eTutor

Modularchitektur und Moduldokumentation

SQL

Georg Nitsche

Version 1.2
Stand März 2006

INSTITUT FÜR WIRTSCHAFTSINFORMATIK
dke
Data & Knowledge Engineering

Versionsverlauf:

Version	***Autor***	***Datum***	***Änderungen***
1.0	gn	06.03.2005	Fertigstellung der ersten Version
1.1	gn	07.03.2005	Umfangreiche Ergänzungen
1.2	gn	10.03.2005	Korrektur

Inhaltsverzeichnis:

Abbildungsverzeichnis:

Tabellenverzeichnis:

Kurzfassung

Das eTutor-System ist eine internetbasierte Plattform, über die Studenten Übungsbeispiele zu Aufgabengebieten ortsunabhängig bearbeiten, analysieren lassen und abgeben können. Während der Kern des eTutor-Systems die Grundfunktionalitäten zur Verfügung stellt, werden die einzelnen Aufgabengebiete als Module in das Kernsystem integriert. Das SQL-Modul steuert dabei die Funktionalitäten zur Bearbeitung von SQL-Übungsbeispielen bei. Die Umsetzung dieser Funktionalitäten und die Systemarchitektur des SQL-Moduls werden in diesem Dokument näher beschrieben.

1. Einleitung

Das SQL-Modul ermöglicht im eTutor-System die Bearbeitung von Übungsbeispielen, bei denen von Studenten SQL-Abfragen zu formulieren sind. Diese werden durch das SQL-Modul ausgeführt und bewertet. Zusätzlich wird Assistenten die Möglichkeit zu Verfügung gestellt, Übungsbeispiele zum Aufgabengebiet SQL zu spezifizieren.

Diese Anforderungen werden vom Kernsystem eTutor-Systems, dem eTutor Core, als Schnittstellen definiert und vorgegeben. Für eine erfolgreiche Integration in das eTutor-System müssen diese Schnittstellen durch das Modul umgesetzt werden.

Die Anforderungen, die vom eTutor-System für die Integration eines Moduls in das eTutor-System vorgegeben werden, sowie deren Umsetzung im SQL-Modul werden in Kapitel 2 präzisiert. Kapitel 3 widmet sich der Systemstruktur des SQL-Moduls, das Datenbankschema für die persistente Speicherung von Übungsaufgaben wird in Kapitel 4 behandelt. Abschließend werden in Kapitel 5 die Konfigurationsmöglichkeiten des Moduls dokumentiert.

2. Funktionsweise

Dieses Kapitel befasst sich mit den Anforderungen, die vom eTutor-System für die Integration von Modulen allgemein vorgegeben werden, sowie die spezielle Ausprägung dieser Anforderungen für das SQL-Modul.

Für die Integration in das eTutor-System muss das SQL-Modul zwei grundsätzliche Funktionalitäten unterstützten, einerseits die Auswertung von SQL-Abfragen, die von Studenten zu einem Übungsbeispiel formuliert werden, und andererseits die Spezifikation von Übungsbeispielen durch Assistenten.

Die Kernfunktionalitäten, die für die Bearbeitung durch Studenten zur Verfügung gestellt werden müssen, lassen sich folgendermaßen zusammenfassen:

- Analyse von SQL-Abfragen,
- Bewertung von SQL-Abfragen, sowie
- entsprechende Rückmeldungen an Studenten.

Die Funktionalitäten für die Spezifikation von Übungsbeispielen durch Assistenten lässt sich vor allem durch die folgenden Punkte charakterisieren:

- Erzeugen eines neuen SQL-Übungsbeispiels,
- Ändern eines bestehenden SQL-Übungsbeispiels,
- Abrufen eines bestehenden SQL-Übungsbeispiels, sowie
- Löschen eines bestehenden SQL-Übungsbeispiels.

Der generelle Ablauf bei der Benutzung eines Moduls durch Studenten wird in Abbildung 2.1 dargestellt: Ein Student wählt über einen Internet-Browser eine konkrete Übungsaufgabe aus, die ihm im Rahmen einer Lehrveranstaltung zugeteilt wurde. Die Anzeige wird dabei von einer Komponente des eTutor Core übernommen, die in der Abbildung als *showTask.jsp* bezeichnet wird. Für die Bearbeitung der Aufgabe muss ein in das eTutor-System integriertes Modul eine Eingabemaske bereitstellen, über die Lösungen eingegeben werden können. Im Falle des SQL-Moduls ist dies die Seite *showEditor.jsp*, über die SQL-Abfragen zu einem Übungsbeispiel eingegeben werden können. Durch den eTutor Core werden in dieser Phase Informationen bereitgestellt, die für die Anzeige und alle folgenden Aufgaben, die bei der Ausarbeitung von Lösungen durch das Modul unterstützt werden, von Bedeutung sind. Diese Informationen werden als Attribute im Session-Kontext gespeichert, wo sie für das jeweilige Modul verfügbar sind. Bei der Anzeige der Eingabemaske zu einem Übungsbeispiel, so wie bei den folgenden Schritten (Ausführung einer Lösung, Bewertung, Anzeige des Feedbacks) sind im speziellen für das SQL-Modul die folgenden Attribute relevant:

- *exerciseID*: Dieses Attribut dient dazu, die modulspezifischen Informationen zu einer Übungsaufgabe, die in der vom Modul verwalteten Datenhaltung gespeichert sind, zu identifizieren.
- *taskID*: Dieses Attribut identifiziert im eTutor-System eine Aufgabe, die zu einem Studenten zugeteilt ist. Im Rahmen einer Aufgabe wird ein Übungsbeispiel, identifiziert durch die *exerciseID*, aus dem Beispielpool des eTutor-Systems zu einem Studenten zugeteilt.
- *userID*: Durch dieses Attribut lässt sich ein Benutzer des eTutor-Systems eindeutig identizieren.
- *actions*: Mit diesem Attribut wird festgelegt, welche Möglichkeiten dem Studenten in der Eingabemaske für die Ausführung einer SQL-Abfrage zur Verfügung gestellt werden. Der eTutor Core definiert in diesem Attribut vom Typ *java.util.Set* alle oder nur eine Teilmenge der folgenden Werte: *run*, *check*, *diagnose* und *submit*. Das SQL-Modul interpretiert diese Werte den Spezifikationen für die Integration in das eTutor-Systems entsprechend und zeigt in der Eingabemaske nur diejenigen Ausführungsmöglichkeiten an, die in diesem Attribut enthalten sind: Die Ausführung einer SQL-Abfrage ohne jegliche Analysen und Bewertungen (*run*), eine kurze Überprüfung, ob die ausgeführte Lösung korrekt ist (*check*), umfangreichere Analysen von eventuellen Fehlern (*diagnose*), sowie die Abgabe einer Lösung *(submit*). Bei der Diagnose werden vom SQL-Modul zusätzlich mehrere Diagnosestufen unterschieden.

