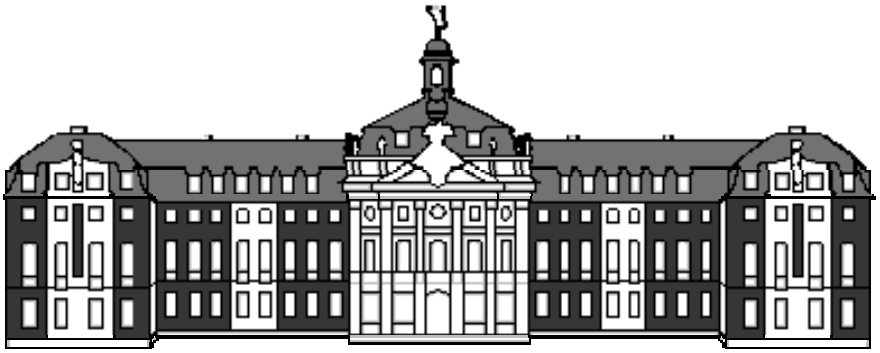

WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

Examensarbeit zum Thema:

Entwicklung eines visuellen Lernprogramms zu SQL-Anfragen an ein Datenbanksystem

Fachbereich 10
Institut für Mathematik und Informatik

Themensteller: **Prof. Dr. Marco Thomas**

Von: **Michael Saul – Matr. Nr.: 315980 – MichaelSaul [at] gmx [dot] de**
Lehramt Mathematik und Informatik Gy/Ge

Münster, den 20.07.2009

Inhaltsverzeichnis Seite

I Einleitung

I.1 Um was geht es?

Das Projekt „SQL-Learner“ entstand im Rahmen meiner Examensarbeit. Einer der Themenvorschläge von Prof. Dr. Thomas war es, ein Exponat für das virtuelle Museum „LIMO - Lernumgebung für informatisches Modellieren“ zu erstellen. Das Ziel des LIMO Projektes ist eine Sammlung von interaktiven Exponaten zu Lernzwecken in einer virtuellen Umgebung zu vereinen. Dabei können verschiedene Exponate thematisch passend zu einer Ausstellung zusammenfasst werden. Der erzielte Lernerfolg wird von jedem Exponat gemessen und an das Museum weitergeleitet, damit für jeden Besucher des virtuellen Museums ein persönliches Leistungsprofil erstellt werden kann.
Der SQL-Learner ist ein in Adobe Flash CS3 programmiertes Lernprogramm, das schwerpunktmäßig in der gymnasialen Oberstufe eingesetzt werden soll, um die Verwendung von SQL-Anfragen zu erlernen und das Ergebnis solcher Anfragen zu visualisieren. In Kapitel II wird auf die Datenbanksprache SQL näher eingegangen. Kapitel III ist als Handbuch gedacht und richtet sich in erster Linie an Personen, die den SQL-Learner zu Lehr- und Lernzwecken einsetzen möchten. Wer hingegen Interesse an den Hintergründen, Designentscheidungen und dem Entwicklungsprozess des SQL-Learners hat, findet dazu in Kapitel IV viele Informationen. In Abschnitt IV.1 bis IV.3 werden zunächst die Entscheidungen begründet, die zum Design der Oberfläche und der Funktionsweise des SQL-Learners geführt haben. Anschließend wird auf technische Entscheidungen, z.B. warum die Wahl der Datenbankanfragesprache auf SQL fiel, eingegangen. Während sich also der erste Teil des Kapitels IV eher mit der Frage nach dem „Warum“ beschäftigt, geht der zweite Teil ab Abschnitt IV.4 auf das „Wie“ ein. Hier werden der genaue Ablauf einiger Funktionen des SQL-Learners, sowie die Funktionsweise der externen Komponenten beschrieben.

I.2 Einbettung in den Unterricht

I.2.1 Lehrplan Informatik in Nordrhein-Westfalen

Der Lehrplan zum Fach Informatik des Bundeslandes Nordrhein-Westfalen (siehe [MS99]) legt vor allem die Grundprinzipien des Informatikunterrichtes fest. Die Pflichtinhalte sind in drei Bereiche untergliedert. Der erste Bereich stellt die fachlichen Inhalte dar. Das Gewinnen eines Informatikmodells, das Abstrahieren von Daten und Algorithmen, Lösungen nach einem Programmierkonzept realisieren, das Einschätzen von Informationssystemen und das Erkennen von Prinzipien von Hard- und Software.
Der zweite Bereich befasst sich mit dem Lernen im Kontext einer Anwendung. Dabei ist zu beachten, dass es nicht darum geht, den Umgang mit einer bestimmten Anwendung zu perfektionieren, sondern die Fachinhalte daraus zu abstrahieren, zu verallgemeinern und die Theorie herauszuarbeiten.
Im dritten Bereich geht es um Methoden und Formen selbstständigen Arbeitens. Es könnte z.B. eine Software erstellt werden, wobei Gruppenweise so gearbeitet wird, dass jede Gruppe ein Modul der Software erstellt. Ebenso können die Schülerinnen und Schüler in Einzel- oder Partnerarbeit Erfahrungen mit einer Programmierumgebung machen.
Der Unterricht in der gymnasialen Oberstufe ist sequenziell aufgebaut. Eine konkrete Festlegung der zu bearbeitenden Themen gibt es nicht. Als Kriterien für die Auswahl von Unterrichtsinhalten stellt der Lehrplan folgendes heraus:

- Das Prinzip des Exemplarischen soll vermittelt werden, das sich auf wesentliche, repräsentative und bedeutsame Fachinhalte beschränkt, die geeignet sind, übertragbare Kenntnisse und Fertigkeiten zu vermitteln.
- Vorwissen der Schülerinnen und Schüler soll berücksichtigt und einbezogen werden.
- Die Inhalte sollen methodisch selbstständiges Arbeiten ermöglichen und entsprechend Kompetenzen aufbauen und sichern.

Das Thema „Datenbanken“ wird im Lehrplan als mögliche Sequenz im anwendungsorientierten Bereich vorgeschlagen. Es werden mögliche Lernsequenzen beschrieben, die als Vorschlag gedacht sind. Somit könnte der SQL-Learner entsprechend dem Lehrplan auf jeden Fall im Unterricht eingesetzt werden.

I.2.2 Erwartetes Vorwissen

Für die Bedienung des Programms muss der Nutzer als Grundvoraussetzung mit einer Computermaus umgehen können und der deutschen Sprache mächtig sein.
Für den Einführungsmodus wird keinerlei Fachwissen zum Thema „Datenbanken“ benötigt. In diesem Fall müssen die Texte und Hilfestellungen im SQL-Learner sehr genau gelesen und nachvollzogen werden. Der weiterführende Modus (Levelmodus) setzt die im Einführungsmodus vermittelten Grundkenntnisse und die Fähigkeit den SQL-Learner zu bedienen voraus.

I.2.3 Zielgruppe / Lernziele

Der SQL-Learner richtet sich primär an Schülerinnen und Schüler der gymnasialen Oberstufe. Er eignet sich sowohl für den Einstieg, als auch als Abschluss für eine Unterrichtseinheit zum Thema „Datenbanken“, da es innerhalb des Programms für beide Anwendungszwecke jeweils einen Modus gibt. Wenn der SQL-Learner als Einstieg in das Thema „Datenbanken“ verwendet wird, sollte in jedem Fall nachträglich noch einmal reflektiert werden, was die Schülerinnen und Schüler gelernt haben, um einen gemeinsamen Standpunkt für das weitere Vorgehen in der Unterrichtsreihe festlegen zu können.
Als Abschluss einer Unterrichtseinheit zum Thema „Datenbanken“ sollte der so genannte „Einführungsmodus“ kein Problem mehr darstellen, sodass direkt die Aufgaben aus dem schwereren Modus bearbeitet werden können. Schülerinnen und Schüler, die bisher Schwierigkeiten mit dem Thema hatten, können aber auch mit dem Einführungsmodus anfangen und damit zu den Anderen aufschließen. Um sich mit der Bedienung des SQL-Learners vertraut zu machen, wird empfohlen, zunächst die ersten zwei bis drei Aufgaben des Einführungsmodus zu durchlaufen. Das durcharbeiten des Handbuchs ist dafür nicht notwendig. Die Lehrperson sollte vor dem Einsatz des Programms selbst einmal alle Aufgaben durchlaufen, um ggf. Hilfestellungen geben zu können.
Folgende Lernziele können erreicht werden:

- Den Inhalt einer Datenbank benennen können (Tabelle, Attribute, usw.)

- Erstellen von SQL-Anfragen zur Abfrage von Informationen aus einer Datenbank
- Verstehen, wie Informationen aus verschiedenen Tabellen der Datenbank kombiniert werden können (Join)
- Selbstständiges Erarbeiten eines neuen Sachverhaltes (Einführungsmodus, falls zu Beginn einer Unterrichtsreihe)
- Den Unterschied zwischen Datenbank, Datenbank-Managementsystem und Datenbankanwendung unterscheiden, bzw. diese Begriffe erklären können (über Diskussion mit einer Lehrperson)
- Notwendigkeit des verantwortungsvollen Umgangs mit Informationen einschätzen. Dies führt zu den Themen Datenschutz und Datensicherheit (über Diskussion mit einer Lehrperson)

I.2.4 Schulbücher zum Thema „Datenbanken“

Ein Schulbuch für den Informatikunterricht der gymnasialen Oberstufe, das sich umfangreich mit Datenbanken beschäftigt, war auch nach längerer Recherche nicht zu finden. Der Diesterweg Verlag bietet überhaupt keine Informatik-Schulbücher an. In der Detail-Suche des Schroedel-Verlages wurden ebenfalls keine Schulbücher für Informatik in der gymnasialen Oberstufe in NRW gefunden. Ein solches Buch müsste zunächst die in Abschnitt I.2.1 genannten Anforderungen erfüllen. Weiterhin dürfte es nicht zu speziell sein, da es zu jedem der zu behandelnden Bereiche unzählige Themen gibt, die dazu möglich wären. Außerdem ist keine zu verwendende Programmiersprache in der Oberstufe festgelegt.
Das einzig annährend passende Schulbuch zum Thema „Datenbanken“ bietet der Klett-Verlag an [Kl07]. Allerdings ist dieses Buch auf den Lehrplan in Bayern zugeschnitten und für Schülerinnen und Schüler der Klasse 9 gedacht, die bereits ab der 6. Klasse Informatik belegen. Die Abfrage von Daten aus einer Tabelle einer Datenbank wird auf drei Seiten behandelt. Auf Abfragen aus mehreren Tabellen wird im letzten Kapitel des Buches eingegangen. Natürlich gibt es mehr fachwissenschaftliche Erklärungen, als sie der SQL-Learner geben könnte. Ein Schulbuch kann aber das praktische Arbeiten mit SQL-Befehlen nicht so gut vermitteln. Die Aufgaben im Buch sind zum Teil sehr theoretisch und es ist ungewiss, ob die erarbeitete Lösung richtig ist. Der beste Weg ist vermutlich eine Kombination aus Unterricht,

Buch und passendem Einsatz des SQL-Learners. Dabei könnten Aufgaben aus dem Buch im SQL-Learner umgesetzt werden.

II Fachwissenschaftliche Grundlagen

II.1 SQL als Datenbanksprache

SQL steht für „Structured Query Language. Der Einsatz von SQL ermöglicht es, Daten in relationalen Datenbanken abzufragen, zu manipulieren oder die Struktur der Daten in der Datenbank zu bestimmen.
Ein Datenbanksystem besteht aus der Datenbank, also der Menge der zu verwaltenden Daten selbst, und einem Datenbankmanagementsystem. Anfragen in SQL werden nicht direkt an die Datenbank, sondern an das Datenbankmanagementsystem gestellt. Dieses organisiert intern die Speicherung der Daten und überprüft alle Zugriffe auf die Datenbank selbst. SQL-Anfragen müssen syntaktisch korrekt formuliert werden, damit sie die gewünschte Wirkung erzielen.
Die Sprache SQL teilt sich in drei Kategorien auf.

- DCL (=Data Control Language)
  Der Zweck dieser Befehle ist die Rechteverwaltung und Transaktionskontrolle.
- DDL (=Data Definition Language)
  Die DDL stellt Befehle zum erstellen des Datenbankschemas bereit.
- DML (=Data Manipulation Language)
  Befehle zum Ändern, Einfügen und Löschen von Daten in Tabellen, sowie das Abfragen von Daten sind in der DML enthalten.

Bei allen Aufgaben des SQL-Learners geht es um die Abfrage von Daten.

II.2 SQL Befehle mit Erklärung

Im Folgenden wird exemplarisch auf drei ausgewählte SQL-Befehle eingegangen. Sämtliche im SQL-Learner enthaltenen Erläuterungen zu SQL-Befehlen hier zu platzieren wäre zu umfangreich gewesen. Die Erklärungen wurden dem SQL-Learner direkt entnommen, richten sich an den Nutzer und sind daher in direkter Rede formuliert.

> Die **SELECT** Anweisung steht für 'Wähle aus'. Wenn du bestimmte Daten aus einer Tabelle abfragen möchtest, steht die SELECT-Anweisung üblicherweise am Anfang der SQL-Anfrage.

Nach der Verwendung der SELECT-Anweisung fragt man sich, was eigentlich ausgewählt werden soll. Das kann z.B. der Name einer oder mehrerer Spalten (die jeweils mit einem "," voneinander getrennt werden) sein oder das Zeichen '*' für alle Spalten.

Beispiele für eine komplette Anweisung:

- SELECT spalte1 FROM tabelle1

(Diese Anfrage zeigt nur spalte1 aus der Tabelle 'tabelle1'.)

- SELECT * FROM tabelle1

(Diese Anfrage zeigt alle Spalten aus der Tabelle 'tabelle1'.)

Der **WHERE** Befehl sorgt dafür, dass bei einer SELECT-Anfrage nicht alle Zeilen ausgegeben werden, sondern nur die Zeilen, die eine bestimmte Bedingung erfüllen.

WHERE steht für gewöhnlich nach der FROM-Anweisung und dem zugehörigen Tabellennamen.

Die Bedingung, die zu der WHERE-Anweisung gehört, kann z.B. sein, dass nur Personen mit dem Namen 'Meier' angezeigt werden.

Beispiele für eine komplette Anweisung:

- SELECT * FROM tabelle1 WHERE name = 'Meier'

(Das zeigt nur die Zeilen an, wo die Spalte name das Attribut 'Meier' besitzt.)

- SELECT kontostand FROM tabelle1 WHERE kontostand > 500

(Nur Kontostände werden aufgelistet, die größer als 500 sind.)

GROUP BY fügt gleiche Elemente einer Spalte zusammen. Das macht Sinn, wenn du z.B. eine Tabelle mit den Spalten 'bank' und 'kontostand' hast und die Kontostände jeder Bank zusammenrechnen möchtest.

Beispiel für eine komplette Anweisung:

- SELECT bank, sum(kontostand) FROM tabelle1 GROUP BY bank

(Das zeigt eine Liste aller Banken mit der zugehörigen Summe der Kontostände.)

III Bedienung

III.1 Systemvoraussetzungen

SQL-Lerner ist in Flash programmiert. Das so genannte Shockwave-Dateiformat „SWF“ wird mit dem von der Firma Adobe kostenlos zur Verfügung gestellten Flash Player 10 angezeigt. Ältere Versionen des Players sind ebenfalls tauglich, insofern sie Action Script 3.0 unterstützen. Das Abspielprogramm ist auch als Browser-Plugin erhältlich und integriert sich in die gängigen Internetbrowser. Die Mindestsystemvoraussetzungen für den Nutzer von SQL-Learner sind somit dieselben, wie für den Flash Player 10:[1] Unter Windows: Intel Pentium II 450 MHz, AMD Athlon 600 MHz, unter Macintosh: PowerPC G3 500 MHz oder schneller und unter Linux ein Prozessor mit 800 MHz. 128 MB Arbeitsspeicher und ebenso viel Speicher auf der Grafikkarte sollten ausreichend sein.
Um die Ausgabefunktion nutzen zu können, muss die Datei „sqllearner.php“ über die in der Variablen „schnittstelle“ eingetragenen URL erreichbar und serverseitig ausführbar sein. Das ist für gewöhnlich das selbe Verzeichnis, in dem sich auch die übrigen Dateien befinden. Der (Web)-Server muss die Programmiersprache PHP ab Version 5 interpretieren können (PHP 5 ist nötig, da die verwendeten Befehle zur XML-Manipulation nicht vollständig kompatibel mit denen aus PHP 4 sind). Weiterhin muss auf dem Server eine MySQL-Datenbank mit dem in der „global.php“ eingetragenen Namen angelegt werden können. Der Grund hierfür ist, dass die SQL-Anfragen, die in SQL-Learner vom Benutzer gemacht werden, an das externe Datenbanksystem weitergeleitet werden, um eine möglichst originale Reaktion auf die SQL-Anfragen zu erhalten.

III.2 Installation

Folgende Schritte sind nötig um den SQL-Learner in Betrieb zu nehmen:

1. Entpacken Sie alle Dateien aus dem Archiv „sqllearner.zip“ in ein

[1] [Ad09a]

beliebiges Verzeichnis. Dazu eignet sich fast jedes beliebige Dateikompressionsprogramm, z.B. das kostenlose 7-Zip, das unter http://7-zip.org/ heruntergeladen werden kann.

2. Öffnen Sie die Datei „global.php“ in einem Texteditor und passen Sie die Variablen (im Beispiel fett gedruckt) entsprechend ihrem (Web)-Server an. Für den lokalen Betrieb wird die Software „Xampp“ (das ist eine Distribution von Apache, MySQL, PHP und Perl, die es ermöglicht diese Programme auf sehr einfache Weise zu installieren)[2] mit folgender Konfiguration der „global.php“ empfohlen:

```
<?php
// Hostname oder IP des MySQL-Servers
$mysqlhost = "localhost";
// Username und Passwort zum einloggen in den
Datenbankserver
$mysqluser = "root";
$mysqlpassword = "";
// Name der Datenbank
$mysqldb = "sqllearner";
?>
```

3. Kopieren Sie die Dateien „sqllearner.swf“, „sqllearner.htm“, „AC_RunActiveContent.js“, „sqllearner.php“ und „global.php“ in ein beliebiges Verzeichnis auf dem (Web)-Server bzw. laden Sie diese Dateien mit einem geeigneten Programm hoch.
   Bei Verwendung von Xampp müssen die Dateien in einem Unterverzeichnis von xampp\htdocs\ abgelegt werden.
4. Rufen Sie in einem beliebigen Browserfenster die URL der Datei „sqllearner.htm“ auf. Dabei ist zu beachten, dass diese Internetseite nicht aus einem lokalen Verzeichnis aufgerufen werden darf, damit die Funktionen der PHP-Schnittstelle vom (Web)-Server ausgeführt werden können.

III.2.1 Offline-Version

Es besteht die Möglichkeit den SQL-Learner zu betreiben, ohne das auf ein

[2] [Ap09]

externes Datenbanksystem zugegriffen werden muss. Zudem entfällt der Installationsprozess.

Laden Sie dazu die Datei „sqllearner-offline.swf“ in den Adobe Flash Player.

In der Offline-Version gibt es keine Ausgabe von Ergebnissen der SQL-Anfragen, es wird lediglich die Richtigkeit der im Einführungsmodus und Levelmodus gemachten Anfragen mit den fest vorgegebenen Lösungen überprüft.

Die Offline-Version eignet sich für den schnellen und mobilen Einsatz, insbesondere wenn die Installation auf einem (Web)-Server nicht möglich ist oder zu viel Zeit in Anspruch nehmen würde. An dieser Stelle muss man den Trade-off zwischen Funktionalität und Umgebungsanforderung in kauf nehmen.

Intern wird die Offline-Funktionalität im „functions“-Layer über die Variable „online“ definiert.

```
var online=1;
```

Dies bedeutet, dass SQL-Learner normal betrieben wird.

```
var online=0;
```

Dies hingegen verhindert, dass ein Datenaustausch mit der Schnittstelle erfolgt.

III.3 Oberfläche

III.3.1 Aufbau und Programmablauf

Das folgende Diagramm beschreibt den Aufbau des SQL-Learners:

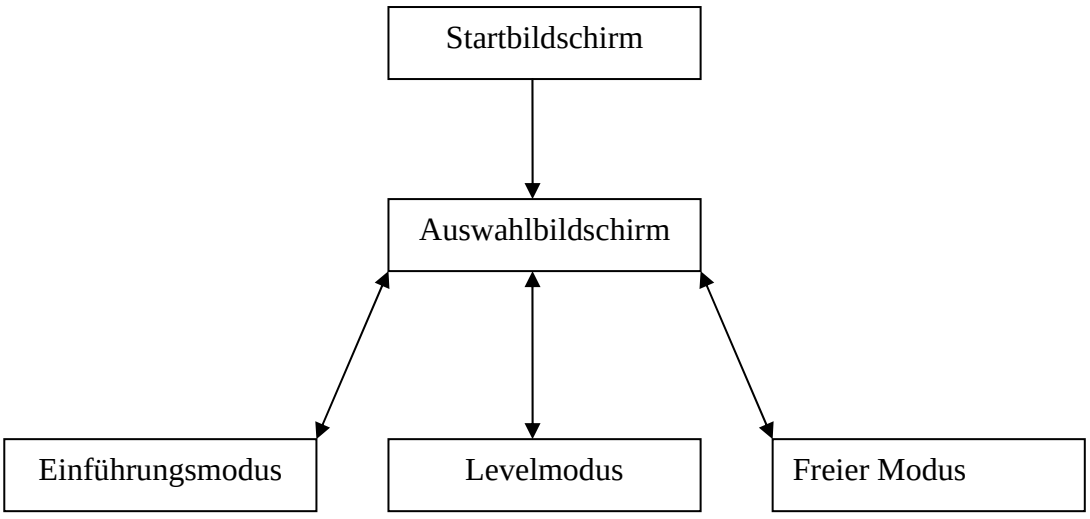

Abb. 1: Aufbau des SQL-Learners

Nach dem Programmstart befindet sich der Benutzer zunächst im Startbildschirm, der den Titel des Programms, sowie einige persönliche Angaben zum Autor enthält. Der Startbildschirm ist nur einmal zum Programmstart sichtbar und kann durch einmaligen Mausklick ausgeblendet werden. In den Startbildschirm kann später nicht mehr zurückgekehrt werden, er erscheint aber bei jedem Programmstart erneut. Der Benutzer wechselt dann zum Auswahlbildschirm.

Der Auswahlbildschirm ist in sofern wichtig, als dass er die Möglichkeit bietet, zwischen den drei verschiedenen Modi hin- und herzuschalten. Genauere Informationen über die Unterschiede der Modi werden später noch aufgezeigt. Weiterhin wird im Auswahlbildschirm das Gespräch zwischen zwei Jugendlichen mittels Sprechblasen simuliert um einen Einstieg in die Thematik zu vermitteln. Ein Mausklick auf das [x] in der Sprechblase oder eine der Figuren führt zum jeweils nächsten Gesprächsabschnitt. Der Benutzer kann mit einem Klick auf die „Zurück“-Schaltfläche zum vorherigen Gesprächsabschnitt wechseln. Im letzten Schritt des Gespräches kann durch einen Mausklick auf eine der Spielfiguren ausgewählt werden, ob der Junge oder das Mädchen im weiteren Verlauf den Platz des Avatars ausfüllen soll. Darauf hin erscheinen Schaltflächen zur Auswahl der Modi.

Die drei verschiedenen Modi sind vom Aufbau her identisch. Im oberen Textfeld wird immer angegeben, wo sich der Benutzer momentan im Programm befindet. Der rechte Teil, die Befehlsliste, zeigt alle in der aktuellen Aufgabe verfügbaren SQL-Befehle an. Das Eingabefeld ist der rechteckige, hellblaue Bereich am unteren Rand. Befehle aus der Befehlsliste können hier abgelegt und in die richtige Reihenfolge gebracht werden. Links daneben ist eine der Spielfiguren aus dem Auswahlbildschirm als Avatar. Der große Bereich mit der Tabelle und den Schaltflächen ist der Datenbereich. Er gibt jeweils alle für die Aufgaben relevanten Tabellen an und ermöglicht es, die Ansicht zwischen den Tabellen zu wechseln. Die Kontrollschaltflächen ermöglichen zum einen die Navigation zwischen den Aufgaben und Modi, zum anderen das schnelle Löschen des Eingabefeldes und ein Senden der im Eingabefeld erstellten SQL-Anfrage.

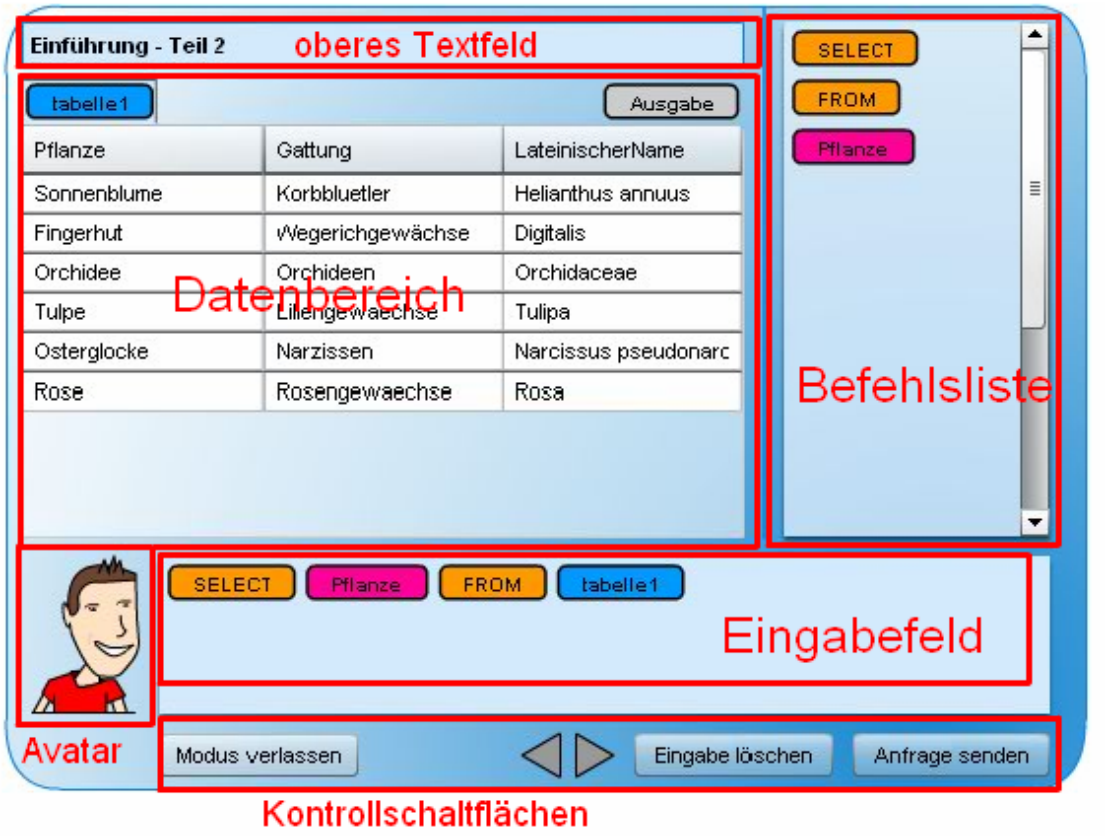

Abb. 2: Aufbau des Hauptfensters

III.3.2 Befehlsliste

Die Befehlsliste ist sozusagen der Wortschatz, der benötigt wird, um in SQL eine Anfrage zu formulieren. Er enthält stets alle nötigen Befehle, mit denen die aktuelle Aufgabe gelöst werden kann. Jeder Befehl ist unendlich oft vorhanden. Auch im Levelmodus sind in der Befehlsliste nur die Befehle eingebaut, die zur Lösung der jeweiligen Aufgabe benötigt werden. Der „Freie Modus“ hingegen, bietet nahezu alle Befehle an, die im Rahmen der Abfrage von Daten aus einer Datenbank sinnvoll verwendet werden können.
Um einen der SQL-Befehle zu verwenden, wird dieser per Drag & Drop in das Eingabefeld gelegt. Mit Drag & Drop ist das Gedrückthalten der linken Maustaste über dem jeweiligen Befehl und das spätere Loslassen der Maustaste gemeint, während sich der Mauszeiger über dem Eingabefeld befindet. Der Befehl wird dann den Befehlen im Eingabefeld an einer beliebigen Stelle hinzugefügt, je nachdem wo der Befehl mit der Maus abgelegt wird.
Jedes Element der Befehlsliste ist eingefärbt. Die Farbe richtet sich nach der jeweiligen Bedeutung des Elementes für eine SQL-Anweisung.

Folgende Unterscheidungen gibt es:

Abb. 3: Unterscheidung der Elemente der Befehlsliste

Tabellennamen sind die Namen der Tabellen (Relationen) in einer relationalen Datenbank. Wenn eine Tabelle in einer Datenbank angelegt wird, wird ihr dieser Name zugewiesen und in SQL-Learner auf einem blauen Hintergrund dargestellt. Eine Tabelle wiederum enthält in der Regel eine oder mehrere Spalten. Die Namen dieser Spalten werden Attribute genannt und hier in pink dargestellt. Jede Zeile besteht aus einer Reihe von Attributwerten. Attributwerte werden in lila dargestellt. Operatoren, z.B. „+, -, AND, OR, …“ und Funktionsnamen wie z.B. „SUM“, „AVG“, usw. sind grün. Sämtliche nicht zu den übrigen Kategorien gehörige verwendete SQL-Anweisungen werden in der Befehlsliste in orange dargestellt. Die farbliche Hinterlegung der SQL-Sprachelemente ist nur zu Lernzwecken implementiert worden und hat keinen Einfluss auf die SQL-Anweisung an sich.

III.3.3 Eingabefeld

Im Eingabefeld wird die Lösung für eine Aufgabe „zusammengesetzt“. Die Lösung ist stets eine Anfrage an das Datenbanksystem. Die Auswahl der richtigen Befehle und die Reihenfolge, in der die Befehle angeordnet sind, sind entscheidend. Das Umsortieren der Befehle ist jederzeit möglich, da ein Befehl mit der Maus angeklickt und per Drag & Drop an eine andere Stelle im Eingabefeld gelegt werden kann. Soll ein Befehl aus dem Eingabefeld entfernt werden, so kann dieser ebenfalls mit Drag & Drop genommen und dann außerhalb des Eingabefeldes fallen gelassen werden. Stellt sich später heraus, dass der Befehl doch noch benötigt wird, so kann dieser wieder neu aus der Befehlsliste geholt werden.

III.3.4 Datenbereich

Dieser Bereich enthält alle Tabellen, die zum Lösen einer Aufgabe nötig sind. Die Tabellennamen sind blau hinterlegt. Bis zu drei Tabellen können in einer

Aufgabe zum Einsatz kommen. Die jeweils aktive Tabelle hat eine zusätzliche farbliche Hinterlegung und einen eckigen Rahmen unter dem Tabellennamen. Um die Ansicht zwischen den Tabellen zu wechseln, wird der Tabellenname einfach angeklickt.

Jedes Mal wenn eine neue Aufgabe aufgerufen oder eine Anfrage gesendet wird, erfolgt eine Aktualisierung des Datenbereichs. Sobald das Ergebnis einer SQL-Anfrage vorliegt, wird eine zusätzliche, graue Schaltfläche mit der Bezeichnung „Ausgabe“ eingeblendet und auf die Ergebnisansicht gewechselt. Um eine der Tabellen wieder anzuzeigen kann der Tabellenname wie gewohnt angeklickt werden.

Attribute (die Spaltenüberschriften), Attributwerte (Inhalt jeder normalen Zelle einer Tabelle) und auch die Tabellennamen können direkt vom Datenbereich per Drag & Drop im Eingabefeld abgelegt werden, um sie in eine SQL-Anfrage einzubauen. Die für eine Aufgabe nötigen Attribute und Attributwerte sind nur in Ausnahmefällen in der Befehlsliste vorhanden.

III.3.5 Avatar-Funktion

Der Avatar, also die Figur im Bereich links neben dem Eingabefeld, informiert am Anfang jeder Aufgabe über die aktuelle Situation. Unter Umständen muss der in einer Sprechblase erscheinende Text heruntergescrollt werden, wenn er zu lang ist. Dieser erste Text, der auch die Aufgabe für dieses Szenario enthält, kann jederzeit durch einen Linksklick auf den Avatar ein und ausgeblendet werden.

Nachdem eine Anfrage gesendet und das Ergebnis intern überprüft wurde, erscheint die Sprechblase erneut und gibt eine positive bzw. negative Rückmeldung, ggf. mit weiteren Hinweisen aus.

Elemente aus der Befehlsliste können per Drag & Drop auf den Avatar gezogen werden. Es erscheint dann eine ausführliche Beschreibung des Befehls, sowie einige Beispiele zur Verwendung.

Der Avatar ist somit der erste Anlaufpunkt, wenn Hilfe benötigt wird. Insbesondere in den ersten Einführungsszenarien informiert er auch über die Verwendung von Funktionen des SQL-Learners.

III.3.6 Fehlermeldungen

Fehlermeldungen des SQL-Learners werden durch den Avatar in einer

Sprechblase ausgegeben. Folgende Fehlermeldungen sind möglich:

```
Die Verbindung zur Datei sqllearner.php, die zur
Ausführung dieses Lernprogramms nötig ist, konnte
nicht hergestellt werden.
```

Bei dieser Fehlermeldung handelt es sich um einen Input-Output-Error. Die Datei sqllearner.php konnte nicht gefunden werden. Zunächst sollte sichergestellt werden, dass die Datei sqllearner.php auf dem (Web)-Server im selben Verzeichnis wie die anderen Komponenten des SQL-Learners liegt (bzw. in dem in der internen Variablen „schnittstelle“ definierten Pfad) und auch vom Server interpretiert wird (siehe Systemvoraussetzungen).

```
Die Verbindung zwischen der Schnittstelle
(sqllearner.php) und der MySQL-Datenbank konnte
nicht aufgebaut werden.
```

Dieser Fehler tritt auf, wenn die Konfiguration in der global.php fehlerhaft ist, die Datenbank „sqllearner“ nicht angelegt, bzw. nicht auf sie zugegriffen werden kann.

```
Da der SQL-Learner derzeit für den Offline-Modus
konfiguriert ist, ist eine Ausgabe der Ergebnisse
deiner SQL-Anfragen nicht möglich.
```

Wenn der SQL-Learner in der Offline-Version betrieben wird, wird diese Meldung im „Freien Modus“ als Startmitteilung angezeigt. Für weitere Informationen siehe in den Abschnitt „Offline-Version“.

III.3.7 Einführungsmodus

Der Einführungsmodus richtet sich an alle, die bisher wenig bis gar keine Erfahrung mit SQL-Befehlen gesammelt haben. Im Gegensatz zu dem Levelmodus werden hier neben der Aufgabe viele nützliche Tipps gegeben, wie die Aufgabe gelöst werden kann bzw. was die Befehle bedeuten. Es wird häufig auf die Hilfefunktion des Avatars aufmerksam gemacht und bei

fehlerhaften SQL-Anfragen sehr hilfreiche Tipps gegeben. Vorausgesetzt werden in späteren Aufgaben des Einführungsmodus nur Befehle, die bereits im selben Modus vorgekommen sind und in der entsprechenden Form genutzt wurden. Unter Berücksichtigung des Gesprächinhalts im Auswahlbildschirm sollte auch jemand, der noch nie etwas mit einem Datenbanksystem zu tun hatte, den Einführungsmodus durchlaufen können. Der Einführungsmodus benutzt für alle Aufgaben die gleichen bzw. sehr ähnlichen Tabellen.

III.3.8 Levelmodus

Im Levelmodus geht es primär um die Lösung von Aufgaben für Personen, die bereits Erfahrung mit SQL-Befehlen haben, also bereits durch Unterrichte, Bücher oder im Einführungsmodus Wissen erworben haben. Der Benutzer sollte mit dem Umgang des SQL-Learners bereits vertraut sein. Der Schwierigkeitsgrad steigt bei jeder Aufgabe leicht an. Hinweise zur Aufgabe gibt es gewöhnlich keine, wohl aber eine Hilfestellung bei falscher Antwort. Im Levelmodus wird problemorientierter gearbeitet. Der gewählte Avatar bittet um Hilfe bei einigen Problemen aus seinem Alltag. Dies hat zur Folge, dass die verwendeten Tabellen sehr unterschiedlich sind.

