die Möglichkeit geben, die verschiedenen SQL Anfragen noch besser kennen zu lernen und zu vertiefen. Dazu wird folgende Beispieldatenbank [http://www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html] verwendet. Behalten Sie diese Übersicht während der Bearbeitung der Übung offen.
Dokumentieren Sie die Lösungen für sich in einem Word- oder PDF-Dokument. Wenn Sie möchten können Sie das Dokument am Schluss auf dem Diskussionsforum unter dem Thema „Datenbankanfragen“ veröffentlichen. Wenn Sie Fragen oder Probleme haben mit dieser Übung können Sie auf dem Diskussionsforum unter dem Thema „SQL Probleme“ eine Nachricht hinterlegen. Gerne dürfen Sie auch eventuelle Fragen Ihrer Mitstudierenden beantworten.

Aufgaben
Finden Sie die korrekten Lösungstabellen für die nachstehenden SQL Anfragen.

```
SELECT vname, nname
FROM angestellter
WHERE salaer = 25000
AND ahvnr > 5000
```

```
SELECT pname
FROM projekt
WHERE abt = 5
OR pnummer = 30
```

```
SELECT nname
FROM angestellter
WHERE salaer BETWEEN 28000 AND 41000
```

```
SELECT ahvnr
FROM angestellter
WHERE adresse LIKE 'Zu%'
```

```
SELECT Angestellter.nname
FROM Angestellter, Arbeitet_an
WHERE Arbeitet_an.projekt = '20'
AND Angestellter.ahvnr = Arbeitet_an.ang
```

```
SELECT name
FROM angehoeriger
WHERE ang IN
(SELECT ahvnr
FROM angestellter
WHERE adresse = 'Kuesnacht')
```

Lösungen zu den Aufgaben

Lösungsdownload [http://www.gitta.info/RelQueryLang/de/download/Loesung_SQLAnfrage.pdf Grösse: 183KB. Typ: pdf.]

eLSQL Übung 'Datenbankanfragen'

Diese Übung befasst sich mit allen bis jetzt kennengelernten SQL Anfragen. Dazu wird die folgende Beispieldatenbank [http://www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html] verwendet. Es ist empfohlen diese Übersicht während der Bearbeitung der Übung offen zu behalten. Die Übersicht finden Sie auch im eLSQL-Tool unter dem Tab 'Datensatz'. Die Übung wird mittels eines Web-Interfaces auf einer MySQL-Datenbank durchgeführt. Die nötigen Informationen dazu (Webadresse und Registrierung) erhalten Sie von ihrem Tutor.

Hinweise zum eLSQL-Interface

- Der Strichpunkt am Ende eines SQL Befehls oder einer SQL Anfrage ist optional.
- Ihre Änderungen an der Datenbank werden gespeichert und sind beim nächsten Login wieder verfügbar.
- Unter dem Tab 'Datensatz' finden Sie Informationen zum Inhalt der Originaldatenbank.
- Unter dem Tab 'Einstellungen' können Sie Ihre Benutzerinformation ändern, sowie die Datenbank in den Originalzustand zurücksetzen. Alle Ihre Änderungen an der Datenbank werden gelöscht. Achtung: Dieser Befehl kann nicht rückgängig gemacht werden.
- Unter dem Tab 'Protokoll' finden Sie eine Auflistung aller von Ihnen gemachten SQL-Anfragen.

Ziel dieser Übung ist, dass Sie selbständig einfache und schwierigere SQL Anfragen formulieren können, um Antworten auf Problemstellung zu bekommen. Dazu sollen Sie die untenstehenden Aufgaben lösen. Die Vorgehensweise, Überlegungen, SQL Anfragen, Lösungen etc. sollen in einem Word oder PDF Dokument zusammengefasst werden (editiertes und kommentiertes Protokoll). Veröffentlichen Sie Ihre Lösung auf dem Diskussionsforum unter dem Thema 'eLSQL' und schauen Sie sich auch Lösungen Ihrer Mitstudierenden an. Wenn Sie Fragen oder Probleme haben mit dieser Übung können Sie auf dem Diskussionsforum unter dem Thema 'SQL Probleme' eine Nachricht hinterlegen. Gerne dürfen Sie auch eventuelle Fragen Ihrer Mitstudierenden beantworten.