Wird eine in der Eingabemaske formulierte SQL-Abfrage ausgeführt, so werden die Eingaben durch das SQL-Modul verarbeitet und die Kontrolle an das *CoreManagerServlet* des eTutor-Systems übergeben. Dabei wird im Request das Attribut *action* festgelegt, das angibt, welcher der oben angeführten Ausführungsmöglichkeiten gewählt wurde. Das *CoreManagerServlet* ruft nun die Analyse-, Bewertungs- und Feedback-Funktionalitäten des SQL-Moduls auf. Diese befinden sich in einer Klasse, die das eTutor-Interface *etutor.core.evaluation.Evaluator* implementiert. Im Falle des SQL-Moduls wird dieses Interface durch die Klasse *etutor.modules.sql.SQLEvaluator* implementiert. Die Ergebnisse werden für den Benutzer letztendlich auf einer Seite *printReport.jsp* dargestellt, die wiederum vom SQL-Modul bereitgestellt wird.

Die Funktionalitäten des SQL-Moduls für die Auswertung von Studentenlösungen, die den Methoden der Klasse

etutor.modules.sql.SQLEvaluator entsprechen, werden in Abschnitt 2.1, Abschnitt 2.2 und Abschnitt 2.3 genauer beschrieben.

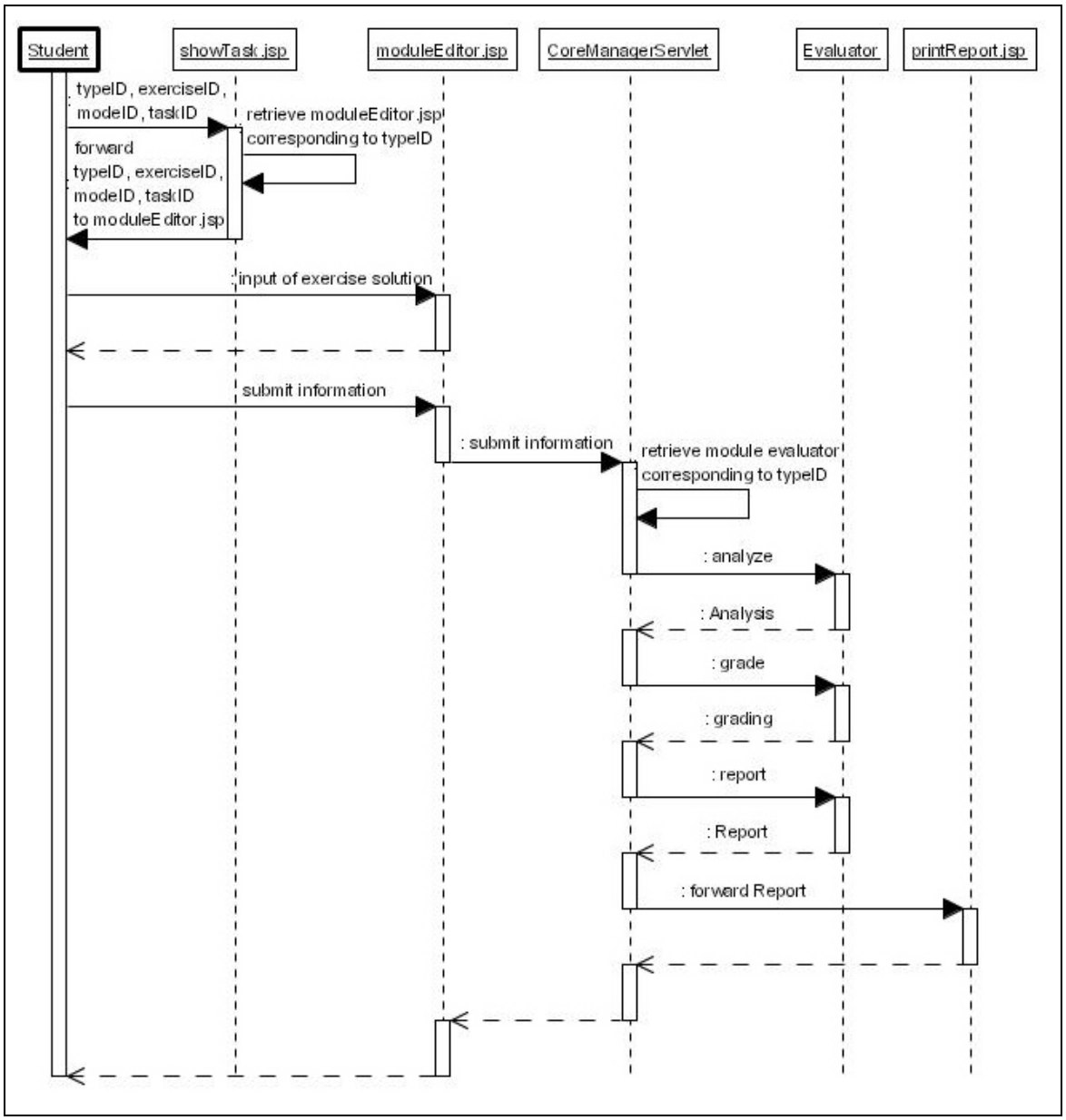

Abbildung 2.1: Aufruf eines Moduls im eTutor

Für die Funktionalitäten, die der Administration und Spezifikation von Übungsbeispielen durch Assistenten dienen, wird vom eTutor Core das Interface *etutor.core.manager.ModuleExerciseManager* vorgegeben. Das Interface und die darin definierten Methoden werden im Falle des SQL-Moduls durch die Klasse *etutor.modules.sql.SQLExerciseManager* implementiert. Die Eingabemaske, die das SQL-Modul für die Spezifikation von Übungsbeispielen zur Verfügung stellt, wird durch eine JSP-Seite realisiert (*exerciseSetting.jsp*). Dieser Seite ist ein Servlet vorgeschaltet, das für den Kontrollfluss bei der Eingabe von

modulspezifischen Informationen zu einem Übungsbeispiel verantwortlich ist, bzw. das die von Assistenten zu einem Übungsbeispiel eingegebenen Informationen verarbeitet (*etutor.modules.sql.ui.SQLExerciseManagerServlet*). Die Funktionalitäten des SQL-Moduls für die Administration von Übungsbeispielen, die den Methoden der Klasse *etutor.core.manager.ModuleExerciseManager* entsprechen, werden in den Abschnitten 2.4, 2.5, 2.6, 2.7, 2.8 und 2.9 genauer beschrieben.