III.3.9 Freier Modus

Der „Freie Modus“ bietet die Möglichkeit, selbstständig die SQL-Befehle zu „entdecken“ bzw. deren Auswirkungen auf einige vorgegebene Tabellen zu beobachten. Natürlich ist dazu Grundwissen nötig, das aber entweder mit Hilfe der zwei anderen Modi oder über ein Fachbuch erarbeitet werden kann. Im Folgenden sind einige interessante Anweisungen aufgelistet, die im „Freien Modus“ ausprobiert werden können, über die dann z.B. im Unterricht gesprochen werden könnte. Es handelt sich dabei immer um ein Abbild des Eingabefeldes

- Alle Telefonnummern auflisten, die dem Anbieter „versafon“ zugeordnet sind

- Alle Telefonnummern auflisten, aber keine doppelt.

- Man möchte wissen, wie viel Geld die Banken jeweils noch auf den Konten der Kunden haben, wenn man davon ausgeht, dass alle Personen ihre Ausgaben abheben und ihre Einnahmen auf den Konten lassen.

```
SELECT Bank , SUM(Einnahmen) - SUM(Ausgaben)
FROM finanzen GROUP BY Bank
```

- Es soll eine Liste mit Personen erstellt werden, deren Name nicht mit ‚mann' endet und deren Name auch nicht ‚Meier' ist.

```
SELECT * FROM person WHERE Name NOT
LIKE '%mann' AND NOT Name = 'Meier'
```

- Alle Daten der drei Tabellen miteinander verbinden. Man beachte die Zeilen, die sich nur in der Telefonnummer unterscheiden, weil einige Personen mehrere Telefonnummern besitzen.

```
SELECT * FROM person NATURAL JOIN finanzen
NATURAL JOIN telefon
```

- Eine Namensliste mit zugehörigen Ein- und Ausgaben jeder Person erstellen

```
SELECT Name , Einnahmen , Ausgaben FROM
person NATURAL JOIN finanzen
```

- Nur die Personen und zugehörigen Ausgaben anzeigen, wenn die Person Ausgaben zwischen 500 bis 1000 Euro besitzt.

```
SELECT Name , Ausgaben FROM person
NATURAL JOIN finanzen WHERE Ausgaben BETWEEN
500 AND 1000
```

- Alle Personen auflisten, die keine Telefonnummer besitzen.

```
SELECT * FROM person WHERE id NOT
IN ( SELECT id FROM telefon )
```

IV Konzept / Entwurf

IV.1 Vorüberlegungen

IV.1.1 Erstes Konzept

Das erste Konzept entstand im Rahmen der Themenfindung für diese Examensarbeit. Dazu ist zu sagen, dass es lediglich als Skizze existierte und sich in einigen Punkten von der tatsächlichen Gestaltung und Funktion unterscheidet (siehe Designentscheidungen in den folgenden Abschnitten). Das erste Konzept sah bereits die Unterteilung des Hauptfensters in die Teile Eingabefeld, Befehlsliste und Datenbereich vor. Der Datenbereich sollte den meisten Platz einnehmen und einen Wechsel zwischen den einzelnen Tabellen und der Ausgabe, die ebenfalls als Tabelle erscheinen sollte, ermöglichen. Die Befehlsliste, also das Verzeichnis mit sämtlichen für eine Aufgabe nötigen Befehlen, war rechts neben dem Datenbereich angesiedelt. Unterhalb des Datenbereiches und der Befehlsliste war das Eingabefeld positioniert. Befehle waren per Drag & Drop von der Befehlsliste in das Eingabefeld zu ziehen. Die Befehle im Eingabefeld sollten zusammen schließlich eine SQL-Anfrage bilden. Überlegungen, was geschehen sollte, wenn zu viele Befehle in der Befehlsliste oder im Eingabefeld vorhanden sind, gab es zu diesem Zeitpunkt noch keine. Ebenso gab es noch keine genaue Planung, welche Schaltflächen im Hauptfenster nötig sind. Lediglich eine Schaltfläche mit der Bezeichnung „Ausgabe“, die ein Verarbeiten der Befehle im Eingabefeld initiieren sollte, war vorhanden. Im oberen, rechten Bereich über der Befehlsliste war noch eine Schaltfläche mit einem „?“ geplant, die eine Hilfestellung zu einer Aufgabe oder der Bedienung des SQL-Learners geben sollte. Diese Funktion hat nun der Avatar übernommen (siehe entsprechendes Kapitel). Für jede Aufgabe sollte das Hauptfenster mit den entsprechenden Einstellungen, also einer bereits vorhandenen Tabelle und den in der Befehlsliste vorhandenen Befehlen initialisiert werden. Wie der Wechsel zwischen den Aufgaben vollzogen werden sollte, war noch nicht durchdacht worden.

Während des Gespräches mit Herrn Prof. Dr. Thomas und Herrn Weigend entstanden einige weitere Ideen, die in das Grundkonzept übernommen wurden. Es handelt sich dabei um die folgenden Punkte:

- eine Möglichkeit für Nutzer schaffen, den Umgang mit dem Programm zu erlernen

- ein levelbasierter Aufgabenmodus mit schwieriger werdenden Aufgaben, der als Motivation gleichzeitig eine Geschichte vorantreibt
- eine Möglichkeit für erfahrenere Nutzer, mit möglichst allen verfügbaren Befehlen frei zu experimentieren
- eine Identifikationsfigur integrieren
- Standards des virtuellen Museums einhalten, damit das Exponat später integriert werden kann (das Projekt sollte aus einer einzelnen SWF-Datei bestehen, eine Rückgabefunktion zur Leistungsbewertung bieten und von verschiedenen Nutzern gleichzeitig ausführbar sein)
- gute Anpassbarkeit der Aufgaben gewährleisten

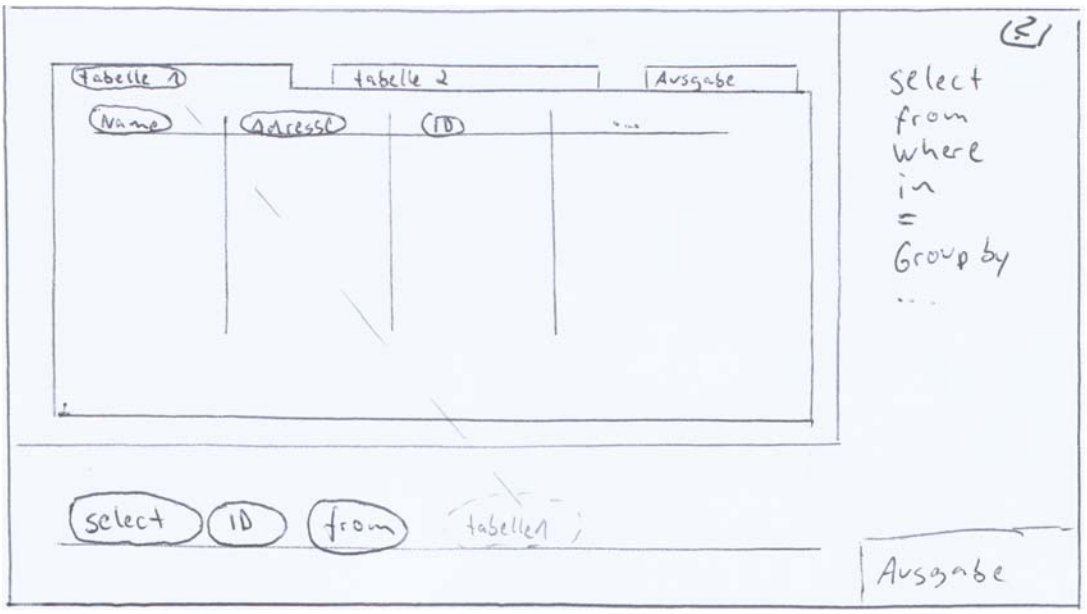

Abb. 4: Erstes Konzept

IV.1.2 Verarbeitung von Anfragen

Zunächst war im ersten Konzept vorgesehen, dass die im Eingabefeld zusammengesetzten Befehle vom Programm selbst vollständig verarbeitet werden, d.h. auf Richtigkeit geprüft, anschließend ausgeführt werden und eine Ausgabe entsprechend der SQL-Anfrage erstellt wird. Es gab an dieser Stelle zwei Möglichkeiten:

1.) Die SQL-Anfrage wird in einen String umgewandelt und direkt mit der in der Aufgabe festgelegten Musterlösung verglichen. Wenn der Inhalt der Strings übereinstimmt, wird jeweils eine vom Aufgabenautor vorab erstellte Lösungstabelle ausgegeben.
   Probleme bei dieser Vorgehensweise:

- Nur bei vollständig richtigen Anfragen gibt es eine Ausgabe.
- Der Nutzer hat keine Möglichkeit zu sehen wo der Fehler liegt. Insbesondere bei komplexeren Anfragen könnte dies zu Problemen führen.
- Ein für erfahrenere Nutzer angedachter Experimentiermodus, bei dem es darum geht, die Wirkung von verschiedenen Anfragen zu testen fällt weg, da man selbst bei vorher festgelegten Datensätzen nicht für jede SQL-Anfrage das richtige Ergebnis im Programm hinterlegen kann.

2.) Die SQL-Anfrage wird in einen String umgewandelt und wie bei 1.) mit einer Musterlösung verglichen. Ungeachtet, ob die Lösung richtig ist, wird die Anfrage in ihre Bestandteile zerlegt („geparst“) und die Syntax überprüft. Wenn die SQL-Anfrage gültig ist, wird sie ausgeführt, ansonsten ein Fehler zurückgegeben. Zur Ausführung der Anfrage sind für jeden SQL-Befehl verschiedene Funktionen nötig, die den Datensatz der Aufgabe manipulieren. Die Wirkungsweise der SQL-Befehle in einem Datenbanksystem wird also nachprogrammiert. Probleme bei dieser Vorgehensweise:
   - Sehr hoher Programmieraufwand, wenn alle SQL-Befehle berücksichtigt werden sollen.
   - Die Gefahr, dass Befehle vom Programm anders interpretiert werden, als von einem Datenbanksystem mit SQL, ist sehr groß. Es ist kaum möglich, sämtliche Eventualitäten abzudecken.
   - Sinn und Zweck ist es nicht, eine Datenbanksprache neu- bzw. nachzuprogrammieren, sondern sie zu visualisieren und die Verwendung zu vermitteln.

Die Wahl stand nun offen zwischen einem sehr statischen Programm, das kein ausprobieren von Befehlen unterstützt, eine eingebaute Lösungstabelle als Ausgabe für jede Aufgabe besitzt, die aber auf jeden Fall richtig ist oder einem Programm, das die Ausgabe selbst erzeugt, jedoch nicht für alle SQL-Anfragen und nicht unbedingt fehlerfrei ist.
Diese Überlegungen zeigten, dass sowohl die eine, als auch die andere Methode schwerwiegende Mängel aufweist. Das führte zu der Entscheidung, ein externes Datenbanksystem zu benutzen. Die Vorteile sind:

- Das exakte Ergebnis jeder Anfrage wird zurückgegeben.
- Jeder Befehl, der in SQL integriert ist, kann theoretisch genutzt werden.
- Rückgabe von Fehlermeldung und Fehlernummer bei allen Fehlern, die auftreten können.
- Der Programmieraufwand beschränkt sich auf das Programmieren einer Schnittstelle zum Datenbanksystem und das Auswerten der zurückgegebenen Daten.

Natürlich entstehen auch Nachteile, bei der Verwendung einer externen Datenbank:

- Erhöhte technische Voraussetzungen für das Betreiben des SQL-Learners
- Erhöhter Installationsaufwand
- Es muss vom Autor der Aufgaben berücksichtigt werden, dass einige SQL-Befehle wie z.B. „DROP DATABASE sqllearner“ (der die Datenbank sqllearner komplett entfernt) nicht nur die Funktion des SQL-Learners beeinträchtigen können, sofern Datenbanken von anderen Anwendungen vorhanden sind.
- Programmieraufwand der Schnittstelle, da Flash CS3 keine eigene Schnittstelle zu einem Datenbanksystem bietet.

Diese Nachteile erschienen vergleichsweise gering, insbesondere weil sie sich zum Teil umgehen lassen könnten. Die Sicherheit wäre auf jeden Fall gegeben, wenn der SQL-Learner mit den vorgegebenen Aufgaben betrieben bzw. keine Aufgaben mit den entsprechend „gefährlichen“ Befehlen (wie z.B. DROP und DATABASE) erstellt werden würden. Sollte der Installationsaufwand zu groß oder die technischen Voraussetzungen zu hoch sein, bestünde trotzdem noch die Möglichkeit, zumindest das Ergebnis auf Richtigkeit zu überprüfen. Eine Ausgabe wäre dann allerdings nicht möglich (siehe III.2.1 Offline-Version). Theoretisch denkbar, könnte die Ausgabe einer vordefinierten Lösungs-Tabelle sein, die aber zu jeder Aufgabe hinterlegt sein müsste.

Aufgrund der Wahl ein externes Datenbanksystem zur Verarbeitung der SQL-Anfragen zu verwenden, war es nötig geworden, Überlegungen zur Kommunikation zwischen SQL-Learner und dem Datenbanksystem anzustellen:

Da es in Flash CS3 mit Actionscript 3.0 keine Möglichkeit gibt, auf ein Datenbanksystem direkt zuzugreifen, müsste eine Schnittstelle zwischen SQL-Learner und Datenbanksystem erstellt werden.
Im Folgenden ist die grundlegende, geplante Vorgehensweise beschrieben, wie eine SQL-Anfrage des SQL-Learners ausgeführt werden soll.
Beim Öffnen einer Aufgabe soll zunächst der Inhalt des Datenfeldes und der Befehlsliste aus dem Programm selbst geladen und angezeigt werden. Die Daten, die im Datenfeld angezeigt werden sollen, sind im Programm in einer Variablen abgelegt. Diese wird zusammen mit ggf. nötigen Parametern an die Schnittstelle versendet, dort verarbeitet und eine Anfrage an das Datenbanksystem gestellt, sodass dort Tabellen erstellt werden, die den im SQL-Learner angezeigten Tabellen entsprechen. Somit wäre die Konsistenz hergestellt und die in SQL-Learner angezeigten Tabellen würden sich so wie man sie dort sieht auch in der Datenbank befinden.
Erfolgt nun eine SQL-Anfrage, so wird diese per Parameter an die Schnittstelle übergeben und von dort an die Datenbank weitergeleitet. Das Rückgabeergebnis der Datenbank soll von der Schnittstelle wieder in ein für SQL-Learner lesbares Format konvertiert und dann zum SQL-Learner zurückgesendet werden.

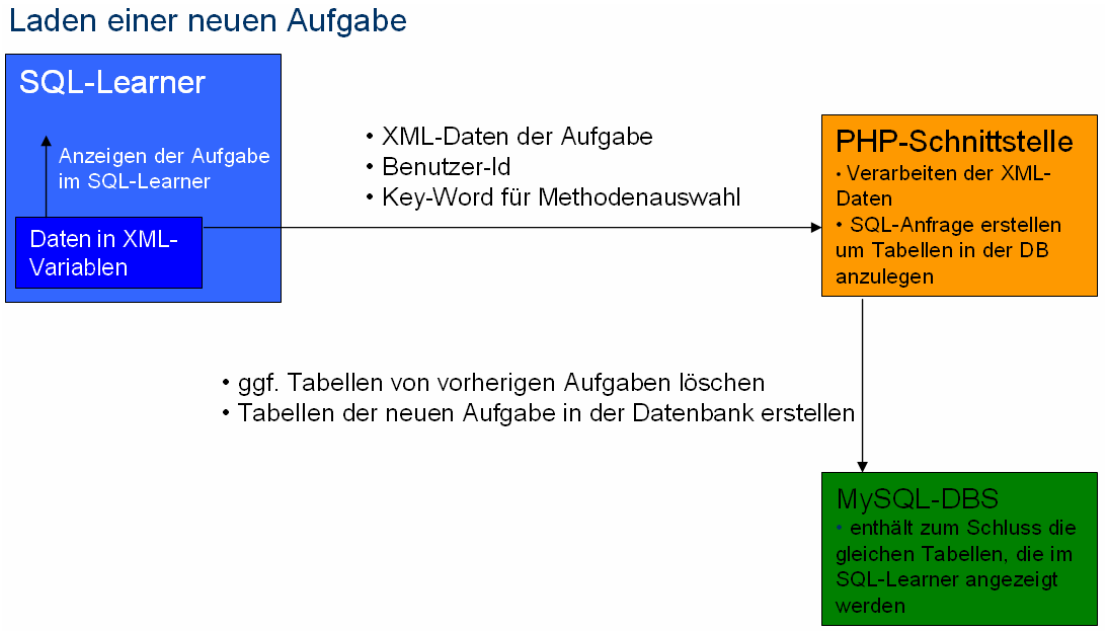

Abb. 5: Schnittstellendiagramm: Laden einer neuen Aufgabe

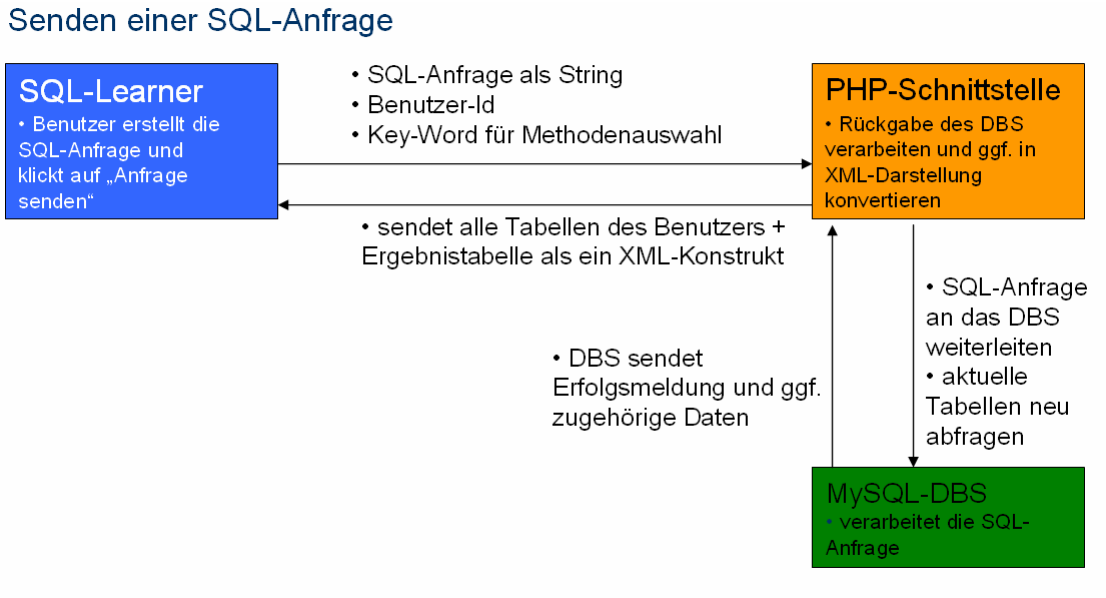

Abb. 6: Schnittstellendiagramm: Senden einer SQL-Anfrage

.

IV.2 Designentscheidungen

In diesem Kapitel geht es um wichtige, die Gestaltung betreffende, Entscheidungen. Im Einzelnen sind dies die Wahl der verschiedenen Steuerelemente, die Benutzerführung, die Strukturierung des Programms, aber auch den Inhalt der Aufgaben. Geklärt werden soll nicht die Frage des „Wie –“, sondern die Frage des „Warum wurde das so gemacht?“. Auf das „Wie“ wird später noch eingegangen.

IV.2.1 Start- und Auswahlbildschirm

Der Nutzer des SQL-Learners sieht beim Programmstart den Startbildschirm. Dieser enthält zunächst einmal den Titel des Programms, damit der Nutzer weiß, dass er sich nun im SQL-Learner befindet. An dieser Stelle schien es ebenfalls sinnvoll, die Informationen zum Autor und zur Verwendung des Programms (Lizenz) zu präsentieren, damit bei möglichen Fragen ein Ansprechpartner zur Verfügung steht und der Nutzer weiß, ob er ein kommerzielles Produkt vor sich hat, ein Open-Source-Projekt oder etwas anderes.

Der Dialog im Auswahlbildschirm soll den Nutzer in die Thematik einführen, damit auch jemand, der sich noch nie mit Datenbanken beschäftigt hat, eine Ahnung davon bekommt, wozu SQL-Befehle nützlich sind und was ihn im Folgenden erwartet. Der Nutzer wird im Rahmen des Dialoges direkt angesprochen und aufgefordert, dem männlichen oder dem weiblichen Avatar

bei den Aufgaben zu helfen. Dies erhöht die Motivation, da ein Schüler oder eine Schülerin eher bereit dazu ist, jemandem zu helfen als selbstständig für sich etwas zu erarbeiten. Zur Entscheidung, warum es sich bei den Avataren um eine Schülerin und einen Schüler handelt, siehe Kapitel IV.2.3.

IV.2.2 Dreiteilung der Modi

Die Aufgaben des SQL-Learners teilen sich wie oben beschrieben in drei verschiedene Kategorien auf:

- Einführungsmodus
- Levelmodus
- Freier Modus

Der Grund für diese Aufteilung sind die verschiedenen Einsatzgebiete des SQL-Learners. Es soll Benutzern, die noch nie etwas von SQL-Befehlen gehört haben, möglich sein, mit dem Programm zu arbeiten und etwas daraus zu lernen. Daher eignet sich dieser Modus besonders für den Einsatz in den ersten Unterrichtsstunden einer Unterrichtsreihe zu dem Thema „Datenbanken“. Insbesondere in Verbindung mit dem Dialog zwischen den beiden Schülern im Auswahlbildschirm bietet der Einführungsmodus eine gute Grundlage, die Verwendung von SQL-Befehlen zu verstehen. Der Anspruch im Levelmodus hingegen ist wesentlich größer, da die Aufgaben an sich komplexer werden, die Fähigkeit, den SQL-Learner zu bedienen vorausgesetzt wird und davon ausgegangen wird, dass der Nutzer bereits Wissen über Datenbanken und insbesondere SQL-Anfragen besitzt. Im „Freien Modus“ gibt es keine konkrete Aufgabenstellung mehr. Er soll dazu dienen, das Wissen selbstständig anzuwenden, um sinnvolle SQL-Anfragen zu erstellen und zu überprüfen, ob das erhaltene Ergebnis dem erwarteten Ergebnis entspricht.

Es wäre möglich gewesen, die drei Modi zusammenzulegen und komplett auf eine Auswahl zu verzichten. Jedoch würde man dann erfahrene Benutzer mit viel zu leichten Anfangsaufgaben unter- und Personen, die noch kein Vorwissen haben ggf. überfordern, wenn nach den Einführungsaufgaben die schwereren Aufgaben aus dem Levelmodus dazukommen. Außerdem wäre es nicht sinnvoll, den „Freien Modus“ erst nach Beendigung aller Aufgaben zuzulassen, da es manchmal besser ist, einen bestimmten SQL-Befehl noch einmal in anderem Kontext zu betrachten, was nur durch die Unterteilung in

verschiedene Modi möglich ist. In einer künftigen Programmversion kann an diese Unterteilung weiter angeknüpft werden, indem neue Modi erstellt würden. Momentan sind nur Befehle der DML (Data Manipulation Language) integriert. Modi mit Aufgaben zu Befehlen aus der DDL (Data Definition Language) oder der DCL (Data Control Language) könnten aber leicht erstellt werden. Ein Modus wäre dann z.B. Einführungsmodus DDL, der den Benutzer mit dem Erstellen von Tabellen in einer Datenbank vertraut macht.

IV.2.3 Avatare und Kommunikation mit dem Nutzer

Zu Beginn ging es um die Frage, wie man dem Nutzer des SQL-Learners zu verstehen gibt, woraus eigentlich die Aufgabe besteht, wenn nun eine oder mehrere Tabellen und eine Menge von möglichen Befehlen angeboten werden.

Eine übergeordnete Aufgabe wie z.B. „Erstelle stets aus den vorhandenen Befehlen eine gültige SQL-Anfrage.“ hätte keinen Sinn gemacht, weil es erstens nicht die Intention des SQL-Learners sein soll, aus fest vorgegebenen SQL-Befehlen eine beliebige gültige Anweisung zusammenzustellen. Der Nutzer soll anhand eines Problems erkennen, welche Befehle er zur Lösung einer konkreten Aufgabenstellung benötigt und damit eine gültige SQL-Anfrage zusammenstellen. Zweitens gibt es zu einer Menge von SQL-Befehlen nicht immer nur eine gültige Anfrage. Es ist also nicht zwangsläufig klar, welches Ergebnis eigentlich erreicht werden soll. Diese Überlegung führt dazu, dass zu jeder Aufgabe eine Aufgabenstellung gehört, die dem Nutzer vor der Bearbeitung der Aufgabe mitgeteilt werden muss. Die Aufgabenstellung sollte auch während der Bearbeitung der Aufgabe noch sichtbar oder zumindest aufrufbar sein, da es auch komplexere Aufgabenstellungen geben könnte, die sich der Nutzer nicht direkt merken kann oder sich der Nutzer zunächst einen Überblick über die vorhandenen Tabellen und Befehle verschaffen muss, um die Aufgabenstellung besser verstehen zu können.

Nun gab es die Möglichkeit, die Aufgabenstellung als festen Bestandteil in die Oberfläche zu integrieren. Dies erschien jedoch aus folgenden Gründen nicht sinnvoll. Aufgabenstellungen können beliebig lang formuliert sein. Daher hätte der Bereich mit der Aufgabenstellung genau wie das Eingabefeld bzw. die Befehlsliste scrollbar sein müssen oder hätte sehr viel Platz von der

Oberfläche weggenommen. Das Hauptfenster des SQL-Learners sollte aber möglichst einfach gestaltet bleiben, damit der Nutzer nicht den Überblick verliert, die Oberfläche nicht überladen wirkt und die wirklich wichtigen Elemente genug Platz haben bzw. groß genug dargestellt werden können. Es hätte den Anschein, als ob man eine Aufgabe nach der anderen bearbeitet, nur weil dort steht, was gemacht werden soll. Damit würde ein persönlicher Bezug fehlen und wenig Motivation erzeugt werden. Das letztere Problem könnte allerdings auch nicht damit gelöst werden, die Aufgabenstellung am Anfang in einem über der eigentlichen Oberfläche liegenden Fenster einzublenden und über eine kleine Schaltfläche ständig wieder abrufbar zu machen. Das erste Problem (Übersichtlichkeit des Hauptfensters) wäre damit aber schon gelöst.

Die Frage war nun, wie sich der persönliche Bezug zum Nutzer herstellen lassen könnte. In vielen Computerspielen wird der persönliche Bezug über eine Spielfigur hergestellt, die vom Spielenden bewegt wird und über die der Spielende mit einer virtuellen Welt innerhalb des Spiels interagiert. Er hat das Gefühl, ein Teil des Geschehens zu sein, wird in die Handlung des Spiels involviert und kann mit seiner Umwelt interagieren. Das Gefühl, ein Teil des Geschehens zu sein, könnte auch von einer Spielfigur erzeugt werden, die Kontakt zum Nutzer des Programms aufnimmt, ihn also z.B. direkt anspricht. Wenn die erzeugte Situation glaubwürdig und in den Augen des Nutzers nachvollziehbar ist, lässt er sich darauf ein. Es wäre somit am besten, wenn die Aufgabenstellung nicht einfach nur sichtbar gemacht werden würde, sondern von einer Spielfigur als persönliches Anliegen vorgetragen würde. Dieses Anliegen könnte sein, dass die Spielfigur den Nutzer um Unterstützung bittet, weil sie die Aufgabe selbst nicht lösen kann. Dieser Ansatz erhöht die Motivation des Nutzers, sich dieses Problems anzunehmen.

Als Spielfiguren eignen sich alle Figuren, mit denen sich der Nutzer gut identifizieren kann. Insbesondere sollte es optimaler Weise für männliche Nutzer eine männliche Figur geben und für weibliche Nutzer eine weibliche Figur, da eine Figur bevorzugt wird, die einem selbst ähnlich ist. Auch ein Tier wäre als Identifikationsfigur möglich, jedoch müsste man sich gut überlegen, was dieses Tier denn mit SQL zu tun hat und warum es z.B. alle Personen mit dem Nachnamen „Müller“ aus einer Tabelle angezeigt haben möchte.

Diese Überlegungen haben dazu geführt, eine männliche und eine weibliche Spielfigur anzubieten, von denen sich der Nutzer eine aussuchen kann, die ihm zur Seite steht. Da es sich bei der Zielgruppe des SQL-Learners in erster Linie um Schülerinnen und Schüler handelt, sind auch diese beiden virtuellen Avatare eine Schülerin und ein Schüler. Die Aufgabenstellung wird in einer Sprechblase angezeigt, um zu suggerieren, dass die Identifikationsfigur den Nutzer um Hilfe bei der Lösung der Aufgabe bittet. Weiterhin wird der Nutzer von dieser Figur für die richtige Lösung einer Aufgabe gelobt oder erhält einen Hinweis bei Fehlern in der Lösung. Dies soll vor allem zum Weitermachen motivieren bzw. dem Nutzer bei Problemen mit der Aufgabe helfen, damit er an dieser Stelle weiterkommt.
Als zusätzliche Funktion des Avatars wurde im Nachhinein implementiert, dass Befehle aus der Befehlsliste per Drag & Drop auf dem Bild des Avatars fallen gelassen werden können um eine Erklärung zu diesem SQL-Befehl zu erhalten. Es soll so aussehen, dass der Avatar (der ja eine Schülerin oder einen Schüler darstellt) nicht allwissend ist (denn sonst könnte er auch die Aufgaben alleine lösen), sondern die Befehle im Schulbuch nachschlägt und dem Nutzer die Informationen dazu zur Verfügung stellt.

IV.2.4 Drag & Drop der Befehlselemente

Die Idee, Befehle aus einer Art Verzeichnis auszuwählen, schien von Anfang an die beste Lösung zu sein. Dennoch hätte man auf die Befehlsliste verzichten und ein Eingabefeld für Text anbieten können, damit der Nutzer nur Befehle erhält, die er mit Hilfe der Tastatur eingibt. Das hätte jedoch den Schwierigkeitsgrad so stark erhöht, dass der SQL-Learner nur noch bedingt für die genannte Zielgruppe geeignet gewesen wäre.
Eine Alternative zum Drag & Drop wäre das bloße Anklicken eines Befehls der Befehlsliste gewesen, sodass dieser einfach am Ende des Eingabefeldes angehängt worden wäre. Allerdings erschien die Verwendung von Drag & Drop geeigneter, da der Nutzer die Befehle so direkt an der richtigen Stelle positionieren kann und auch sieht, woher die Befehle kommen, wenn er z.B. die Tabellennamen, die im Datenbereich angeordnet sind, in das Eingabefeld zieht. Die Tabellennamen bieten im Gegensatz zu den anderen Drag & Drop Elementen zusätzlich noch die Möglichkeit, zwischen den Tabellen hin- und herzuschalten. Diese Doppelverwendung schien am besten geeignet, um

dem Nutzer zu zeigen, dass die Tabellennamen eine wichtige Bedeutung für die SQL-Anfragen haben und jeweils für eine Tabelle in der Datenbank stehen. Der Hintergrund der Schaltfläche der Ausgabe-Tabelle ist grau, um klar abzugrenzen, dass es sich bei der Schaltfläche nicht um ein Element handelt, dass in eine SQL-Anfrage eingebunden werden soll.

IV.2.5 Datagrid

Aus dem ersten Konzept ging hervor, dass die Tabellen aus einer Datenbank als Tabellen, zwischen denen man hin- und herwechseln kann, visualisiert werden. Folgende Möglichkeiten wurden zur Darstellung der Tabellen in Betracht gezogen:

- Dynamische Textfelder, die wie eine Tabelle angeordnet sind
- Datagrid

Bei der Idee, viele Textfelder zu verwenden wäre ein MovieClip erstellt worden, der als Klassennamen z.B. „my_tabelle“ erhalten hätte, sodass dieser MovieClip mehrfach (für jede Tabelle in der Datenbank, die zu einem bestimmten Benutzer gehört) auf der Bühne eingebunden werden kann. Der MovieClip selbst hätte eine ScrollPane enthalten und während der Laufzeit beliebig viele Textfelder unter- und nebeneinander erzeugt, um die Spalten und Zeilen einer Tabelle nachzubilden. Auch eine feste Spalten- oder Zeilenanzahl wäre denkbar gewesen. Dies hätte jedoch eine zusätzliche Einschränkung im Aufgabendesign zur Folge gehabt. Der Bau einer solchen Struktur schien nicht sinnvoll, da es für genau diesen Zweck ein schon vorgefertigtes Steuerelement gibt. Das Datagrid ist sehr geeignet, da es viele Funktionen bietet, die in der zuerst aufgezeigten Lösung zusätzlich hätten programmiert werden müssen. Das wäre unter anderem die Anpassung der Spaltenbreite, integrierte Scrollbars und der direkte Zugriff auf einzelne Zellen einer Tabelle gewesen. Die Daten für die Tabellen können im Datagrid mit einem Objekt der Klasse DataProvider bereitgestellt werden. Diese Klasse unterstützt XML direkt, sodass es nicht nötig ist, den Inhalt der XML-Variable zu zerlegen und dann elementweise einzusortieren. Mehr zur Entscheidung warum XML verwendet wird und wie es verwendet wird siehe Abschnitt IV.3.3 und IV.4.4.

IV.2.6 Design der Aufgaben

Die Aufgaben des Einführungsmodus basieren hauptsächlich auf einer einzigen Tabelle, da der Nutzer, der vielleicht bisher nie etwas mit Datenbanken und SQL zu tun hatte, nicht verwirrt werden soll. In den ersten Aufgaben geht es darum, dass der Nutzer die Funktionen des SQL-Learners kennen lernt und sich an die Programmoberfläche gewöhnt. Wichtige Elemente sind hier das Zusammensetzen von SQL-Anfragen, die Bedienung der Hilfe-Funktion (SQL-Befehle auf den Avatar ziehen und die Maustaste loslassen), sowie das Wechseln der Ansichten zwischen den normalen Tabellen und der Ausgabe. Die Entscheidung, den Einführungsmodus auf diese Weise zu gestalten ist damit begründet, dass jemand, der noch nicht mit dem SQL-Learner gearbeitet hat, nicht erst das Handbuch lesen müssen soll, sondern direkt in das Programm einsteigen kann. Die weiteren Aufgaben im Einführungsmodus decken fast alle Themen bzw. SQL-Bausteine ab, die auch im Levelmodus auftauchen. So kann das im Einführungsmodus gelernte später im Levelmodus weiter vertieft werden. Sowohl im Einführungs- als auch im Levelmodus enthält die Befehlsliste stets alle nötigen Befehle zum Lösen einer Aufgabe. Auf zusätzliche Befehle, die z.B. den Nutzer verwirren sollen wurde aus zwei Gründen verzichtet. Wie bereits beschrieben richtet sich der SQL-Learner in erster Linie an Schüler und Schülerinnen der gymnasialen Oberstufe. Die Sprache SQL ist dort vom Schwierigkeitsgrad her bereits den komplexeren Themen zuzuordnen. Somit ist der Schwierigkeitsgrad insbesondere in den Aufgaben des Levelmodus auf jeden Fall ausreichend hoch. Zum anderen würde eine erweiterte bzw. eine vollständige Befehlsliste dazu führen, dass in einigen Aufgaben alternative, vielleicht abwegige Lösungen gefunden werden, die jedoch syntaktisch korrekt sind und somit als richtig erkannt werden müssten. Der SQL-Learner ist aber darauf ausgelegt, nur die vordefinierte Lösung (mit Ausnahme der Vertauschung der Tabellennamen beim „NATURAL JOIN“) als richtig zu erkennen.

Die Aufgaben des Levelmodus sind im Gegensatz zu Aufgaben des Einführungsmodus komplexer und beschäftigen sich nicht nur mit einer oder zwei Tabellen, die in sämtlichen Aufgaben auftauchen, sondern mit verschiedenen Tabellen, die in der Welt des Avatars vorkommen. Nachdem der Einführungsmodus absolviert wurde, ist davon auszugehen, dass sich der

Nutzer nun nicht mehr durch verschiedene Tabellen verwirren lässt und mit der Bedienung des Programms vertraut ist. Die Abwechslung durch die unterschiedlichen Tabellen und Aufgabenstellungen, sowie der Gedanke den Avatar zu unterstützen sollten für eine erhöhte Motivation sorgen.
Der „Freie Modus“ schließlich besteht aus drei Tabellen mit dem typischen Inhalt einer Adressdatenbank. Dabei können die Tabellen über die Spalte ‚id’ miteinander verbunden werden. Dieser Inhalt wurde gewählt um dem Nutzer ein Beispiel für einen möglichst realen Datenbankinhalt zu geben, sodass viele sinnvolle und unterschiedliche SQL-Anfragen gestellt werden können. Eine Datenbank, die verschiedene Personen mit zugehörigen Informationen enthält, sollte auf den ersten Blick zu verstehen sein, wenn zuvor mit den anderen Modi gearbeitet wurde.

