eTutor

Modularchitektur und Moduldokumentation

Relationale Algebra

Georg Nitsche

Version 1.0
Stand März 2006

INSTITUT FÜR WIRTSCHAFTSINFORMATIK

dke
Data & Knowledge Engineering

Versionsverlauf:

Version	***Autor***	***Datum***	***Änderungen***
1.0	gn	09.03.2005	Fertigstellung der ersten Version

Inhaltsverzeichnis:

Abbildungsverzeichnis:

Tabellenverzeichnis:

1. Einleitung

Im Relationenmodell werden Daten in Form von Relationen gespeichert, die sich als Tabellen darstellen lassen. Die Zeilen einer Tabelle entsprechen jeweils einem Objekt einer Anwendungsdomäne, das eine Menge bestimmter Merkmalsausprägungen besitzt. Die Relationale Algebra definiert Operationen, die auf einer oder mehreren Relationen durchgeführt werden können, um Daten abzufragen, die wiederum als Relation repräsentiert werden.

Das Modul Relationale Algebra stellt ein Expertensystem dar, das im eTutor-System die Bearbeitung von Übungsbeispielen, bei denen von Studenten Abfragen in Relationaler Algebra zu formulieren sind, ermöglicht. Das Modul unterstützt die Ausführung dieser Abfragen auf einem vordefinierten Schema einer relationalen Datenbank. Darüber hinaus kann das Ergebnis einer Abfrage analysiert und bewertet werden. Letztlich werden die Ergebnisse einer durch einen Studenten ausgearbeiteten Lösung, die die zurückgelieferten Daten, die Analyse und die Bewertung betreffen, aufbereitet und präsentiert.

Während Studenten durch das Modul Relationale Algebra Übungsbeispiele ausarbeiten können, stellt das Modul auch eine Schnittstelle für die Spezifikation und Administration dieser Übungsbeispiele durch Assistenten zur Verfügung.

Kapitel 2 enthält Informationen über die Anforderungen, die für die Integration eines Moduls in das eTutor-System spezifiziert sind, sowie die Umsetzung dieser Anforderungen im Modul Relationale Algebra. Kapitel 3 widmet sich der Systemstruktur des Moduls Relationale Algebra. Inhalt dieses Kapitels ist unter anderem die Abhängigkeit des Moduls Relationale Algebra vom SQL-Modul. 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

Das eTutor-System besteht im wesentlichen aus einem Kernsystem (eTutor Core), das die grundsätzlichen Funktionalitäten als E-Learning-Plattform bereitstellt. Einzelne Wissensgebiete, mit denen die Ausarbeitung entsprechender Übungsbeispiele ermöglicht wird, werden als Module in das eTutor-System integriert. Für die Integration werden vom eTutor Core Schnittstellen spezifiziert, die von einem zu integrierenden Modul unterstützt werden müssen. Die Schnittstellen werden in Form von Java-Interfaces vorgegeben, die Umsetzung in einem Modul erfolgt durch Klassen, die diese Interfaces implementieren. Details zu den vorgegebenen Schnittstellen und zu sonstigen Voraussetzungen für eine Integration können der Dokumentation der Modularchitektur des eTutor-Systems entnommen werden.

Dieses Kapitel befasst sich mit der Umsetzung des Moduls Relationale Algebra, wobei für eine Integration in das eTutor-System zwei grundsätzliche Funktionalitäten unterstützt werden. Einerseits ermöglicht das Modul die Auswertung von Abfragen in Relationaler Algebra, die von Studenten zu einem Übungsbeispiel formuliert werden, und andererseits die Spezifikation von Übungsbeispielen durch Assistenten.

Der generelle Ablauf bei der Benutzung eines Moduls durch Studenten wird in der Dokumentation der Modularchitektur des eTutor-Systems beschrieben. Im Rahmen dieses Ablaufes werden im Falle des Moduls Relationale Algebra die folgenden Komponenten aufgerufen:

- *Eingabemaske*: Für die Bearbeitung einer Aufgabe muss ein in das eTutor-System integriertes Modul eine Eingabemaske bereitstellen, über die Lösungen eingegeben werden können. Diese Eingabemaske wird durch eine JSP-Seite realisiert, in der eine Abfrage in Relationaler Algebra eingegeben werden kann (*showEditor.jsp*). Der Aufruf der Seite wird durch den eTutor Core initiiert, wenn ein Student eine ihm zugeteilte Aufgabe zum Aufgabengebiet Relationale Algebra auswählt. Nach der

Eingabe der Lösung wird die Kontrolle vom Modul Relationale Algebra zurück an den eTutor Core gegeben.

- *Rückmeldung*: Das Modul Relationale Algebra stellt eine JSP-Seite zur Verfügung, in der die Ergebnisse der Ausführung einer Abfrage in Relationaler Algebra, sowie Analyse- und Bewertungsergebnisse aufbereitet werden (*printReport.jsp*). Das Ergebnis einer Abfrage ist dabei eine Relation, die, sofern die Abfrage ausführbar war, als Tabelle dargestellt wird.
- *Auswertungskomponente*: Die Auswertungskomponente übernimmt die Ausführung und Analyse einer Abfrage, die Bewertung, sowie die Generierung eines Feedback-Objektes. Das Feedback-Objekt enthält alle für die Aufbereitung des Feedbacks (*printReport.jsp*) benötigte Informationen. Die Auswertungskomponente wird durch die Klasse *etutor.modules.ra2sql.RAEvaluator* realisiert, die das vom eTutor Core vorgegebene Interface *etutor.core.evaluation.Evaluator* implementiert. Die Funktionalitäten, die durch die Auswertungskomponente unterstützt werden, werden in Abschnitt 2.1, Abschnitt 2.2 und Abschnitt 2.3 genauer beschrieben.