2.1. Analyse

Die Analysefunktion wird durch die Methode *analyze* im eTutor-Interface *etutor.core.evaluation.Evaluator* definiert, und in der Klasse *etutor.modules.sql.SQLEvaluator* implementiert:

```
analyze(int exerciseID, int userID, Map passedAttributes, Map passedParameters)
```

2.1.1. Funktionalität

Zwei SQL-Abfragen werden auf einem vorgegebenen Datenbankschema ausgewertet. Die erste SQL-Abfrage repräsentiert die Musterlösung, die zu einem Übungsbeispiel im modulspezifischen Datenbestand gespeichert ist. Die zweite SQL-Abfrage hingegen stellt die Lösung eines Studenten dar, die es auszuführen und zu analysieren gilt. Bei der Ausführung einer SQL-Abfrage liefert die Datenbank Ergebnisse in Form von Ergebnistupeln. Bei einer erfolgreichen Ausführung beider SQL-Abfragen werden die dabei erhaltenen Ergebnistupeln miteinander verglichen. Aus den Abweichungen vom geforderten Ergebnis werden Schlüsse über mögliche Fehler in der SQL-Abfrage des Studenten gezogen. Die folgenden Fehlerkategorien werden bei einer Analyse in der angeführten Reihenfolge in Betracht gezogen:

- *Syntax*: In erster Linie muss die SQL-Abfrage syntaktisch korrekt formuliert und ausführbar sein.
- *Kartesisches Produkt*: Werden in einer SQL-Abfrage Daten mehrerer Datenbanktabellen abgefragt, ohne zumindest ein Attribut anzugeben, über das diese Tabellen miteinander verbunden werden (Datenbankbefehl JOIN), so enthält das Ergebnis ein kartesisches Produkt der Tupel beider Ausgangstabellen. Das SQL-Modul geht von der Bildung eines kartesischen Produkts aus, wenn die Anzahl der Ergebnistupel der

Studentenlösung die Anzahl der Ergebnistupel der Musterlösung signifikant übersteigt.

- *Fehlende oder überflüssige Spalten*: In den Ergebnistupeln der Studentenlösung dürfen weder mehr noch weniger Spalten enthalten sein als in den Ergebnistupeln der Musterlösung.
- *Fehlende oder überflüssige Ergebnistupel*: Die Studentenlösung darf weder mehr noch weniger Ergebnistupel enthalten als die Musterlösung.
- *Sortierung*: Die Ergebnistupel der Studentenlösung müssen exakt die selbe Reihenfolge aufweisen wie die Ergebnistupel der Musterlösung.

2.1.2. Eingabedaten

Neben den Parametern *exerciseId* und *userId*, die zur Identifizierung des Übungsbeispiels und des Studenten dienen, werden zusätzliche Parameter für die Analyse benötigt. Diese Parameter werden in der Map *passedAttributes* erwartet, in die vom eTutor Core vor Aufruf der Methode *analyze* alle Attribute aus dem Session- und dem Request-Kontext gespeichert werden. Die folgenden zusätzlichen Parameter werden für die Analyse benötigt:

- *action*: Wie weiter oben beschrieben wird in diesem Attribut gespeichert, welche der zur Verfügung stehenden Ausführungsmöglichkeiten vom Studenten gewählt wurden (*run, check, diagnose* oder *submit*).
- *submission*: Dieses Attribut enthält die vom Studenten formulierte SQL-Abfrage.
- *diagnoseLevel*: In diesem Attribut wird die gewählte Diagnosestufe übermittelt, die darüber Aufschluss gibt, wie detailliert das Feedback sein soll, das an den Studenten zurückgeliefert wird.

Die übergebene Map *passedParameters*, in die vom Core alle Request-Parameter gespeichert werden, ist für die Analyse nicht relevant.

2.1.3. Ausgabedaten

Die Ergebnisse der Analyse werden in einem Objekt vom Typ *etutor.modules.sql.analysis.SQLAnalysis* zusammengefasst. Die Klasse *SQLAnalysis* implementiert das eTutor-Interface *etutor.core.evaluation.Analysis*. In diesem Objekt sind Informationen über die analysierte Lösung und bei der Analyse eventuell aufgedeckte Fehler gespeichert.

2.1.4. Systemfehler

Die folgenden Systemfehler können bei einer Analyse im SQL-Modul auftreten:

- *Fehlende Parameter*: Die Analyse kann nicht durchgeführt werden, wenn die in Abschnitt 2.1.2 beschriebenen Parameter nicht vollständig übergeben werden.
- *Ungültige Parameter*: Systemfehler werden beispielsweise ausgelöst, wenn einer der übergebenen Parameter nicht den erwarteten Typ besitzt, oder etwa anhand der übergebenen *exerciseID* kein Übungsbeispiel gefunden wird.
- *Ungültiges Übungsbeispiel*: Die modulspezifischen Informationen zu einem Übungsbeispiel, die anhand der *exerciseID* identifiziert werden, sind ungültig. Davon betroffen ist beispielsweise eine SQL-Abfrage, die die Musterlösung darstellt, aber nicht ausführbar ist, da sie syntaktische Fehler enthält. Fehlerhaft ist eine SQL-Abfrage auch dann, wenn das für das Übungsbeispiel definierte Datenbankschema nicht alle Tabellen enthält, auf die sich die SQL-Abfrage bezieht.
- *Datenbankfehler*: Datenbankfehler sind Fehler, die nicht unmittelbar aufgrund der Einstellungen des SQL-Moduls bedingt sind, wie etwa eine abgebrochene Datenbankverbindung.

Im Gegensatz zur Musterlösung wird bei syntaktischen Fehlern in der SQL-Abfrage von Studenten kein Systemfehler angezeigt, da dieser Fehler als Teil der Analyse behandelt wird.

2.2. Bewertung

Die Bewertungsfunktion wird durch die Methode *grade* im eTutor-Interface *etutor.core.evaluation.Evaluator* definiert, und in der Klasse *etutor.modules.sql.SQLEvaluator* implementiert:

```
grade(Analysis analysis, int taskID, Map passedAttributes, Map
passedParameters)
```

Dabei werden die Ergebnisse einer Analyse herangezogen, um eine Studentenlösung zu bewerten.

2.2.1. Funktionalität

Die Bewertung einer Lösung basiert auf den Analyseergebnissen, die in Form eines Analyseobjektes übergeben werden. Eine Lösung wird in der derzeitigen Implementierung des SQL-Moduls nur dann als korrekt bewertet, wenn kein Fehler identifiziert wurde und das Ergebnis mit dem Ergebnis der Musterlösung übereinstimmt. Andernfalls wird die Lösung unter Abzug aller maximal erreichbaren Punkte bewertet.