IV.3 Technische Entscheidungen

IV.3.1 Flash CS3

Die Verwendung von Flash CS3 ergab sich aus mehreren Gründen. Zum einen ist der SQL-Learner als Exponat des Virtuellen Museums (LIMO) ausgelegt und soll demnächst entsprechend darin integriert werden. Das Virtuelle Museum selbst besteht aus einer Weboberfläche und aus einem so genannten „Presenter“, einem Flash-Film, der die Auswahl eines Exponats des Museums ermöglicht und die Exponate aufruft. Dieser Presenter ist in Flash CS3 erstellt worden. Exponate müssen ebenfalls Flash-Filme sein, um in das Virtuelle Museum integriert werden zu können. Eine Verwendung der Version CS3 bot sich besonders an, da es die neuste Version von Flash ist und das didaktische Institut der Informatik der WWU Münster einen Arbeitsraum zur Verfügung stellt, wo auf jedem Rechner Flash CS3 vorinstalliert ist. Die Anforderungen, die sich aus dem ersten Konzept ergeben haben, sind durch die Möglichkeiten, die Flash CS3 bietet, voll abgedeckt.

IV.3.2 Action Script 3.0

Action Script 3.0 wurde laut Adobe mit dem Ziel entwickelt, das Erstellen hochkomplexer Anwendungen mit umfangreichen Datensätzen und einer objektorientierten, wieder verwendbaren Codebasis zu ermöglichen.[3]

[3] [Ad09b]

Die Wiederverwendbarkeit war der ausschlaggebende Faktor für die Bevorzugung von Action Script 3.0 gegenüber Action Script 2.0. Objekte, die häufig wieder verwendet werden, sind z.B. die Objekte der Klasse „MyButton“, die von der MovieClip-Klasse abgeleitet werden und die SQL-Befehle auf der Bühne visualisieren sollen.
Weiterhin kann Action Script 3.0 Code bis zu zehn Mal schneller ausgeführt werden als älterer Action Script-Code.

IV.3.3 Datenhaltung innerhalb des SQL-Learners

Zunächst war zu ermitteln, welche Daten im SQL-Learner gespeichert werden müssen. Dies sind vor allem die Daten jeder Aufgabe, also Startbelegung der Tabellen, die verfügbaren SQL-Befehle in der Befehlsliste, die Aufgabe selber, ein Hinweistext, der bei Eingabe einer falschen Lösung erscheint, ein Text, der bei der richtigen Lösung erscheint und die richtige Lösung als SQL-Anweisung selbst.
Es schien hierfür am einfachsten, eine Klasse „Aufgabe“ zu erstellen, die zur Speicherung dieser Elemente geeignet ist. Dabei bestehen alle Angaben außer der Startbelegung und der Elemente der Befehlsliste aus Strings. (siehe dazu IV.4.10)
Die Elemente der Befehlsliste sind Objekte der Klasse „MyButton“ und werden zusammen als ein Array in der Aufgaben-Klasse definiert.
Für die Startbelegung wurde eine XML-Variable gewählt. XML steht für „Extensible Markup Language“ und wurde zur Darstellung von hierarchisch strukturierten Daten entwickelt. Die Daten liegen dabei als Text vor, wobei bestimmte Regeln, wie z.B. die Existenz genau eines Wurzelelementes, eingehalten werden müssen.
XML ist also sehr geeignet um mehrere Tabellen abzubilden (hierarchische Struktur / Baumstruktur: Wurzel – Tabellen als Knoten – Zeilen als Blätter). Außerdem werden XML-Variablen von Flash gut unterstützt, sodass der Inhalt der Variablen leicht in einem Datagrid angezeigt werden kann. Auf diese Weise muss kein Array für die Startbelegung der Tabellen durchlaufen werden, was die Definition der Startbelegung stark erleichtert (lediglich die XML-Tags sind zu beachten). Viele Schleifendurchläufe über die sonst nötigen Arrays der einzelnen Tabellen brauchen also nicht gemacht werden. Man hat also mehr Geschwindigkeit und weniger Implementierungsaufwand.

Weiterhin muss beachtet werden, dass die Startbelegung in die Datenbank übertragen werden muss. Dies geschieht über eine Schnittstelle zwischen SQL-Learner und Datenbank. Die Schnittstelle muss natürlich die Startbelegung verarbeiten können. Definiert man die Startbelegung als XML-Variable, ist die Anzahl der Variablen, die an die Schnittstelle geschickt werden müssen sehr gering, da auf diese Weise ansonsten nur die Identifikationskennung des aktuellen Benutzers für das Einfügen der Startbelegung in die Datenbank benötigt wird.
Die Rückgabe der Datenbank muss sich ebenfalls verarbeiten lassen. Durch die konsequente Verwendung von XML-Variablen für die Tabelleninhalte lässt sich auch die Rückgabe sehr leicht wieder im SQL-Learner verarbeiten ohne das diverse Arrays neu angelegt und jede Zelle der Datagrids einzeln durchlaufen werden müssten.
Die oben genannte Struktur ist besonders für die Datenhaltung der Aufgaben geeignet, da sie neben der guten Wiederverwendbarkeit, der einfachen Definition eines Objektes der Klasse „Aufgabe“ auch noch den einfachen Zugriff auf die Daten einer Aufgabe ermöglicht. Sind z.B. alle Objekte der Klasse „Aufgaben“, die Aufgaben für den Einführungsmodus enthalten in einem Array „tut_mode“ abgelegt, könnte man den Hinweistext der ersten Aufgabe mit folgender Anweisung erhalten:

```
var hinweistext = tut_mode[0]['hinweis'];
```

IV.3.4 PHP Schnittstelle

Da Flash keine Möglichkeit bietet, direkt auf ein SQL-Datenbanksystem zuzugreifen, ist es unumgänglich, eine Schnittstelle zwischen Flash-Anwendung und Datenbanksystem zu installieren. Für die Programmierung der Schnittstelle eignen sich generell alle gängigen serverseitigen Programmiersprachen, die die HTTP-Anforderungen des URLRequest-Objektes von Flash akzeptieren und ihrerseits mit einem Datenbanksystem kommunizieren können. Zudem ist eine gute Unterstützung von XML nötig, da der Programmieraufwand der Schnittstelle ansonsten weitaus höher wäre. Die Wahl fiel aufgrund der weiten Verbreitung, der kostenlosen Verfügbarkeit und aufgrund persönlicher Vorkenntnisse auf PHP („PHP: Hypertext Preprocessor“). Insbesondere zeichnet sich PHP 5 durch Erweiterungen bei XML- und DOM-Handhabung aus.

IV.3.5 MySQL Datenbank

Durch die Wahl von PHP als Schnittstelle ergab sich fast schon automatisch, dass ein MySQL-Server als Datenbanksystem verwendet wird. Dies ist wiederum begründet durch die kostenlose Verfügbarkeit, die weite Verbreitung und gute Unterstützung, die PHP dafür anbietet. Eine Access-Datenbank wäre auch denkbar gewesen, allerdings mit spezielleren technischen Anforderungen (Windows-Webserver). Weiterhin ist Access nicht kostenlos verfügbar.
Der SQL-Learner kann durch die Verwendung von MySQL auch Offline mit Datenbankunterstützung betrieben werden, wenn er auf einem so genannten LAMP bzw. WAMP wie mit dem oben beschriebenen XAMPP installiert wird. Diese Softwaredistributionen vereinen Apache-Server, PHP-Unterstüztung und MySQL-Datenbanksystem in einem Paket, sodass ein echter Webserver nicht zwingend benötigt wird um alle Funktionen des SQL-Learners zu nutzen.

IV.4 Entwicklungsprozess

Der Entwicklungsprozess des SQL-Learners erfolgte in der Reihenfolge, in der die Abschnitte IV.4.1 bis IV.4.12 angeordnet sind. Dabei werden jeweils auch einzelne, später hinzugefügte Funktionen berücksichtigt, da eine Einordnung an anderer Stelle wenig Sinn machen würde.
Zunächst wurden nur einige Steuerelemente (ein Datagrid, einige Schaltflächen) auf die Bühne gezogen, um sie nach und nach mit Funktion zu füllen. Der Entwicklungsprozess begann also direkt am Hauptfenster. Dabei war das Projekt zunächst darauf ausgelegt, dass erst der „Freie Modus“ mit einer konkreten Startkonfiguration (Tabellen, verfügbare Befehle) implementiert wird und der Einführungs-, sowie der Levelmodus, die intern vom Prinzip her dem „Freien Modus“ sehr ähnlich sind, später folgen sollten. Folgende Schritte wurden in dieser Reihenfolge vollzogen: Grobes Design der Oberfläche, Eingaben des Nutzers entgegennehmen, Einfügen der Tabellen in die Datenbank mit Hilfe der PHP-Schnittstelle, Rückgabe des Datenbanksystems verarbeiten und anzeigen. Erst an dieser Stelle begann der Prozess, die bisher eher auf den „Freien Modus“ ausgelegten Funktionen noch weiter zu verallgemeinern und das Programm neu zu strukturieren

(siehe IV.4.11). Start- und Auswahlbildschirm wurden hinzugefügt und die Aufgaben für die Verschiedenen Modi gestaltet.

IV.4.1 Elemente in Befehlsliste und Eingabefeld

In diesem Abschnitt geht es um die Elemente, die die SQL-Befehle symbolisieren sollen und in der Befehlsliste bzw. im Eingabefeld erzeugt werden, sowie um deren Verhalten bei Aktionen des Benutzers. Jedes der Elemente ist eine Instanz der eigenen Klasse „MyButton“. Das Erstellen einer eigenen Klasse war notwendig, da viele gleichartige Objekte, nämlich die Elemente, die die SQL-Befehle darstellen, gebraucht werden und während der Laufzeit des SQL-Learners ständig erzeugt und wieder zerstört werden müssen. Die Klasse ist folgendermaßen aufgebaut:

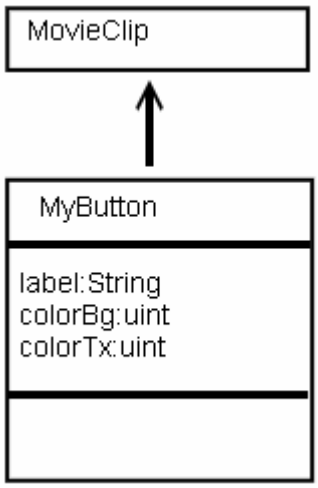

Abb. 7: Klassendiagramm, Klasse „MyButton“

Eine Instanz der Klasse „MyButton“ wird z.B. mit dem Befehl

```
addChild(new MyButton(0xFF9900,0x000000,"SELECT"));
```

erzeugt und auf der Bühne hinzugefügt. Dabei stellt der erste Parameter, der an den Konstruktor übergeben wird, die Hintergrundfarbe des Elements dar, der zweite Parameter die Textfarbe und der dritte einen String, der den SQL-Befehl selbst angibt. In diesem Fall wäre also ein Element mit der schwarzen Aufschrift „SELECT“ auf orangem Hintergrund entstanden und hinzugefügt worden. Diese Eigenschaften sind auch nach der Erstellung des Objektes über die öffentlichen Variablen „colorBg“, „colorTx“ und „label“ abrufbar.
Da die Klasse „MyButton“ eine Unterklasse von „MovieClip“ ist, werden alle Eigenschaften, die ein MovieClip hat vererbt. Das ist wichtig, da das erzeugte Element auf der Bühne angezeigt, verschoben und auch skaliert wird. Ohne

eine Vererbung wäre diese Funktionalität nicht von selbst gegeben. Der Konstruktor der Klasse „MyButton“ erstellt also genau genommen einen MovieClip, zeichnet ein Rechteck mit abgerundeten Ecken mit der im Parameter vorgegebenen Füllfarbe, passend zu der Länge des übergebenen Strings, erstellt ein Textfeld mit vorgegebenen Eigenschaften (Textgröße, Ausrichtung, Farbe, Text usw.) und fügt es anschließend dem MovieClip hinzu.

Ganz wichtig ist, dass die Eigenschaft

```
mouseChildren = false;
```

bereits im Konstruktor gesetzt wird, da es sonst zu Problemen beim Verschieben des gesamten Elements kommt, da nur, wenn diese Eigenschaft auf „false“ gesetzt wird, der gesamte Inhalt des MovieClips eine Einheit bildet. Wird diese Eigenschaft nicht gesetzt, würde bei Drag & Drop-Aktionen nur das im MovieClip erstellte Textfeld verschoben, selbst dann, wenn sich der Event Listener auf den gesamten MovieClip bezieht.

Bei der Erstellung einer Aufgabe legt der Autor alle Befehle fest, die für die entsprechende Aufgabe verfügbar sein sollen. Dabei werden die Objekte der Klasse „MyButton“ wie folgt in einem Array abgelegt:

```
var ex_A1_btn:Array = new Array();
ex_A1_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A1_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
//usw... es lassen sich beliebig viele Elemente hinzufügen
```

Dieses Array wird zum Programmstart gefüllt und beim Öffnen einer Aufgabe verwendet, um die Befehlsliste zu füllen. Für jede Aufgabe gibt es ein solches Array. Jedem Element wird beim Einfügen in die Befehlsliste ein Event Listener zugewiesen.

```
addEventListener(MouseEvent.MOUSE_DOWN, mousedownHandleraSpAll);
```

Das Ereignis, auf das sich der Event Listener bezieht, ist das Drücken der linken Maustaste, während sich der Mauszeiger über dem Element befindet. Wenn dies geschieht, wird die Funktion

```
function mousedownHandleraSpAll(event:MouseEvent):void
```

aufgerufen. Diese Funktion erstellt eine Kopie des Elements, sodass das Element selbst in der Befehlsliste verbleibt und später noch einmal verwendet werden kann.

```
var dragbutton = new MyButton(event.target.colorBg,
event.target.colorTx, event.target.label);
```

Die Kopie bewegt sich mit dem Mauszeiger, solange die linke Maustaste

gedrückt wird.

```
dragbutton.startDrag();
```

Was geschehen soll, wenn die Maustaste losgelassen oder die Kopie des Elementes bewegt wird, wird wiederum durch zwei Event Listener gesteuert, die entsprechende Funktionen dafür aufrufen.

```
dragbutton.addEventListener(MouseEvent.MOUSE_MOVE,
mousemoveHandleraSpAll);
dragbutton.addEventListener(MouseEvent.MOUSE_UP,
mouseupHandleraSpAll);
```

Die Funktion “mousemoveHandleraSpAll”, die vom Event Listener beim Bewegen der Kopie des Elementes aufgerufen wird, sorgt dafür, dass bei Bewegung über das Eingabefeld jeweils eine Einfügemarke zwischen zwei schon vorhandenen Elementen im Eingabefeld sichtbar wird, damit der Benutzer weiß, an welcher Stelle der Befehl eingefügt wird.
Die Funktion „mouseupHandleraSpAll“ wird aufgerufen, wenn die linke Maustaste bei einer Drag & Drop-Aktion eines Elementes losgelassen wird. Die Wirkung dieser Funktion ist unterschiedlich in Abhängigkeit davon, an welcher Stelle sich der Mauszeiger zu diesem Zeitpunkt befindet. Folgende Möglichkeiten werden berücksichtigt:

- Das kopierte Element wird über dem Eingabefeld fallen gelassen. Diese Aktion wird so verstanden, dass dem Eingabefeld ein neues Element hinzugefügt werden soll, nämlich ein Element mit den gleichen Eigenschaften, die das ursprünglich ausgewählte Element und auch die Kopie dieses Elementes (die mit dem Mauszeiger bewegt wurde) besitzt. Dabei wird für jede Einfügemarke im Eingabefeld geprüft, ob die Kopie des Elementes diese Marke berührt. Falls ja, ist die erste gefundene Stelle die Stelle, an der das neue Element im Eingabefeld erstellt werden muss. Wurde das kopierte Element nicht über einer Einfügemarke fallen gelassen, so wird das Element nach allen im Eingabefeld enthaltenen Elementen erstellt, also am Ende hinzugefügt.
  Anschließend wird die Kopie des Elementes zerstört und die Event Listener entfernt.
  Elemente, die dem Eingabefeld hinzugefügt worden sind, besitzen eine andere Funktion, die beim Start einer Drag & Drop-Aktion ausgeführt wird. Diese Funktion erstellt eine Kopie des Elements im Eingabefeld,

die sich mit dem Mauszeiger bewegt, löscht aber das Original im Eingabefeld. Da dies der einzige Unterschied ist, können auch bei der Drag & Drop-Aktion aus dem Eingabefeld heraus dieselben, hier aufgeführten Punkte für die Wirkung von Drag & Drop-Aktionen verwendet werden, wie für Elemente aus der Befehlsliste. Es werden ansonsten dieselben Funktionen verwendet.

- Das kopierte Element wird im Datenfeld fallen gelassen.
  Diese Aktion ist dafür gedacht, dass der Nutzer die Ansicht zwischen den einzelnen Tabellen wechseln kann. Er kann also durch Drag & Drop-Aktionen der Elemente mit den Tabellennamen bzw. des Elementes mit der Aufschrift „Ausgabe“ entscheiden, welche Tabelle im Datenfeld dargestellt werden soll.
  Falls die Aufschrift des kopierten Elementes einem Tabellennamen entspricht, wird die gewünschte Tabelle sichtbar und die vorherige Tabelle ausgeblendet. Ist die Aufschrift des kopierten Elementes von den Tabellennamen verschieden, bleibt die aktuelle Tabelle sichtbar. In beiden Fällen wird anschließend die Kopie des Elementes zerstört und die Event Listener werden entfernt.
- Das kopierte Element wird auf dem Avatar fallen gelassen.
  Eine Sprechblase wird erstellt und der Inhalt als String aus dem Array „avaexpl“, das im Layer „ava_expl“ definiert ist, abgefragt. Das Array ist so aufgebaut, dass z.B.

  ```
  avaexpl[SELECT]
  ```

  einen Sting mit der kompletten Beschreibung der SELECT-Anweisung enthält. Das kopierte Element wird auch in diesem Fall zerstört und die Event Listener entfernt.
- Das kopierte Element wird an einer anderen Stelle fallen gelassen.
  Dies hat keine weiteren Auswirkungen, außer dass die Kopie des Elementes zerstört und die Event Listener entfernt werden. Somit wird der Drag & Drop Vorgang beendet.

Neben den Elementen aus der Befehlsliste verhalten sich die Tabellennamen im Datenbereich, sowie die Spaltenüberschriften und der Inhalt jeder Zelle einer Tabelle, ebenfalls wie oben beschrieben. Bei den Tabellennamen ist dies sofort einsehbar, da sie vom Aussehen her von vornherein schon Objekte der Klasse „MyButton“ sind. Anders als die Objekte der Klasse

„MyButton“, die aus den Spaltenüberschriften und Inhalten jeder Zelle erzeugt werden. Wird die Maustaste über dem Datenfeld gedrückt gehalten, ermittelt die im zugehörigen Event Listener eingetragene Funktion zunächst, ob auf eine Spaltenüberschrift (Objekt der Klasse HeaderRenderer) oder auf eine normale Zelle der Tabelle (Objekt der Klasse CellRenderer) geklickt wurde. In beiden Fällen dann das erstellt, was bisher als Kopie eines Elementes bezeichnet wurde, wobei der Text aus der Eigenschaft „event.target.label“ ausgelesen und die Farbe des Objektes der Klasse „MyButton“ in Abhängigkeit davon ausgewählt wird, ob es sich um eine Spaltenüberschrft oder eine normale Zelle der Tabelle handelt. Da es sich bei den Zelleninhalten um Attributwerte handelt, werden noch Hochkommata vor und nach dem Inhalt der label-Eigenschaft des erzeugten Objektes der Klasse „MyButton“ angehängt.

```
function mousedownHandleraDg(event:MouseEvent):void {
[...]
if (event.target.toString() == "[object HeaderRenderer]"){
      dragbutton = new MyButton(0xFF0099, 0x000000,
event.target.label);
}else{
      dragbutton = new MyButton(0x9900FF, 0x000000, "'" +
event.target.label + "'");
}
[...]
}
```

IV.4.2 Befehlsliste und Eingabefeld / Scrollpanes

Die Befehlsliste enthält je nach Gestaltung einer Aufgabe unterschiedliche Befehle (als Objekt der Klasse „MyButton“ erstellte MovieClips), die wie im vorigen Kapitel beschrieben, aus dem zu jeder Aufgabe zugehörigen Array geladen werden.

Die Befehlsliste ist selbst ein MovieClip-Objekt, das zum Programmstart erzeugt wird.

```
var mAll:MovieClip = new MovieClip();
```

Beim Laden einer Aufgabe werden alle im Array angegebenen Elemente als MovieClips untereinander, in der Reihenfolge wie sie im Array angegeben wurden, in den MovieClip „mAll“ eingefügt. „mAll“ besitzt also keine feste Höhe, da sonst die Anzahl der Elemente begrenzt werden müsste. Jedes

Element erhält den oben beschriebenen Event Listener, der das Gedrückthalten der linken Maustaste über einem Element verarbeitet.
Der so entstandene MovieClip ist natürlich häufig zu groß, um einfach auf der Bühne hinzugefügt werden zu können. Schon bei mehr als 10 Elementen ist der für die Befehlsliste vorgegebene Platz aufgebraucht. Daher ist auf der Bühne eine Instanz von „ScrollPane“ mit dem Instanznamen „aSpAll“ für die Befehlsliste enthalten, die eine feste Größe besitzt und vertikale Scrollbars bereitstellt, wenn dies nötig ist. Mit dem folgenden Befehl wird der erstellte MovieClip eingebunden und angezeigt:

```
aSpAll.source = mAll;
```

Bei dem Eingabefeld verhält es sich ähnlich. Das Eingabefeld besteht ebenfalls zunächst aus einer MovieClip-Instanz.

```
var mCom:MovieClip = new MovieClip();
```

Es werden beim Laden einer Aufgabe selbstverständlich noch keine Elemente hinzugefügt, da das Eingabefeld dafür vorgesehen ist, dass es die Befehle, die der Nutzer per Drag & Drop von der Befehlsliste in das Eingabefeld zieht, entgegennimmt. Da die Anzahl der Elemente im Eingabefeld variabel sein muss um auch längere SQL-Befehle zu ermöglichen, bietet sich auch hier die Verwendung einer Instanz des Steuerelementes „ScrollPane“ an:

```
aSpCom.source = mCom;
```

So wie es für den Inhalt der Befehlsliste ein Array gibt, das die Objekte der Klasse „MyButton“, die ja im übertragenen Sinne die Befehls-Elemente der SQL-Sprache enthält, die für eine Aufgabe vorhanden sind, enthält, gibt es auch für das Eingabefeld ein Array für sämtliche Objekte der Klasse „MyButton“, die sich aktuell im Eingabefeld befinden.

```
var arrCom:Array = new Array();
```

Dabei ist hier ebenfalls die Reihenfolge der Elemente dieses Arrays die gleiche Reihenfolge, in der die Elemente im Eingabefeld angeordnet sind. Das Array wird durch die im vorherigen Kapitel beschriebenen Drag & Drop-Aktionen verändert. Folgende Anweisung wird ausgeführt, wenn ein Element an einer „Stelle i“ erstellt werden soll:

```
arrCom.splice(i,0,new  MyButton(event.target.colorBg,
event.target.colorTx, event.target.label));
```

Zitat:

“splice() Methode

> AS3 function splice(startIndex:int, deleteCount:uint, ... values):Array Fügt einem Array Elemente hinzu bzw. entfernt diese. Bei dieser Methode werden Änderungen am Array vorgenommen, ohne eine Kopie zu erstellen.“[4]

Der Parameter „startIndex“ gibt an, an welcher Stelle der Einfüge- bzw. Löschvorgang beginnen soll. deleteCount gibt die Anzahl der zu löschenden Elemente an. „values“ ist ein optionaler Parameter und enthält Elemente, die an der Stelle „startIndex“ eingefügt werden sollen. Die splice-Methode ist mit Action-Script 3.0 eingeführt worden.
Weiterhin wird eine neue Einfügemarke erstellt. Einfügemarken sind Instanzen des MovieClips „spacer“, die sichtbar werden, sobald der Nutzer die Kopie eines Elementes des Eingabefeldes oder der Befehlsliste über einen Zwischenraum von zwei Elementen des Eingabefeldes bewegt. Alle Einfügemarken befinden sich im Array „arrComSpacer“, das jederzeit genau so viele Instanzen des MovieClips „spacer“ enthält, wie sich Elemente der Klasse „MyButton“ im Array „arrCom“ befinden.

```
arrComSpacer.push(new spacer());
```

Eine Änderung der Arrays allein bewirkt selbstverständlich noch keine Änderung der sichtbaren Elemente auf der Bühne. Dafür gibt es die Funktion

```
function updateCom():void {…}
```

Diese Funktion löscht zunächst sämtliche MovieClips von der Bühne, die in „mCom“ enthalten sind. Dann werden nacheinander wieder sämtliche Elemente des Arrays „arrCom“ hinzugefügt, wobei vor jedem Element eine Einfügemarke platziert wird.
Die Elemente werden nebeneinander eingefügt, wobei bei jedem Element überprüft werden muss, ob das Element eventuell zu lang für die aktuelle Zeile ist und dadurch erst in der nächsten Zeile eingefügt werden kann. Auf diese Weise bleibt die Breite des MovieClips „mCom“ konstant und die Höhe ist genau wie in der Befehlsliste variabel.

[4] [Ad09b]

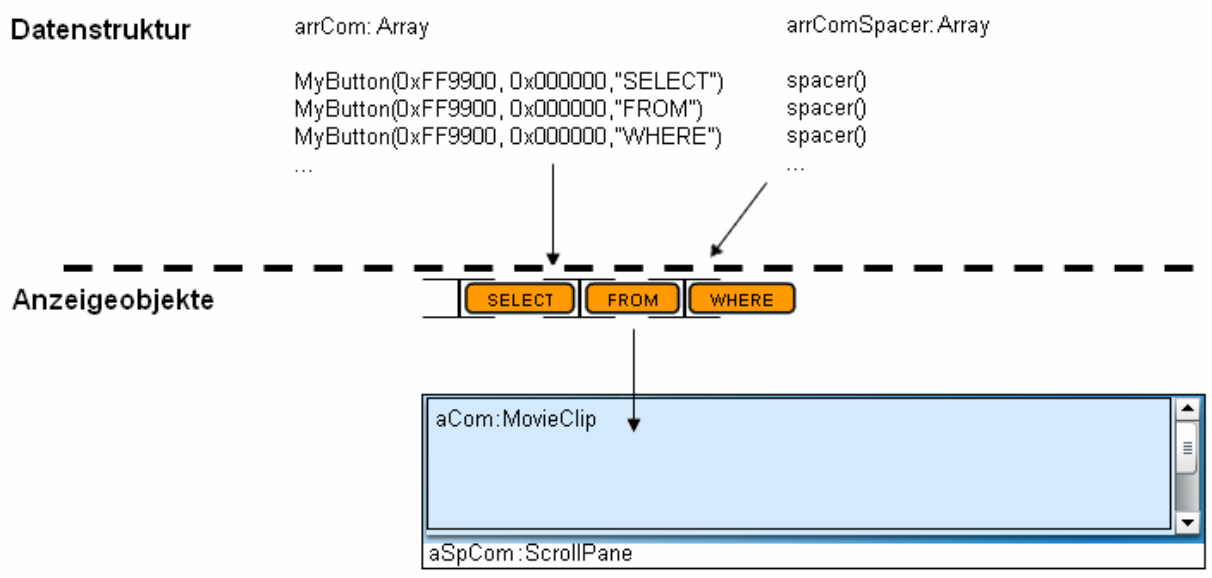

Abb. 8: Aufbau des Eingabefeldes mit interner Datenstruktur

IV.4.3 Datagrid

Für die Visualisierung der Tabellendaten / Ausgabe und zur Datenhaltung innerhalb einer Aufgabe werden Instanzen der Klasse Datagrid verwendet. Zur besseren Handhabung gibt es ein Array, das beliebig viele Datagrid-Instanzen aufnehmen kann, sowie ein Array mit den zugehörigen DataProvider-Instanzen (Die DataProvider-Klasse stellt Methoden und Eigenschaften bereit, mit denen Daten in einer listenbasierten Komponente (wie z.B. DataGrid) abrufen und geändert werden können)

```
//DataProvider Array
var arrDp:Array = new Array();
//DataGrid Array mit aktuellen DataGrids
var arrDg:Array = new Array();
```

Wenn eine Aufgabe geladen wird, werden die Instanzen der Klasse Datagrid und die zugehörigen DataProvider-Instanzen erstellt. Dazu wird die XML-Variable aus der Aufgabe benutzt, die genau angibt, wie viele der drei möglichen Tabellen genutzt werden und welchen Inhalt sie haben. Die XML-Variable gibt also sozusagen die Startbelegung der Datagrids an. Die Funktion

```
function insertData(xml):void { … }
```

wird allgemein dazu genutzt den Inhalt einer XML-Variable, in der im nächsten Kapitel beschriebenen Form, richtig in die Datagrids einzufügen. Anschließend werden die Datagrids dem MovieClip „mData“ (dem sichtbaren Datenbereich) hinzugefügt und alle Datagrids außer dem Ersten mit der

visible-Eigenschaft unsichtbar gemacht. Nebenbei werden die Schaltflächen erstellt, mit denen zwischen den Tabellen gewechselt werden kann.

IV.4.4 XML in Flash

XML-Variablen werden im SQL-Learner an mehreren Stellen verwendet.

- Zu jeder Aufgabe gehört eine XML-Variable, die die Startbelegung der maximal drei vorgefertigten Tabellen enthält. Die Ausgabe-Tabelle kann ebenfalls vorab definiert werden und ist dann direkt aufrufbar.
- Die Funktion „insertData(xml)“ erhält eine XML-Variable als Parameter. Der Inhalt der Variablen wird von der Funktion im Datenbereich visualisiert.
- Beim Senden und Empfangen in der Funktion „sendAndGetData(…)“ werden XML-Variablen verwendet.

Eine XML-Variable im SQL-Learner hat stets das gleiche Format. Das Beispiel wurde so gewählt, dass drei Tabellen erzeugt werden würden. Das Wurzelelement ist stets „<sqllearn>“. Jede Tabelle stellt ein Element dar, das mit einem „<tab>“ als Start-Tag eingeleitet wird. Alles bis zum End-Tag „</tab>“ bzw. bis zum entsprechenden „/>“ gehört zu dieser Tabelle. Informationen über Elemente lassen sich in XML als Attribute im Start-Tag unterbringen. Das „<tab>“-Tag besitzt als Attribute „name“, das den Tabellennamen enthält, sowie „h1“, …. , „hn“, wobei n die Anzahl der Spalten der Tabelle ist. Die Attribute „h1“, …. , „hn“ enthalten jeweils die Überschrift der jeweiligen Spalte. Die Zeilen einer Tabelle wiederum werden jeweils durch ein Element mit dem Start-Tag „<item>“ eingeleitet. Dabei ist die Anzahl der Attribute des „<item>“-Tags abhängig von der Anzahl der Spalten der Tabelle, da jeder Zelle einer Zeile ein Wert zugewiesen werden soll, der in dem entsprechenden Attribut steht. Die Attributnamen sind „c1“,…,“cn“, wobei n wieder die Anzahl der Spalten der Tabelle ist. Die vierte Tabelle ist die Startbelegung für die Ausgabetabelle. Die im SQL-Learner enthaltenen Aufgaben sind so gestaltet, dass die Ausgabetabelle anfangs immer leer ist. Man könnte aber bei besonders komplexen Aufgaben in Erwägung ziehen, die Ausgabe, die bei der richtigen Lösung erscheinen würde, zu Anfang sichtbar zu machen, damit der Benutzer besser versteht, was genau das Ziel der Aufgabe ist. In diesem Fall würde der Autor der Aufgabe einfach auch die

vierte Tabelle analog zu den vorherigen Tabellen in der XML-Variablen gestalten. Das Beispiel zeigt drei Tabellen und eine leere Ausgabetabelle. Die erste Tabelle hat drei Spalten und drei Zeilen. Die zweite Tabelle mit dem Namen „tabelle2“ besitzt ebenfalls drei Spalten, hat aber nur zwei Zeilen. „tabelle3“ besitzt nur zwei Spalten, hat aber drei Zeilen. Die Attributsnamen sind fest, die Attribute selbst lassen sich anpassen. Es ist darauf zu achten, dass die Bezeichnungen der Spalten keine SQL-Befehle enthalten, da das Datenbankmanagementsystem Tabellen mit solchen Spalten nicht akzeptiert. Die Tabellen sind durch hinzufügen bzw. löschen von weiteren Attributsnamen und Attributen bzw. Elementen in ihrer Größe bezüglich Spalten- und Zeilenanzahl frei veränderbar.

```
var beispiel:XML =
<sqllearn>
 <tab name="tabelle1" h1="sp11" h2="sp12" h3="sp13">
  <item c1="a11" c2="a12" c3="a13" />
  <item c1="a21" c2="a22" c3="a23" />
  <item c1="a31" c2="a32" c3="a33" />
  </tab>
 <tab name="tabelle2" h1="sp21" h2="sp22" h3="sp23">
  <item c1="a11" c2="a12" c3="a13" />
  <item c1="a21" c2="a22" c3="a23" />
</tab>
 <tab name="tabelle3" h1="sp31" h2="sp32">
  <item c1="a11" c2="a12" />
  <item c1="a21" c2="a22" />
  <item c1="a31" c2="a32" />
 </tab>
 <tab name="" />
</sqllearn>;
```

Das Einfügen einer XML-Variable geschieht wie im vorherigen Kapitel beschrieben in der Funktion:

```
function insertData(xml):void { … }
```

Dabei werden in der Funktion zunächst alle vorhandenen Datagrids aus dem MovieClip „mData“ entfernt und die Arrays „arrDg“, „arrDp“, das Array, das Objekte der Klasse „MyButton“ für die Tabellennamen enthält und ein Array das Elemente für die grafische Hervorhebung der jeweils aktiven Tabelle enthält, geleert, da ggf. von einer vorherigen Aufgabe noch Tabellen

vorhanden sein könnten.

```
      while (mData.numChildren>0) {
            mData.removeChildAt(0);
      }
      arrDg.splice(0,arrDg.length);
      arrDp.splice(0,arrDp.length);
      arrTabs.splice(0,arrTabs.length);
      arrTabsBack.splice(0,arrTabsBack.length);
```

Anschließend wird für jede der drei normalen Tabellen und die Ausgabetabelle zunächst ein DataProvider Objekt im Array „arrDp“ angelegt, das jeweils den Teil von „<tab ….“ bis „</tab>“ bzw. bis „</>“ für die entsprechende Tabelle zugewiesen bekommt. Dann wird das zugehörige Datagrid im Array „arrDg“ erstellt.
Über die Children-Eigenschaft (xml.children()[i].@*) der übergebenen XML-Variable lassen sich alle Attribute vom „<tab …“-Tag jeder Tabelle auslesen. Da die Anzahl der Attribute -1 (das erste Attribut ist der Name der Tabelle) die Anzahl der Spalten dieser Tabelle ist, können entsprechend viele Spalten im Datagrid erstellt und die jeweiligen Überschriften der Spalten aus der XML-Variablen ausgelesen und im Datagrid verwendet werden. Im Folgenden wird jedes Datagrid mit dem zugehörigen DataProvider verknüpft, das Datagrid unsichtbar gemacht und dem MovieClip „mData“ hinzugefügt.

```
//Für jede Tabelle
for (var i=0; i<4; i++) {
      arrDp.push(new DataProvider(xml.children()[i]));
      arrDg.push(new DataGrid());
      […]
      var attNamesList:XMLList = xml.children()[i].@*;
      //Für jede Spalte
      for (var j=0; j<attNamesList.length()-1; j++) {
      arrDg[i].addColumn(new DataGridColumn("c"+(j+1)));
      arrDg[i].columns[j].headerText =
xml.children()[i].attributes()[1+j];
      }
      […]
      arrDg[i].dataProvider = arrDp[i];
      arrDg[i].visible = false;
      mData.addChild(arrDg[i]);
```

Natürlich wird das „visible“-Attribut eines der Datagrids (idr. des Datagrids das die erste Tabelle anzeigen soll) anschließend auf „true“ gesetzt, damit der Benutzer nicht erst den Tabellennamen einer Tabelle im Datenbereich anklicken muss.
Eine weitere Verwendung von XML-Variablen findet sich in der Funktion „sendAndGetData(methode, ausgabe:String, userid:int, xmldata:XML)“. Diese Funktion initiiert je nach übergebenem Parameter den Startzustand für jede Aufgabe in der Datenbank bzw. sendet eine SQL-Anfrage an die PHP-Schnittstelle. Diese Entscheidung wird vom ersten Parameter („methode“) abhängig gemacht. Der zweite Parameter enthält ggf. die SQL-Anfrage selbst. Zusätzlich muss in jedem Fall eine Benutzer-ID übergeben werden, damit die Tabellen eindeutig einem Benutzer zugeordnet werden können. Über den Parameter „xmldata“ wird die Startbelegung der Tabellen an die PHP-Schnittstelle gesendet. Diese Daten kommen direkt aus den im SQL-Learner gespeicherten Aufgaben. Falls nicht die Startbelegung in der Datenbank erzeugt werden soll, reicht es aus, auf eine leere XML-Variable zu verweisen. Das Senden und Empfangen der Daten wird über ein URLLoader-, sowie ein URLRequest- und ein URLVariables Objekt realisiert. Alle an die PHP-Schnittstelle zu sendenden Variablen werden in das URLVariables-Objekt integriert. Das URLRequest-Objekt legt fest, auf welche Art die Variablen übergeben werden und bekommt die URL der PHP-Schnittstelle, sowie ein URLVariables-Objekt (mit den oben genannten Parametern) zugewiesen. Die Zugriffsart ist entweder POST oder GET, wobei POST geeigneter ist, da die Daten in einem Datenstrom an die PHP-Schnittstelle versendet werden. Bei GET wird die Schnittstelle zusammen mit den Parametern direkt aufgerufen, was dazu führt, dass die Länge einer solchen Anfrage stark begrenzt ist.
Das URLLoader-Objekt sorgt schließlich dafür, dass eine Anfrage mit den in einem URLRequest spezifizierten Einstellungen gesendet wird.

```
function sendAndGetData(methode, ausgabe:String, userid:int,
xmldata:XML):void {
[...]
      var sqlquery:URLLoader = new URLLoader();
      var url:String = schnittstelle;
```

```
      var request:URLRequest = new URLRequest(url);
      request.method = URLRequestMethod.POST;
      var variables:URLVariables = new URLVariables();
      variables.userid = "u"+userid;
      variables.query = ausgabe;
      variables.meth = methode;
      variables.startdata = aktxml;
      request.data = variables;
      try {
            sqlquery.load(request);
      } catch (error:Error) {
      [...]
      }
      if (methode == "startdata") {
            sqlquery.addEventListener(Event.COMPLETE,
completeHandlerStart);
      } else {
            sqlquery.addEventListener(Event.COMPLETE,
completeHandlerGetAll);
      }
```

In Abhängigkeit davon, ob die Startbelegung hergestellt wurde bzw. ob eine SQL-Anfrage gesendet wurde, wird ein EventListener erstellt, der ausgelöst wird, wenn die Parameter an die PHP-Schnittstelle gesendet und eine Rückgabe empfangen wurden. Dies ist nötig, da erst weitergearbeitet werden kann, wenn die Daten der Schnittstelle vorliegen. Sobald das Ergebnis da ist, wird entweder die empfange XML-Variable mit der oben beschriebenen Funktion „insertData(xml)“ eingelesen oder, falls es sich bei der Anfrage um eine Erstellung der Startbelegung einer Aufgabe gehandelt hat, stattdessen die in der Aufgabe im SQL-Learner eingetragene XML-Variable eingelesen, also im Datenbereich angezeigt.