- Selektieren Sie alle Nachnamen der Angestellten und sortieren Sie diese absteigend (Z–A).
- Finden Sie die Vornamen aller Angestellten, die in Dübendorf wohnen und mehr als 30'000 Franken verdienen.
- Selektieren Sie alle Kinder der in Zürich wohnhaften Angestellten und sortieren Sie diese aufsteigend nach dem Geburtsdatum.
- Finden Sie die gesamte Arbeitszeit an allen Projekten der Forschungsabteilung heraus. (in einer Abfrage!)
- Finden Sie heraus wie viele Kinder der Leiter der Abteilung Verwaltung hat.
- Welche Projekte sind in Zürich beheimatet an denen schon mindestens 50 Stunden gearbeitet wurde?
- Finden Sie selber eine einfache oder schwierigere SQL-Abfrage (in Textform als Beschreibung) und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema „eLSQL“. Allfällige Lösungen zu solchen Aufgaben können im selben Diskussionsthema veröffentlicht werden. Verwenden Sie für die beiden Nachrichten den gleichen Titel jeweils mit der Endung „- Abfrage“ oder „- Lösung“.

- Lösen Sie einige Aufgaben Ihrer Mitstudierenden.

Mögliche Lösungen

Lösungsdownload [http://www.gitta.info/RelQueryLang/de/download/Loesung_eLSQL1.pdf Grösse: 179KB. Typ: pdf.]

1.4. Einfügen, Löschen und Ändern

Um eine Datenbank immer auf dem aktuellen Stand zu halten, müssen von Zeit zu Zeit die Daten nachgeführt werden (alte, nicht mehr gebrauchte Einträge müssen gelöscht, neue Einträge erfasst oder Einträge müssen geändert werden). In SQL gibt es drei Kommandos, um die Datenbasis zu manipulieren. Diese Kommandos sind Einfügen (INSERT), Löschen (DELETE) und Ändern (UPDATE).

1.4.1. Einfügen von Tupeln

In seiner einfachsten Form wird durch das Kommando „Einfügen“ (INSERT INTO) ein Tupel in eine Relation hinzugefügt.
Form des Statements:

```
INSERT INTO <Relationenname> (<Liste von Attributen>)
VALUES (<Liste von Werten>);
```

Wobei die Liste der Attribute nur dann angegeben werden muss, wenn keine vollständigen Tupel eingegeben werden oder wenn die Werte in einer anderen Reihenfolge eingegeben werden als die zugehörigen Attribute bei der Definition der Relation.

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Mit der oben gezeigten Form des INSERT Kommandos werden neue Tupel in die Datenbank eingefügt. Es kommt auch häufig vor, dass die Werte die eingefügt werden müssen, als Resultatrelation einer Anfrage vorhanden sind. Dafür kann das Kommando folgendermassen geänderte werden.

```
INSERT INTO <Relationenname> (<Liste von Attributen>)
SELECT ...
```

(normale Anfrage)
VALUES wird also durch eine SQL-Anfrage ersetzt. Das Resultat dieser Anfrage wird in die angegebene Tabelle eingefügt. Falls die Anfrage eine komplettes Tupel mit der richtigen Reihenfolge liefert, muss auch hier die Liste der Attribute nicht angegeben werden.

1.4.2. Löschen von Tupeln

Das Kommando „Löschen“ (DELETE FROM) entfernt ein oder mehrere Tupel aus einer Relation. Die WHERE-Klausel spezifiziert, welche Tupel betroffen sind. Tupel werden explizit immer nur aus einer Relation gleichzeitig gelöscht. Bei einer fehlenden WHERE-Klausel werden alle Tupel aus der angegebenen Relation gelöscht.
Form des Statements:

```
DELETE FROM <Relationenname>
WHERE <Bedingung>;
```

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Achtung: Wird `DELETE FROM` ohne `WHERE` verwendet, werden alle Datensätze der Relation gelöscht. Dies kann in den meisten Systemen nicht rückgängig gemacht werden.