2.2.2. Eingabedaten

Die für die Bewertung benötigten Informationen werden aus den Parametern ermittelt, die über die Methode *grade* übergeben werden:

- *analysis*: Die Bewertung wird anhand des Analyseobjektes durchgeführt, das vom SQL-Modul in der Methode *analyze* zurückgeliefert wird (siehe Abschnitt 2.1.3).
- *taskID*: Hiermit wird die Aufgabe identifiziert, die dem Studenten zugeteilt wurde. Dieser Parameter ist für die Bewertung im SQL-Modul irrelevant.
- *passedAttributes*: In Form eines Objektes vom Typ *java.util.Map* werden hier alle vom eTutor Core alle Attribute übergeben, die im Session-, und im Request-Kontext gespeichert sind. Für die Bewertung relevant ist hier ein Attribut mit der Bezeichnung *action*. Dieser Parameter kennzeichnet die Ausführungsmöglichkeit, die vom Studenten gewählt wurde. Eine Bewertung wird nicht durchgeführt, wenn die SQL-Abfrage lediglich ausgeführt wird (*run*). In allen anderen Fällen (*check*, *diagnose*, *submit*) wird die Lösung auf Basis der gefundenen Fehler bewertet.
- *passedParameters*: In diesem Objekt werden vom eTutor Core alle Parameter aus dem Request-Kontext übermittelt, wobei diese für die Bewertung irrelevant sind.

2.2.3. Ausgabedaten

Das Ergebnis der Bewertung ist ein Objekt, in dem alle wichtigen Informationen über die Bewertung gespeichert sind. Das Objekt ist vom Typ *etutor.modules.sql.grading.SQLGrading*, wobei die Klasse *SQLGrading* von der Klasse *etutor.modules.evaluation.DefaultGrading* abgeleitet wird. Mit der Klasse

DefaultGrading wird vom eTutor-System eine Standardimplementierung für das eTutor-Interface *etutor.core.evaluation.Grading* zur Verfügung gestellt.

2.2.4. Systemfehler

Fehler können auftreten wenn die in Abschnitt 2.2.2 beschriebenen Parameter, die für die Bewertung benötigt werden, ungültige Werte haben. Dies ist etwa dann der Fall, wenn das Attribut *action* in der übergebenen *Map* der Session- und Request-Attribute nicht enthalten ist. In diesen Fällen wird die Bewertung gestoppt und ein Systemfehler ausgelöst.

2.3. Feedback

Die Feedbackfunktion wird durch die Methode *report* im eTutor-Interface *etutor.core.evaluation.Evaluator* definiert, und in der Klasse *etutor.modules.sql.SQLEvaluator* implementiert:

```
report(Analysis analysis, Grading grading, Map passedAttributes, Map
passedParameters)
```

2.3.1. Funktionalität

Mithilfe der Feedback-Funktionalität wird aus den Erkenntnissen, die bei der Analyse und der Bewertung einer Studentenlösung gewonnen werden, ein Objekt generiert, das in einem späteren Schritt für die Generierung einer HTML-Seite verwendet werden kann, die dem Studenten angezeigt wird. In dem Objekt werden zu diesem Zweck Informationen gespeichert, die das Ergebnis der Abfrage, eventuell analysierte Fehler, Hinweise auf mögliche Fehlerquellen, Verbesserungsvorschläge und die Bewertung umfassen.

2.3.2. Eingabedaten

Die Informationen werden aus den Eingabedaten ermittelt, die über die Methode *report* übergeben werden:

- *analysis*: Die Generierung der Rückmeldung wird anhand des Analyseobjektes durchgeführt (siehe Abschnitt 2.1.3).
- *grading*: Neben dem Analyseobjekt wird auch das dazugehörige Bewertungsobjekt übergeben (siehe Abschnitt 2.2.3).

- *passedAttributes*: Attribute, die im Session-, bzw. dem Request-Objekt gespeichert sind. Relevant ist hier ein Attribut mit der Bezeichnung *diagnoseLevel*, mit dem das Ausmaß an Information über die Bewertungs- und Analyseergebnisse, das an den Benutzer zurückgeliefert werden soll, festgelegt wird. Das Attribut *diagnoseLevel* wird im Zusammenhang mit der Analyse verwendet, die dann durchgeführt wird, wenn der Student die Ausführungsmöglichkeit *diagnose* wählt. Diese Information ist im Attribut *action* enthalten, das neben *diagnose* die Werte *run*, *check* und *submit* haben kann.
- *passedParameters*: Parameter, die im Request-Objekt gespeichert sind, wobei diese für die Bewertung irrelevant sind.

2.3.3. Ausgabedaten

Das Ergebnis ist ein Objekt, in dem alle wichtigen Feedback-Elemente gespeichert sind, und aus denen sich das Dokument zusammensetzen lässt, das dem Studenten als Ergebnis präsentiert wird. Das Objekt ist vom Typ *eutor.modules.sql.report.SQLReport*, wobei die Klasse *SQLReport* das eTutor-Interface *etutor.core.evaluation.Report* implementiert.

2.3.4. Systemfehler

Bei der Generierung des Feedbacks im SQL-Modul sind keine Systemfehler vorgesehen.

2.4. Abfrage eines neuen Übungsbeispiels

Die Abfrage eines Objektes, das ein neu initialisiertes Übungsbeispiel repräsentiert, wird durch die Methode *fetchExerciseInfo* im eTutor-Interface *etutor.core.manager.ModuleExerciseManager* definiert, und in der Klasse *etutor.modules.sql.SQLExerciseManager* implementiert:

```
fetchExerciseInfo()
```

Das dabei zurückgelieferte Objekt wird vom eTutor Core an die Eingabemaske für die Spezifizierung eines neuen Übungsbeispiels übergeben.

2.4.1. Funktionalität

Der eTutor initiiert die Abfrage eines neuen Übungsbeispiels, wenn der Assistent in einer Reihe von Schritten zur Erzeugung eines neuen Übungsbeispiels auf die Seite gelangt, in der modulspezifische Informationen zum Übungsbeispiel einzugeben sind. Das SQL-Modul liefert in diesem Fall ein Objekt, das mit Informationen, wie v.a. einer SQL-Abfrage anzureichern ist. Zusätzlich werden in diesem Objekt Informationen gespeichert, die für die Benutzereingaben notwendig sind. Davon betroffen sind vor allem die Datenbankschemata, die grundsätzlich für SQL-Abfragen im SQL-Modul verfügbar sind. Aus den verfügbaren Schemata ist ein Schema zu wählen, auf dem die formulierte SQL-Abfrage ausführbar ist.