IV.4.5 XML in PHP

PHP bietet für die Manipulation von XML verschiedene Möglichkeiten an. Aufgrund der hohen Flexibilität wird in der PHP-Schnittstelle direkt auf die Funktionen für das so genannte „Document Object Modell“ – kurz DOM – zur XML-Manipulation zurückgegriffen. Dieses Modell ist vom World Wide Web Consortium (W3C) standardisiert (dieses Gremium legt unter Anderem auch die Standards für andere, das World Wide Web betreffende Techniken wie

z.B. HTML, CSS, usw. fest). Das Document Object Modell erlaubt den dynamischen Zugriff und Modifikation von Inhalt und Struktur von Dokumenten (HTML / XML) und stellt damit eine Schnittstelle zwischen Dokument und Programmierung dar.

Um in PHP mit einem DOM-Dokument zu arbeiten, muss dieses zunächst als Objekt erzeugt und der Inhalt des eigentlichen Dokumentes geladen werden.

```
$dom = new DOMDocument('1.0', 'iso-8859-1');
$dom->loadXML($xml);
```

Die Variable „$xml“ enthält dabei ein wohlgeformtes XML-Dokument. Wohlgeformt bedeutet, dass sämtliche vom W3C festgelegte Regeln eingehalten werden, z.B. dass es genau ein Wurzelelement gibt. Diese Kriterien werden vom SQL-Learner bei der Definition der Aufgabenstellungen berücksichtigt. (Siehe dazu IV.4.4)

Auf jedes Element des DOM-Dokument kann nun zugegriffen werden. Folgendes Beispiel veranschaulicht den Zugriff auf den Namen der ersten Tabelle einer als XML-Variable übergebenen Aufgabe des SQL-Learners. Dabei wird von „$dom“, also dem dem gesamten Dokument ausgegangen auf das Wurzelelement verwiesen (documentElement), auf die Liste aller Elemente, die mit dem Tag <tab> beginnen, davon das erste Element genommen (item(0)) und wiederum davon auf das Attribut mit der Bezeichnung „name“ verwiesen.

```
$name=$dom->documentElement->getElementsByTagName('tab')-
>item(0)->getAttribute('name');
```

Wäre die Struktur der der zuvor übergebenen XML-Variable folgendermaßen

```
<sqllearn>
 <tab name="tabelle1" h1="sp11" h2="sp12" h3="sp13"> </tab>
 <tab name="tabelle2" h1="sp21" h2="sp22" h3="sp23"> </tab>
</sqllearn>;
```

Würde der Variablen „$name“ nun der String „tabelle1“ zugewiesen werden. Auf ähnliche Weise lässt sich nun jedes Element des DOM-Dokumentes referenzieren.

Falls kein XML-Dokument vorliegt, sondern neu erstellt werden soll, muss zunächst wieder ein neues DOM-Dokument als Objekt erstellt werden.

```
$dom = new DOMDocument('1.0', 'iso-8859-1');
```

Das erste Element, das erstellt wird, ist das Wurzelelement. Es wird zunächst erstellt und dann dem Dokument selbst hinzugefügt.

```
$wurzel = $dom->createElement('sqllearn');
```

```
$wurzel = $dom->appendChild($wurzel);
```

Um ein dem Wurzelelement ein unterordnetes Element zu erstellen, kann mit der gleichen Anweisung ein Knoten erzeugt werden.

```
$knoten = $dom->createElement('tab');
$knoten = $root->appendChild($knoten);
```

Beliebig viele Attribute lassen sich dann an den Knoten anhängen.

```
$knoten->setAttribute('name','tabelle1');
```

Um Text zwischen zwei Tags zu setzen wird ein Text-Knoten erstellt und hinzugefügt. Text-Knoten werden im SQL-Learner jedoch nicht verwendet, da sämtliche Informationen bereits in den Attributen der Knoten enthalten sind. (Siehe IV.4.4 zum Aufbau der XML-Variablen)

```
$wert = $doc->createTextNode('text');
$wert = $child->appendChild($wert);
```

Zum Schluss kann der Inhalt des DOM-Dokumentes z.B. mit der „print“-Anweisung als XML ausgegeben werden.

```
print $dom->saveXML();
```

Das Gesamtergebnis hätte nun die folgende Gestalt:

```
<sqllearn>
 <tab name="tabelle1">text</tab>
</sqllearn>;
```

IV.4.6 Einfügen der Daten in die Datenbank

Das Einfügen der Tabellen jeder Aufgabe in die Datenbank wird über die PHP-Schnittstelle realisiert. Die relevanten an die Schnittstelle übergebenen Daten bestehen aus der Startbelegung der Aufgabe als XML-Variable, einer Benutzer-ID und der Variablen „methode“ aus dem SQL-Learner selbst, die entweder den Wert „startdata“ oder den Wert „getall“ enthält. Falls der übergebene Wert „startdata“ ist, wird dies von der Schnittstelle so interpretiert, dass eine neue Aufgabe im SQL-Learner aufgerufen wurde und die in der Aufgabe spezifizierten Tabellen in der Datenbank erzeugt werden müssen. Damit der SQL-Learner von mehreren Benutzern mit einem einzigen angeschlossenen Datenbanksystem gleichzeitig verwendet werden kann, ist es nötig, dass die Tabellen, die vom Datenbanksystem erzeugt werden, verschiedene Namen erhalten, damit es nicht zu Namenskonflikten kommt oder auf einmal zwei Benutzer an derselben Tabelle arbeiten. Um dieses Problem zu umgehen, werden vor dem Senden an die PHP-Schnittstelle an den eigentlichen Datenbanknamen ein „u“ und die Identifikationsnummer des

aktuellen Benutzers angehängt. Daher können nach dem Senden der Informationen an die Schnittstelle zunächst alle Tabellen eines Benutzers in der Datenbank gefunden und ohne Bedenken gelöscht werden, da seitens des Benutzers ein Wechsel der Aufgabe veranlasst wurde und die Tabellen von einer möglichen vorherigen Aufgabe nicht mehr benötigt werden. Für jegliche Veränderungen der Datenbank muss zunächst mit dem folgenden Befehl die Verbindung zum MySQL-Server hergestellt werden, insofern die Verbindung noch nicht aufgebaut ist:

```
$db = @MYSQL_CONNECT($mysqlhost,$mysqluser,$mysqlpassword) or
die ("Konnte keine Verbindung zur Datenbank herstellen");
```

Die Variablen „$mysqlhost“, „$mysqluser“ und „$mysqlpassword“ werden in der Datei „global.php“ definiert. (Siehe dazu Kapitel III.2 Installation). Anschließend muss die aktive Datenbank ausgewählt werden. Die Variable „`$mysqldb`“ enthält standardmäßig die Bezeichnung „sqllearner“ für den Namen der Datenbank und wird ebenfalls in der Datei „global.php“ definiert. Sollte die eingetragene Datenbank nicht existieren, wird sie zuvor erzeugt.

```
mysql_select_db($mysqldb,$db);
```

Nun ist es möglich, Operationen auf der Datenbank auszuführen. Jeder nachfolgende Aufruf der „mysql_query()“-Funktion, die Anfragen an das Datenbanksystem sendet, wird auf der aktiven Datenbank ausgeführt.
Wie oben beschrieben werden nun sämtliche Tabellen des aktuellen Benutzers ermittelt und gelöscht. Dazu wird eine Anfrage gesendet, die als Ergebnis die Namen der Tabellen zurückgibt, in deren Namen zum Schluss die ID des aktuellen Benutzers vorkommt. Die Funktion „mysql_fetch_array()“ liefert ein Array, das einer Ergebniszeile entspricht – in diesem Fall ist das jeweils der Name einer Tabelle. Ist keine Ergebniszeile mehr vorhanden, die abgefragt werden könnte, wird „false“ zurückgegeben und die while-Schleife endet. Ansonsten wird die Tabelle mit dem gefundenen Tabellennamen mit Hilfe der „mysql_query()“-Funktion über eine Anfrage an das Datenbanksystem gelöscht.

```
$todelete = mysql_query("SHOW TABLES LIKE '%$userid'");
while ($row = mysql_fetch_array($todelete)) {
      mysql_query("DROP TABLE $row[0]");
}
```

Die Datenbank enthält zu diesem Zeitpunkt keine Tabellen des aktuellen Benutzers. Da aber der Startzustand einer neuen Aufgabe hergestellt werden

soll, müssen nun die maximal drei Tabellen der neuen Aufgabe in der Datenbank erzeugt werden. Die benötigten Informationen dazu sind in der übergebenen XML-Variablen verzeichnet und werden wie in Abschnitt IV.4.5 „XML in PHP“ beschrieben ermittelt.

```
//Name der aktuellen Tabelle ermitteln
$dbname=$dom->documentElement->
getElementsByTagName('tab')->item($i)->
getAttribute('name');
//Spalten der aktuellen Tabelle ermitteln
$spalten = $dom->documentElement->
getElementsByTagName('tab')->item($i);
//Zeilen der aktuellen Tabelle ermitteln
$zeilen = $dom->documentElement->
getElementsByTagName('tab')->item($i)->
getElementsByTagName('item');
```

Da die Anzahl der Spalten einer Tabelle seitens des SQL-Learners nicht beschränkt ist, ist die Verwendung von einer Schleife nötig um eine SQL-Anfrage zu erstellen, die die Tabelle zunächst ohne Inhalt in der Datenbank erzeugt. Es wird die Struktur der Tabelle ermittelt, indem alle Attribute außer dem „name“-Attribut, das den Namen der zu erzeugenden Tabelle enthält, durchlaufen und der Variablen „$sp“ so hinzugefügt, dass „$sp“ später als Teil der SQL-Anfrage zum Erstellen der Tabelle nutzbar ist. Das Array „$zl“ wird später verwendet, um die Zeilen der Tabelle zu erstellen.

```
$sp="";
$zl=array();
 foreach ($spalten->attributes as $attribute){
  if ($attribute->name != "name"){
      array_push($zl, $attribute->value);
      $sp = $sp."`".$attribute->value."` VARCHAR( 150 ) NOT
NULL ,";
  }
 }
[…]
mysql_query("CREATE TABLE `$mysqldb`.`$dbname$userid` (".$sp.")
ENGINE = MYISAM");
```

Ist die Tabelle erstellt, wird dann für jede Zeile der Tabelle eine SQL-Anfrage erstellt. Dabei sind in der Variablen „$n“ die Attribute (=Spaltennamen) und in

„$v“ die zugehörigen Attributwerte enthalten.

```
mysql_query("INSERT INTO `$mysqldb`.`$dbname$userid` (".$n.")
VALUES (".$v.")");
```

Diese Vorgehensweise wird für jede der maximal drei Tabellen durchgeführt. Sobald die Tabellen erstellt wurden, wird die Kommunikation mit dem MySQL-Server durch den folgenden Befehl beendet:

```
$db_close = @MYSQL_CLOSE($db);
```

Die im SQL-Learner angezeigten und die in der SQL-Datenbank abgelegten Tabellen stimmen nun überein. Diese Konsistenz ist wichtig und muss stets bestehen, damit SQL-Anfragen, die im SQL-Learner gemacht werden auch die erwarteten Ergebnisse liefern.

IV.4.7 Parsen der Datenbankdaten per PHP in XML

Wie im vorherigen Kapitel beschrieben, kann die Variable „methode“ des SQL-Learners den Wert „getall“ enthalten. Dies bewirkt den Aufruf der Funktion

```
function getAll($userid, $query){ […] }
```

der PHP-Schnittstelle. Diese Funktion

- sendet die im SQL-Learner erstellte SQL-Anfrage („$query“) eines bekannten Benutzers an den MySQL-Server.
- gibt als Ergebnis ein XML-Dokument zurück. Das XML-Dokument enthält die maximal drei zu der Aufgabe gehörenden Tabellen (die unter Umständen durch die SQL-Anfrage von den anfangs erzeugten Tabellen verschieden sind), sowie eine aus der SQL-Anfrage erstellte Ergebnistabelle.

Zuerst wird jedoch eine Hilfsfunktion aufgerufen, um die Namen der Tabellen festzustellen, die in der aktuellen Aufgabe eines Benutzers gerade aktiv sind. Die Hilfsfunktion stellt also eine Verbindung zum Datenbanksystem her und fordert mit Hilfe der PHP-Funktion „mysql_list_tables()“ eine Liste aller Tabellen an. Da die Tabellen in der Datenbank aus dem jeweiligen eigenen Namen und der Benutzer-ID zusammengesetzt sind, reicht es, aus der Liste aller Tabellen genau die Tabellennamen zurückzugeben, die auf die Benutzer-ID enden. Dabei wird nur der jeweilige eigene Name der Tabelle zurückgegeben (damit später der Benutzer nicht erfährt, dass die Tabelle, die er bearbeitet in Wirklichkeit unter anderem Namen in der Datenbank geführt wird). Falls weniger als drei Tabellen gefunden werden, wird das Rückgabe-

Array mit leeren Einträgen auf drei Elemente vergrößert.
Nachdem die Namen der Tabellen festgestellt sind, wird die SQL-Anfrage des Benutzers nach möglicherweise verwendeten Tabellennamen durchsucht. Wird der Name einer Tabelle gefunden, muss die Benutzer-ID angehängt werden, damit sich die SQL-Anfrage auf die richtige Tabelle bezieht. Nach dieser Prozedur kann die SQL-Anfrage gesendet werden. Da die Anfrage vom Benutzer erzeugt wurde ist es natürlich möglich, dass diese fehlerhaft ist. In jedem Fall wird die Rückgabe des MySQL-Servers (Fehlernummer und Fehlermeldung) in zwei Variablen gespeichert.

```
$sqlerrno = mysql_errno($db);
$sqlerror = mysql_error($db);
```

Ab diesem Punkt beginnt das Zusammenstellen des XML-Dokuments, das genau so aufgebaut sein muss, wie die im SQL-Learner definierten XML-Variablen für die Startbelegung der Aufgaben. Dazu wird zunächst mit der oben beschriebenen Hilfsfunktion die Liste der vorhandenen Tabellen aktualisiert, da durch die SQL-Anfrage theoretisch über die „DROP TABLE“-Anweisung Tabellen verloren gegangen sein könnten. Wie in IV.4.5 beschrieben wird nun ein neues DOM-Dokument zur Generierung der XML-Ausgabe erstellt. Eine Schleife über die maximal drei Tabellen sorgt für die Vervollständigung des DOM-Dokumentes. Für jede in der Datenbank gefundene Tabelle werden die Namen der Spalten und später der Inhalt der Zeilen aus der Datenbank abgefragt und dem DOM-Dokument hinzugefügt. Nicht vorhandene oder leere Tabellen werden dabei als Sonderfälle betrachtet, damit das DOM-Dokument wohlgeformt bleibt, falls keine inneren Knoten hinzugefügt werden können.
Das Erstellen der Ausgabetabelle ist um einiges komplexer, da es viele Arten von SQL-Anfragen mit unterschiedlichen Rückgaben bzw. Fehlern gibt. Es ist z.B. das Ergebnis einer Anfrage, die mit SELECT beginnt, für gewöhnlich eine Menge von Tabellendaten, während das Durchführen einer „DROP TABLE“-Anweisung zur Folge hat, dass eine Tabelle gelöscht wird und der Benutzer eine Meldung dazu erhalten sollte.
Als erstes wird überprüft, ob die SQL-Anfrage erfolgreich ausgeführt worden ist. Dies ist bereits der Fall, wenn nach dem Senden der Anfrage

```
$myquery = mysql_query($query);
```

der Wert der Variablen “$myquery” = “true” ist, da sie dann Daten enthält. Danach wird die SQL-Anfrage selbst noch einmal verwendet um zu

bestimmen, ob es sich um eine SELECT- oder SHOW-Anfrage gehandelt hat. War dies der Fall, wird mit Hilfe der PHP Funktion „mysql_fetch_assoc()“ der Inhalt der Rückgabe des Datenbanksystems entsprechend für das DOM-Dokument verarbeitet. Als Sonderfall gibt es die Möglichkeit, dass MySQL ein leeres Resultat zurückgibt. In diesem Fall wird eine entsprechende Tabelle erstellt, sodass der Benutzer darüber informiert wird. Von SELECT- bzw. SHOW-Anfragen unterschiedliche SQL-Anfragen informieren den Benutzer lediglich darüber, dass seine Anfrage erfolgreich ausgeführt wurde. Insbesondere bei DELETE- oder INSERT-Anweisungen ist das Ergebnis dann an den normalen Tabellen im SQL-Learner ersichtlich. Sollte die SQL-Anweisung nicht erfolgreich ausgeführt worden sein, werden die Fehlernummer und die vom MySQL-Server zurückgegebene Fehlermeldung entsprechend in das DOM-Dokument eingetragen.
Abschließend wird die Verbindung zum MySQL-Server getrennt und aus dem DOM-Dokument die XML-Rückgabe erzeugt.

IV.4.8 Fehlerbehandlung beim Senden und Laden

Das Senden und Laden von Daten über die PHP-Schnittstelle stellt viele technische Anforderungen. Auf der einen Seite steht der SQL-Learner selbst, auf der anderen das Datenbanksystem. Fällt nun z.B. das Datenbanksystem aus, muss der Benutzer informiert werden das etwas nicht korrekt funktioniert. Die Fehlerbehandlung befindet sich in der Funktion „sendAndGetData()“ des SQL-Learners (siehe dazu IV.4.4). Sollten Fehler auftreten, werden die zugehörigen Fehlermeldungen mit Hilfe des Avatars als Sprechblase visualisiert.
Der erste Fehler, der auftreten könnte wäre, dass der SQL-Learner erst gar nicht dazu kommt, Daten an die PHP-Schnittstelle zu senden, weil z.B. die Rechte auf dem lokalen Rechner eingeschränkt wurden, sodass SWF-Dateien keine Daten versenden können. Der Fehler wird über einen try-catch-Block festgestellt, der den Versuch unternimmt, die „load“-Funktion des URLLoader-Objektes aufzurufen. Schlägt dies fehl, erhält der Benutzer die Meldung

```
Programm-Fehler
Der SQL-Learner konnte keine Daten versenden.
Bitte die Anwendung neu starten und die Rechte von
```

```
SWF-Dateien überprüfen.
```

Weiterhin wird mit einem EventListener überwacht ob ein IOErrorEvent-Objekt vom Flash-Player gesendet wird. Dies ist der Fall, wenn ein Sende- oder Ladevorgang fehlschlägt und hier gleichbedeutend damit, dass die Verbindung zur PHP-Schnittstelle nicht aufgebaut werden konnte.

```
sqlquery.addEventListener(IOErrorEvent.IO_ERROR,
ioErrorHandler);
[…]
function ioErrorHandler(event:Event):void {
      getBubble("<b>Programm-Fehler</b><br><br>Die Verbindung
zur Datei "+schnittstelle+", die zur Ausführung dieses
Lernprogramms nötig ist, konnte nicht hergestellt werden.");
}
```

Programm-Fehler

Die Verbindung zur Datei http://localhost/sql/sqllearner.php, die zur Ausführung dieses Lernprogramms nötig ist, konnte nicht hergestellt werden.

Die beiden bisher vorgestellten Fehler können bereits beim Aufrufen einer Aufgabe auftreten, da die Kommunikation mit der PHP-Schnittstelle initiiert wird, um den Startzustand in der Datenbank herzustellen. Dabei wird zunächst keine Erfolgsmeldung an den SQL-Learner zurückgegeben, ob das Datenbanksystem die Tabellen der aktuellen Aufgabe des SQL-Learners akzeptiert hat, bzw. ob das Datenbanksystem überhaupt erreichbar war. Dies wird erst überprüft, sobald der Benutzer das Datenbanksystem aktiv verwendet, indem er eine Anfrage sendet. Liefert die Anfrage kein XML-Dokument als Ergebnis, ist die Kommunikation mit dem Datenbanksystem nicht möglich und eine Fehlermeldung wird ausgegeben.

```
if (xml.hasComplexContent() == true) {
      //alles ok […]
}else{
      getBubble("<b>Programm-Fehler</b>[…]");
}
```

```
Programm-Fehler
Die Verbindung zwischen der Schnittstelle
(http://localhost/sql/sqllearner.php) und der MySQL-
Datenbank konnte nicht aufgebaut werden.
```

Die URL der Schnittstelle wird aus der Variablen „schnittstelle“ des SQL-Learners ermittelt. Standardmäßig wird davon ausgegangen, dass sich die Schnittstelle im selben Verzeichnis wie die SWF-Datei befindet.

IV.4.9 Sprechblasen

Die zentrale Funktion die für die Sprechblasen zuständig ist, ist die Funktion „getBubble()“. Als Parameter wird ein String erwartet. Die übergebene Zeichenkette wird in der erstellten Sprechblase angezeigt. HTML-Tags werden unterstützt. Der anzuzeigende Text kann also direkt formatiert übergeben werden. Bevor die Funktion getBubble()“ aufgerufen werden kann, muss erst einmalig die Sprechblase initialisiert werden. Es wird dazu ein Objekt des MovieClips „bubble“ erstellt, der Bühne hinzugefügt und zunächst unsichtbar geschaltet.

```
var bbl = new bubble();
addChild(bbl);
bbl.visible = false;
```

Der MovieClip selbst enthält ein Textfeld (als Instanz der Klasse „TextArea“, benannt als „bubbletext“) und einen weiteren MovieClip der das X zum Schließen der Sprechblase zeigt (als Instanz der Klasse bubble_x, benannt als „btn_close“).
Das erstellte Objekt des MovieClips „bubble“ bleibt ständig bestehen und wird auf der Bühne bei Bedarf ein- bzw. wieder ausgeblendet.
Die bereits angesprochene Funktion „getBubble()“ sorgt nun dafür, dass der Text aktualisiert, die Scrollleiste wieder zurückgesetzt, die Sprechblase angezeigt wird und entsprechende EventListener bereit gestellt werden um die Sprechblase wieder auszublenden.

IV.4.10 Aufgabenklasse

Wie bereits in Kapitel IV.3.3 angerissen, wurde eine Klasse definiert, die zur Speicherung der einzelnen Aufgaben dient. Nachdem die Kommunikation für eine bestimmte Aufgabe funktionsfähig war, galt es, eine Vorgehensweise für

beliebig viele Aufgaben zu finden und dabei sicherzustellen, dass die Aufgaben einfach angepasst und strukturiert im Programm abgelegt werden. Die sehr simple Klasse „Aufgabe“ bietet allen nötigen Variablen für jeweils eine Aufgabe Platz. Mit Hilfe des Konstruktors und den übergebenen Variablen kann ein Objekt dieser Klasse angelegt werden.

```
package klassen{

public class Aufgabe extends Object{

       public var xml:XML;
       public var buttons:Array;
       public var aufgabe:String;
       public var hinweis:String;
       public var loesungSQL:String;
       public var loesungstext:String;

       public function Aufgabe( xml_p:XML, buttons_p:Array,
aufgabe_p:String, hinweis_p:String, loesungSQL_p:String,
loesungstext_p:String):void
   {
      xml = xml_p;
      buttons = buttons_p;
      aufgabe = aufgabe_p;
      hinweis = hinweis_p;
      loesungSQL = loesungSQL_p;
      loesungstext = loesungstext_p;
   }
}
}
```

Für jeden Modus existiert ein Array, das aus einer Menge von Objekten der Klasse „Aufgabe“ zusammengesetzt ist. Jede Aufgabe wiederum enthält alle zu ihr gehörenden Informationen.

- „xml:XML“
  Startbelegung der Tabellen einer Aufgabe. Siehe dazu IV.4.4.
- „buttons:Array“
  Enthält alle Schaltflächen als Objekte der Klasse „MyButton“, die in der Befehlsliste erscheinen sollen.
- „aufgabe:String“

Dieser Text erscheint dem Benutzer, wenn er die Aufgabe aufruft.

- „hinweis:String“
  Der Hinweis-Text erscheint, wenn der Benutzer eine SQL-Anfrage gesendet hat, die nicht der richtigen Lösung entspricht.
- „loesungSQL:String“
  Enthält die erwartete Lösung der jeweiligen Aufgabe, also die SQL-Anfrage selbst.
- „loesungstext:String“
  Dieser Text wird angezeigt, wenn der Benutzer eine Aufgabe richtig löst.

IV.4.11 Strukturierung

Um bessere Anpassbarkeit und Übersichtlichkeit sicherzustellen wurden die Elemente des SQL-Learners auf verschiedene Ebenen (Layers) innerhalb von Flash CS 3 verteilt. Dabei enthalten sämtliche Ebenen Quelltext außer die Ebene mit der Bezeichnung „static“. Folgende Ebenen mit ihrer jeweiligen Funktion wurden angelegt:

- „import“
  Diese Ebene importiert alle Klassen, die für die Ausführung des SQL-Learners nötig sind. Insbesondere sind dies die zwei selbst erstellten Klassen (zur Anzeige der Schaltflächen und für die Datenhaltung der Aufgaben), sowie die Klassen für die Verwendung von Datagrids.
- „functions“
  In der „functions“ Ebene werden zunächst alle globalen Variablen definiert. Des Weiteren finden sich in dieser Ebene sämtliche Funktionen des SQL-Learners mit Ausnahme der Funktionen, die für die Navigation im Intro und im Auswahlbildschirm zuständig sind.
- „timeline“
  Diese Ebene definiert die Abfolge von Intro und Auswahlbildschirm und enthält die dazu nötigen Funktionen. Die „timeline“-Ebene stellt den Einstieg ins Programm dar. Zunächst wird der Intro-MovieClip hinzugefügt und entsprechend mit einem Event-Listener ausgestattet, damit der Benutzer durch Anklicken des Intros in den Auswahlbildschirm gelangt. Dem Auswahlbildschirm werden daraufhin Event-Listener hinzugefügt, die dafür sorgen, dass beim Anklicken der

entsprechenden Schaltfläche der jeweilige Modus aufgerufen wird. Die Verbindung zu den Funktionen der „functions“-Ebene besteht im Aufruf der zu der „functions“-Ebene gehörenden Funktion „changeAufgabe()“.

- „free_mode“
  In dieser Ebene werden Variablen definiert, die für den „Freien Modus“ genutzt werden. Das ist die XML-Variable für die Startbelegung der Tabellen, das Hinzufügen und Erstellen von Objekten der Klasse „MyButton“ für alle benötigten Schaltflächen in einem Array und die Variablen für Textmeldungen innerhalb des „Freien Modus“. Der „Freie Modus“ ist genau so aufgebaut wie die beiden anderen Modi mit dem Unterschied, dass der „Freie Modus“ nur eine „Aufgabe“ enthält.
- „ex_mode“
  Diese Ebene enthält die Aufgaben für den Levelmodus.
- „tut_mode“
  Diese Ebene enthält die Aufgaben für den Einführungsmodus.
- „ava_expl“
  Auf der Ebene „ava_expl“ werden die Erklärungen, die der Avatar bei Drag&Drop-Aktionen von Befehlselementen auf den Avatar selbst gibt, definiert.
- „static“
  Die „static“ Ebene dient dazu, die festen Elemente auf der Bühne einzubinden. Alle MovieClips und Formen, die nicht dynamisch erstellt werden, sind auf dieser Ebene positioniert.

IV.4.12 Umschaltmechanismus zwischen den Modi

Was geschieht, wenn der Benutzer in den Auswahlbildschirm zurückgeht und einen anderen Modus auswählt? In den globalen Variablen, die in der Ebene „functions“ definiert sind, existieren unter anderem folgende Variablen, die den aktuellen Modus und die Nummer der Aufgabe enthalten, die der Benutzer momentan geöffnet hat.

```
var modus:String;
var aktaufgabennummer:int;
```

Wird die Aufgabe oder der Modus geändert, erkennt das die Funktion „changeAufgabe()“. Diese Funktion sorgt dafür, dass in die globalen Variablen, die für jede Eigenschaft einer Aufgabe existieren mit den Werten

gefüllt werden, die im Aufgabendesign festgelegt wurden.

```
aktxml = tut_mode[aktaufgabennummer]['xml'];
hinweis = tut_mode[aktaufgabennummer]['hinweis'];
[…]
```

Artefakte von der vorherigen Aufgabe werden entfernt und anschließend die Inhalte der neuen Aufgabe hinzugefügt, sowie der Startzustand in der Datenbank hergestellt.

IV.5 Mögliche Fehler

In der heutigen Zeit gibt es wenige Programmautoren, die von Ihrer Software behaupten können, dass sie komplett fehlerfrei ist. Dies ist bedingt durch die enorme Komplexität, die Unmöglichkeit absolut jede Situation abzusichern und die nahezu unendlichen Möglichkeiten, die eine Programmiersprache wie Action Script 3.0 dem Programmautor anbietet. Dazu kommen im Fall des SQL-Learners die PHP-Schnittstelle und das Verwenden eines SQL-Servers dazu, welche zusätzliche Komplexität und mögliche Fehlerquellen bedeuteten.

IV.5.1 Fehler, die durch das Aufgabendesign entstehen können

Manchmal ist es möglich, durch verschiedene SQL-Anfragen ein identisches Ergebnis zu erhalten. Der SQL-Learner akzeptiert für jede Aufgabe jedoch immer nur genau eine mögliche Lösung. Daher ist beim Design der Aufgaben zu beachten, dass sich diese möglichst nur auf eine Weise (die auch die naheliegendste sein sollte) lösen lassen. Sollte es in einer Aufgabe vorkommen, dass zwei Tabellen mit der „NATURAL JOIN“-Anweisung verbunden werden, wird die vom Nutzer eingegebene Lösung auch als richtig erkannt, wenn die Tabellennamen vor und nach der „NATURAL JOIN“-Anweisung vertauscht sind.

Weiterhin ist darauf zu achten, dass die XML-Variable zur Definition der Aufgabe wohlgeformt und der für westeuropäische Sprachen üblichen Kodierung ISO-8859-1 entsprechen muss. Damit lassen sich die deutschen Umlaute und gewisse relevante Sonderzeichen ohne Umschreibung notieren.

Ein weiterer Punkt sind die in den Aufgaben angebotenen SQL-Befehle. Sie wurden in verschiedenen Situationen getestet. Die Rückgabe vom Datenbanksystem sollte also in der Regel von der PHP-Schnittstelle richtig interpretiert werden. Jedoch ist es aufgrund der Menge an SQL-Befehlen und

Kombination dieser nicht auszuschließen, dass es SQL-Anfragen gibt, die zu einer fehlerhaften Interpretation seitens der PHP-Schnittstelle führen und ein ungültiges XML-Dokument zur Folge haben.
Bei der Festlegung der möglichen SQL-Befehle, die in der Befehlsliste erscheinen sollen, muss bedacht werden, dass es Befehle gibt, die die Struktur der Datenbank verändern können. Würden z.B. die Befehle „DROP“, „DATABASE“ und ein Tabellenname „sqllearner“ angeboten werden, könnte daraus die Anfrage „DROP DATABASE sqllearner“ erstellt werden, die zur Folge hätte, dass alle aktuellen Nutzer für die momentan bearbeitete Aufgabe keine Ausgabe des Datenbanksystems erhalten würden.
Innerhalb der Aufgaben muss darauf geachtet werden, dass keine SQL-Befehle in den Spaltenüberschriften vorkommen, da diese vom MySQL-Server nicht korrekt interpretiert werden. Die Spaltenüberschrift „Alter“ wäre z.B. ungültig, da „ALTER“ als SQL-Befehl benutzt wird um eine Tabellenstruktur zu verändern. Ebenso sind Umlaute in Spaltennamen ungültig.

IV.5.2 Generelle Fehler

Folgende Fehler sind bekannt:

- Die Tabellen der letzten Aufgabe, die ein Benutzer bearbeitet hat, verbleiben nach dem Beenden des SQL-Learners in der Datenbank. Dies beeinträchtigt die Funktion des SQL-Learners in keiner Weise, jedoch sollte die Datenbank ab und zu geleert werden, wenn viele Nutzer den SQL-Learner mit einem einzigen angeschlossenen SQL-Server verwenden.
- Wenn der Text, der in einer Zelle einer Tabelle angezeigt werden soll sehr lang ist, kann selbst durch verschieben der Spalten der Text nicht komplett sichtbar gemacht werden. Dies kann unter Umständen zur Folge haben, dass Fehlermeldungen seitens des SQL-Servers abgeschnitten werden und der Benutzer mit Hilfe der angezeigten Fehlernummer nach dem genauen Fehler suchen muss.