1.4.3. Ändern von Tupeln

Das Kommando „Ändern“ (UPDATE) wird gebraucht, um Werte von Attributen von einem oder mehreren Tupeln zu ändern. Die WHERE-Klausel spezifiziert, welche Tupel von der Änderung betroffen sind. Die SET-Klausel spezifiziert die Attribute, die geändert werden sollen und bestimmt deren neue Werte.
Form des Statements:

```
UPDATE <Relationenname>
SET  einem oder mehreren Attributen einen Wert zuweisen
WHERE <Bedingung>;
```

Dieses Element (Animation, Video etc.) kann in der PDF version nicht dargestellt werden und ist nur in der online Version sichtbar. [link]

Die Zuweisung des neuen Wertes erfolgt wie im Beispiel ersichtlich folgendermassen:

```
<Attributname> = <Wert>
```

Mehrer Zuweisungen werden durch Komma getrennt. Der zugewiesen Wert kann entweder ein Konstante sein oder das Resultat einer Anfrage, d.h. es kann auch eine Anfrage rechts vom Gleichheitszeichen stehen.
Wird eine ganze Zeile geändert sieht die Zuweisung wie folgt aus:

```
ROW = ROW(<Liste von Werten>)
```

1.4.4. eLSQL Übung 'Insert, Delete und Update'

Diese Übung befasst sich vorwiegend mit den SQL Befehlen INSERT , DELETE und UPDATE . Um die Übung lösen zu können, müssen aber auch die SQL Anfragen aus der Unit Datenbankanfragen bekannt sein. Während der folgenden Übung arbeiten Sie mit dieser Beispieldatenbank [http://www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html] Es ist empfohlen diese Übersicht während der Bearbeitung der Übung offen zu behalten. Die Übersicht finden Sie auch im eLSQL-Tool unter dem Tab 'Datensatz'. Die Übung wird mittels eines Web-Interfaces auf einer MySQL-Datenbank durchgeführt. Die nötigen Informationen dazu (Webadresse und Registrierung) erhalten Sie von ihrem Tutor.

Hinweise zum eLSQL-Interface

- Der Strichpunkt am Ende eines SQL Befehls oder einer SQL Anfrage ist optional.
- Ihre Änderungen an der Datenbank werden gespeichert und sind beim nächsten Login wieder verfügbar.
- Unter dem Tab 'Datensatz' finden Sie Informationen zum Inhalt der Originaldatenbank.
- Unter dem Tab 'Einstellungen' können Sie Ihre Benutzerinformation ändern, sowie die Datenbank in den Originalzustand zurücksetzen. Alle Ihre Änderungen an der Datenbank werden gelöscht. Achtung: Dieser Befehl kann nicht rückgängig gemacht werden.
- Unter dem Tab 'Protokoll' finden Sie eine Auflistung aller von Ihnen gemachten SQL-Anfragen.