2.4.2. Eingabedaten

Für die Initialisierung eines Objektes, das die Basisinformationen für ein neu zu erstellendes Übungsbeispiel enthält, werden keine Eingabedaten benötigt.

2.4.3. Ausgabedaten

Die Methode liefert ein Objekt vom Typ *etutor.modules.sql.SQLExerciseBean*. Um dem Rückgabewert der vom eTutor Core vorgegebenen Schnittstelle zu entsprechen, wird zumindest das Interface *java.io.Serialiable* implementiert. Der eTutor Core nimmt beim Aufruf der Methode *fetchExerciseInfo* nur die Rolle eines Übermittlers ein, da das zurückgegebene Objekt an die Eingabemaske des SQL-Moduls für die Spezifizierung eines neuen Übungsbeispiels überreicht wird. Dort werden Benutzereingaben im Objekt gespeichert und bei Bestätigung durch den Assistenten durch Aufruf der in Abschnitt 2.6 beschriebenen Methode *createExercise* erzeugt. Der eTutor Core nimmt auch hier wieder die Rolle des Übermittlers ein, indem er das Objekt im Web-Container übernimmt und an die Methode *createExercise* des SQL-Moduls zurückgibt.

Die Methode liefert *null*, falls ein schwerwiegender Systemfehler aufgetreten ist. Systemfehler können beim Aufruf dieser Methode in erster Linie dann auftreten, wenn die Abfrage der dazu benötigten Informationen aus der vom SQL-Modul verwalteten Datenbank fehlschlägt.

2.4.4. Systemfehler

Systemfehler werden in der Log-Datei des SQL-Moduls protokolliert und durch einen Rückgabewert, der *null* ist, signalisiert (siehe Abschnitt 2.4.3).

2.5. Abfrage eines existierenden Übungsbeispiels

Die Abfrage eines Objektes, das ein bereits existierendes Übungsbeispiel repräsentiert, wird durch die Methode *fetchExercise* im eTutor-Interface *etutor.core.manager.ModuleExerciseManager* definiert, und in der Klasse *etutor.modules.sql.SQLExerciseManager* implementiert:

```
fetchExercise(int exerciseID)
```

Das dabei zurückgelieferte Objekt wird vom eTutor Core an die Eingabemaske für die Anpassung der Informationen zum Übungsbeispiel übergeben.

2.5.1. Funktionalität

Der eTutor initiiert die Abfrage eines existierenden Übungsbeispiels, wenn der Assistent in einer Reihe von Schritten zur Änderung eines Übungsbeispiels auf die Seite gelangt, in der modulspezifische Informationen zum Übungsbeispiel modifiziert werden können. Das SQL-Modul liefert in diesem Fall ein Objekt, das dem Objekt entspricht, wie es zuletzt zur Speicherung eines Übungsbeispiels an die Methode *createExercise* (siehe Abschnitt 2.6) oder die Methode *modifyExercise* (siehe Abschnitt 2.7) übergeben wurde.

2.5.2. Eingabedaten

Das SQL-Modul benötigt zum Abrufen eines Übungsbeispiels einen Parameter, anhand dessen sich das Übungsbeispiel identifizieren lässt. Dieser Parameter entspricht dem Parameter, der beim Erzeugen des Übungsbeispiels übergeben wurde (siehe Abschnitt 2.6).

2.5.3. Ausgabedaten

Die Methode liefert ein Objekt vom Typ *etutor.modules.sql.SQLExerciseBean*. Der eTutor Core nimmt beim Aufruf der Methode *fetchExercise* nur die Rolle eines Übermittlers ein, da das zurückgegebene Objekt an die Eingabemaske des SQL-Moduls für die Änderung des Übungsbeispiels überreicht wird. Dort werden

Benutzereingaben im Objekt gespeichert und bei Bestätigung durch den Assistenten durch Aufruf der in Abschnitt 2.6 beschriebenen Methode *createExercise* erzeugt. Der eTutor Core nimmt auch hier wieder die Rolle des Übermittlers ein, indem er das Objekt im Web-Container übernimmt und an die Methode *modifyExercise* des SQL-Moduls zurückgibt.

Die Methode liefert *null*, falls ein schwerwiegender Systemfehler aufgetreten ist. Systemfehler können beim Aufruf dieser Methode in erster Linie dann auftreten, wenn die Abfrage der dazu benötigten Informationen aus der vom SQL-Modul verwalteten Datenbank fehlschlägt.

2.5.4. Systemfehler

Systemfehler werden in der Log-Datei des SQL-Moduls protokolliert und durch einen Rückgabewert, der *null* ist, signalisiert (siehe Abschnitt 2.5.3).

2.6. Erzeugen eines neuen Übungsbeispiels

Das Erzeugen eines Übungsbeispiels wird durch die Methode *createExercise* im eTutor-Interface *etutor.core.manager.ModuleExerciseManager* definiert, und in der Klasse *etutor.modules.sql.SQLExerciseManager* implementiert:

```
createExercise(int exerciseID, Serializable exercise, Map passedAttributes, Map
passedParameters)
```

2.6.1. Funktionalität

Die Erzeugung eines neuen Übungsbeispiels setzt ein Objekt voraus, in dem alle dazu benötigten Informationen enthalten sind. Das Objekt wird zu diesem Zweck in einem ersten Schritt vom eTutor Core von der Methode *fetchExercise* angefordert (siehe Abschnitt 2.4). Nachdem das Objekt imzuge der Interaktionen des Benutzers mit der vom SQL-Modul bereitgestellten Eingabemaske mit Informationen befüllt wurde, wird es bei Bestätigung durch den Benutzer erzeugt.

2.6.2. Eingabedaten

Beim Erzeugen eines neuen Übungsbeispiel werden die folgenden Informationen an das SQL-Modul übergeben:

- *exerciseID*: Das Übungsbeispiel ist bei erfolgreicher Speicherung zu einem späteren Zeitpunkt über die hier übergebene *exerciseID* wieder abrufbar (siehe Abschnitt 2.5).
- *exercise*: Das SQL-Modul erwartet hier ein Objekt, das einem Objekt entspricht, wie es beim Abrufen eines neu initialisierten Übungsobjektes zurückgegeben wird (siehe Abschnitt 2.4).
- *passedAttributes* und *passedParameters*: Diese Parameter entsprechen den in Abschnitt 2.1.2 beschriebenen Parametern, sind hier allerdings nicht nicht relevant.