V Lizenz

SQL-Learner wird unter der GNU General Public License veröffentlicht.

```
SQL-Learner - Lernprogramm zur Visualisierung von SQL-Anfragen
----------
Diese Software wurde im Sommersemester 2009 an der Universität
Münster, Institut für Didaktik der Mathematik und der
Informatik entwickelt.
Entwickler: Michael Saul
Betreuer: Prof. Dr. Marco Thomas, Michael Weigend
SQL-Learner - educational environment for visualizing SQL
language
----------
This software has been developed at the University of Münster
(Germany), department of didactics of mathematics and computer
science.
Developer: Michael Saul
Supervisors: Prof. Dr. Marco Thomas, Michael Weigend
----------
Copyright (c) 2009 Michael Saul
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public
License along with this program; if not, see
http://www.gnu.org/licenses/
```

VI Quellenverzeichnis

[Ad09a] Adobe: Flash Player 10: System requirements, 2009. Online verfügbar: http://www.adobe.com/products/flashplayer/systemreqs/index.html (20.07.09)

[Ad09b] Adobe: Flash CS3 Professional Help Resource Center, 2009. Online verfügbar: http://livedocs.adobe.com/flash/9.0_de/UsingFlash/ (20.07.09)

[Ap09] Apache Friends – XAMPP, 2009. Online verfügbar: http://www.apachefriends.org/de/xampp.html (20.07.09)

[Kl07] Klett: Informatik 2 - Tabellenkalkulationssysteme, Datenbanken. Klett Verlag, Stuttgart 2007.

[Ma04] Marston, T.: Using PHP 5's DOM functions to create XML files from SQL data, 2004. Online verfügbar: http://www.tonymarston.net/php-mysql/dom.html (20.07.09)

[MS99] Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen: Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen Informatik. Ritterbach Verlag GmbH, Frechen 1999.

[PP09] PHP.net: Document Object Model, 2009. Online verfügbar: http://de.php.net/manual/en/book.dom.php (20.07.09)