Aufgaben

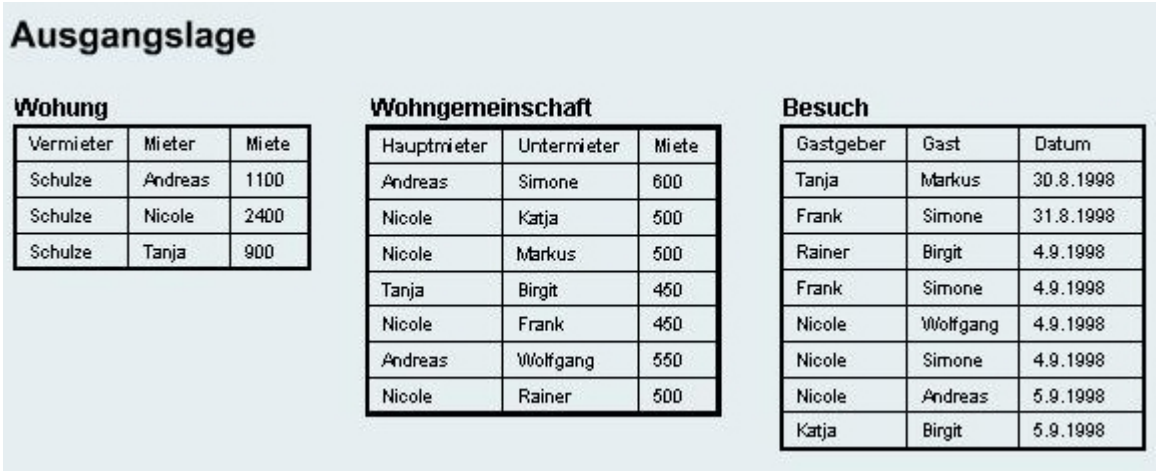
- Stellen Sie sich vor, Sie seien neu angestellt bei der Firma welche durch die Beispieldatenbank repräsentiert wird. Fügen Sie Ihren Eintrag in die Tabelle 'Angestellter' ein und fügen Sie mindestens einen Ihrer Angehöriger in die Tabelle 'Angehoeriger' ein. Danach schreiben Sie in Ihrem Namen eine Anzahl von Projektstunden auf eines der Projekte auf.
  Hinweis: Lassen Sie die Geburtsdaten weg, da diese das Format Datum verwenden und kompliziert zu definieren sind.
- Selektieren Sie die von Ihnen gemachten Einträge in den entsprechenden Tabellen.
- Die Arbeit an Projekt 20 wurde eingestellt. Löschen Sie alle Hinweise auf dieses Projekt aus der Datenbank.
  Wie sind Sie vorgegangen und weshalb?
- Beate Tell verlässt die Firma. Ihre Leitungsposition in der Abteilung Forschung wird nun von Sonja Maradona übernommen. Ändern Sie die Datenbank und löschen Sie nicht mehr benötigte Einträge, so dass alle Tabellen auf dem aktuellsten Stand sind.
  Wie sind Sie vorgegangen und weshalb?
- Erhöhen Sie das Salär von allen Angestellten, die Boris Frisch zum Vorgesetzten haben, um 10'000 Fr.
- Stellen Sie sich eine weitere mögliche Aufgabe und führen Sie die entsprechende Änderungen in der Datenbank durch.

Mögliche Lösungen

Lösungsdownload [http://www.gitta.info/RelQueryLang/de/download/Loesung_eLSQL2.pdf Grösse: 265KB. Typ: pdf.]

1.5. Lernkontrolle

Die nachfolgende Interaktion kann das bisher gelernte an einem konkreten Beispiel angewendet werden. Um die Flash-Animation zu starten, klicken Sie bitte auf die Grafik.



Anwendung Datenbankanfragen

Die Daten der Übung können als Access-Datenbank [http://www.gitta.info/RelQueryLang/de/multimedia/SQL-SelfAss2.mdb Grösse: 180 KB. Typ: mdb.] heruntergeladen werden. Damit können alle in der Übung vorhanden Anfragen ausprobiert werden, um deren Unterschiede festzustellen.

1.6. Zusammenfassung

SQL ist eine standardisierte Sprache zur Kommunikation mit relationalen und objektrelationalen Datenbanken. Mittels SQL können Datenbankschemata erzeugt und abgeändert werden. Tabellen und Abhängigkeiten können beispielsweise mit dem `CREATE TABLE`-Befehl erzeugt und mit dem `DROP TABLE`-Befehl gelöscht werden. Dieser Teil der SQL-Sprache wird auch als Datendefinitionssprache (DDL) bezeichnet. Im täglichen Umgang mit Datenbanken viel wichtiger sind die SQL-Befehle zur Datenabfrage und -manipulation (DML). Damit können Anfragen an die Datenbank gestellt und Tupel in Tabellen mutiert werden (Befehle `INSERT`, `DELETE` und `UPDATE`). Ausserdem können mittels SQL Zugriffsrechte auf die Datenbank vergeben werden.

1.7. Literaturempfehlungen

- **ELMASRI, R.; NAVATHE, S.B.**, 1994. *Fundamentals of Database Systems*. 2nd. Redwood City, California: Addison-Wesley.
  Einführung ins Thema Datenbanken und SQL, auf Englisch
- **GULUTZAN, P.; PELZER, T.**, 1999. *SQL-99 Complete, Really*. 1nd. Lawrence, USA: R&D Books.
  An Example-Based Refernce Manual of the New Standard

1.8. Bibliographie

- **ELMASRI, R.; NAVATHE, S.B.**, 1994. *Fundamentals of Database Systems*. 2nd. Redwood City, California: Addison-Wesley.
- **GULUTZAN, P.; PELZER, T.**, 1999. *SQL-99 Complete, Really*. 1nd. Lawrence, USA: R&D Books.