2.6.3. Ausgabedaten

Falls die Speicherung der Informationen zum neuen Übungsbeispiel erfolgreich durchgeführt werden konnte, wird der *boolean*-Wert *true* zurückgeliefert. Im Gegensatz dazu wird mit *false* angezeigt, dass die Speicherung aufgrund von Fehlern nicht möglich war. Hier wird zwischen leichteren Fehlern und Systemfehlern unterschieden. Ein Beispiel für den ersten Fall ist die Übergabe eines Objektes, das das Übungsbeispiel repräsentiert und das *null* ist. Systemfehler können beim Aufruf dieser Methode in erster Linie dann auftreten, wenn der Zugriff auf die vom SQL-Modul verwalteten Datenbank fehlschlägt.

2.6.4. Systemfehler

Systemfehler werden in der Log-Datei des SQL-Moduls protokolliert und durch einen Rückgabewert, der *false* ist, signalisiert (siehe Abschnitt 2.6.3).

2.7. Änderung eines existierenden Übungsbeispiels

Das Speichern einer geänderten Version zu einem Übungsbeispiel wird durch die Methode *modifyExercise* im eTutor-Interface *etutor.core.manager.ModuleExerciseManager* definiert, und in der Klasse *etutor.modules.sql.SQLExerciseManager* implementiert:

```
modifyExercise(int exerciseID, Serializable exercise, Map passedAttributes, Map
passedParameters)
```

2.7.1. Funktionalität

Die Änderung eines bereits existierenden Übungsbeispiels erfolgt analog zur in Abschnitt 2.6 beschriebenen Vorgehensweise zur Erzeugung eines neuen

Übungsbeispiels. Ein bereits existierendes Übungsbeispiel wird geändert, indem ein Objekt, das das Übungsbeispiel repräsentiert, vom SQL-Modul angefordert und in der Benutzerschnittstelle angepasst wird, und die Änderungen schließlich übernommen werden, indem das modifizierte Objekt wieder an das SQL-Modul übergeben wird.

2.7.2. Eingabedaten

Für die Speicherung einer geänderten Version eines Übungsbeispiels werden die folgenden Informationen an das SQL-Modul übergeben, die den in Abschnitt 2.6 beschriebenen Parametern entsprechen. Die *exerciseID* wird hier dazu verwendet, den Eintrag in der Datenbank zu identifizieren, der mit der geänderten Version überschrieben werden soll.

2.7.3. Ausgabedaten

Falls die Speicherung der Informationen des geänderten Übungsbeispiels erfolgreich durchgeführt werden konnte, wird der *boolean*-Wert *true* zurückgeliefert. Im Gegensatz dazu wird mit *false* angezeigt, dass die Speicherung aufgrund von Fehlern nicht möglich war. Hier wird zwischen leichteren Fehlern und Systemfehlern unterschieden. Ein Beispiel für den ersten Fall ist die Übergabe eines Objektes, das das Übungsbeispiel repräsentiert und das *null* ist. Systemfehler können beim Aufruf dieser Methode in erster Linie dann auftreten, wenn der Zugriff auf die vom SQL-Modul verwalteten Datenbank fehlschlägt.

2.7.4. Systemfehler

Systemfehler werden in der Log-Datei des SQL-Moduls protokolliert und durch einen Rückgabewert, der *false* ist, signalisiert (siehe Abschnitt 2.7.3).

2.8. Löschen eines Übungsbeispiels

Das Löschen eines Übungsbeispiels wird durch die Methode *deleteExercise* im eTutor-Interface *etutor.core.manager.ModuleExerciseManager* definiert, und in der Klasse *etutor.modules.sql.SQLExerciseManager* implementiert:

```
modifyExercise(int exerciseID)
```

2.8.1. Funktionalität

Das Löschen eines Übungsbeispiels durch Assistenten hat zur Folge, dass alle modulspezifischen Informationen zu diesem Übungsbeispiel aus dem vom SQL-Modul verwalteten Datenbestand gelöscht werden.

2.8.2. Eingabedaten

Der übergebene Parameter *exerciseID* dient zur Identifikation des zu löschenden Übungsbeispiels.

2.8.3. Ausgabedaten

Falls das Löschen der Informationen zum ausgewählten Übungsbeispiel erfolgreich war, wird der *boolean*-Wert *true* zurückgeliefert. Im Gegensatz dazu wird mit *false* angezeigt, dass der Löschvorgang nicht durchgeführt wurde. Ursache dafür kann sein, dass das entsprechende Übungsbeispiel nicht existiert, oder dass der Zugriff auf die vom SQL-Modul verwaltete Datenbank fehlschlägt.

2.8.4. Systemfehler

Systemfehler werden in der Log-Datei des SQL-Moduls protokolliert und durch einen Rückgabewert, der *false* ist, signalisiert (siehe Abschnitt 2.8.3).

2.9. Generierung eines Angabetextes zu einem Übungsbeispiel

Das eTutor-System sieht bei der Spezifikation von Übungsbeispielen die Möglichkeit vor, aus den modulspezifischen Informationen automatisch einen Angabetext zu generieren, der die Aufgabenstellung für die Ausarbeitung eines Übungsbeispiels beinhaltet. Diese Funktionalität wird allerdings vom SQL-Modul derzeit nicht unterstützt. Die für die Umsetzung dieser Funktionalität vorgesehene Methode liefert deshalb in der Klasse *etutor.modules.sql.SQLExerciseManager* immer *null*:

```
generateHtml(Serializable exercise, Locale locale)
```

3. Systemstruktur

In diesem Kapitel wird der Entwurf des SQL-Moduls beschrieben. Behandelt wird zum einen die Systemarchitektur hinsichtlich der Verteilung der Komponenten des SQL-Moduls und der Integration in das eTutor-System, und zum anderen die interne Struktur des SQL-Moduls in Form von Java-Packages.

3.1. Architektur

In diesem Abschnitt werden die Komponenten, aus denen sich das SQL-Modul zusammensetzt, ihr Zusammenwirken untereinander, sowie das Zusammenwirken mit dem eTutor Core beschrieben. Abbildung 3.1 zeigt die Verteilung der Komponenten des SQL-Moduls im Kontext des eTutor-Systems. Zu beachten ist hierbei, dass das SQL-Modul zusammen mit dem eTutor Core im Web-Container installiert wird (siehe Abschnitt 3.1.3). Zu sehen ist außerdem, dass einerseits auf einen modulspezifischen Datenbestand zugegriffen wird, in dem allgemeine Informationen zu Übungsbeispielen enthalten sind (*Exercise DB*), sowie auf Datenbankschemata, auf denen SQL-Abfragen ausgeführt werden (*Referenced DBs*).

Die Informationen zu einem Übungsbeispiel setzen sich aus einer SQL-Abfrage, die die Musterlösung repräsentiert, sowie aus Parametern für den Aufbau von Datenbankverbindung zu Datenbankschemata zusammen, auf denen die Musterlösungen und SQL-Abfragen von Studenten ausgeführt werden.