VIII Anhang: Vollständiger Quelltext des SQL-Learners

```
//benoetige Klassen importieren
import klassen.MyButton;
import klassen.Aufgabe;
import fl.controls.DataGrid;
import fl.data.DataProvider;
import fl.events.ListEvent;
import fl.events.DataGridEvent;
import fl.controls.dataGridClasses.DataGridColumn;

////////////////////////////////////////////////
// Folgende Werte koennen angepasst werden:
////////////////////////////////////////////////
// gibt die Adresse zur PHP-Datei an, die als Schnittstelle zum Datenbanksystem
fungiert
var pfad = stage.loaderInfo.url.substr(0,stage.loaderInfo.url.lastIndexOf("/"));
var schnittstelle=pfad + "/sqllearner.php";
// die Möglichkeit geben, dass der Einfuehrungsmodus und der Levelmodus auch ohne
// externe Datenbank funktioniert. Allerdings gibt es dann keine Ausgabe, sondern nur
// die Prüfung des Befehls.
var online=1;
//Fortschritt festhalten, jeweils ist die am weiten zugängliche Lektion eingetragen
//durch setzen dieser Variablen auf einen entsprechenden Wert ist es moeglich, dass
bereits zu
//Begin alle oder ein Teil der Aufgaben zugaenglich ist.
var ex_modeFortschritt:int=0;
var tut_modeFortschritt:int=0;
// zufällige User-Id generieren, damit das Programm bei mehreren Benutzern mit einer
DB arbeiten kann.
// sollte eine Einbettung in das Museum erfolgen, waere es sinnvoll hier die User-ID
des Museumsbenutzers
// zu hinterlegen.
var userid=Math.round(Math.random()*10000);

////////////////////////////////////////////////
// ab hier sollte nichts mehr geaendert werden
////////////////////////////////////////////////
//Aktuelles Startschema für die Tabellen
var aktxml:XML;
//DataProvider Array
var arrDp:Array = new Array();
//DataGrid Array mit aktuellen DataGrids
var arrDg:Array = new Array();
//Array fuer die zugehoerigen Buttons um die Ansicht zu wechseln
var arrTabs:Array = new Array();
//Array fuer Hintergrund der Tabs
var arrTabsBack:Array = new Array();
//Array das alle zu einem Moment verfuegbaren Buttons fuer Befehle enthält
var arrAll:Array = new Array();
//Array fuer alle Objekte im Eingabefeld
var arrCom:Array = new Array();
//Array fuer alle Zwischenraeume im Eingabefeld, damit neue Befehle
//an eine bestimmte Stelle gelegt oder bestehende umsortiert werden koennen
var arrComSpacer:Array = new Array();
//String der die aktuelle Aufgabe enthaelt
var aufgabe:String;
//String der den aktuellen Hinweis enthaelt
var hinweis:String;
//String der die aktuelle loesung als SQL-Query enthaelt
var loesungSQL:String;
//Ein Kommentar der erscheint wenn die richtige loesung eingegeben wurde bei der
aktuellen Aufgabe
var loesungstext:String;
//MovieClips erstellen die als Inhalt fuer die ScrolPanes dienen
//bzw. fuer die DataGrids
var mAll:MovieClip = new MovieClip();
var mCom:MovieClip = new MovieClip();
var mData:MovieClip = new MovieClip();
//Sprechblase initialisieren
var bbl = new bubble();
addChild(bbl);
bbl.visible = false;
//Variaben um anzuzeigen welcher modus und welche Aufgabe gerade aktiv ist
var modus:String;
var aktaufgabennummer:int;
//Wird gesetzt beim Wechseln einer Aufgabe, gibt die Gesamtanzahl der Aufgaben im
aktuellen Modus an
var aufgabenInModus:int;
```

```
////////////////////////////////////////////////
//statische Buttons
////////////////////////////////////////////////
//EventListener hinzufuegen fuer alle Buttons
btn_go.addEventListener(MouseEvent.CLICK, btngo);
btn_del.addEventListener(MouseEvent.CLICK, btndel);
btn_next.addEventListener(MouseEvent.CLICK, btnnext);
btn_next.addEventListener(MouseEvent.MOUSE_OVER, btnnextMouseOver);
btn_next.addEventListener(MouseEvent.MOUSE_OUT, btnnextMouseOut);
btn_prev.addEventListener(MouseEvent.CLICK, btnprev);
btn_prev.addEventListener(MouseEvent.MOUSE_OVER, btnprevMouseOver);
btn_prev.addEventListener(MouseEvent.MOUSE_OUT, btnprevMouseOut);
ava.addEventListener(MouseEvent.CLICK, avatarClickHandler);

//Gibt den Fortschritt des Nutzers in Prozent zurück.
//100% ist erreicht, wenn alle Aufgaben erfolgreich bearbeitet wurden.
function museumErgebnis():int {
	return Math.floor((ex_modeFortschritt + tut_modeFortschritt) /
(tut_mode.length + ex_mode.length))*100
}

// Funktion für den Auswerten-Button
function btngo(event:MouseEvent):void {
	var ausgabe:String = "";
	var swap:String = "";
	var leeresXML:XML;
	var loesungrichtig:Boolean = false;
	var arrAusg = new Array;
	var arrTest = new Array;
	//Ausgabe-String erstellen
	for (var i=0; i<arrCom.length; i++) {
		ausgabe+=arrCom[i].label + ((i+1==arrCom.length) ? "" : " ");
	}
	trace(ausgabe);
	//Bestimmen ob die gesendete Lösung mit der im Programm hinterlegten
übereinstimmt
	if (ausgabe == loesungSQL) {
		loesungrichtig = true;
		//Falls die Lösung nicht der hinterlegten Lösung entspricht
		//soll überprüft werden, ob die Lösung trotzdem korrekt ist.
		//das ist der Fall, wenn die Tabellennamen beim Join vertauscht wurden
	} else {
		//Arrays der Lösungen erstellen, " " ist das Trennzeichen
		arrAusg = loesungSQL.split(" ");
		arrTest = ausgabe.split(" ");
		//Wenn die Anzahl der Befehlsbausteine übereinstimmt...
		if (arrAusg.length == arrTest.length) {
			//Alle Befehle durchlaufen
			for (var j=0; j<arrAusg.length; j++) {
				//Wenn 'NATURAL' gefunden wird, tausche das Wort davor
mit dem 2. danach
				if ((arrAusg[j] == 'NATURAL') && (j<(arrAusg.length-2)))
{
					swap = arrTest[j-1];
					arrTest[j-1] = arrTest[j+2];
					arrTest[j+2] = swap;
				}
			}
			//Lösung ist richtig, falls sich die Elemente der Arrays nach
der Vertauschung nicht mehr unterscheiden
			loesungrichtig = true;
			for (var k=0; k<arrAusg.length; k++) {
				if (arrAusg[k] != arrTest[k]) {
					loesungrichtig = false;
				}
			}
		}
	}
	//Query senden und saemtliche tabellen aktualisieren
	sendAndGetData("getall",ausgabe,userid,leeresXML);
	if (modus != "free_mode") {
		if (loesungrichtig == true) {
			getBubble("Deine Lösung ist RICHTIG!<br>"+loesungstext);
			thumb.gotoAndStop(2);
			if ((modus=="tut_mode") &&
(tut_modeFortschritt<=aktaufgabennummer)) {
				tut_modeFortschritt=(aktaufgabennummer+1);
			}
```

```
                        if ((modus=="ex_mode") &&
(ex_modeFortschritt<=aktaufgabennummer)) {
                                ex_modeFortschritt=(aktaufgabennummer+1);
                        }
                        if (aktaufgabennummer<(aufgabenInModus-1)) {
                                btn_next.visible = true;
                        }
                } else {
                        getBubble("Deine Lösung ist leider Falsch.<br>"+hinweis);
                }
        }

}

//Loescht alle Elemente im Eingabefeld
function btndel(event:MouseEvent):void {
        //array mit Elementen entfernen
        arrCom.splice(0,arrCom.length);
        updateCom();
}

//Funktion fuer den Button um eine Aufgabe weiterzuspringen
function btnnext(event:MouseEvent):void {
        changeAufgabe(-1);//wechsel zur naechsten Aufgabe
}
function btnnextMouseOver(event:MouseEvent):void {
        event.target.gotoAndStop(2);
}
function btnnextMouseOut(event:MouseEvent):void {
        event.target.gotoAndStop(1);
}

//Funktion fuer den Button um eine Aufgabe zurueckzuspringen
function btnprev(event:MouseEvent):void {
        changeAufgabe(-2);//wechsel zur vorherigen Aufgabe
}
function btnprevMouseOver(event:MouseEvent):void {
        event.target.gotoAndStop(2);
}
function btnprevMouseOut(event:MouseEvent):void {
        event.target.gotoAndStop(1);
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////


////////////////////////////////////////////////
//Sprechblase
//t:String:wenn t="standard" wird der normale Aufgabentext verwendet, ansonsten
//wird t ausgegeben. html-code erlaubt.
////////////////////////////////////////////////
function getBubble(t:String):void {
        bbl.x = 105;
        bbl.y = 105;
        if (t=="standard") {
                bbl.bubbletext.htmlText = "<font size='12'>" + aufgabe + "</font>";
        } else {
                bbl.bubbletext.htmlText = "<font size='12'>" + t + "</font>";
        }
        //Bubble neu einfuegen, damit sie ueber den anderen Elementen angeordnet wird
        removeChild(bbl);
        addChild(bbl);
        bbl.visible = true;
        bbl.bubbletext.verticalScrollPosition = 0;
        bbl.btn_close.addEventListener(MouseEvent.CLICK,closeBubbleHandler);
        ava.removeEventListener(MouseEvent.CLICK, avatarClickHandler);
        ava.addEventListener(MouseEvent.CLICK, closeBubbleHandler);

        function closeBubbleHandler(event:MouseEvent):void {
                bbl.visible = false;
                bbl.btn_close.removeEventListener(MouseEvent.CLICK,closeBubbleHandler);
                ava.removeEventListener(MouseEvent.CLICK, closeBubbleHandler);
                ava.addEventListener(MouseEvent.CLICK, avatarClickHandler);
        }
}

////////////////////////////////////////////////
//
////////////////////////////////////////////////
```

```
////////////////////////////////////////////////
//Aufgabe wechseln. Erwarteter Paramenter: Nummer der Aufgabe im aktuellen Modus
//oder -1 fuer naechste Aufgabe, -2 fuer vorherige Aufgabe
//benutzt die globalen Variablen: "modus" und "aktaufgabennummer"
////////////////////////////////////////////////
function changeAufgabe(a:int) {
	switch (a) {
		case -2 :
			aktaufgabennummer -=1;
			break;
		case -1 :
			aktaufgabennummer +=1;
			break;
		default :
			aktaufgabennummer=a;
			break;
	}
	//abhaenig vom Modus
	switch (modus) {
		case "tut_mode" :
			aufgabenInModus = tut_mode.length;
			if ((aktaufgabennummer==(aufgabenInModus-1)) ||
(tut_modeFortschritt<=aktaufgabennummer)) {
				btn_next.visible=false;
			} else {
				btn_next.visible=true;
			}
			if (aktaufgabennummer==0) {
				btn_prev.visible=false;
			} else {
				btn_prev.visible=true;
			}
			titletext.htmlText="<b>Einführung - Teil " +
(aktaufgabennummer+1)+" (von "+ aufgabenInModus +")</b>";
			aktxml = tut_mode[aktaufgabennummer]['xml'];
			arrAll = tut_mode[aktaufgabennummer]['buttons'];
			aufgabe = tut_mode[aktaufgabennummer]['aufgabe'];
			hinweis = tut_mode[aktaufgabennummer]['hinweis'];
			loesungSQL = tut_mode[aktaufgabennummer]['loesungSQL'];
			loesungstext = tut_mode[aktaufgabennummer]['loesungstext'];
			initUI();
			sendAndGetData("startdata","",userid,aktxml);
			getBubble("standard");
			break;
		case "ex_mode" :
			aufgabenInModus = ex_mode.length;
			if ((aktaufgabennummer==(aufgabenInModus-1))||
(ex_modeFortschritt<=aktaufgabennummer)) {
				btn_next.visible=false;
			} else {
				btn_next.visible=true;
			}
			if (aktaufgabennummer==0) {
				btn_prev.visible=false;
			} else {
				btn_prev.visible=true;
			}
			titletext.htmlText="<b>Levelmodus - Aufgabe " +
(aktaufgabennummer+1)+" (von "+ aufgabenInModus +")</b>";
			aktxml = ex_mode[aktaufgabennummer]['xml'];
			arrAll = ex_mode[aktaufgabennummer]['buttons'];
			aufgabe = ex_mode[aktaufgabennummer]['aufgabe'];
			hinweis = ex_mode[aktaufgabennummer]['hinweis'];
			loesungSQL = ex_mode[aktaufgabennummer]['loesungSQL'];
			loesungstext = ex_mode[aktaufgabennummer]['loesungstext'];
			initUI();
			sendAndGetData("startdata","",userid,aktxml);
			getBubble("standard");
			break;
		case "free_mode" :
			aufgabenInModus=1;
			aktaufgabennummer=0;
			btn_next.visible=false;
			btn_prev.visible=false;
			titletext.htmlText="<b>Freier Modus</b>";
			aktxml = free_mode[aktaufgabennummer]['xml'];
			arrAll = free_mode[aktaufgabennummer]['buttons'];
			aufgabe = free_mode[aktaufgabennummer]['aufgabe'];
			hinweis = free_mode[aktaufgabennummer]['hinweis'];
```

```
                        loesungSQL = free_mode[aktaufgabennummer]['loesungSQL'];
                        loesungstext = free_mode[aktaufgabennummer]['loesungstext'];
                        initUI();
                        sendAndGetData("startdata","",userid,aktxml);
                        if (online==1) {
                                getBubble("standard");
                        } else {
                                getBubble("<b>Programm-Fehler</b><br><br>Da der SQL-
Learner derzeit für den Offline-Modus konfiguriert ist, ist eine Ausgabe der
Ergebnisse deiner SQL-Anfragen nicht möglich.");
                                trace("Variable 'var online=1' setzen um die Ausgabe der
Ergebnisse zu aktivieren.");
                        }
                        break;
        }
        //array mit veralteten Elementen entfernen
        arrCom.splice(0,arrCom.length);
        //sicherstellen, dass der Daumen (Aufgabe geschafft) versteckt ist
        thumb.gotoAndStop(1);
        updateCom();
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

////////////////////////////////////////////////
//initialisierung der MovieClips für die ScrollPanes
////////////////////////////////////////////////
function initUI():void {
        //mAll neu erstellen => alte Befehlsliste wird entfernt
        mAll = new MovieClip;
        //Unsichtbaren MovieClip erstellen, der spaeter den Inhalt der ScrollPane
        //"aSpAll" darstellt (rechte Seite)
        mAll.graphics.beginFill(0xD2EAFB, 0.1);
        mAll.graphics.drawRect(0, 0, aSpAll.width, ((arrAll.length*25>260) ?
(arrAll.length*25) : 260));
        mAll.graphics.endFill();
        for (var j=0; j<=arrAll.length; j++) {
                if (arrAll[j] != null) {
                        mAll.addChild(arrAll[j]);
                        arrAll[j].x= 5;
                        arrAll[j].y= j*25+5;
                        arrAll[j].addEventListener(MouseEvent.MOUSE_DOWN,
mousedownHandleraSpAll);
                }
        }
        // Unsichtbaren MovieClip erstellen, der spaeter den Inhalt der ScrollPane
        // "aSpCom" darstellt (unten)
        mCom.graphics.beginFill(0xD2EAFB, 0.1);
        mCom.graphics.drawRect(0, 0, aSpCom.width, aSpCom.height);
        mCom.graphics.endFill();
        // MovieClips mit den ScrollPanes verbinden
        aSpAll.source = mAll;
        aSpCom.source = mCom;
        // Scrollbar-Position zurücksetzen
        aSpAll.verticalScrollPosition = 0;
        aSpCom.verticalScrollPosition = 0;
        // MovieClip mit Datagrids hinzufügen
        addChild(mData);
}

////////////////////////////////////////////////
//Befehlszeilenelemente in der unteren ScrollPane neu Zeichnen
////////////////////////////////////////////////
function updateCom():void {
        //Alle Elemente von der Buehne entfernen
        while (mCom.numChildren>0) {
                mCom.removeChildAt(0);
        }
        //Array neu in Befehlszeile einfügen
        if (arrCom.length>0) {
                var nextpos=5;//gibt die Position des x-wertes an, an dem das nächste
Element eingefuegt werden kann
                var zeile=0;//aktuelle Zeile in der geschrieben wird

                for (var j=0; j<arrCom.length; j++) {
                        if (arrCom[j] != null) {
                                mCom.addChild(arrCom[j]);
                                arrCom[j].width = arrCom[j].width;
                                arrCom[j].height = arrCom[j].height;
```

```
                                if (nextpos+arrCom[j].width > aSpCom.width-15) {
                                        nextpos=5;
                                        zeile+=1;
                                }

                                mCom.addChild(arrComSpacer[j]);
                                arrComSpacer[j].x=nextpos-4;
                                arrComSpacer[j].y=5+zeile*25;
                                arrComSpacer[j].gotoAndStop(1);

                                arrCom[j].x= nextpos;
                                arrCom[j].y= 5+zeile*25;
                                arrCom[j].addEventListener(MouseEvent.MOUSE_DOWN,
mousedownHandleraSpCom);
                                nextpos +=arrCom[j].width+5;
                        }
                }
        }
        //Hintergrund
        mCom.graphics.clear();
        mCom.graphics.beginFill(0xD2EAFB, 1);
        mCom.graphics.drawRect(0, 0, aSpCom.width, aSpCom.height + ((zeile>2) ?
((zeile-2)*25)+5 : 0));
        mCom.graphics.endFill();
        aSpCom.update();
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

////////////////////////////////////////////////
//Sende und empfange XML Daten abhaenig von den uebergebenen Parametern
////////////////////////////////////////////////
function sendAndGetData(methode, ausgabe:String, userid:int, xmldata:XML):void {
        if (online ==1) {
                var sqlquery:URLLoader = new URLLoader();
                var url:String = schnittstelle;
                var request:URLRequest = new URLRequest(url);
                request.method = URLRequestMethod.POST;
                var variables:URLVariables = new URLVariables();
                variables.userid = "u"+userid;//u anhaengen fuer eindeutig
identifizierbare Tabellennamen
                variables.query = ausgabe;//Ausgabe der SQL-Zeile, wird nicht benoetigt
beim Startzustand herstellen
                variables.meth = methode;//"getall" oder "startdata"
                variables.startdata = aktxml;//wird nur benoetigt bei Startzustand
herstellen wemm meth="startdata"
                request.data = variables;
                try {
                        sqlquery.load(request);
                } catch (error:Error) {
                        getBubble("<b>Programm-Fehler</b><br><br>Der SQL-Learner konnte
keine Daten versenden.<br>Bitte die Anwendung neu starten und die Rechte von SWF-
Dateien überprüfen.");
                }
                sqlquery.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);

                if (methode == "startdata") {
                        sqlquery.addEventListener(Event.COMPLETE, completeHandlerStart);
                } else {
                        sqlquery.addEventListener(Event.COMPLETE,
completeHandlerGetAll);
                }
        } else {
                var xml:XML = new XML();
                xml = aktxml;
                //Die im Programm vorgegebenen Daten einfuegen
                insertData(xml);
                //Erste Tabelle sichtbar
                arrDg[0].visible = true;
                arrTabsBack[0].visible = true;
        }
        function ioErrorHandler(event:Event):void {
                getBubble("<b>Programm-Fehler</b><br><br>Die Verbindung zur Datei
"+schnittstelle+", die zur Ausführung dieses Lernprogramms nötig ist, konnte nicht
hergestellt werden.");
                trace("IO Error: sqllearner.php nicht gefunden. Variable
\"schnittstelle\" anpassen oder sicherstellen, dass die Datei sqllearner.php vom
Webserver interpretiert wird.");
        }
```

```
        function completeHandlerStart(event:Event):void {
                var xml:XML = new XML();
                xml = aktxml;
                //Die im Programm vorgegebenen Daten einfuegen
                insertData(xml);
                //Erste Tabelle sichtbar
                arrDg[0].visible = true;
                arrTabsBack[0].visible = true;
        }
        function completeHandlerGetAll(event:Event):void {
                var xml:XML = new XML(event.target.data);
                var leeresXML:XML;
                if (xml.hasComplexContent() == true) {
                        insertData(xml);
                        var result:String = String(sqlquery.data);
                        trace(result);
                        arrDg[3].visible = true;
                        arrTabsBack[3].visible = true;
                } else {
                        getBubble("<b>Programm-Fehler</b><br><br>Die Verbindung zwischen
der Schnittstelle ("+schnittstelle+") und der MySQL-Datenbank konnte nicht aufgebaut
werden.");
                        trace("Konfiguration in der global.php überprüfen und
sicherstellen, dass die Datenbank \"sqllearner\" angelegt ist.");
                }
        }
        function insertData(xml):void {

                while (mData.numChildren>0) {
                        mData.removeChildAt(0);
                }
                arrDg.splice(0,arrDg.length);
                arrDp.splice(0,arrDp.length);
                arrTabs.splice(0,arrTabs.length);
                arrTabsBack.splice(0,arrTabsBack.length);

                //3 moegliche tabellen + 1 ausgabetabelle
                for (var i=0; i<4; i++) {
                        arrDp.push(new DataProvider(xml.children()[i]));
                        arrDg.push(new DataGrid());
                        arrDg[i].addEventListener(MouseEvent.MOUSE_DOWN,
mousedownHandleraDg);
                        // Attribute vom Tag <table[i] holen. Dies sind immer
                        // name, h1, h2 ... (h1 usw. fuer spaltenueberschriften
                        // Anzahl der Spalten also immer attNamesList.length()-1
                        var attNamesList:XMLList = xml.children()[i].@*;
                        // Fuer jede Spalte
                        for (var j=0; j<attNamesList.length()-1; j++) {
                                arrDg[i].addColumn(new DataGridColumn("c"+(j+1)));
                                arrDg[i].columns[j].headerText =
xml.children()[i].attributes()[1+j];
                        }
                        //Button fuer zu zugehoerige Tabelle erstellen
                        if (i != 3) {
                                arrTabs.push(new MyButton(0x0099FF,
0x000000,xml.children()[i].attributes()[0]));
                        } else {
                                arrTabs.push(new MyButton(0xCCCCCC,
0x000000,xml.children()[i].attributes()[0]));
                        }
                        //Einstellungen am jeweiligen DataGrid
                        arrDg[i].dataProvider = arrDp[i];
                        arrDg[i].setSize(365,208);
                        arrDg[i].move(10,62);
                        arrDg[i].sortableColumns = false;
                        arrDg[i].visible = false;
                        mData.addChild(arrDg[i]);
                        arrTabs[i].x =13+((i>0) ? arrTabs[i-1].x+arrTabs[i-1].width :
0);
                        arrTabs[i].y =42;
                        arrTabs[i].addEventListener(MouseEvent.MOUSE_DOWN,
mousedownHandleraSpAll);
                        //Hintergrund MovieClip erstellen, der anzeigen soll, welches
Tab gerade aktiv ist
                        arrTabsBack.push(new MovieClip);
                        arrTabsBack[i].graphics.beginFill(0x585F63, 1);
                        arrTabsBack[i].graphics.drawRect(0, 0, arrTabs[i].width+4,
arrTabs[i].height+4);
                        arrTabsBack[i].graphics.endFill();
                        arrTabsBack[i].graphics.beginFill(0xD2EAFB, 1);
```

```
                        arrTabsBack[i].graphics.drawRect(1, 1, arrTabs[i].width+2,
arrTabs[i].height+3);
                        arrTabsBack[i].graphics.endFill();
                        arrTabsBack[i].x = arrTabs[i].x-3;
                        arrTabsBack[i].y = arrTabs[i].y-3;
                        arrTabsBack[i].visible = false;
                        //Tabs und Tab Hintergruende einfuegen
                        if (arrTabs[i].label != "") {
                                mData.addChild(arrTabsBack[i]);
                                mData.addChild(arrTabs[i]);
                        }
                }
                //Ausgabebutton am rechten Rand
                arrTabs[3].x = 374 - arrTabs[3].width;
                arrTabsBack[3].x = arrTabs[3].x-3;
        }
}

////////////////////////////////////////////////
//Handler-Funktionen
////////////////////////////////////////////////
////////////////////////////////////////////////
//Drag: von Befehlsliste
////////////////////////////////////////////////
function mousedownHandleraSpAll(event:MouseEvent):void {
        //sicherstellen, dass Sprechblase ausgeblendet ist
        bbl.visible = false;
        ava.addEventListener(MouseEvent.CLICK, avatarClickHandler);
        Mouse.hide();
        //Kopie von dem Element in der Befehlsliste erstellen
        //Dieses Objekt wird nur verwendet, damit das Element vom Benutzer
        //in das Eingabefeld per Drag&Drop gelegt werden kann und das Element
        //in der Befehlsliste bleibt (falls es nochmal verwendet werden soll)
        var dragbutton = new MyButton(event.target.colorBg, event.target.colorTx,
event.target.label);
        dragbutton.addEventListener(MouseEvent.MOUSE_UP, mouseupHandleraSpAll);
        dragbutton.addEventListener(MouseEvent.MOUSE_MOVE, mousemoveHandleraSpAll);
        addChild(dragbutton);
        dragbutton.x = mouseX-10;
        dragbutton.y = mouseY-5;
        dragbutton.startDrag();
}
////////////////////////////////////////////////
//Drag: von Datagrid
////////////////////////////////////////////////
function mousedownHandleraDg(event:MouseEvent):void {
        //sicherstellen, dass Sprechblase ausgeblendet ist
        bbl.visible = false;
        ava.addEventListener(MouseEvent.CLICK, avatarClickHandler);
        //Abfrage, falls in einen leeren Bereich im Datagrid geklickt wird oder die
Scrollbar bzw. Buttons der Scrollbar angeklickt wurden
        //zusätzlich: Daten dürfen nicht aus dem Ausgabe-Fenster entnommen werden.
        if ((event.target.toString() != "[object LabelButton]") &&
(event.target.toString() != "[object Sprite]") && (event.target.toString() != "[object
BaseButton]")&& (event.target.label != undefined) && (arrDg[3].visible == false)) {
                var dragbutton;
                Mouse.hide();
                //Kopie von dem Element aus dem Datagrid erstellen
                //Dieses Objekt wird nur verwendet, damit das Element vom Benutzer
                //in das Eingabefeld per Drag&Drop gelegt werden kann
                if (event.target.toString() == "[object HeaderRenderer]") {
                        dragbutton = new MyButton(0xFF0099, 0x000000,
event.target.label);
                } else {
                        dragbutton = new MyButton(0x9900FF, 0x000000, "'" +
event.target.label + "'");
                }
                dragbutton.addEventListener(MouseEvent.MOUSE_UP, mouseupHandleraSpAll);
                dragbutton.addEventListener(MouseEvent.MOUSE_MOVE,
mousemoveHandleraSpAll);
                addChild(dragbutton);
                dragbutton.x = mouseX-10;
                dragbutton.y = mouseY-5;
                dragbutton.startDrag();
        }
}
////////////////////////////////////////////////
//Drag: von Eingabefeld in Eingabefeld
//dazu: Element loeschen und per Drop neu einfuegen
////////////////////////////////////////////////
```

```
function mousedownHandleraSpCom(event:MouseEvent):void {
	Mouse.hide();
	//Ausgewaehltes Element aus dem Array arrCom entfernen
	arrComSpacer.pop();
	for (var i=0; i<arrCom.length; i++) {
		if (event.target === arrCom[i]) {
			arrCom.splice(i,1);
		}
	}
	//Kopie von dem Element im erstellen, das beim Cursor bleibt, solange
	//Die linke Maustaste gedrueckt wird
	var dragbutton = new MyButton(event.target.colorBg, event.target.colorTx,
event.target.label);
	//hier koennen die gleichen Listener verwendet werden wie beim neuen
	//einfuegen, da das Element nicht verschoben, sondern geloescht und neu
	//eingefuegt wird
	dragbutton.addEventListener(MouseEvent.MOUSE_UP, mouseupHandleraSpAll);
	dragbutton.addEventListener(MouseEvent.MOUSE_MOVE, mousemoveHandleraSpAll);
	addChild(dragbutton);
	dragbutton.x = mouseX-10;
	dragbutton.y = mouseY-5;
	dragbutton.startDrag();
	updateCom();
}

////////////////////////////////////////////////
// Wenn beim Drag&Drop ein Element über einen Zwischenraum bewegt wird, soll
// dieser sichtbar werden
////////////////////////////////////////////////
function mousemoveHandleraSpAll(event:MouseEvent):void {
	var found:Boolean = false;
	var animAt:int;
	//fuer jeden Zwischenraum
	for (var i=0; i<arrComSpacer.length; i++) {
		arrComSpacer[i].alpha = 0;
		//sicherstellen, dass jeweils nur ein Abstandshalter sichtbar ist
		if (event.target.hitTestObject(arrComSpacer[i]) && (found == false)) {
			arrComSpacer[i].alpha = 1;
			if (arrComSpacer[i].currentFrame==1) {
				arrComSpacer[i].gotoAndPlay(2);
			}
			animAt = i;
			found = true;
		}
		//Alle Filme zuruecksetzen ausser den aktuellen
		if (animAt != i) {
			arrComSpacer[i].gotoAndStop(1);
		}
	}
}

////////////////////////////////////////////////
// Drop-Aktion in Eingabefeld
////////////////////////////////////////////////
function mouseupHandleraSpAll(event:MouseEvent):void {
	var inserted:Boolean=false;
	event.target.stopDrag();
	Mouse.show();
	if (event.target.hitTestObject(aSpCom)) {
		//Durchlaufe alle Zwischenraeume um festzustellen ob das Objekt
		//dort abgelegt werden soll. Falls keine Uebereinstimmtung => Objekt am
Ende einfuegen
		for (var i=0; i<=arrComSpacer.length; i++) {
			if ((i==arrComSpacer.length) ||
event.target.hitTestObject(arrComSpacer[i])) {
				//Element an Position i im Array arrCom einfuegen
				arrCom.splice(i,0,new  MyButton(event.target.colorBg,
event.target.colorTx, event.target.label));
				//Zwischenraum MovieClip hinzufuegen
				arrComSpacer.push(new spacer());
				arrComSpacer[arrComSpacer.length-1].alpha = 0;
				arrComSpacer[i].alpha = 0;
				//Elemente neu zeichnen
				updateCom();
				//Drag&Drop Element entfernen
				event.target.removeEventListener(MouseEvent.MOUSE_UP,
mouseupHandleraSpAll);
				event.target.removeEventListener(MouseEvent.MOUSE_MOVE,
mousemoveHandleraSpAll);
				removeChild(event.target as MovieClip);
```

```
                                inserted = true;
                                break;
                        }
                }
        }
        //Falls ein Tabellenname auf dem Datenbereich fallen gelassen wird
        if (event.target.hitTestObject(mData)) {
                for (var j=0; j<4; j++) {
                        if (event.target.label == arrTabs[j].label) {
                                for (var k=0; k<4; k++) {
                                        if (k==j) {
                                                arrDg[k].visible=true;
                                                arrTabsBack[k].visible = true;
                                        } else {
                                                arrDg[k].visible=false;
                                                arrTabsBack[k].visible = false;
                                        }
                                }
                        }
                }
        }
        if (event.target.hitTestObject(ava)) {
                //Sprechblase anzeigen, wenn ein SQL-Befehlselement auf den Avatar
gezogen wird
                if (avaexpl[event.target.label] != null) {
                        getBubble(avaexpl[event.target.label]);
                } else {
                        getBubble("Zu diesem Befehl finde ich leider nichts.");
                }
        }
        //falls das Element irgendwo fallen gelassen wurde
        if (inserted==false) {
                event.target.removeEventListener(MouseEvent.MOUSE_UP,
mouseupHandleraSpAll);
                event.target.removeEventListener(MouseEvent.MOUSE_MOVE,
mousemoveHandleraSpAll);
                removeChild(event.target as MovieClip);
        }

}
////////////////////////////////////////////////
//Handler: Avatar wird angeklickt
////////////////////////////////////////////////
function avatarClickHandler(event:MouseEvent):void {
        getBubble("standard");
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

var auswahlmovie = new auswahl();

////////////////////////////////////////////////
//Intro anzeigen
var intromovie = new intro();
addChild(intromovie);
intromovie.addEventListener(MouseEvent.CLICK, introClickHandler);

////////////////////////////////////////////////
//Auswahlbildschirm starten
////////////////////////////////////////////////
function introClickHandler(event:MouseEvent):void {
        //Intro entfernen
        if (intromovie != null) {
                intromovie.removeEventListener(MouseEvent.CLICK, introClickHandler);
                removeChild(intromovie);
        }

        addChild(auswahlmovie);
        auswahlmovie.btn_tut.addEventListener(MouseEvent.CLICK, btntut);
        auswahlmovie.btn_ex.addEventListener(MouseEvent.CLICK, btnex);
        auswahlmovie.btn_free.addEventListener(MouseEvent.CLICK, btnfree);
        auswahlmovie.initauswahl(0);

        ////////////////////////////////////////////////
        // Einführungsmodus
        function btntut(event:MouseEvent):void {
                removeChild(auswahlmovie);
                modus = "tut_mode";
                changeAufgabe(0);
```

```
		}
		////////////////////////////////////////////////
		// Levelmodus
		function btnex(event:MouseEvent):void {
			removeChild(auswahlmovie);
			modus = "ex_mode";
			changeAufgabe(0);
		}

		////////////////////////////////////////////////
		// Freier Modus
		function btnfree(event:MouseEvent):void {
			removeChild(auswahlmovie);
			modus = "free_mode";
			changeAufgabe(0);
		}
		////////////////////////////////////////////////
		// Funktion für den Modus verlassen Button im Hauptfenster um
		// wieder in die Auswahl zu kommen
		btn_back.addEventListener(MouseEvent.CLICK, btnback);
		function btnback(event:MouseEvent):void {
			addChild(auswahlmovie);
			auswahlmovie.initauswahl(1);
		}

}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

var free_mode = new Array();
////////////////////////////////////////////////////
//Free Mode
////////////////////////////////////////////////////
var free_xml:XML =
<sqllearn>
<tab name="person" h1="id" h2="Vorname" h3="Name" h4="Adresse">
  <item c1="1" c2="Peter" c3="Schulz" c4="Waldweg 1" />
  <item c1="2" c2="Michael" c3="Schmitt" c4="Poststr. 28" />
  <item c1="3" c2="Julia" c3="Meier" c4="Bahnhofstr. 16" />
  <item c1="4" c2="Teresa" c3="Grote" c4="Kreuzweg. 19" />
  <item c1="5" c2="Heinrich" c3="Niemann" c4="Windmuehlenberg 32" />
  <item c1="6" c2="Ludger" c3="Weber" c4="Feldweg 11" />
  <item c1="7" c2="Marc" c3="Steinmann" c4="Kurfuerstendamm 12" />
  <item c1="8" c2="Richard" c3="Elsberger" c4="Badstr. 7" />
  <item c1="9" c2="Alwin" c3="Mueller" c4="Seestr. 9" />
  <item c1="10" c2="Ute" c3="Werner" c4="Rathausplatz 41" />
  <item c1="11" c2="Gerda" c3="Schulz" c4="Hochfeldstr. 37" />
  <item c1="12" c2="Paula" c3="Meier" c4="Hohe Str. 3" />
  <item c1="13" c2="Heinz" c3="Schulz" c4="Gutenbergallee 11" />
  <item c1="14" c2="Waltraud" c3="Elsmann" c4="Zentrumsplatz 2" />
  <item c1="15" c2="Zoe" c3="Trautmann" c4="Adlerweg 8" />
  </tab>
 <tab name="telefon" h1="id" h2="Telefon" h3="Anbieter">
  <item c1="1" c2="025941234511" c3="T1" />
  <item c1="2" c2="023463464511" c3="D-Mobile" />
  <item c1="3" c2="049394854534" c3="O3" />
  <item c1="4" c2="049394854534" c3="T1" />
  <item c1="5" c2="049394854534" c3="versafon" />
  <item c1="5" c2="165465488884" c3="O3" />
  <item c1="5" c2="026669998888" c3="versafon" />
  <item c1="6" c2="024343434441" c3="O3" />
  <item c1="6" c2="017729323223" c3="D-Mobile" />
  <item c1="7" c2="340534534534" c3="O3" />
  <item c1="8" c2="349034034904" c3="versafon" />
  <item c1="8" c2="012548787888" c3="versafon" />
  <item c1="9" c2="025941234511" c3="O3" />
  <item c1="11" c2="32403434344" c3="T1" />
  <item c1="12" c2="349034034904" c3="T1" />
  <item c1="12" c2="017845454521" c3="D-Mobile" />
  <item c1="12" c2="4959494994944" c3="T1" />
  <item c1="12" c2="2222333332222" c3="versafon" />
  <item c1="13" c2="9056044554644" c3="versafon" />
  <item c1="13" c2="0177777778888" c3="D-Mobile" />
  <item c1="14" c2="4959494994944" c3="O3" />
  <item c1="15" c2="4959494994944" c3="T1" />
  <item c1="15" c2="6546565548888" c3="D-Mobile" />
  </tab>
 <tab name="finanzen" h1="id" h2="Ausgaben" h3="Einnahmen" h4="Bank">
  <item c1="1" c2="700" c3="1132" c4="Spasskasse" />
```

```
  <item c1="2" c2="1700" c3="1520" c4="Vollksbank" />
  <item c1="3" c2="1243" c3="874" c4="Daten Bank" />
  <item c1="4" c2="100" c3="190" c4="Daten Bank" />
  <item c1="5" c2="2700" c3="5433" c4="Spasskasse" />
  <item c1="6" c2="1243" c3="1243" c4="Vollksbank" />
  <item c1="7" c2="233" c3="847" c4="Vollksbank" />
  <item c1="8" c2="631" c3="3422" c4="Hypo Unreal" />
  <item c1="9" c2="7642" c3="14522" c4="Spasskasse" />
  <item c1="10" c2="3346" c3="5433" c4="Daten Bank" />
  <item c1="11" c2="123" c3="321" c4="Daten Bank" />
  <item c1="12" c2="745" c3="1520" c4="Daten Bank" />
  <item c1="13" c2="1964" c3="2341" c4="Spasskasse" />
  <item c1="14" c2="2344" c3="2532" c4="Vollksbank" />
  <item c1="15" c2="544" c3="478" c4="Vollksbank" />
 </tab>
  <tab name="" />

  </sqllearn>;

var free_btn:Array = new Array();
free_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"ASC"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"LIKE"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"HAVING"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"BETWEEN"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"DISTINCT"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"IN"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"ALL"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"UNION"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"DROP"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"TABLE"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"JOIN"));
free_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"NOT"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"AND"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"OR"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"+"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"-"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"( "));
free_btn.push(new MyButton(0x99FF00, 0x000000,") "));
free_btn.push(new MyButton(0x99FF00, 0x000000,"="));
free_btn.push(new MyButton(0x99FF00, 0x000000,">"));
free_btn.push(new MyButton(0x99FF00, 0x000000,">="));
free_btn.push(new MyButton(0x99FF00, 0x000000,"<"));
free_btn.push(new MyButton(0x99FF00, 0x000000,"<="));
free_btn.push(new MyButton(0x99FF00, 0x000000,","));
free_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
free_btn.push(new MyButton(0x99FF00, 0x000000,"AVG("));
free_btn.push(new MyButton(0x9900FF, 0x000000,"'%mann'"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"500"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"1000"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"15"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"16"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"17"));
free_btn.push(new MyButton(0x9900FF, 0x000000,"18"));

var free_aufg="<b>Willkommen im 'Freien Modus' !</b><br><br>Hier sind fast alle
Befehle verfügbar, die du schon aus den anderen Modi kennst. Probiere einfach etwas
herum. <br>Hier lassen sich besonders gut verschiedene SELECT Anweisungen
ausprobieren, z.B. Alle Telefonnummern einer bestimmten Person anzeigen.";

var free_hin="";

var free_lsql="";

var free_ltext="";

free_mode.push(new Aufgabe(free_xml, free_btn, free_aufg, free_hin, free_lsql,
free_ltext));
//////////////////////////////////////////////////
//Ende Free Mode
//////////////////////////////////////////////////

var ex_mode = new Array();
```

```
////////////////////////////////////////////////////
//Ex Mode - Aufgabe 1
////////////////////////////////////////////////////
var ex_A1_xml:XML =
<sqllearn>
<tab name="tabelle1" h1="Vorname" h2="Name" h3="Adresse">
  <item c1="Peter" c2="Schulz" c3="Lessingstr. 3" />
  <item c1="Heinz" c2="Schmitt" c3="Ahornweg 11" />
  <item c1="Peter" c2="Stock" c3="Waldweg 2" />
  <item c1="Peter" c2="Vogelsang" c3="Erdweg 15" />
  <item c1="Julia" c2="Uckelmann" c3="Winkelgasse 7" />
  <item c1="Franz" c2="Meier" c3="Taubenstr. 17a" />
  <item c1="Marie" c2="Ziegler" c3="Hauptstr. 8" />

  </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var ex_A1_btn:Array = new Array();
ex_A1_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A1_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A1_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A1_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
ex_A1_btn.push(new MyButton(0x99FF00, 0x000000,"="));

var ex_A1_aufg="<b>Willkommen im Levelmodus!</b><br><br>Hey, danke dass du mir helfen
willst! Wir hatten jetzt inzwischen einige Unterrichtsstunden zu dem Thema Datenbanken
und ich habe hier einige Aufgaben...<br>Wenn du wissen willst, wie ein bestimmter SQL-
Befehl funktioniert, halte die linke Maustaste über dem Befehl gedrückt und lasse ihn
auf mein Bild fallen. Ich zeige dir dann, was in meinem Schulbuch dazu steht.
<br><br><b>Aufgabe 1</b><br><ul><li>Erstelle eine Anfrage, die alle Personen mit dem
Vornamen 'Peter' ausgibt!</li></ul>";

var ex_A1_hin="<br><b>Hinweis:</b><br><br>Wähle alle Datensätze aus der Tabelle
'tabelle1', wo der Vorname=Peter ist.";

var ex_A1_lsql="SELECT * FROM tabelle1 WHERE Vorname = 'Peter'";

var ex_A1_ltext="<br>Genau diese Anfrage ist es! Ich glaube wir sind ein gutes Team!";

ex_mode.push(new Aufgabe(ex_A1_xml, ex_A1_btn, ex_A1_aufg, ex_A1_hin, ex_A1_lsql,
ex_A1_ltext));
////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 1
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//Ex Mode - Aufgabe 2
////////////////////////////////////////////////////
var ex_A2_xml:XML =<sqllearn>
<tab name="pflanzen" h1="Pflanze" h2="Gattung" h3="LateinischerName">
  <item c1="Sonneblume" c2="Korbbluetler " c3="Helianthus annuus" />
  <item c1="Fingerhut" c2="Wegerichgewächse" c3="Digitalis" />
  <item c1="Wiesen-Rose" c2="Rosengewächse" c3="Rosa pendulina" />
  <item c1="Orchidee" c2="Orchideen " c3="Orchidaceae" />
  <item c1="Alpen-Rose" c2="Rosengewächse" c3="Rosa carolina" />
  <item c1="Tulpe" c2="Liliengewächse" c3="Tulipa" />
  <item c1="Osterglocke" c2="Narzissen" c3="Narcissus pseudonarcissus" />
  <item c1="Rose" c2="Rosengewächse" c3="Rosa" />
   </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var ex_A2_btn:Array = new Array();
ex_A2_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A2_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A2_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A2_btn.push(new MyButton(0x99FF00, 0x000000,"="));

var ex_A2_aufg="<b>Aufgabe 2</b><br>Erstelle eine Anfrage, die folgendes
erfüllt:<ul><li>Es soll nur die Spalte 'Pflanze' ausgegeben werden.</li><li>Alle
Pflanzen müssen zur Gattung 'Rosengewächse' gehören.</li></ul>";

var ex_A2_hin="<br><b>Hinweis:</b><br><br>Wenn du nicht weiterkommst, ziehe mal den
SELECT-Befehl mit der Maus auf mich und lasse dann die Maustaste los. In der Erklärung
zum SELECT-Befehl findest du Hilfe.<br><br>An die Namen der Spalten kommst du, wenn du
sie mit der Maus von der Tabelle in das Eingabefeld unten neben mir ziehst.";
```

```
var ex_A2_lsql="SELECT Pflanze FROM pflanzen WHERE Gattung = 'Rosengewächse'";

var ex_A2_ltext="<br>Mir fällt gerade ein, dass ich noch eine Erdkunde-Aufgabe lösen
muss...";

ex_mode.push(new Aufgabe(ex_A2_xml, ex_A2_btn, ex_A2_aufg, ex_A2_hin, ex_A2_lsql,
ex_A2_ltext));
////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 2