Nähere Informationen zur Datenverwaltung im SQL-Modul sind in Kapitel 4 zu finden.

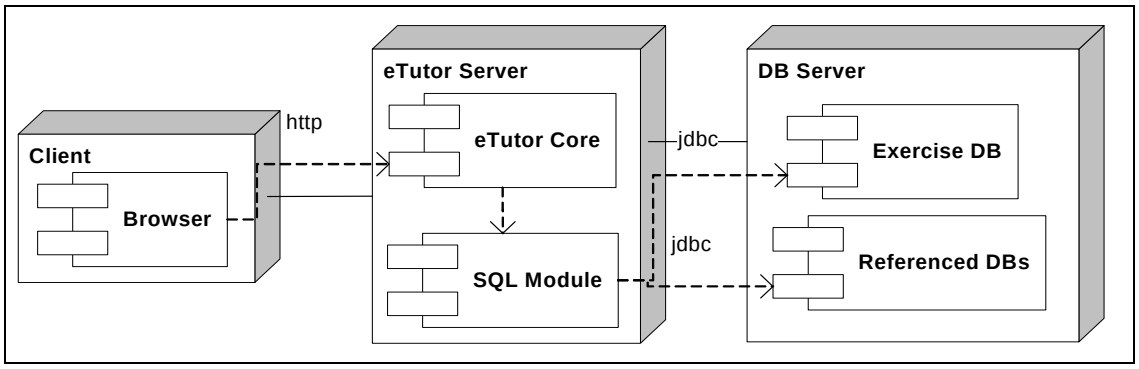

Abbildung 3.1: Verteilungsdiagramm des SQL-Moduls

3.1.1. Klassenbibliotheken (Libraries)

Die in Tabelle 3.1 aufgelisteten Java-Klassenbibliotheken werden vom SQL-Modul verwendet und müssen sich dementsprechend bei der Ausführung des Moduls im Klassenpfad befinden.

JAR-Datei	**Zweck**
cos.jar	Hilfsbibliothek, die für die Verarbeitung von HTTP-Requests zum Upload von Dateien verwendet wird [Hunt06].
servlet.jar	Enthält eine Teilmenge von Klassen der J2EE API, die für die Kompilierung von Servlets des eTutor-Systems benötigt werden. Zur Laufzeit werden diese Klassen hingegen vom Web-Container bereitgestellt.
ojdbc14.jar	JDBC-Driver für die Verbindung zur Datenbank des eTutor Core.

Tabelle 3.1: Klassenbibliotheken

3.1.2. Packages

Die Klassen des SQL-Moduls werden zu den in Tabelle 3.2 angeführten Java-Packages zusammengefasst.

Packages	**Zweck**
etutor.modules.sql	Dieses Package enthält mit den Klassen *SQLEvaluator* und *SQLExerciseManager* vor allem die Schnittstellen für die Auswertung

	und Spezifikation von Übungen. Eine zentrale Rolle nimmt die Klasse *SQLConstants* ein, die die wichtigsten Konstanten für Java-Klassen und JSP-Seiten des SQL-Moduls definiert.
etutor.modules.sql.analysis	Klassen, die für die Analysefunktionalität des SQL-Moduls verwendet werden, befinden sich in diesem Package. Die Klasse *SQLAnalyzer*, sowie die Klasse *SQLAnalysis*, die das Ergebnis einer Analyse repräsentiert, stellen in diesem Package die zentrale Klasse dar.
etutor.modules.sql.grading	Klassen, die für die Bewertungsfunktionalität des SQL-Moduls verwendet werden, befinden sich in diesem Package. Die Klasse *SQLGrader* stellt in diesem Package die zentrale Klasse dar.
etutor.modules.sql.report	Hier findet die Aufbereitung der Analyse- und Bewertungsergebnisse für die Rückmeldung an den Benutzer statt. Für die Aufbereitung verantwortlich ist die Klasse *SQLReporter*, das dabei erzeugte Ergebnis wird durch die Klasse *SQLReport* repräsentiert.
etutor.modules.sql.ui	Dieses Package enthält Klassen, die als Servlets im Web-Container registriert werden und die Anzeige der modulspezifischen Seiten in der Benutzeroberfläche übernehmen.

Tabelle 3.2: Java-Packages

3.1.3. RMI

Die für das eTutor-System konzipierten Module sollten im Sinne einer verteilten Architektur über *Remote Method Invocation* (RMI) eingebunden werden. Zu den dabei verfolgten Zielen zählen unter anderem die Unabhängigkeit der Teilsysteme untereinander, die isolierte Wartbarkeit einzelner Teilsysteme, sowie die Lastverteilung auf mehrere Rechner.

Im Falle einer Kommunikation über RMI wird das Modul im Idealfall auf einem eigenen Rechner ausgeführt. Bestimmte modulspezifische Ressourcen müssen zwar gemeinsam mit dem eTutor-Core im Web-Container installiert sein (v.a. Servlets, JSPs und darin benötigte, modulspezifische Klassen), die Ausführung der Funktionalitäten, die die Ausführung und Bewertung von Lösungen und die Administration von Übungsbeispielen betrifft, wird aber über RMI an das Modul delegiert.

Das SQL-Modul unterstützt die Kommunikation über RMI derzeit noch nicht, weshalb es mit dem eTutor Core im Web-Container installiert sein und dort ausgeführt werden muss.

3.2. Ordnerstruktur

In Abbildung 3.2 wird die Struktur des SQL-Moduls dargestellt. Der Aufbau entspricht den Konventionen, die für die Integration von Modulen in das eTutor-System vorgegeben sind. Demnach gibt es ein gemeinsames Java-Package *etutor.modules*, in dem sich die untergeordneten Packages des SQL-Moduls befinden. Im Ordner *lib* sind alle benötigten Klassenbibliotheken enthalten, die in Abschnitt 3.1.1 genauer beschrieben werden. Auf die Packages des SQL-Moduls wird hingegen in Abschnitt 3.1.2 eingegangen.

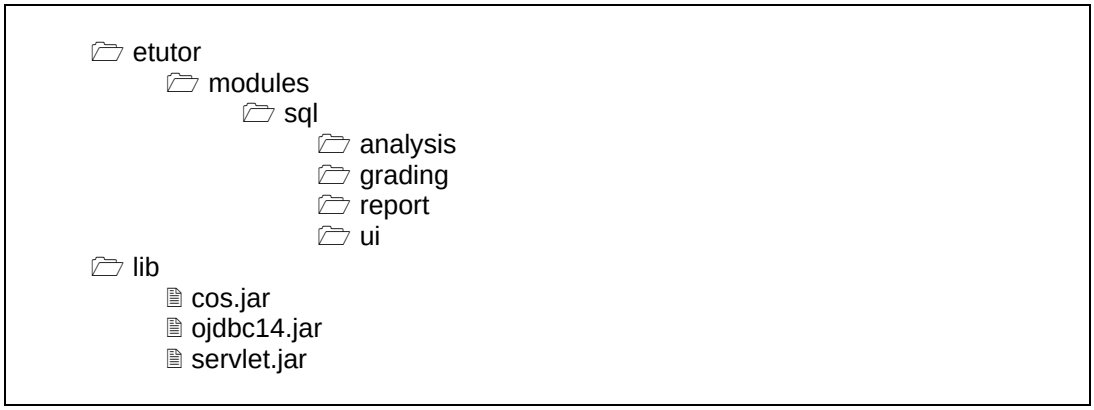

Abbildung 3.2: Ordnerstruktur des SQL-Moduls

4. Definition von Übungsaufgaben

Im folgenden werden das Datenbankschema und die Bedeutung der Tabellen erläutert, die für die Definition von Übungsaufgaben verwendet werden. Im hier beschriebenen Datenbankschema wird für jedes Übungsbeispiel im wesentlichen die SQL-Abfrage gespeichert, die die Musterlösung darstellt, sowie die Parameter für den Aufbau einer Verbindung zu einem Datenbankschema, auf dem die Abfrage ausgeführt werden kann. Die im allgemeinen verfügbaren Datenbankschemata, auf denen SQL-Abfragen im SQL-Modul ausgeführt werden können, werden in einer eigenen Tabelle gespeichert. Die Datenbankschemata werden somit nur über die Parameter für den Aufbau einer Datenbankverbindung referenziert. Ein solches Datenbankschema kann beliebig strukturiert sein. Ein Datenbankschema kann beispielsweise ein Banksystem repräsentieren, während im anderen Tabellen eines Flughafensystems enthalten sind. Die Administration eines solchen Datenbankschemas für die Zwecke des SQL-Moduls, d.h. für die Ausführung von SQL-Abfragen, setzt umfangreichere und komplexere Verwaltungsmöglichkeiten voraus, die von den Tools des jeweiligen Datenbankmanagementsystems besser erfüllt werden, als es im SQL-Modul realisierbar ist.

In den folgenden Abschnitten erfolgt die Beschreibung der Datenbanktabellen, die benötigt werden, um Informationen für konkrete Übungsaugaben für das SQL-Modul definieren zu können. Das dazugehörige Datenbankschema wird in Form eines UML-Diagramms in Abbildung 4.1 gezeigt.

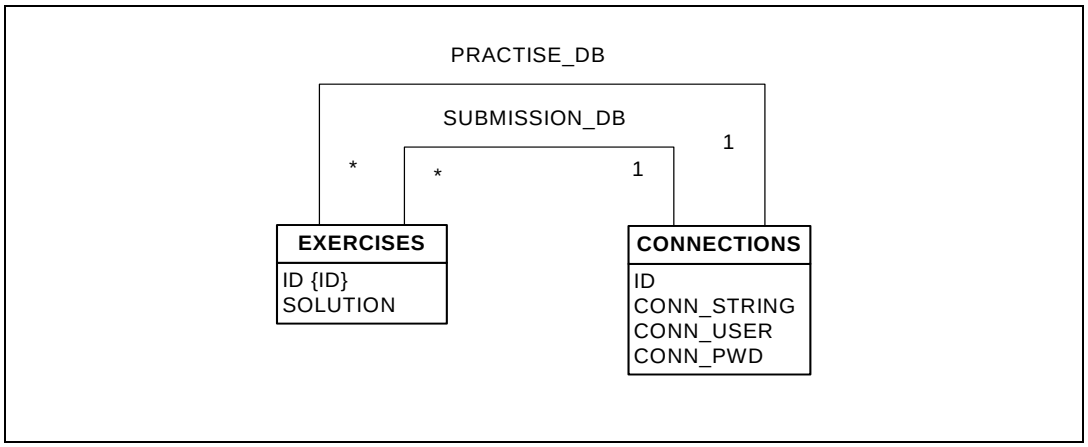

Abbildung 4.1: Datenbankschema für Übungsbeispiele

4.1.1. Übungsbeispiele

Die Datenbanktabelle *EXERCISES* enhält alle für ein Übungsbeispiel relevante Informationen (siehe Tabelle 4.1).

Spaltenbezeichnung	**Erklärung**
ID	Eindeutiger Schlüssel für ein Übungsbeispiel.
SOLUTION	SQL-Abfrage, die die Musterlösung darstellt.
PRACTISE_DB	Referenz auf das Datenbankschema, auf dem SQL-Abfragen ausgeführt werden, wenn der Student die Lösung nicht abgeben, sondern nur testen möchte (siehe Ausführungsmöglichkeiten in Kapitel 2).
SUBMISSION_DB	Referenz auf das Datenbankschema, auf dem SQL-Abfragen ausgeführt werden, wenn der Student die Lösung abgeben und bewerten lassen möchte (siehe Ausführungsmöglichkeiten in Kapitel 2).

Tabelle 4.1: Tabellenspalten für Übungsbeispiele

4.1.2. Datenbankverbindungen

Die Tabelle *CONNECTIONS* enthält die Parameter für den Aufbau von Datenbankverbindungen zu allen Datenbankschemata, in denen SQL-Abfragen ausgeführt werden können (siehe Tabelle 4.2).

Spaltenbezeichnung	**Erklärung**
ID	Eindeutiger Schlüssel für ein Datenbankschema
CONN_STRING	URL, über die die Datenbank erreicht werden kann. Dieser Eintrag muss ein Format haben, das vom im SQL-Modul verwendeten JDBC-Driver verarbeitet werden kann (siehe Abschnitt 3.1.1).
CONN_USER	Benutzer für die Anmeldung zur Datenbank
CONN_PWD	Passwort für die Anmeldung zur Datenbank

Tabelle 4.2: Tabellenspalten für Datenbankschemata

5. Konfiguration

Konstanten, die im gesamten SQL-Modul Gültigkeit haben, sind in der Java-Klasse *etutor.modules.sql.SQLConstants* zusammengefasst. Eine Änderung dieser Konstanten setzt eine neuerliche Kompilierung des SQL-Moduls voraus, aus diesem Grund sollten diese Parameter bei einer Weiterentwicklung des Moduls in eine Konfigurationsdatei ausgelagert werden.

Literaturverzeichnis

[Hunt06] J. Hunter. com.oreilly.servlet. http://servlets.com/cos, Februar 2006.