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//Ex Mode - Aufgabe 3
////////////////////////////////////////////////////
var ex_A3_xml:XML = <sqllearn>
<tab name="eu_laender" h1="Land" h2="Hauptstadt" h3="Einwohner_in_Mio">
  <item c1="Belgien" c2="Brüssel " c3="10.3" />
  <item c1="Bulgarien" c2="Sofia" c3="7.3" />
  <item c1="Dänemark" c2="Kopenhagen" c3="5.4" />
  <item c1="Deutschland" c2="Berlin" c3="82.4" />
  <item c1="Estland" c2="Tallinn" c3="1.3" />
  <item c1="Finnland" c2="Helsinki " c3="5.2" />
  <item c1="Frankreich" c2="Paris" c3="63.7" />
  <item c1="Griechenland" c2="Athen" c3="10.7" />
  <item c1="Großbritannien" c2="London" c3="60.8" />
  <item c1="Irland" c2="Dublin" c3="4.1" />
  <item c1="Island" c2="Reykjavik" c3="0.3" />
  <item c1="Italien" c2="Rom" c3="58.1" />
  <item c1="Lettland" c2="Riga" c3="2.2" />
  <item c1="Littauen" c2="Vilnius" c3="3.6" />
  <item c1="Luxemburg" c2="Luxemburg" c3="0.4" />
  <item c1="Malta" c2="Valletta" c3="0.4" />
  <item c1="Niederlande" c2="Amsterdam" c3="16.6" />
  <item c1="Österreich" c2="Wien" c3="8.3" />
  <item c1="Polen" c2="Warschau" c3="38.1" />
  <item c1="Portugal" c2="Lissabon" c3="10.4" />
  <item c1="Rumänien" c2="Bukarest" c3="21.5" />
  <item c1="Schweden" c2="Stockholm" c3="9.2" />
  <item c1="Slowakei" c2="Bratislava" c3="5.4" />
  <item c1="Slowenien" c2="Ljubljana" c3="2.0" />
  <item c1="Spanien" c2="Madrid" c3="45.3" />
  <item c1="Tschechische Republik" c2="Prag" c3="10.3" />
  <item c1="Ungarn" c2="Budapest" c3="10.1" />
  <item c1="Zypern" c2="Nikosia" c3="0.8" />
   </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var ex_A3_btn:Array = new Array();
ex_A3_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A3_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A3_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A3_btn.push(new MyButton(0x99FF00, 0x000000,"ORDER BY"));
ex_A3_btn.push(new MyButton(0x99FF00, 0x000000,"DESC"));
ex_A3_btn.push(new MyButton(0x99FF00, 0x000000,"="));
ex_A3_btn.push(new MyButton(0x99FF00, 0x000000,","));
ex_A3_btn.push(new MyButton(0x99FF00, 0x000000,">=10.0"));

var ex_A3_aufg="Ich soll aus der Tabelle der Länder der Europäischen Union alle Länder
heraussuchen, die mehr als 10 Millionen Einwohner haben. Kennst du eine SQL-Anweisung
mit der wir die Aufgabe schnell erledigen können?<br><br><b>Aufgabe 3</b><br>Erstelle
eine Anfrage, die folgendes erfüllt:<ul><li>Es soll nur die Spalte 'Land' und die
Spalte 'Einwohner_in_Mio' ausgegeben werden.</li><li>Alle Länder, die in der Ausgabe
aufgenommen werden, müssen mehr als 10 Millionen Einwohner besitzen.</li><li>Sortiere
die absteigend nach Einwohnerzahl</li></ul>";

var ex_A3_hin="<br><b>Hinweis:</b><br><br>Wenn du nicht weiterkommst, ziehe einen
Befehl mit der Maus auf mich und lasse dann die Maustaste los. In der Erklärung
findest du Hilfe zu den meisten Befehlen.<br><br>An die Namen der Spalten kommst du,
wenn du sie mit der Maus von der Tabelle in das Eingabefeld unten neben mir ziehst.";

var ex_A3_lsql="SELECT Land , Einwohner_in_Mio FROM eu_laender WHERE Einwohner_in_Mio
>=10.0 ORDER BY Einwohner_in_Mio DESC";

var ex_A3_ltext="<br>Super, das ging viel schneller als es per Hand aufzuschreiben!";

ex_mode.push(new Aufgabe(ex_A3_xml, ex_A3_btn, ex_A3_aufg, ex_A3_hin, ex_A3_lsql,
ex_A3_ltext));
```

```
////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 3
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//Ex Mode - Aufgabe 3
////////////////////////////////////////////////////
var ex_A4_xml:XML = <sqllearn>
<tab name="eu_laender" h1="Land" h2="Hauptstadt" h3="Einwohner_in_Mio">
  <item c1="Belgien" c2="Brüssel " c3="10.3" />
  <item c1="Bulgarien" c2="Sofia" c3="7.3" />
  <item c1="Dänemark" c2="Kopenhagen" c3="5.4" />
  <item c1="Deutschland" c2="Berlin" c3="82.4" />
  <item c1="Estland" c2="Tallinn" c3="1.3" />
  <item c1="Finnland" c2="Helsinki " c3="5.2" />
  <item c1="Frankreich" c2="Paris" c3="63.7" />
  <item c1="Griechenland" c2="Athen" c3="10.7" />
  <item c1="Großbritannien" c2="London" c3="60.8" />
  <item c1="Irland" c2="Dublin" c3="4.1" />
  <item c1="Island" c2="Reykjavik" c3="0.3" />
  <item c1="Italien" c2="Rom" c3="58.1" />
  <item c1="Lettland" c2="Riga" c3="2.2" />
  <item c1="Littauen" c2="Vilnius" c3="3.6" />
  <item c1="Luxemburg" c2="Luxemburg" c3="0.4" />
  <item c1="Malta" c2="Valletta" c3="0.4" />
  <item c1="Niederlande" c2="Amsterdam" c3="16.6" />
  <item c1="Österreich" c2="Wien" c3="8.3" />
  <item c1="Polen" c2="Warschau" c3="38.1" />
  <item c1="Portugal" c2="Lissabon" c3="10.4" />
  <item c1="Rumänien" c2="Bukarest" c3="21.5" />
  <item c1="Schweden" c2="Stockholm" c3="9.2" />
  <item c1="Slowakei" c2="Bratislava" c3="5.4" />
  <item c1="Slowenien" c2="Ljubljana" c3="2.0" />
  <item c1="Spanien" c2="Madrid" c3="45.3" />
  <item c1="Tschechische Republik" c2="Prag" c3="10.3" />
  <item c1="Ungarn" c2="Budapest" c3="10.1" />
  <item c1="Zypern" c2="Nikosia" c3="0.8" />
   </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var ex_A4_btn:Array = new Array();
ex_A4_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A4_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A4_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A4_btn.push(new MyButton(0x99FF00, 0x000000,") "));

var ex_A4_aufg="Die Einwohnerzahl aller Länder der Europäischen Union sollte ich auch
noch herausbekommen...<br><br><b>Aufgabe 4</b><br>Erstelle eine Anfrage,
die<ul><li>die Spalte Einwohner_in_Mio aufsummiert</li></ul>";

var ex_A4_hin="<br><b>Hinweis:</b><br><br>Benutze die Summe wie eine Spalte.";

var ex_A4_lsql="SELECT SUM( Einwohner_in_Mio )  FROM eu_laender";

var ex_A4_ltext="<br>Sehr gut, das wäre geschafft!";

ex_mode.push(new Aufgabe(ex_A4_xml, ex_A4_btn, ex_A4_aufg, ex_A4_hin, ex_A4_lsql,
ex_A4_ltext));
////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 4
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//Ex Mode - Aufgabe 5
////////////////////////////////////////////////////
var ex_A5_xml:XML = <sqllearn>
<tab name="telefonbuch" h1="Vorname" h2="Name" h3="Adresse" h4="Telefon">
       <item c1="Peter" c2="Schulz" c3="Waldweg 23" c4="98234"/>
       <item c1="Julia" c2="Schmitt" c3="Hauptstr. 33" c4="567566"/>
       <item c1="Frank" c2="Meier" c3="Seeweg 19" c4="78977"/>
       <item c1="Jörg" c2="Pieper" c3="Seeweg 8" c4="5654546"/>
       <item c1="Thorsten" c2="Weber" c3="Wernerstr. 3" c4="855667"/>
       <item c1="Maike" c2="Lohmann" c3="Zentralplatz 12" c4="125654"/>
       <item c1="Janine" c2="Mueller" c3="Amselweg 11" c4="4831233"/>
       <item c1="Otto" c2="Zöller" c3="Waldweg 23" c4="98234"/>
       <item c1="Ingo" c2="Polster" c3="Wernerstr. 3" c4="4333455"/>
       <item c1="Rolf" c2="Reinermann" c3="Zeckenweg 8" c4="333123"/>
       <item c1="Petra" c2="Feldmann" c3="Kreuzweg 4" c4="987878"/>
       <item c1="Gerd" c2="Elskemper" c3="Windmühlenberg 30" c4="3345543"/>
```

```
        <item c1="Sarah" c2="Vogt" c3="Eisenhütte 3" c4="121135"/>
        <item c1="Katharina" c2="Ziegler" c3="Hermannsweg 15a" c4="669994"/>
        <item c1="Ewald" c2="Nordmann" c3="Hauptstr. 33" c4="324545"/>
        <item c1="Lisa" c2="Artmann" c3="Teichsmühle 12" c4="1231212"/>
        <item c1="Otto" c2="Nordmann" c3="Siebstr. 33" c4="5487894"/>
        <item c1="Tobias" c2="Gellmann" c3="Bahnhofsstr. 1" c4="978544"/>
        <item c1="Birgit" c2="Enseling" c3="Kinoplatz 23" c4="3554456"/>
        <item c1="Lukas" c2="Zarting" c3="Badstr. 6" c4="13112123"/>
        <item c1="Emil" c2="Terhart" c3="Seeweg 9" c4="887878"/>
        <item c1="Frank" c2="Ostmann" c3="Vogelallee" c4="4545456"/>
        <item c1="Heike" c2="Schmitt" c3="Waldweg 23" c4="8778554"/>
        </tab>
 <tab name="" />
 <tab name="" />
 <tab name="" />
</sqllearn>;

var ex_A5_btn:Array = new Array();
ex_A5_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A5_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A5_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A5_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
ex_A5_btn.push(new MyButton(0x99FF00, 0x000000,"="));
ex_A5_btn.push(new MyButton(0x99FF00, 0x000000,"AND"));
ex_A5_btn.push(new MyButton(0x9900FF, 0x000000,"'Otto'"));
ex_A5_btn.push(new MyButton(0x9900FF, 0x000000,"'Nordmann'"));

var ex_A5_aufg="Wir sollen uns anschauen, wie eine Internetseite funktioniert, die
Telefonnummern findet.<br><br><b>Aufgabe 5</b><br>Erstelle eine Anfrage, die
<ul><li>alle Daten von Otto Nordmann ermittelt.</li></ul>";

var ex_A5_hin="<br><b>Hinweis:</b><br><br>Benutze in der WHERE-Bedingung, dass der
Vorname 'Otto' UND der Name 'Nordmann' sein muss.";

var ex_A5_lsql="SELECT * FROM telefonbuch WHERE Vorname = 'Otto' AND Name =
'Nordmann'";

var ex_A5_ltext="<br>Man hätte Otto Nordmanns Telefonnummer natürlich auch selbst
heraussuchen können, aber stell dir mal vor, es wären nicht 30 Personen, sondern
hunderttausende Personen im Telefonbuch!";

ex_mode.push(new Aufgabe(ex_A5_xml, ex_A5_btn, ex_A5_aufg, ex_A5_hin, ex_A5_lsql,
ex_A5_ltext));
//////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 5
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//Ex Mode - Aufgabe 6
//////////////////////////////////////////////////////
var ex_A6_xml:XML = ex_A5_xml;

var ex_A6_btn:Array = new Array();
ex_A6_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A6_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A6_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A6_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
ex_A6_btn.push(new MyButton(0x99FF00, 0x000000,"="));
ex_A6_btn.push(new MyButton(0x9900FF, 0x000000,"'887878'"));

var ex_A6_aufg="Wie kann ich zu einer Telefonnummer die zugehörige Person finden?
(Rückwärtssuche)<br><br><b>Aufgabe 6</b><br>Erstelle eine Anfrage, die<ul><li>zu der
Telefonnummer '887878' den Teilnehmer findet.</li></ul>";

var ex_A6_hin="<br><b>Hinweis:</b><br><br> Wenn du nicht weiterkommst, schau nochmal
nach, wie die WHERE Anweisung funktioniert.";

var ex_A6_lsql="SELECT * FROM telefonbuch WHERE Telefon = '887878'";

var ex_A6_ltext="<br>Du bist gut! Vielleicht bekommst du folgendes auch hin... ";

ex_mode.push(new Aufgabe(ex_A6_xml, ex_A6_btn, ex_A6_aufg, ex_A6_hin, ex_A6_lsql,
ex_A6_ltext));
//////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 6
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//Ex Mode - Aufgabe 7
//////////////////////////////////////////////////////
var ex_A7_xml:XML =<sqllearn>
<tab name="haushaltsplan" h1="Position" h2="Kategorie" h3="Betrag" h4="Person">
```

```
        <item c1="Arbeit Vater" c2="Einkommen" c3="2853" c4="Vater"/>
        <item c1="Arbeit Mutter" c2="Einkommen" c3="832" c4="Mutter"/>
        <item c1="Steuern Vater" c2="Steuer" c3="-978" c4="Vater"/>
        <item c1="Steuern Mutter" c2="Steuer" c3="-214" c4="Mutter"/>
        <item c1="Sonstige" c2="Steuer" c3="-121" c4="Allgemein"/>
        <item c1="Wohnung" c2="Miete" c3="-521" c4="Allgemein"/>
        <item c1="Hausrat" c2="Versicherungen" c3="-100" c4="Allgemein"/>
        <item c1="Krankenkasse" c2="Versicherungen" c3="-341" c4="Allgemein"/>
        <item c1="Sonstige" c2="Versicherungen" c3="-241" c4="Allgemein"/>
        <item c1="Auto" c2="Freizeit" c3="-221" c4="Vater"/>
        <item c1="Woche 1" c2="Lebensmittel" c3="-121" c4="Allgemein"/>
        <item c1="Woche 2" c2="Lebensmittel" c3="-108" c4="Allgemein"/>
        <item c1="Woche 3" c2="Lebensmittel" c3="-115" c4="Allgemein"/>
        <item c1="Woche 4" c2="Lebensmittel" c3="-137" c4="Allgemein"/>
        <item c1="Computer" c2="Sonstiges" c3="-50" c4="Tom"/>
        <item c1="Führerschein" c2="Sonstiges" c3="-78" c4="Tina"/>
        <item c1="Taschengeld" c2="Sonstiges" c3="-30" c4="Tom"/>
        <item c1="Taschengeld" c2="Sonstiges" c3="-30" c4="Tina"/>
        </tab>
 <tab name="" />
 <tab name="" />
 <tab name="" />
</sqllearn>;

var ex_A7_btn:Array = new Array();
ex_A7_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A7_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A7_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A7_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
ex_A7_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
ex_A7_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A7_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A7_btn.push(new MyButton(0x99FF00, 0x000000,","));

var ex_A7_aufg="Meine Eltern haben im letzten Monat aufgeschrieben, für was wir Geld
ausgegeben haben. Leider ist die Tabelle etwas unübersichtlich und es wäre schön, wenn
du mir dabei helfen könntest, sie zu ordnen.<br><br><b>Aufgabe 7</b><br>Erstelle eine
Anfrage, die folgenden Kriterien genügt:<ul><li>Gruppiere die Tabelle nach
Kategorie</li><li>Verwende als Spalten Kategorie und die Summe der Einnahmen/Ausgaben
dieser Kategorie.</li><li>Ordne absteigend nach der Summe der
Einnahmen/Ausgaben</li></ul>";


var ex_A7_hin="<br><b>Hinweis:</b><br><br>Gruppiere nach Kategorie, dann wird der
aufsummierte Betrag für jede Kategorie berechnet.<br><br>Pass bei der Sortierung auf,
dass du nach der <b>Summe</b> des Betrages sortierst.";

var ex_A7_lsql="SELECT Kategorie , SUM( Betrag )  FROM haushaltsplan GROUP BY
Kategorie ORDER BY SUM( Betrag )  DESC";

var ex_A7_ltext="<br>Da werden sich meine Eltern aber freuen!";

ex_mode.push(new Aufgabe(ex_A7_xml, ex_A7_btn, ex_A7_aufg, ex_A7_hin, ex_A7_lsql,
ex_A7_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 7
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 8
//////////////////////////////////////////////////
var ex_A8_xml:XML = ex_A7_xml;

var ex_A8_btn:Array = new Array();
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
ex_A8_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
ex_A8_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A8_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A8_btn.push(new MyButton(0x99FF00, 0x000000,","));
ex_A8_btn.push(new MyButton(0x99FF00, 0x000000,"="));

var ex_A8_aufg="<b>Aufgabe 8</b><br>Erstelle eine Anfrage, die folgenden Kriterien
genügt:<ul><li>Berücksichtige nur Ausgaben, die der Allgemeinheit zugeordnet
sind.</li><li>Gruppiere die Tabelle nach Kategorie</li><li>Verwende als Spalten
Kategorie und die Summe der Einnahmen/Ausgaben dieser Kategorie.</li><li>Ordne
absteigend nach der Summe der Einnahmen/Ausgaben</li></ul>";
```

```
var ex_A8_hin="<br><b>Hinweis:</b><br><br> Mit der WHERE-Anweisung erhältst du alle
relevanten Zeilen.";

var ex_A8_lsql="SELECT Kategorie , SUM( Betrag )  FROM haushaltsplan WHERE Person =
'Allgemein' GROUP BY Kategorie ORDER BY SUM( Betrag )  DESC";

var ex_A8_ltext="<br>Super, so ist das ganze schon viel übersichtlicher. Mit der SQL-
Anfrage \"SELECT SUM(Betrag) FROM haushaltsplan\" habe ich gesehen, dass wir noch über
200 Euro im letzten Monat übrig hatten! Da ist doch bestimmt eine Taschengelderhöhung
für die gute Arbeit drin :)";

ex_mode.push(new Aufgabe(ex_A8_xml, ex_A8_btn, ex_A8_aufg, ex_A8_hin, ex_A8_lsql,
ex_A8_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 8
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 9
//////////////////////////////////////////////////
var ex_A9_xml:XML = <sqllearn>
<tab name="filialen" h1="Ort" h2="Einnahmen" h3="Ausgaben">
       <item c1="Filiale Berlin" c2="3402" c3="2853" />
       <item c1="Filiale Hamburg" c2="3822" c3="2132" />
       <item c1="Filiale New York" c2="3412" c3="1121" />
       <item c1="Filiale Essen" c2="2333" c3="478" />
       <item c1="Filiale Berlin" c2="1402" c3="1853" />
       <item c1="Filiale Essen" c2="1893" c3="214" />
       <item c1="Filiale Köln" c2="2544" c3="2821" />
       <item c1="Filiale New York" c2="5412" c3="1521" />
       <item c1="Filiale Hamburg" c2="1822" c3="3632" />
       <item c1="Filiale Berlin" c2="1844" c3="1821" />
       </tab>
 <tab name="" />
 <tab name="" />
 <tab name="" />
</sqllearn>;

var ex_A9_btn:Array = new Array();
ex_A9_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A9_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A9_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A9_btn.push(new MyButton(0xFF9900, 0x000000,"HAVING"));
ex_A9_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A9_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A9_btn.push(new MyButton(0x99FF00, 0x000000,","));
ex_A9_btn.push(new MyButton(0x99FF00, 0x000000,">"));

var ex_A9_aufg="Mein Vater arbeitet in einem Laden, der in verschiedenen Städten
Filialen betreibt.<br><br><b>Aufgabe 9</b><br>Erstelle eine Anfrage, die<ul><li>als
Spalten den Filialnamen, die summierten Einkünfte und die summierten Ausgaben
enthält.</li><li>Es sollen nur Filialen aufgelistet werden, die mehr Einkünfte als
Ausgaben haben.</li></ul>";

var ex_A9_hin="<br><b>Hinweis:</b></br><br>Vergiss in der HAVING-Bedingung nicht, dass
du die Summe der Einnahmen mit der Summe der Ausgaben vergleichen musst.</b><br>";

var ex_A9_lsql="SELECT Ort , SUM( Einnahmen )  , SUM( Ausgaben )  FROM filialen GROUP
BY Ort HAVING SUM( Einnahmen )  > SUM( Ausgaben ) ";

var ex_A9_ltext="<br>Klasse! Wenn du das kannst, dann bestimmt auch das, was jetzt
kommt...";

ex_mode.push(new Aufgabe(ex_A9_xml, ex_A9_btn, ex_A9_aufg, ex_A9_hin, ex_A9_lsql,
ex_A9_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 9
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 10
//////////////////////////////////////////////////
var ex_A10_xml:XML =<sqllearn>
<tab name="filialen" h1="Ort" h2="Einnahmen" h3="Ausgaben">
       <item c1="Filiale Berlin" c2="3402" c3="2853" />
       <item c1="Filiale Hamburg" c2="3822" c3="2132" />
       <item c1="Filiale New York" c2="3412" c3="1121" />
       <item c1="Filiale Essen" c2="2333" c3="478" />
       <item c1="Filiale Berlin" c2="1402" c3="1853" />
       <item c1="Filiale Essen" c2="1893" c3="214" />
       <item c1="Filiale Köln" c2="2544" c3="2821" />
       <item c1="Filiale New York" c2="5412" c3="1521" />
```

```
        <item c1="Filiale Hamburg" c2="1822" c3="3632" />
        <item c1="Filiale Berlin" c2="1844" c3="1821" />
        </tab>
 <tab name="filialinfo" h1="Ort" h2="Mitarbeiter" h3="Quadratmeter">
        <item c1="Filiale Berlin" c2="12" c3="320" />
        <item c1="Filiale Hamburg" c2="14" c3="366" />
        <item c1="Filiale New York" c2="25" c3="821" />
        <item c1="Filiale Essen" c2="7" c3="190" />
        <item c1="Filiale Köln" c2="9" c3="225" />
        </tab>
 <tab name="" />
 <tab name="" />
</sqllearn>;

var ex_A10_btn:Array = new Array();
ex_A10_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A10_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A10_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
ex_A10_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A10_btn.push(new MyButton(0x99FF00, 0x000000,"-"));
ex_A10_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A10_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A10_btn.push(new MyButton(0x99FF00, 0x000000,","));

var ex_A10_aufg="Hier ist noch eine Tabelle dazugekommen. Sie enthält die Anzahl der
Mitarbeiter jeder Filiale und die Größe der Filiale in
Quadratmetern.<br><br><b>Aufgabe 10</b><br>Erstelle eine Anfrage, die folgendes
Ergebnis liefert:<ul><li>Die Spalten sind Filialname, Anzahl der Mitarbeiter, die
Größe der Filiale und die Differenz aus den summierten Einkünfte und summierten
Ausgaben.</li><li>Die Ausgabe soll nach dem Filialnamen gruppiert sein.</li></ul>";

var ex_A10_hin="<br><b>Hinweis:</b><br><br>Die vorletzte Spalte enthält die Summe der
Einnahmen - minus - Summe der Ausgaben.<br>Beachte außerdem genau die Reihenfolge der
Spalten.";

var ex_A10_lsql="SELECT Ort , Mitarbeiter , Quadratmeter , SUM( Einnahmen )  - SUM(
Ausgaben )  FROM filialen NATURAL JOIN filialinfo GROUP BY Ort";

var ex_A10_ltext="<br>Du wirst immer besser! Bald ist es geschafft, aber ich brauche
noch etwas Hilfe ";

ex_mode.push(new Aufgabe(ex_A10_xml, ex_A10_btn, ex_A10_aufg, ex_A10_hin, ex_A10_lsql,
ex_A10_ltext));
//////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 10
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//Ex Mode - Aufgabe 11
//////////////////////////////////////////////////////
var ex_A11_xml:XML = <sqllearn>
<tab name="umfragepersoenlich" h1="Umfragebogen" h2="Lebensalter" h3="Geschlecht">
        <item c1="1" c2="12" c3="männlich" />
        <item c1="2" c2="11" c3="männlich" />
        <item c1="3" c2="18" c3="weiblich" />
        <item c1="4" c2="14" c3="männlich" />
        <item c1="5" c2="22" c3="weiblich" />
        <item c1="6" c2="19" c3="männlich" />
        <item c1="7" c2="17" c3="weiblich" />
        <item c1="8" c2="17" c3="männlich" />
        <item c1="9" c2="16" c3="weiblich" />
        <item c1="10" c2="15" c3="weiblich" />
        <item c1="11" c2="14" c3="männlich" />
        <item c1="12" c2="16" c3="männlich" />
        <item c1="13" c2="14" c3="weiblich" />
        <item c1="14" c2="19" c3="männlich" />
        <item c1="15" c2="12" c3="weiblich" />
        <item c1="16" c2="15" c3="männlich" />
        <item c1="17" c2="17" c3="weiblich" />
        <item c1="18" c2="17" c3="männlich" />
        <item c1="19" c2="19" c3="weiblich" />
        <item c1="20" c2="15" c3="weiblich" />
        <item c1="21" c2="21" c3="männlich" />
        <item c1="22" c2="18" c3="männlich" />
        <item c1="23" c2="16" c3="weiblich" />
        <item c1="24" c2="18" c3="männlich" />
        <item c1="25" c2="18" c3="weiblich" />
        <item c1="26" c2="19" c3="männlich" />
        <item c1="27" c2="21" c3="weiblich" />
        <item c1="28" c2="20" c3="männlich" />
        <item c1="29" c2="17" c3="weiblich" />
```

```
        <item c1="30" c2="12" c3="weiblich" />
        </tab>
 <tab name="umfrageinhalt" h1="Umfragebogen" h2="Taschengeld" h3="Kontostand">
        <item c1="1" c2="20" c3="125" />
        <item c1="2" c2="85" c3="0" />
        <item c1="3" c2="65" c3="354" />
        <item c1="4" c2="44" c3="1242" />
        <item c1="5" c2="60" c3="644" />
        <item c1="6" c2="20" c3="33" />
        <item c1="7" c2="35" c3="232" />
        <item c1="8" c2="17" c3="125" />
        <item c1="9" c2="40" c3="442" />
        <item c1="10" c2="150" c3="4772" />
        <item c1="11" c2="60" c3="1244" />
        <item c1="12" c2="80" c3="15" />
        <item c1="13" c2="70" c3="322" />
        <item c1="14" c2="30" c3="1242" />
        <item c1="15" c2="25" c3="644" />
        <item c1="16" c2="45" c3="33" />
        <item c1="17" c2="38" c3="232" />
        <item c1="18" c2="30" c3="125" />
        <item c1="19" c2="40" c3="442" />
        <item c1="20" c2="110" c3="772" />
        <item c1="21" c2="130" c3="1125" />
        <item c1="22" c2="85" c3="0" />
        <item c1="23" c2="18" c3="354" />
        <item c1="24" c2="80" c3="122" />
        <item c1="25" c2="92" c3="644" />
        <item c1="26" c2="49" c3="33" />
        <item c1="27" c2="26" c3="0" />
        <item c1="28" c2="147" c3="1250" />
        <item c1="29" c2="25" c3="142" />
        <item c1="30" c2="10" c3="472" />
        </tab>
 <tab name="" />
 <tab name="" />
</sqllearn>;

var ex_A11_btn:Array = new Array();
ex_A11_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A11_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A11_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A11_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
ex_A11_btn.push(new MyButton(0x99FF00, 0x000000,"AVG("));
ex_A11_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A11_btn.push(new MyButton(0x99FF00, 0x000000,","));

var ex_A11_aufg="Wir haben letzte Woche eine Umfrage durchgeführt, die noch
ausgewertet werden muss.<br><br><b>Aufgabe 11</b><br>Erstelle eine Anfrage,
die<ul><li>als Spalten das Geschlecht, durchschnittliches Alter, das durchschnittliche
Taschengeld im Monat und den durchschnittlichen Kontostand liefert.</li><li>Unterteile
nach männlich und weiblich!</li></ul>";

var ex_A11_hin="<br><b>Hinweis:</b><br><br>AVG steht für average=durchschnittlich und
wird so verwendet wie die SUM-Anweisung.";

var ex_A11_lsql="SELECT Geschlecht , AVG( Lebensalter )  , AVG( Taschengeld )  , AVG(
Kontostand )  FROM umfrageinhalt NATURAL JOIN umfragepersoenlich GROUP BY Geschlecht";

var ex_A11_ltext="<br>Genial! Per Hand hätte es ewig gedauert das zu berechnen. Stell
dir mal vor, wir hätten noch mehr Umfragebögen gehabt...";

ex_mode.push(new Aufgabe(ex_A11_xml, ex_A11_btn, ex_A11_aufg, ex_A11_hin, ex_A11_lsql,
ex_A11_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 11
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 12
//////////////////////////////////////////////////
var ex_A12_xml:XML = ex_A11_xml;

var ex_A12_btn:Array = new Array();
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"))
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"UNION"));
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"BETWEEN"));
ex_A12_btn.push(new MyButton(0xFF9900, 0x000000,"AND"));
```

```
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,"14"));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,"16"));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,"<13"));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,">17"));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,"AVG("));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A12_btn.push(new MyButton(0x99FF00, 0x000000,","));

var ex_A12_aufg="Ich brauche noch eine Liste, die bei der nach Altersgruppen sortiert
wird.<br><br><b>Aufgabe 12</b><br>Erstelle eine Anfrage, die<ul><li>als Spalte nur das
durchschnittliche Taschengeld für die Altersgruppen  unter 13, von 14-16 und über 16
ausgibt.</li><li>Die resultierende Ausgabe hat damit eine Spalte und genau drei
Zeilen.</li></ul>";

var ex_A12_hin="<br><b>Hinweis:</b><br>Beachte, dass du bei der mehrfachen Verwendung
der NATURAL JOIN-Anweisung immer die erste Tabelle mit der zweiten Tabelle verbindest,
damit die Lösung richtig erkannt wird.<br>Die SQL-Anfrage dieser Aufgabe ist sehr
lang.";

var ex_A12_lsql="SELECT AVG( Taschengeld )  FROM umfrageinhalt NATURAL JOIN
umfragepersoenlich WHERE Lebensalter <13 UNION SELECT AVG( Taschengeld )  FROM
umfrageinhalt NATURAL JOIN umfragepersoenlich WHERE Lebensalter BETWEEN 14 AND 16
UNION SELECT AVG( Taschengeld )  FROM umfrageinhalt NATURAL JOIN umfragepersoenlich
WHERE Lebensalter >17";

var ex_A12_ltext="<br>Wow, was für eine lange Anfrage. Das nächste Mal wirds wieder
kürzer, versprochen!";

ex_mode.push(new Aufgabe(ex_A12_xml, ex_A12_btn, ex_A12_aufg, ex_A12_hin, ex_A12_lsql,
ex_A12_ltext));
//////////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 12
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//Ex Mode - Aufgabe 13
//////////////////////////////////////////////////////
var ex_A13_xml:XML = <sqllearn>
<tab name="bankkunden" h1="Name" h2="Kontostand" h3="Institut">
  <item c1="Ulrich" c2="1200" c3="Deutsche Vereinsbank" />
  <item c1="Reinermann" c2="542" c3="Daten Bank" />
  <item c1="Laube" c2="2391" c3="Sparbank Westmuensterland" />
  <item c1="Oensel" c2="239" c3="Sparbank Westmuensterland" />
  <item c1="Peters" c2="844" c3="Hypo Unrealestate" />
  <item c1="Eismann" c2="1200" c3="Daten Bank" />
  <item c1="Schirmer" c2="420" c3="Daten Bank" />
  <item c1="Möllers" c2="2232" c3="Sparbank Westmuensterland" />
  <item c1="Wolter" c2="0" c3="Deutsche Vereinsbank" />
  <item c1="Kaminski" c2="1233" c3="Sparbank Westmuensterland" />
  </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var ex_A13_btn:Array = new Array();
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
ex_A13_btn.push(new MyButton(0xFF9900, 0x000000,"HAVING"));
ex_A13_btn.push(new MyButton(0x99FF00, 0x000000,">"));
ex_A13_btn.push(new MyButton(0x99FF00, 0x000000,","));
ex_A13_btn.push(new MyButton(0x99FF00, 0x000000,"SUM("));
ex_A13_btn.push(new MyButton(0x99FF00, 0x000000,") "));
ex_A13_btn.push(new MyButton(0x9900FF, 0x000000,"'1000'"));

var ex_A13_aufg="Jetzt hast du mir soviel bei meinen persönlichen Dingen geholfen,
dass ich glatt die letzten drei Aufgaben aus meinem Informatik-Buch vergessen habe...
<br><br><b>Aufgabe 13</b><br>Erstelle eine Anfrage, die<ul><li>als Spalten alle
Banken, sowie die dazugehörige Summe der Kontostände der Bank besitzt.</li><li>Es
sollen nur Banken aufgeführt werden, die einen Gesamtkontostand von >1000
besitzen.</li><li>Ordne absteigend nach dem Gesamtkapital.</li></ul>";

var ex_A13_hin="<br><b>Hinweis:</b> Vergiss nicht, dass der Kontostand mit SUM
aufsummiert werden muss (auch bei der Sortierung).</b><br>";

var ex_A13_lsql="SELECT Institut , SUM( Kontostand )  FROM bankkunden GROUP BY
Institut HAVING SUM( Kontostand )  > '1000' ORDER BY SUM( Kontostand )  DESC";
```

```
var ex_A13_ltext="<br>Jetzt haben wir es fast geschafft! Zwei Aufgaben noch.";

ex_mode.push(new Aufgabe(ex_A13_xml, ex_A13_btn, ex_A13_aufg, ex_A13_hin, ex_A13_lsql,
ex_A13_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 13
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 14
//////////////////////////////////////////////////
var ex_A14_xml:XML = ex_A13_xml;

var ex_A14_btn:Array = new Array();
ex_A14_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A14_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A14_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A14_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
ex_A14_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
ex_A14_btn.push(new MyButton(0x99FF00, 0x000000,">"));
ex_A14_btn.push(new MyButton(0x99FF00, 0x000000,","));
ex_A14_btn.push(new MyButton(0x99FF00, 0x000000,"AVG("));
ex_A14_btn.push(new MyButton(0x99FF00, 0x000000,"( "));
ex_A14_btn.push(new MyButton(0x99FF00, 0x000000,") "));

var ex_A14_aufg="<b>Aufgabe 14</b><br>Erstelle eine Anfrage, die<ul><li>als Spalten
den Namen des Kontoinhabers und den Kontostand ausgibt.</li><li>Es sollten nur
Kontostände angezeigt werden, die über dem überdurchschnittlich hoch sind.</li><li>Das
Ergebnis ist absteigend nach Kontostand zu sortieren.</li></ul>";

var ex_A14_hin="<br><b>Hinweis:</b><br>Hier soll man eine SELECT Anfrage
verschachteln, d.h. du erhälst den Wert, den du in der WHERE-Bedingung brauchst durch
eine weitere SELECT-Anfrage. Diese zweite Anfrage muss eingeklammert werden.";

var ex_A14_lsql="SELECT Name , Kontostand FROM bankkunden WHERE Kontostand > (  SELECT
AVG( Kontostand )  FROM bankkunden )  ORDER BY Kontostand DESC";

var ex_A14_ltext="<br>Prima, die letzte Aufgabe schaffst du auch noch!";

ex_mode.push(new Aufgabe(ex_A14_xml, ex_A14_btn, ex_A14_aufg, ex_A14_hin, ex_A14_lsql,
ex_A14_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 14
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//Ex Mode - Aufgabe 15
//////////////////////////////////////////////////
var ex_A15_xml:XML = ex_A13_xml;

var ex_A15_btn:Array = new Array();
ex_A15_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
ex_A15_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
ex_A15_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
ex_A15_btn.push(new MyButton(0xFF9900, 0x000000,"UNION"));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,">"));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,"="));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,"AVG("));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,"( "));
ex_A15_btn.push(new MyButton(0x99FF00, 0x000000,") "));


var ex_A15_aufg="Danke für deine Hilfe! Nur noch diese Aufgabe, dann ist es
geschafft!<br><br><b>Aufgabe 15</b><br>Erstelle eine Anfrage, die<ul><li>als Spalten
Name, den zugehörigen Kontostand und das Bankinstitut enthält.</li><li>Es sollten nur
Personen berücksichtigt werden, die überdurchschnittlich viel Geld haben oder Kunde
der Bank 'Deutsche Vereinsbank' sind.</li></ul>";

var ex_A15_hin="<br><b>Hinweis:</b><br>Beachte, dass der Teil, bei dem die Kunden
aufgenommen werden, deren Kontostand über dem durchschnittlichen Kontostand liegt,
zuerst abgefragt werden muss, damit die Lösung korrekt angenommen wird.";

var ex_A15_lsql="SELECT * FROM bankkunden WHERE Kontostand > (  SELECT AVG( Kontostand
)  FROM bankkunden )  UNION SELECT * FROM bankkunden WHERE Institut = 'Deutsche
Vereinsbank'";

var ex_A15_ltext="<br>Du hast alle Aufgaben gelöst! Herzlichen Glückwunsch!<br>Wenn du
noch etwas herumprobieren möchtest, empfehle ich dir den 'Freien Modus'.";
```

```
ex_mode.push(new Aufgabe(ex_A15_xml, ex_A15_btn, ex_A15_aufg, ex_A15_hin, ex_A15_lsql,
ex_A15_ltext));
//////////////////////////////////////////////////
//Ende Ex Mode - Aufgabe 15
//////////////////////////////////////////////////

var tut_mode = new Array();
//////////////////////////////////////////////////
//tut mode - Aufgabe 1
//////////////////////////////////////////////////
var tut_A1_xml:XML =
<sqllearn>
<tab name="tabelle1" h1="Vorname" h2="Name" h3="Lebensjahr">
  <item c1="Peter" c2="Schulz" c3="15" />
  <item c1="Julia" c2="Schmitt" c3="19" />
  <item c1="Frank" c2="Meier" c3="17" />
  <item c1="Jörg" c2="Pieper" c3="23" />
  <item c1="Thorsten" c2="Weber" c3="25" />
  <item c1="Maike" c2="Lohmann" c3="23" />
  <item c1="Janine" c2="Mueller" c3="15" />
  <item c1="Lisa" c2="Zöller" c3="23" />
  <item c1="Ingo" c2="Polster" c3="20" />
  <item c1="Rolf" c2="Reinermann" c3="45" />
  <item c1="Petra" c2="Feldmann" c3="37" />
  <item c1="Gerd" c2="Elskemper" c3="28" />
  <item c1="Sarah" c2="Vogt" c3="28" />
  <item c1="Katharina" c2="Ziegler" c3="49" />
  <item c1="Ewald" c2="Leitzen" c3="51" />
  <item c1="Lisa" c2="Artmann" c3="18" />
  <item c1="Tobias" c2="Gellmann" c3="17" />
  <item c1="Birgit" c2="Enseling" c3="16" />
  <item c1="Lukas" c2="Zarting" c3="18" />
  <item c1="Emil" c2="Terhart" c3="55" />
  <item c1="Frank" c2="Ostmann" c3="25" />
  <item c1="Heike" c2="Schmitt" c3="23" />
  </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;

var tut_A1_btn:Array = new Array();
tut_A1_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A1_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A1_btn.push(new MyButton(0x99FF00, 0x000000,"*"));

var tut_A1_aufg="<b>Willkommen im Einführungsmodus!</b><br><br>Eine Datenbank besteht
aus einer oder mehreren Tabellen.<br>Die Namen der Tabellen (z.B. 'tabelle1') kannst
du mit der Maus im Eingabefeld unten neben mir ablegen um sie in eine Abfrage
einzubauen.<br>Die Tabelle in dieser Aufgabe ist eine Liste mit meinen Freunden und
Bekannten.<br><br><b>Aufgabe 1</b><br>Erstelle eine Anfrage an die Datenbank, indem du
nacheinander die Befehle \"SELECT\", \"*\", \"FROM\", \"tabelle1\" in das Eingabefeld
ziehst und dann auf \"Anfrage senden\" klickst!<br><br>Wenn du die Aufgabe nach dem
Schließen der Sprechblase noch einmal sehen möchtest, dann klick mich einfach an!";

var tut_A1_hin="<br><b>Hinweis:</b><br>Bewege die Maus über einen der in der Aufgabe
genannten Befehle im rechten Teil des Fensters und halte die linke Maustaste gedrückt,
während du den Befehl in das Eingabefeld ziehst.<br><br>Beachte auch die Reihenfolge,
in der du die Befehle im unteren Fenster ablegst.<br><br>Wenn du die Aufgabenstellung
noch einmal sehen möchtest, schließe diese Sprechblase und klicke mich dann an.";

var tut_A1_lsql="SELECT * FROM tabelle1";

var tut_A1_ltext="<br>Super, du hast deine erste SQL-Anfrage richtig gestellt!<br>Die
<b>SELECT</b> Anweisung bedeutet, dass Daten aus der Datenbank abgerufen werden
sollen.<br>Der <b>*</b> steht dafür, dass sämtliche Spalten, hier also Vorname, Name
und das Lebensjahr, abgerufen werden sollen.<br><b>FROM tabelle1</b> bewirkt
schließlich, dass die Daten aus der Tabelle mit dem Namen \"tabelle1\" abgerufen
werden.<br><br><b>Klicke jetzt auf den Pfeil nach rechts, der am unteren Rand
erschienen ist, um zur nächsten Einführungsaufgabe zu gelangen!";

tut_mode.push(new Aufgabe(tut_A1_xml, tut_A1_btn, tut_A1_aufg, tut_A1_hin,
tut_A1_lsql, tut_A1_ltext));
//////////////////////////////////////////////////
//Ende tut mode - Aufgabe 1
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//tut mode - Aufgabe 2
//////////////////////////////////////////////////
var tut_A2_xml:XML = tut_A1_xml;
```

```
var tut_A2_btn:Array = new Array();
tut_A2_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A2_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A2_btn.push(new MyButton(0xFF0099, 0x000000,"Vorname"));

var tut_A2_aufg="Wenn du wissen willst, wie bestimmte SQL-Befehle verwendet werden,
kannst du die Befehle mit der gedrückten, linken Maustaste auf mein Bild ziehen. Ich
werde sie dir dann erklären!<br><br><b>Aufgabe 2</b><br><br>Manchmal braucht man nicht
alle Daten aus einer Tabelle, sondern nur einen Teil. Mit welcher SQL-Anfrage bekommen
wir nur die Spalte mit den Vornamen angezeigt?";

var tut_A2_hin="<br><b>Hinweis:</b><br>Dieses Mal wollen wir nicht alle Spalten (das
wäre der * nach SELECT) sondern nur die Spalte mit den Vornamen haben. Ansonsten sieht
die SQL-Anfrage aus wie in der ersten Einführungsaufgabe.<br><br>Nutze die Hilfe zu
den einzelnen SQL-Befehlen, die ich dir gebe, wenn du die Befehle mit der Maus auf
mich fallen lässt.";

var tut_A2_lsql="SELECT Vorname FROM tabelle1";

var tut_A2_ltext="<br>Es ist auch möglich mehrere Spalten ausgeben zu lassen, wenn die
Namen der Spalten nach SELECT jeweils mit einem \",\" voneinander getrennt werden.";

tut_mode.push(new Aufgabe(tut_A2_xml, tut_A2_btn, tut_A2_aufg, tut_A2_hin,
tut_A2_lsql, tut_A2_ltext ));
////////////////////////////////////////////////////
//Ende tut mode - Aufgabe 2
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut mode - Aufgabe 3
////////////////////////////////////////////////////
var tut_A3_xml:XML = tut_A1_xml;

var tut_A3_btn:Array = new Array();
tut_A3_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A3_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A3_btn.push(new MyButton(0x99FF00, 0x000000,","));

var tut_A3_aufg="Vielleicht ist dir eben aufgefallen, dass der Name der Spalte eine
andere Farbe (pink) hatte. Die Farben dienen nur zur besseren Unterscheidung, was ein
SQL-Befehl, ein Tabellenname (blau), eine Spalte (pink), ein Element einer Zeile
(lila) oder ein sonstiger Ausdruck (grün) ist. Du kannst jedes Element einer Tabelle
mit der Maus in eine SQL-Anfrage einbauen, auch wenn es nicht in der Liste mit den
Befehlen steht. Halte die linke Maustaste über der Tabelle gedrückt und probier es
aus!<br><br><b>Aufgabe 3</b><br><br>Versuch mal, die Spalte mit den Vornamen und die
Spalte mit den Lebensjahren gleichzeitig abzufragen!";

var tut_A3_hin="<br><b>Hinweis:</b><br>Wenn du nicht weiterkommst, ziehe mal den
SELECT-Befehl mit der Maus auf mich und lasse dann die Maustaste los. In der Erklärung
zum SELECT-Befehl findest du Hilfe.<br><br>An die Namen der Spalten kommst du, wenn du
sie mit der Maus von der Tabelle in das Eingabefeld unten neben mir ziehst.";

var tut_A3_lsql="SELECT Vorname , Lebensjahr FROM tabelle1";

var tut_A3_ltext="<br>Die Daten sind ziemlich durcheinander und sollten sortiert
werden...<br>Besonders bei größeren Datensätzen ist dies sinnvoll. Mehr dazu in der
nächsten Einführungsaufgabe.";

tut_mode.push(new Aufgabe(tut_A3_xml, tut_A3_btn, tut_A3_aufg, tut_A3_hin,
tut_A3_lsql, tut_A3_ltext ));
////////////////////////////////////////////////////
//Ende tut mode - Aufgabe 3
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut mode - Aufgabe 4
////////////////////////////////////////////////////
var tut_A4_xml:XML = tut_A1_xml;


var tut_A4_btn:Array = new Array();
tut_A4_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A4_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A4_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
tut_A4_btn.push(new MyButton(0xFF9900, 0x000000,"ASC"));
tut_A4_btn.push(new MyButton(0x99FF00, 0x000000,"*"));

var tut_A4_aufg="Das Ergebnis einer Datenbank-Anfrage kann sortiert ausgegeben werden.
Dazu muss an die Anfrage noch <b>ORDER BY</b>, gefolgt von dem Spaltennamen nach dem
sortiert werden soll angehängt werden. Zusätzlich folgt darauf <b>ASC</b> (für
aufsteigende Sortierung) oder <b>DESC</b> (für absteigende
```

```
Sortierung).<br><br><b>Aufgabe 4</b><br><br>Was muss ich nun machen um die Liste
meiner Freunde nach ihrem Alter aufsteigend zu sortieren?";

var tut_A4_hin="<br><b>Hinweis:</b><br>ORDER BY [Spaltenname] ASC für aufsteigende,
DESC für absteigende Sortierung.";

var tut_A4_lsql="SELECT * FROM tabelle1 ORDER BY Lebensjahr ASC";

var tut_A4_ltext="<br>Super! Mit einer sortierten Liste kann ich schon viel mehr
anfangen!";

tut_mode.push(new Aufgabe(tut_A4_xml, tut_A4_btn, tut_A4_aufg, tut_A4_hin,
tut_A4_lsql, tut_A4_ltext ));
////////////////////////////////////////////////////
//Ende tut mode - Aufgabe 4
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 5
////////////////////////////////////////////////////
var tut_A5_xml:XML =

<sqllearn>
<tab name="tabelle1" h1="Vorname" h2="Name" h3="Lebensjahr">
  <item c1="Peter" c2="Schulz" c3="15" />
  <item c1="Julia" c2="Schmitt" c3="19" />
  <item c1="Julia" c2="Schmitt" c3="19" />
  <item c1="Frank" c2="Meier" c3="17" />
  <item c1="Jörg" c2="Pieper" c3="23" />
  <item c1="Thorsten" c2="Weber" c3="25" />
  <item c1="Maike" c2="Lohmann" c3="23" />
  <item c1="Janine" c2="Mueller" c3="15" />
  <item c1="Janine" c2="Mueller" c3="15" />
  <item c1="Lisa" c2="Zöller" c3="23" />
  <item c1="Ingo" c2="Polster" c3="20" />
  <item c1="Rolf" c2="Reinermann" c3="45" />
  <item c1="Petra" c2="Feldmann" c3="37" />
  <item c1="Gerd" c2="Elskemper" c3="28" />
  <item c1="Sarah" c2="Vogt" c3="28" />
  <item c1="Sarah" c2="Vogt" c3="28" />
  <item c1="Katharina" c2="Ziegler" c3="49" />
  <item c1="Ewald" c2="Leitzen" c3="51" />
  <item c1="Ewald" c2="Leitzen" c3="51" />
  <item c1="Lisa" c2="Artmann" c3="18" />
  <item c1="Tobias" c2="Gellmann" c3="17" />
  <item c1="Birgit" c2="Enseling" c3="16" />
  <item c1="Lukas" c2="Zarting" c3="18" />
  <item c1="Emil" c2="Terhart" c3="55" />
  </tab>
  <tab name="" />
  <tab name="" />
  <tab name="" />
</sqllearn>;


var tut_A5_btn:Array = new Array();
tut_A5_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A5_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A5_btn.push(new MyButton(0xFF9900, 0x000000,"DISTINCT"));
tut_A5_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
tut_A5_btn.push(new MyButton(0xFF9900, 0x000000,"DESC"));
tut_A5_btn.push(new MyButton(0x99FF00, 0x000000,"*"));

var tut_A5_aufg="In meine Tabelle habe ich aus Versehen einige Namen doppelt
aufgenommen. Zum Glück gibt es die SQL-Anweisung \"DISTINCT\", die identische Zeilen
aussortiert. \"DISTINCT\" steht immer direkt nach der \"SELECT\"-
Anweisung.<br><br><b>Aufgabe 5</b><br>Wie kann ich nun die Liste ohne doppelte
Einträge ausgeben und sie dabei absteigend nach dem Nachnamen sortieren?";

var tut_A5_hin="<br>Ziehe die Befehle, die du nicht verstehst zu mir und lasse sie auf
mich fallen.";

var tut_A5_lsql="SELECT DISTINCT * FROM tabelle1 ORDER BY Name DESC";

var tut_A5_ltext="<br>Sehr schön. Alle doppelten Einträge sind verschwunden, wie du
sehen kannst - und sortiert wurde auch noch!";

tut_mode.push(new Aufgabe(tut_A5_xml, tut_A5_btn, tut_A5_aufg, tut_A5_hin,
tut_A5_lsql, tut_A5_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 5
```

```
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 6
////////////////////////////////////////////////////
var tut_A6_xml:XML = tut_A1_xml;

var tut_A6_btn:Array = new Array();
tut_A6_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A6_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A6_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
tut_A6_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
tut_A6_btn.push(new MyButton(0x99FF00, 0x000000,"="));

var tut_A6_aufg="Nachdem wir nun ganze Spalten abfragen können, probieren wir doch mal
aus, bestimmte Zeilen abzufragen. Benutze dazu den \"WHERE\"-Befehl. Wie dieser Befehl
funktioniert siehst du, wenn du ihn mit der Maus über mich ziehst und auf mich fallen
lässt.<br><br><b>Aufgabe 6</b><br>Wie kann ich alle Zeilen ausgeben, bei denen der
Vorname \"Frank\" lautet?";

var tut_A6_hin="<br>Du musst den \"WHERE\"-Befehl in deine Anfrage
einbauen.<br><br>Den Vornamen \"Frank\" erhältst du, wenn du den Namen selbst aus der
Tabelle in das Eingabefeld ziehst! Die Spaltenüberschrift \"Vorname\" erhältst du
ebenso.";

var tut_A6_lsql="SELECT * FROM tabelle1 WHERE Vorname = 'Frank'";

var tut_A6_ltext="<br>Wie du siehst, sind nicht viele Einträge übrig geblieben.";

tut_mode.push(new Aufgabe(tut_A6_xml, tut_A6_btn, tut_A6_aufg, tut_A6_hin,
tut_A6_lsql, tut_A6_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 6
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 7
////////////////////////////////////////////////////
var tut_A7_xml:XML = tut_A1_xml;

var tut_A7_btn:Array = new Array();
tut_A7_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A7_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A7_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
tut_A7_btn.push(new MyButton(0x99FF00, 0x000000,"OR"));
tut_A7_btn.push(new MyButton(0x99FF00, 0x000000,"*"));
tut_A7_btn.push(new MyButton(0x99FF00, 0x000000,"="));

var tut_A7_aufg="Die \"WHERE\"-Anweisung von eben können wir noch erweitern. Man kann
mehrere \"WHERE\"-Bedingungen z.B. mit \"AND\" oder \"OR\" verbinden - es müssen dann
beide Bedingungen bzw. eine der Bedingungen erfüllt sein, damit eine Zeile im Ergebnis
hinzugefügt wird.<br><br><b>Aufgabe 7</b><br>Wie kann ich alle Zeilen ausgeben, bei
denen der Vornamen \"Frank\" lautet ODER der Nachname \"Schmitt\" lautet?";

var tut_A7_hin="<br>Benutze dein Vorwissen aus der vorherigen Aufgabe! <br><br>Tipp:
WHERE spalte1 = 'abc' OR spalte2 = 'xyz'";

var tut_A7_lsql="SELECT * FROM tabelle1 WHERE Vorname = 'Frank' OR Name = 'Schmitt'";

var tut_A7_ltext="<br>Klasse! Nur weiter so!";

tut_mode.push(new Aufgabe(tut_A7_xml, tut_A7_btn, tut_A7_aufg, tut_A7_hin,
tut_A7_lsql, tut_A7_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 7
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 8
////////////////////////////////////////////////////
var tut_A8_xml:XML = tut_A1_xml;

var tut_A8_btn:Array = new Array();
tut_A8_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A8_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A8_btn.push(new MyButton(0xFF9900, 0x000000,"WHERE"));
tut_A8_btn.push(new MyButton(0xFF9900, 0x000000,"LIKE"));
tut_A8_btn.push(new MyButton(0x99FF00, 0x000000,","));
tut_A8_btn.push(new MyButton(0x9900FF, 0x000000,"'%mann'"));

var tut_A8_aufg="Eine \"WHERE\"-Anweisung kann als Bedingung auch z.B. \"Vorname LIKE
'%el'\" enthalten. Dann werden nur Zeilen berücksichtigt, bei denen in der Spalte
\"Vorname\" der jeweilige Name auf \"el\" endet (z.B. Michael oder
```

```
Daniel).<br><br><b>Aufgabe 8</b><br>Ich möchte nun eine Liste mit allen Personen,
deren Name auf \"mann\" endet haben. Dabei sollen nur die Spalten \"Name\" und
\"Lebensjahr\" ausgeben werden.";

var tut_A8_hin="<br>Versuche es mit der Hilfe zum \"LIKE\"-Befehl";

var tut_A8_lsql="SELECT Name , Lebensjahr FROM tabelle1 WHERE Name LIKE '%mann'";

var tut_A8_ltext="<br>Gleich wirst du sehen, dass ein Datenbanksystem auch rechnen
kann!";

tut_mode.push(new Aufgabe(tut_A8_xml, tut_A8_btn, tut_A8_aufg, tut_A8_hin,
tut_A8_lsql, tut_A8_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 8
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 9
////////////////////////////////////////////////////
var tut_A9_xml:XML = tut_A1_xml;

var tut_A9_btn:Array = new Array();
tut_A9_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A9_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A9_btn.push(new MyButton(0x99FF00, 0x000000,"SUM(Lebensjahr) "));

var tut_A9_aufg="SQL besitzt Anweisungen, die z.B. alle Einträge einer Spalte
aufsummieren können.<br><br><b>Aufgabe 9</b><br>Wähle als Spalte \"SUM(Lebensjahr)\"
von \"tabelle1\" und überlege dir, wie das Ergebnis zustande kommt.";

var tut_A9_hin="<br>Versuche mal: SELECT SUM(Lebensjahr) FROM tabelle1";

var tut_A9_lsql="SELECT SUM(Lebensjahr)  FROM tabelle1";

var tut_A9_ltext="<br>Das Ergebnis das du hier siehst ist dadurch entstanden, dass die
Lebensjahre jeder Person addiert wurden.";

tut_mode.push(new Aufgabe(tut_A9_xml, tut_A9_btn, tut_A9_aufg, tut_A9_hin,
tut_A9_lsql, tut_A9_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 9
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 10
////////////////////////////////////////////////////
var tut_A10_xml:XML = tut_A1_xml;

var tut_A10_btn:Array = new Array();
tut_A10_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A10_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A10_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
tut_A10_btn.push(new MyButton(0x99FF00, 0x000000,","));
tut_A10_btn.push(new MyButton(0x99FF00, 0x000000,"COUNT(Lebensjahr) "));

var tut_A10_aufg="Man kann eine Tabelle anhand der Attributwerte in Gruppen aufteilen:
Z.B. könnte man für jedes Alter eine Gruppe anlegen und die Personen in dieser Gruppe
zählen (COUNT).<br><br><b>Aufgabe 10</b><br>Erstelle die folgende SQL-Anfrage und
überlege dir, was sie genau bewirkt.<br>SELECT Lebensjahr, COUNT(Lebensjahr) FROM
tabelle1 GROUP BY Lebensjahr";

var tut_A10_hin="<br>Verwende die SQL-Anweisung aus der Aufgabe:<br>\"SELECT
Lebensjahr, COUNT(Lebensjahr) FROM tabelle1 GROUP BY Lebensjahr\"";

var tut_A10_lsql="SELECT Lebensjahr , COUNT(Lebensjahr)  FROM tabelle1 GROUP BY
Lebensjahr";

var tut_A10_ltext="<br>Als Spalten werden das Lebensjahr und die Anzahl der Personen,
die dieses Lebensjahr besitzen verwendet. \"GROUP BY Lebensjahr\" bewirkt, dass
gleiche Lebensjahre zusammengefasst werden. COUNT(Lebensjahr) gibt die Anzahl der
Leute in dieser Gruppe (mit dem gleichen Lebensjahr) an.";

tut_mode.push(new Aufgabe(tut_A10_xml, tut_A10_btn, tut_A10_aufg, tut_A10_hin,
tut_A10_lsql, tut_A10_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 10
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 11
////////////////////////////////////////////////////
var tut_A11_xml:XML = tut_A1_xml;
```

```
var tut_A11_btn:Array = new Array();
tut_A11_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A11_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A11_btn.push(new MyButton(0xFF9900, 0x000000,"GROUP BY"));
tut_A11_btn.push(new MyButton(0xFF9900, 0x000000,"HAVING"));
tut_A11_btn.push(new MyButton(0x99FF00, 0x000000,","));
tut_A11_btn.push(new MyButton(0x99FF00, 0x000000,"COUNT(Lebensjahr) "));
tut_A11_btn.push(new MyButton(0x99FF00, 0x000000,">1"));

var tut_A11_aufg="Häufig hat eine \"GROUP BY\"-Anweisung noch eine Bedingung dabei.
Die Bedingung sorgt dafür, dass z.B. nicht alle verschiedenen Lebensalter aufgelistet
werden, sondern z.B. nur die, die mindestens zweimal vorkommen. Die Bedingung wird mit
\"HAVING\" angegeben. Ziehe die \"HAVING\"-Anweisung auf mich und lasse die Maus los
um ein Beispiel zu erhalten.<br><br><b>Aufgabe 11</b><br>Liste die Lebensjahre und die
Anzahl der Vorkommen der Lebensjahre wie in der vorherigen Aufgabe auf. Dieses Mal
aber zusätzlich mit der Bedingung, dass das Lebensjahr bei mind. 2 Personen
vorkommt.";

var tut_A11_hin="<br>Wenn du nicht weiterkommst, schaue noch einmal in der vorherigen
Aufgabe, wie die Anweisung war. Hänge an diese Anweisung die \"HAVING\"-Bedingung an.
Diese wäre \"HAVING COUNT(Lebensjahr) >1\".";

var tut_A11_lsql="SELECT Lebensjahr , COUNT(Lebensjahr)  FROM tabelle1 GROUP BY
Lebensjahr HAVING COUNT(Lebensjahr)  >1";

var tut_A11_ltext="<br>Super! Das wäre geschafft, ab zur nächsten Aufgabe!";

tut_mode.push(new Aufgabe(tut_A11_xml, tut_A11_btn, tut_A11_aufg, tut_A11_hin,
tut_A11_lsql, tut_A11_ltext));
//////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 11
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//tut Mode - Aufgabe 12
//////////////////////////////////////////////////////
var tut_A12_xml:XML =
<sqllearn>
<tab name="tabelle1" h1="Id" h2="Vorname" h3="Name" h4="Lebensjahr">
  <item c1="1" c2="Peter" c3="Schulz" c4="15" />
  <item c1="2" c2="Julia" c3="Schmitt" c4="19" />
  <item c1="3" c2="Frank" c3="Meier" c4="17" />
  <item c1="4" c2="Jörg" c3="Pieper" c4="23" />
  <item c1="5" c2="Thorsten" c3="Weber" c4="25" />
  <item c1="6" c2="Maike" c3="Lohmann" c4="23" />
  <item c1="7" c2="Janine" c3="Mueller" c4="15" />
  <item c1="8" c2="Lisa" c3="Zöller" c4="23" />
  <item c1="9" c2="Ingo" c3="Polster" c4="20" />
  <item c1="10" c2="Rolf" c3="Reinermann" c4="45" />
  <item c1="11" c2="Petra" c3="Feldmann" c4="37" />
  <item c1="12" c2="Gerd" c3="Elskemper" c4="28" />
  <item c1="13" c2="Sarah" c3="Vogt" c4="28" />
  <item c1="14" c2="Katharina" c3="Ziegler" c4="49" />
  <item c1="15" c2="Ewald" c3="Leitzen" c4="51" />
  <item c1="16" c2="Lisa" c3="Artmann" c4="18" />
  <item c1="17" c2="Tobias" c3="Gellmann" c4="17" />
  <item c1="18" c2="Birgit" c3="Enseling" c4="16" />
  <item c1="19" c2="Lukas" c3="Zarting" c4="18" />
  <item c1="20" c2="Emil" c3="Terhart" c4="55" />
  <item c1="21" c2="Frank" c3="Ostmann" c4="25" />
  <item c1="22" c2="Heike" c3="Schmitt" c4="23" />
  </tab>
  <tab name="tabelle2" h1="Id" h2="Adresse" h3="Telefon" h4="Lieblingsessen">
  <item c1="1" c2="Waldweg 23" c3="98234"c4="Pizza" />
  <item c1="2" c2="Hauptstr. 33" c3="567566"c4="Schnitzel" />
  <item c1="3" c2="Seeweg 19" c3="78977"c4="Salat" />
  <item c1="4" c2="Seeweg 8" c3="5654546"c4="Pizza" />
  <item c1="5" c2="Wernerstr. 3" c3="855667"c4="Thunfisch" />
  <item c1="6" c2="Zentralplatz 12" c3="125654"c4="Braten" />
  <item c1="7" c2="Amselweg 11" c3="4831233"c4="Suppe" />
  <item c1="8" c2="Waldweg 23" c3="98234"c4="Spiegelei" />
  <item c1="9" c2="Wernerstr. 3" c3="4333455"c4="Spargel" />
  <item c1="10" c2="Zeckenweg 8" c3="333123"c4="Schnitzel" />
  <item c1="11" c2="Kreuzweg 4" c3="987878"c4="Pizza" />
  <item c1="12" c2="Windmühlenberg 30" c3="3345543"c4="Salat" />
  <item c1="13" c2="Eisenhütte 3" c3="121135"c4="Suppe" />
  <item c1="14" c2="Hermannsweg 15a" c3="669994"c4="Pizza" />
  <item c1="15" c2="Hauptstr. 33" c3="324545"c4="Eintopf" />
  <item c1="16" c2="Teichsmühle 12" c3="1231212"c4="Eintopf" />
  <item c1="17" c2="Bahnhofsstr. 1" c3="978544"c4="Grillfleisch" />
```

```
  <item c1="18" c2="Kinoplatz 23" c3="3554456"c4="Baguettes" />
  <item c1="19" c2="Badstr. 6" c3="13112123"c4="Pizza" />
  <item c1="20" c2="Seeweg 9" c3="887878"c4="Salat" />
  <item c1="21" c2="Vogelallee" c3="4545456"c4="Fisch" />
  <item c1="22" c2="Waldweg 23" c3="8778554"c4="Salat" />
  </tab>
  <tab name="" />
  <tab name="" />
</sqllearn>;

var tut_A12_btn:Array = new Array();
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Pizza'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Schnitzel'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Braten'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Suppe'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Eintopf'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Grillfleisch'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Baguettes'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Salat'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Fisch'"));
tut_A12_btn.push(new MyButton(0x9900FF, 0x000000,"'Spargel'"));

var tut_A12_aufg="Ich habe noch eine weitere Tabelle gefunden! Du kannst sie
anschauen, indem du auf \"tabelle2\" klickst. Die Spalte \"Id\" ist dazugekommen. Sie
ordnet jeder Person eine Nummer zu. Das ist wichtig, damit du erkennen kannst, welche
Daten aus der zweiten Tabelle zu welcher Person aus der ersten Tabelle
gehören.<br><br><b>Aufgabe 12</b><br>Finde heraus, was die Lieblingsspeise von Lisa
Artmann ist! ";

var tut_A12_hin="<br>Schaue in \"tabelle2\" nach, welche Zeile zu Lisa Artmann gehört.
Sie hat die gleiche Id wie in \"tabelle1\".";

var tut_A12_lsql="'Eintopf'";

var tut_A12_ltext="<br>Du hast verstanden, wie die beiden Tabellen
zusammenhängen.<br>Kümmere dich in diesem Fall nicht um die Rückmeldung des
Datenbanksystems.";

tut_mode.push(new Aufgabe(tut_A12_xml, tut_A12_btn, tut_A12_aufg, tut_A12_hin,
tut_A12_lsql, tut_A12_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 12
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 13
////////////////////////////////////////////////////
var tut_A13_xml:XML = tut_A12_xml;

var tut_A13_btn:Array = new Array();
tut_A13_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A13_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A13_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
tut_A13_btn.push(new MyButton(0x99FF00, 0x000000,"*"));

var tut_A13_aufg="Wir verknüpfen nun beide Tabellen miteinander. dafür gibt es die
Anweisung \"NATURAL JOIN\".<br><b>Aufgabe 13</b><br>Ziehe den Befehl \"NATURAL JOIN\"
über mich und lasse dann die Maus los. Du findest dort ein Beispiel. Verknüpfe dann
\"tabelle1\" mit \"tabelle2\".";

var tut_A13_hin="<br>Ziehe die Befehle, die du nicht verstehst zu mir und lasse sie
auf mich fallen.";

var tut_A13_lsql="SELECT * FROM tabelle1 NATURAL JOIN tabelle2";

var tut_A13_ltext="<br>Schau dir die entstandene Ausgabe-Tabelle noch einmal genau an!
Da beide Tabellen die Spalte \"Id\" hatten, können sie passend zusammengefügt
werden.";

tut_mode.push(new Aufgabe(tut_A13_xml, tut_A13_btn, tut_A13_aufg, tut_A13_hin,
tut_A13_lsql, tut_A13_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 13
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 14
////////////////////////////////////////////////////
var tut_A14_xml:XML = tut_A12_xml;

var tut_A14_btn:Array = new Array();
tut_A14_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
```

```
tut_A14_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A14_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
tut_A14_btn.push(new MyButton(0x99FF00, 0x000000,","));

var tut_A14_aufg="Wenn du zwei Tabellen miteinander verknüpft hast, werden sie so
behandelt als wären sie eine einzige Tabelle. Daher kannst du dann z.B. den Nachnamen
und die Telefonnummer aller Personen angeben, obwohl beides in verschiedenen Tabellen
abgelegt ist.<br><br><b>Aufgabe 14</b><br>Verbinde wieder die beiden Tabellen
miteinander und wähle als Spalten nur den Namen und die Telefonnummer aus.";

var tut_A14_hin="<br>Verwende die SQL-Anfrage mit \"NATURAL JOIN\" von gerade und
ersetze den \"*\" durch die Spaltennamen.";

var tut_A14_lsql="SELECT Name , Telefon FROM tabelle1 NATURAL JOIN tabelle2";

var tut_A14_ltext="<br>Super, du hast es fast geschafft - nun noch eine Aufgabe in der
vieles vorkommt was du bisher gelernt hast!";

tut_mode.push(new Aufgabe(tut_A14_xml, tut_A14_btn, tut_A14_aufg, tut_A14_hin,
tut_A14_lsql, tut_A14_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 14
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//tut Mode - Aufgabe 15
////////////////////////////////////////////////////
var tut_A15_xml:XML = tut_A12_xml;

var tut_A15_btn:Array = new Array();
tut_A15_btn.push(new MyButton(0xFF9900, 0x000000,"SELECT"));
tut_A15_btn.push(new MyButton(0xFF9900, 0x000000,"FROM"));
tut_A15_btn.push(new MyButton(0xFF9900, 0x000000,"NATURAL JOIN"));
tut_A15_btn.push(new MyButton(0xFF9900, 0x000000,"ORDER BY"));
tut_A15_btn.push(new MyButton(0xFF9900, 0x000000,"ASC"));
tut_A15_btn.push(new MyButton(0x99FF00, 0x000000,","));

var tut_A15_aufg="<b>Aufgabe 15</b><br>Ich brauche eine Liste mit den Spalten
\"Name\", \"Adresse\" und \"Telefon\". Diese Liste soll aufsteigend nach dem Namen
sortiert sein. Bitte helfe mir, sie zu erstellen!";

var tut_A15_hin="Ziehe die Befehle, die du nicht verstehst zu mir und lasse sie auf
mich fallen.";

var tut_A15_lsql="SELECT Name , Adresse , Telefon FROM tabelle1 NATURAL JOIN tabelle2
ORDER BY Name ASC";

var tut_A15_ltext="<br>Glückwunsch! Du hast alle Aufgaben aus dem Einführungsmodus
bestanden!";

tut_mode.push(new Aufgabe(tut_A15_xml, tut_A15_btn, tut_A15_aufg, tut_A15_hin,
tut_A15_lsql, tut_A15_ltext));
////////////////////////////////////////////////////
//Ende tut Mode - Aufgabe 15
////////////////////////////////////////////////////

//Hier werden die Erklärungen, die der Avatar zu den jeweiligen Befehlen hat angepasst
var avaexpl= new Array();
avaexpl['SELECT'] = "Die <b>SELECT</b> Anweisung steht für 'Wähle aus'. Wenn du
bestimmte Daten aus einer Tabelle abfragen möchtest, steht die SELECT-Anweisung
üblicherweise am Anfang der SQL-Anfrage.<br><br>Nach der Verwendung der SELECT-
Anweisung fragt man sich, was eigentlich ausgewählt werden soll. Das kann z.B. der
Name einer oder mehrerer Spalten (die jeweils mit einem \",\" voneinander getrennt
werden) sein oder das Zeichen '*' für alle Spalten.<br><br><b>Beispiele für eine
komplette Anweisung:</b><br><br><ul><li>SELECT spalte1 FROM tabelle1</li>(Diese
Anfrage zeigt nur spalte1 aus der Tabelle 'tabelle1'.)<br><br><li>SELECT * FROM
tabelle1</li>(Diese Anfrage zeigt alle Spalten aus der Tabelle
'tabelle1'.)</li></ul>";
avaexpl['DISTINCT'] = "Die <b>DISTINCT</b> Anweisung folgt direkt auf die 'SELECT'-
Anweisung und sorgt dafür, dass gleiche Zeilen im Ergebnis aussortiert
werden.<br><br><b>Beispiel für eine komplette Anweisung:</b><br><br><ul><li>SELECT
DISTINCT * FROM tabelle1</li>('tabelle1' ohne doppelte Zeilen wird ausgegeben.)</ul>";
avaexpl['FROM'] = "Mit <b>FROM</b> gibst du an, aus welcher Tabelle die Daten
abgefragt werden sollen. Der FROM-Befehl mit Tabellennamen steht nach der SELECT-
Anweisung und der Angabe was ausgewählt werden soll.<br><br>In manchen Aufgaben gibt
es mehrere Tabellen. Die blauen Elemente über dem Datenfeld zeigen jeweils die Namen
der Tabellen einer Aufgabe an.<br><br><b>Beispiel für eine komplette
Anweisung:</b><br><br><ul><li>SELECT * FROM tabelle1</li>(Hier wurde die Tabelle mit
dem Namen 'tabelle1' benutzt.)</li></ul>";
avaexpl['WHERE'] = "Der <b>WHERE</b> Befehl sorgt dafür, dass bei einer SELECT-Anfrage
nicht alle Zeilen ausgegeben werden, sondern nur die Zeilen, die eine bestimmte
```

```
Bedingung erfüllen.<br><br>WHERE steht für gewöhnlich nach der FROM-Anweisung und dem
zugehörigen Tabellennamen.<br><br>Die Bedingung, die zu der WHERE-Anweisung gehört
kann z.B. sein, dass nur Personen mit dem Namen 'Meier' angezeigt
werden.<br><br><b>Beispiele für eine komplette Anweisung:</b><br><br><ul><li>SELECT *
FROM tabelle1 WHERE name = 'Meier'</li>(Das zeigt nur die Zeilen an, wo die Spalte
name das Attribut 'Meier' besitzt.)<br><br><li>SELECT kontostand FROM tabelle1 WHERE
kontostand > 500</li>(Nur Kontostände werden aufgelistet, die größer als 500
sind.)</li></ul>";
avaexpl['ORDER BY'] = "<b>ORDER BY</b> sortiert eine Ausgabe. Du musst angeben nach
welcher Spalte sortiert werden soll und ob aufsteigend (ASC) oder absteigend (DESC)
sortiert werden soll.<br><br>Die ORDER BY-Anweisung mit den zugehörigen Angaben steht
immer ganz am Ende von einer kompletten Anfrage.<br><br><b>Beispiele für eine
komplette Anweisung:</b><br><br><ul><li>SELECT spalte1 FROM tabelle1 ORDER BY spalte1
ASC</li>(Das zeigt nur Spalte1 aus der Tabelle 'tabelle1' und sortiert sie
aufsteigend.)<br><br><li>SELECT * FROM tabelle1 ORDER BY spalte2 DESC</li>(Das zeigt
alle Spalten aus der Tabelle 'tabelle1' und sortiert absteigend nach
'spalte2'.)</li></ul>";
avaexpl['ASC'] = "<b>ASC</b> gehört zu der ORDER BY Anweisung und bedeutet, dass
aufsteigend sortiert wird. Siehe ORDER BY.";
avaexpl['DESC'] = "<b>DESC</b> gehört zu der ORDER BY Anweisung und bedeutet, dass
absteigend sortiert wird. Siehe ORDER BY.";
avaexpl['GROUP BY'] = "<b>GROUP BY</b> fügt gleiche Elemente einer Spalte zusammen.
Das macht Sinn, wenn du z.B. eine Tabelle mit den Spalten 'bank' und 'kontostand' hast
und die Kontostände jeder Bank zusammenrechnen möchtest.<br><br><b>Beispiel für eine
komplette Anweisung:</b><br><br><ul><li>SELECT bank, sum(kontostand) FROM tabelle1
GROUP BY bank</li>(Das zeigt eine Liste aller Banken mit der zugehörigen Summe der
Kontostände.)</li></ul>";
avaexpl['HAVING'] = "<b>HAVING</b> steht nach einer GROUP BY Anweisung und der dazu
zugehörigen Spaltenangabe. HAVING sorgt dafür, dass nur bestimmte Ergebnisse der
Gruppierung angezeigt werden (siehe GROUP BY). Die Bedingung ist so aufgebaut wie bei
einer WHERE Anweisung.<br><br><b>Beispiel für eine komplette
Anweisung:</b><br><br><ul><li>SELECT bank, sum(kontostand) FROM tabelle1 GROUP BY bank
HAVING sum(kontostand) > 1000</li>(Das zeigt eine Liste der Banken mit der zugehörigen
Summe der Kontostände, falls diese Summe größer als 1000 ist.)</li></ul>";
avaexpl['LIKE'] = "<b>LIKE</b> gehört meist zu einer WHERE Anweisung. Es sucht die
Zeilen heraus, in denen sich in der angegebenen Spalte etwas befindet, das in das nach
LIKE angegebene Muster passt.<br><br><b>Beispiele für eine komplette
Anweisung:</b><br><br><ul><li>SELECT name FROM tabelle1 WHERE name LIKE
'%mann%'</li>(Das zeigt alle namen aus der Tabelle 'tabelle1' die mit 'mann'
enthalten, z.B. Petermann oder Hermanns. Das '%' ist ein Platzhalter für beliebig
viele Zeichen.)<br><br><li>SELECT * FROM tabelle1 WHERE spalte1 LIKE 'Me_er'</li>(Der
'_' funktioniert als Platzhalter für genau ein Zeichen. Es werden also alle 'Meyer',
aber auch alle 'Meier' angezeigt.)</li></ul>";
avaexpl['*'] = "Der <b>*</b> bedeutet in Verbindung mit einer SELECT Anfrage, dass
sämtliche Spalten einer Tabelle ausgewählt werden. Siehe SELECT.";
avaexpl[','] = "Das <b>,</b> wird z.B. in einer SELECT Anfrage verwendet um mehrere
Spalten, die ausgewählt werden sollen, voneinander zu trennen.";
avaexpl['NATURAL JOIN'] = "<b>NATURAL JOIN</b> Verknüpft mehrere Tabellen über die
Gleichheit aller gleichlautenden Spalten. Gleichlautende Spalten werden im Ergebnis
nur einmal angezeigt. Gibt es keine gleichlautenden Spalten, wird jede Zeile der
ersten Tabelle mit jeder Zeile der zweiten Tabelle verbunden.<br><br><b>Beispiel für
eine komplette Anweisung:</b><br><br><ul><li>SELECT * FROM tabelle1 NATURAL JOIN
tabelle2</li>(Verbindet zwei Tabellen wie oben angegeben.)</li></ul>";
avaexpl['UNION'] = "<b>UNION</b> Verknüpft mehrere SELECT-Anfragen miteinander. Dabei
ist zu beachten, dass die Ergebnismenge jeder SELECT-Anweisung gleich viele Spalten
haben muss.<br><br><b>Beispiel für eine komplette Anweisung:</b><br><br><ul><li>SELECT
* FROM tabelle1 where Lebensjahr <15 UNION SELECT * FROM tabelle1 where Lebensjahr
>15</li>(Gibt als Ergebnis die gesamte Tabelle zurück außer wenn Lebensjahr = 15
ist.<br>Anmerkung: Dieses Beispiel ließe sich auf andere Weise geschickter
lösen.)</li></ul>";
avaexpl['BETWEEN'] = "Der <b>BETWEEN</b> Befehl ermöglicht es, z.B. in WHERE-
Anweisungen einen Bereich anzugeben. Alles was in diesem Bereich ist wird
berücksichtigt.<br><br><b>Beispiel für eine komplette
Anweisung:</b><br><br><ul><li>SELECT * FROM tabelle1 WHERE Lebensjahr BETWEEN 14 AND
19</li>(Das zeigt nur die Zeilen an, wo die Spalte Lebensjahr einen Wert zwischen 14
und 19 enthält.)</ul>";

//Aufgabenspezifische Erklärungen
avaexpl['Vorname'] = "<b>\"Vorname\"</b> ist kein SQL-Befehl als solches, sondern die
Bezeichnung einer Spalte von \"tabelle1\". Du kannst die Spaltennamen oder
Zelleninhalte auch direkt aus der Tabelle holen, indem du die linke Maustaste über dem
gewünschten Element der Tabelle gedrückt hältst und über dem Eingabefeld die Maustaste
loslässt.";

// Folgendes stammt aus dem im Projekt integrieren MovieClips „Auswahl“

var dialog = new Array();
//Dialog festlegen, Sprechblasen wechseln sich immer ab, Start beim linken Avatar
dialog.push("<font size=\"14\"><i><b>Tom und Tina treffen sich in der Pause. Du kennst
die beiden und bekommst ihr Gespräch mit.<br><br>Durch einfaches Klicken mit der
```

```
linken Maustaste auf Tom oder Tina gelangst du zum nächsten
Gesprächsabschnitt.</b></i></font>");
dialog.push("<font size=\"14\"><b>Hi Tina!</b><br><br>Wie gehts dir? Hast du auch
schon gehört welches Thema in Informatik kommen soll?<br><b>Datenbanken</b>... aber so
richtig kann ich damit nichts anfangen...</font>");
dialog.push("<font size=\"14\"><b>Hi Tom</b><br><br>Mir geht's super! Hab auch schon
gehört, dass das kommen soll. Soweit ich weiß benutzen wir jeden Tag Datenbanken, z.B.
im Internet bei der Suche nach einer Telefonnummer.</font>");
dialog.push("<font size=\"14\">Achso... Kann man sich das so vorstellen, dass jede
Person mit ihrer Telefonnummer zusammen in einer ganz langen Liste steht? Wenn ich
dann etwas suche wird automatisch das richtige Ergebnis angezeigt.</font>");
dialog.push("<font size=\"14\">Ja genau! Aber automatisch geht das nur, wenn sich
jemand die Mühe gemacht hat dazu etwas zu programmieren. Wenn du nur das
Datenbanksystem hast, musst du Befehle kennen um Daten abzufragen.</font>");
dialog.push("<font size=\"14\">Das müssen diese SQL-Befehle sein! Aber da gibt's ja
ziemlich viele... </font>");
dialog.push("<font size=\"14\">Keine Sorge, das machen wir schon! Vielleicht hilft uns
ja noch jemand!</font>");
dialog.push("<font size=\"14\"><i><b>Entscheide, ob du lieber Tom oder Tina helfen
möchtest, indem du ihn oder sie anklickst!</b><br>Du wirst ihm oder ihr zur Seite
stehen.<br><br><b>Entscheide dich dann für einen Modus. Im Einführungsmodus wird dir
viel geholfen, im Levelmodus vertrauen Tom bzw. Tina auf deine Fähigkeiten. Der 'Freie
Modus' ist zum experimentieren gedacht!</b></i></font>");

stop();
//gibt die aktuelle Position im Gespraech zwischen den Avataren an
var i;
//2. Bild fuer rechten Avatar anzeigen (Maedchen)
girl.gotoAndStop(2);
//Handler
boy.addEventListener(MouseEvent.CLICK, nextBubble);
girl.addEventListener(MouseEvent.CLICK, nextBubble);
btn_back.addEventListener(MouseEvent.CLICK, btnback);

////////////////////////////////////////////////
//Wechselt zum vorherigen gesprächsabschnitt
////////////////////////////////////////////////
function btnback(event:MouseEvent):void {
	if (i > 0) {
		boy.removeEventListener(MouseEvent.CLICK, choseAvatar);
		girl.removeEventListener(MouseEvent.CLICK, choseAvatar);
		star.visible=false;
		btn_tut.visible = false;
		btn_ex.visible = false;
		btn_free.visible = false;
		//vorheriger Gesprächsabschnitt
		i=i-2;
		//Falls durch den Zurück-Klick im ersten Bild gelandet: Zurück-Button
ausblenden
		if (i<=0) { btn_back.visible = false; }
		//richtige Sprechblase anzeigen
		if (i!=0){
			if ((i%2) == 0){
				gotoAndStop(3);
				} else {
				gotoAndStop(2);
			}
		}else{ gotoAndStop(4); }
	}
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

////////////////////////////////////////////////
//Dialog zuruecksetzen falls in die Auswahl zurueckgewechselt werden muss
//m=0 => Programmstart, sonst nur Modus neu wählen
////////////////////////////////////////////////
function initauswahl(m:int):void {
 if (m==0){
	star.visible=false;
	btn_tut.visible = false;
	btn_ex.visible = false;
	btn_free.visible = false;
	btn_back.visible = false;
	i=0;
	gotoAndStop(4);
 }else{
	star.visible=true;
	btn_tut.visible = true;
```

```
        btn_ex.visible = true;
        btn_free.visible = true;
        btn_back.visible = true;
        i=dialog.length-1;
        gotoAndStop(1);
        gotoAndStop(4);
 }
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

////////////////////////////////////////////////
//Avatar auswaehlen durch Anklicken
////////////////////////////////////////////////
function choseAvatar(event:MouseEvent):void {
        //Auswahl des Avatars ermoeglichen
        if (event.target.name=="boy") {
                event.target.root.ava.gotoAndStop(1);
                event.target.root.thumb.x = 15;
                event.target.root.thumb.y = 322;
                star.x=0;
                star.visible=true;
        } else {
                event.target.root.ava.gotoAndStop(2);
                event.target.root.thumb.x = 67;
                event.target.root.thumb.y = 322;
                star.x=255;
                star.visible=true;
        }
        btn_tut.visible = true;
        btn_ex.visible = true;
        btn_free.visible = true;
        btn_back.visible = true;
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

////////////////////////////////////////////////
//Handler fuer Klick auf den Hintergrund oder auf x von einer Bubble
////////////////////////////////////////////////
function nextBubble(event:MouseEvent):void {
        if (i < dialog.length) {
                if ((i%2) == 0) {
                        if ((i == 0) || (i==dialog.length-1)) {
                                boy.addEventListener(MouseEvent.CLICK, choseAvatar);
                                girl.addEventListener(MouseEvent.CLICK, choseAvatar);
                                gotoAndStop(4);
                        } else {
                                gotoAndStop(3);
                        }
                } else {
                        if (i==dialog.length-1) {
                                boy.addEventListener(MouseEvent.CLICK, choseAvatar);
                                girl.addEventListener(MouseEvent.CLICK, choseAvatar);
                                gotoAndStop(4);
                        } else {
                                gotoAndStop(2);
                        }
                }
        }
        btn_back.visible = true;
}
////////////////////////////////////////////////
//
////////////////////////////////////////////////

stop();
boybubble.btn_close.addEventListener(MouseEvent.CLICK, nextBubble);
boytext.verticalScrollPosition = 0;
boytext.htmlText = dialog[i];
i+=1;

stop();
girlbubble.btn_close.addEventListener(MouseEvent.CLICK, nextBubble);
girltext.verticalScrollPosition = 0;
girltext.htmlText = dialog[i];
i+=1;
```

```
stop();
kasten.btn_close.addEventListener(MouseEvent.CLICK, nextBubble);
kastentext.verticalScrollPosition = 0;
kastentext.htmlText = dialog[i];
i+=1;

////////////////////////////////////////////////
//Klasse "MyButton"
////////////////////////////////////////////////
package klassen{

import flash.text.TextField;
import flash.text.TextFormat;
import flash.display.MovieClip;

public class MyButton extends MovieClip{

       private var txt:TextField;
       private var format:TextFormat;
       private var offsetX:int;
       private var offsetY:int;

       public var label:String;
       public var colorBg:uint;
       public var colorTx:uint;

 public function MyButton(colorBack:uint, colorText:uint, labelButton:String):void
   {
        label = labelButton;
  colorBg = colorBack;
  colorTx = colorText;

  //Abstände zwischen Textfeld und Rahmen festlegen
       offsetX = 15;
       offsetY = 1;

       //Textformat erstellen
       format = new TextFormat;
       format.font = "Arial";
       format.size = 12;
       format.bold = true;
       format.color = colorTx;

       //Textfeld erstellen und definiertes Textformat benutzen
       txt = new TextField();
       txt.autoSize ="center";
       txt.text = label;
       txt.setTextFormat(format);
       txt.selectable = false;
       txt.x = offsetX;
       txt.y = offsetY;

       //Rechteck mit abgerundeten Ecken zeichnen
       graphics.lineStyle(2,0x000000,1);
       graphics.beginFill(colorBg,1);
       graphics.drawRoundRect(0,0,txt.width+(offsetX*2),txt.height+(offsetY*2),10);
       graphics.endFill();

       //Textfeld einfügen
       addChild( txt );
       //Damit das Textfeld beim Verschieben des Buttons nicht ausgewählt werden kann
       mouseChildren = false;
       //Gesamtgroesse verringern
       this.width = this.width*0.8;
       this.height = this.height*0.8;
   }
}
}

////////////////////////////////////////////////
//Klasse "Aufgabe"
////////////////////////////////////////////////

package klassen{

public class Aufgabe extends Object{

               public var xml:XML;
               public var buttons:Array;
               public var aufgabe:String;
```

```
                public var hinweis:String;
                public var loesungSQL:String;
                public var loesungstext:String;

   public function Aufgabe( xml_p:XML, buttons_p:Array, aufgabe_p:String,
hinweis_p:String, loesungSQL_p:String, loesungstext_p:String):void
   {
               xml = xml_p;
               buttons = buttons_p;
               aufgabe = aufgabe_p;
               hinweis = hinweis_p;
               loesungSQL = loesungSQL_p;
               loesungstext = loesungstext_p;
   }
}
}

////////////////////////////////////////////////
// PHP-Schnittstelle "sqllearner.php"
////////////////////////////////////////////////

<?php

$querystring=$_POST["query"];
$userid=$_POST["userid"];
$xmldata = $_POST["startdata"];

//Ersten Zustand in der DB erstellen
function startzustandHerstellen($userid, $xmldata){

        require 'global.php';

        $xml = str_replace("\\", "", $xmldata);

        // Neues DomDocument
        $dom = new DOMDocument('1.0', 'iso-8859-1');
        $dom->preserveWhiteSpace = FALSE;

        // Lade Daten und parse
        $dom->loadXML($xml);

        $db = @MYSQL_CONNECT($mysqlhost,$mysqluser,$mysqlpassword) or die ("Konnte
keine Verbindung zur Datenbank herstellen");
        //Falls Datenbank noch nicht existiert => anlegen
        $createdb = mysql_query("CREATE DATABASE IF NOT EXISTS ".$mysqldb);

        mysql_select_db($mysqldb,$db);

        //Loesche alle Tabellen von diesem User aus vorherigen Sitzungen
        $todelete = mysql_query("SHOW TABLES LIKE '%$userid'");
        while ($row = mysql_fetch_array($todelete)) {
                mysql_query("DROP TABLE $row[0]");
        }

        //mysql_query("DROP TABLE LIKE '%$userid'");

        //fuer alle 3 moeglichen Tabellen
        for($i=0;$i<3;$i++) {

        $dbname=$dom->documentElement->getElementsByTagName('tab')->item($i)-
>getAttribute('name');
        $spalten = $dom->documentElement->getElementsByTagName('tab')->item($i);
        $zeilen = $dom->documentElement->getElementsByTagName('tab')->item($i)-
>getElementsByTagName('item');
        $sp="";
        $zl=array();

         foreach ($spalten->attributes as $attribute){
                if ($attribute->name != "name"){
                        array_push($zl, $attribute->value);
                        $sp = $sp."`".$attribute->value."` VARCHAR( 150 ) NOT NULL ,";
                }
         }
        //letztes ',' loeschen
        $sp = substr($sp, 0, strrpos($sp, ','));

        //Tabelle neu erstellen
        if ($dbname!=""){
                //Falls noch eine Datenbank vom gleichen Benutzer existiert => Loeschen
                mysql_query("DROP TABLE IF EXISTS `$mysqldb`.`$dbname$userid`");
```

```
                mysql_query("CREATE TABLE `$mysqldb`.`$dbname$userid` (".$sp.") ENGINE
= MYISAM");

                //Inhalt der Tabelle ermitteln
                foreach($zeilen as $items){
                        $j=0; $n = ""; $v = "";
                        foreach ($items->attributes as $attribute){
                                $n=$n.$zl[$j].","; $j++;
                                $v=$v."'".$attribute->value."',";
                        }
                $n = substr($n, 0, strrpos($n, ','));
                $v = substr($v, 0, strrpos($v, ','));
                mysql_query("INSERT INTO `$mysqldb`.`$dbname$userid` (".$n.") VALUES
(".$v.")");
                //print "INSERT INTO `$mysqldb`.`$dbname$userid` (".$n.") VALUES
(".$v.")";
                }
        }
        }
$db_close = @MYSQL_CLOSE($db);
}

//gibt ein Array mit den Namen aller tabellen einer userid zurueck
function getTableData($userid){
        require 'global.php';

        $db = @MYSQL_CONNECT($mysqlhost,$mysqluser,$mysqlpassword) or die ("Konnte
keine Verbindung zur Datenbank herstellen");
        mysql_select_db($mysqldb,$db);

        $tabname=Array();
        $alltables = mysql_list_tables($mysqldb);
        while ($row = mysql_fetch_row($alltables)) {
                if (strpos($row[0], $userid)!=false) {
                        array_push($tabname, substr($row[0], 0, strpos($row[0],
$userid)));
                }
        }
        if (count($tabname)==2) { array_push($tabname, ""); }
        if (count($tabname)==1) { array_push($tabname, ""); array_push($tabname, "");}
        if (count($tabname)==0) { array_push($tabname, ""); array_push($tabname, "");
array_push($tabname, "");}
        //$db_close = @MYSQL_CLOSE($db);

return $tabname;
}

function getAll($userid, $query){
        require 'global.php';

        //vorhandene tabellennamen aus DB ermitteln *$userid
        $tabname=Array();
        $tabname=getTableData($userid);


        //Datenbankzugriff: uebergebener SQL-String
        $db = @MYSQL_CONNECT($mysqlhost,$mysqluser,$mysqlpassword) or die ("Konnte
keine Verbindung zur Datenbank herstellen");
        mysql_select_db($mysqldb,$db);

        //query durchsuchen nach Tabellennamen. falls welche gefunden werden => id
anhaengen.
        for($i=0;$i<count($tabname);$i++){
                $query = str_replace($tabname[$i], $tabname[$i].$userid, $query);
        }
        $query = str_replace("\'", "'", $query);

        $myquery = mysql_query($query);

        //moegliche Fehler speichern
        $sqlerrno = mysql_errno($db);
        $sqlerror = mysql_error($db);

        //Tabellennamen neu einlesen fuer den Fall dass sich etwas veraendert hat
        $tabname=getTableData($userid);

        // Neues DomDocument
        $dom = new DOMDocument('1.0', 'iso-8859-1');
        $dom->preserveWhiteSpace = FALSE;
        //Wurzel
```

```
	$root = $dom->createElement('sqllearn');
	$root = $dom->appendChild($root);

	//fuer alle 3 moeglichen Tabellen
	for($i=0;$i<3;$i++) {
		if ($tabname[$i] != ""){
			//Abfrage fuer [tab]
			$data=mysql_query("SELECT * FROM $tabname[$i]$userid");
			if (mysql_num_rows($data)>0){
				$row = mysql_fetch_assoc($data);

				$outer = $dom->createElement('tab');
				$outer = $root->appendChild($outer);
				$outer->setAttribute('name',$tabname[$i]);
				$j=1;
				foreach ($row as $fieldname => $fieldvalue) {
					$outer->setAttribute('h'.$j,$fieldname); $j++;
				}
				//neue Abfrage fuer die [item]-Bloecke
				$data=mysql_query("SELECT * FROM $tabname[$i]$userid");

				// für alle Items (Zeilen in der DB)
				while ($row = mysql_fetch_assoc($data)) {
					// add node for each record
					$inner = $dom->createElement('item');
					$inner = $outer->appendChild($inner);
					// Fuege als Attribute hinzu
					$j=1;
					foreach ($row as $fieldname => $fieldvalue) {
					$inner->setAttribute('c'.$j,$fieldvalue); $j++;
					}
				}
			//Falls die Tabelle keine Daten enthaelt
			}else{
				$data = mysql_query("SHOW COLUMNS FROM
$tabname[$i]$userid");

				$nameArray=Array();
				while ($row = mysql_fetch_assoc($data)) {
					array_push($nameArray, $row["Field"]);
				}
				$outer = $dom->createElement('tab');
				$outer = $root->appendChild($outer);
				$outer->setAttribute('name','');
				$outer->setAttribute('name',$tabname[$i]);
				for ($k=0;$k<count($nameArray);$k++){
					$nexth = $k+1;
					$outer->setAttribute("h".$nexth,$nameArray[$k]);
				}
			}
		//Falls Tabelle nicht existiert
		}else{
			$outer = $dom->createElement('tab');
			$outer = $root->appendChild($outer);
			$outer->setAttribute('name','');

		}

	}

	$outer = $dom->createElement('tab');
	$outer = $root->appendChild($outer);
	$outer->setAttribute('name','Ausgabe');

////////
	//Erfolgreiche Anfrage
	if ($myquery == true){
		//SELECT anweisungen
		if ((substr($query,0,6)=="SELECT") | (substr($query,0,4)=="SHOW")){
		    if (mysql_num_rows($myquery)>0){
			$myrow = mysql_fetch_assoc($myquery);
			$j=1;
			foreach ($myrow as $fieldname => $fieldvalue) {
				$outer->setAttribute('h'.$j,$fieldname); $j++;
			}
			//Abfrage erneuern, damit alle Rows durchlaufen werden
			$myquery = mysql_query($query);
			// process all rows of the inner/many/child table
			while ($myrow = mysql_fetch_assoc($myquery)) {
				// Items hinzufuegen
```

```
                                $inner = $dom->createElement('item');
                                $inner = $outer->appendChild($inner);
                                // Attribute für die Items hinzufuegen
                                $j=1;
                                foreach ($myrow as $fieldname => $fieldvalue) {
                                $inner->setAttribute('c'.$j,$fieldvalue); $j++;
                                }
                            }
                        }else{
                            //Leerer Datensatz durch SELECT Anfrage
                            $sqlerror="MySQL lieferte ein leeres Resultat zurueck. (d.h.
Null Zeilen)";
                            $outer->setAttribute('h1','Ausgabe');
                            $inner = $dom->createElement('item');
                            $inner = $outer->appendChild($inner);
                            $inner->setAttribute('c1',$sqlerror);

                            }
                    }else{
                    //DELETE / INSERT / REPLACE / UPDATE Anweisung
                    if ($sqlerror == "") $sqlerror="Die Anfrage wurde erfolgreich
ausgefuehrt.";
                    $outer->setAttribute('h1','Ausgabe');
                    $inner = $dom->createElement('item');
                    $inner = $outer->appendChild($inner);
                    $inner->setAttribute('c1',$sqlerror);
                    }
            }else{
            //Fehler oder keine Berechtigung
            if ($sqlerrno == "") $sqlerrno="0";
            if ($sqlerror == "") $sqlerror="Die Anfrage wurde ausgefuehrt";
            $outer->setAttribute('h1','Nr');
            $outer->setAttribute('h2','Fehler');
            $inner = $dom->createElement('item');
            $inner = $outer->appendChild($inner);
            $inner->setAttribute('c1',$sqlerrno);
            $inner->setAttribute('c2',$sqlerror);
            }

$db_close = @MYSQL_CLOSE($db);

print $dom->saveXML();

}


if ($_POST["meth"]=="startdata"){ startzustandHerstellen($userid, $xmldata); }

if ($_POST["meth"]=="getall"){ getAll($userid, $querystring); }

?>
```

VIII Erklärung zur Selbstständigkeit

Ich versichere, dass ich die schriftliche Hausarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Das Gleiche gilt auch für die beigegebenen Zeichnungen, Kartenskizzen und Darstellungen.

_____

Münster, den 20.07.2009