Die Kommunikation zwischen den Web-Ressourcen, d.h. zwischen Servlets und JSP-Seiten des eTutor Core und des Moduls Relationale Algebra basiert auf Parametern und Attributen. Parameter können nur im Request-Kontext übergeben werden, während Attribute vor dem Weiterleiten eines Requests an eine weitere Web-Ressource entweder im Request- oder im Session-Kontext gespeichert werden. Bei der Beispielauswertung durch das Modul Relationale Algebra werden die in Tabelle 2.1 Parameter und Attribute verwendet.

Name	**Beschreibung**
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 identifizieren.
actions	Mit diesem Attribut wird vom eTutor Core festgelegt, welche Möglichkeiten dem Studenten in der Eingabemaske für die Ausführung einer 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*. Diese Werte werden vom Modul Relationale Algebra verwendet, um die Eingabemaske anzupassen. Je nachdem, welche Werte in diesem Attribut enthalten sind, werden in der Eingabemaske die entsprechenden Ausführungsmöglichkeiten zur Verfügung gestellt: die Ausführung einer 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 Modul Relationale Algebra zusätzlich mehrere Diagnosestufen unterschieden.
action	Hiermit wird durch das Modul Relationale Algebra festgelegt, welche der zur Verfügung stehenden Ausführungsmöglichkeiten in der Eingabemaske bei der Ausführung einer Abfrage durch den Studenten auswählt wurde. Diese Information wird einerseits vom eTutor Core benötigt, und andererseits vom Modul Relationale Algebra selbst.

Tabelle 2.1: Parameter und Attribute für die Beispielauswertung

Neben dem Ablauf bei der Benutzung eines Moduls durch Studenten werden auch die Abläufe bei der Benutzung durch Assistenten in der Dokumentation der Modularchitektur des eTutor-Systems beschrieben. Je nach Anwendungsfall werden dabei die folgenden Komponenten des Moduls Relationale Algebra aufgerufen:

- *Eingabemaske*: Die Eingabemaske, die das 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.ra2sql.RAExerciseManagerServlet*).

- *Administrationskomponente*: 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 Moduls Relationale Algebra durch die Klasse *etutor.modules.ra2sql.RAExerciseManager* implementiert. Die in dieser Klasse enthaltenen Methoden, bzw. die Funktionalitäten, die dadurch unterstützt werden, 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.ra2sql.RAEvaluator* implementiert:

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

2.1.1. Funktionalität

Das Modul Relationale Algebra wertet zwei Abfragen auf einem vorgegebenen Datenbankschema aus, wobei die erste Abfrage die Musterlösung, die zu einem Übungsbeispiel im modulspezifischen Datenbestand gespeichert ist, repräsentiert, während die zweite Abfrage die Lösung eines Studenten darstellt. Das Modul bedient sich dabei der Analysefunktion des SQL-Moduls (siehe Moduldokumentation des SQL-Moduls). Dazu werden die Abfragen, die in Relationaler Algebra vorliegen, in SQL-Abfragen transformiert. Die bei der Analyse berücksichtigten Fehlerkategorien entsprechen somit den im SQL-Modul berücksichtigten Fehlerkategorien. Ausgenommen davon ist die Überprüfung der syntaktischen Korrektheit einer Abfrage, die bereits bei der Transformation einer Abfrage von Relationaler Algebra zu SQL analysiert wird. Entspricht die Abfrage in Relationaler Algebra der vom Modul Relationale Algebra vorgegebenen Syntax, so muss auch die daraus generierte SQL-Abfrage ausführbar sein.

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*: Dieses Attribut gibt darüber Aufschluss, welche der zur Verfügung stehenden Ausführungsmöglichkeiten vom Studenten gewählt wurden (*run*, *check*, *diagnose* oder *submit*).
- *submission*: Dieses Attribut enthält die vom Studenten formulierte Abfrage in Relationaler Algebra.
- *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

Das Rückgabeobjekt entspricht dem in der Modulbeschreibung des SQL-Moduls beschriebenen Rückgabewert der Analysemethode.

2.1.4. Systemfehler

Die folgenden Systemfehler können bei einer Analyse im Modul Relationale Algebra 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 Abfrage, die die Musterlösung darstellt, aber nicht ausführbar ist, da sie syntaktische Fehler enthält.

- *Datenbankfehler*: Datenbankfehler sind Fehler, die nicht unmittelbar aufgrund der Einstellungen des Moduls bedingt sind, wie etwa eine abgebrochene Datenbankverbindung.

2.2. Bewertung

Die Bewertungsfunktion wird durch die Methode *grade* im eTutor-Interface *etutor.core.evaluation.Evaluator* definiert, und in der Klasse *etutor.modules.ra2sql.RAEvaluator* 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

Das Modul bedient sich dabei der Bewertungsfunktion des SQL-Moduls (siehe Moduldokumentation des SQL-Moduls).

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 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 Modul Relationale Algebra 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 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 Rückgabeobjekt entspricht dem in der Modulbeschreibung des SQL-Moduls beschriebenen Rückgabewert der Bewertungsmethode.

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.ra2sql.RAEvaluator* 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 Rückgabeobjekt entspricht dem in der Modulbeschreibung des SQL-Moduls beschriebenen Rückgabewert der Feedbackmethode.

2.3.4. Systemfehler

Bei der Generierung des Feedbacks im Modul Relationale Algebra 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.ra2sql.RAExerciseManager* 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. Zurückgeliefert wird hier ein Objekt das in einem späteren Schritt zum Speichern eines neuen Übungsbeispiels in der vom Modul verwalteten Datenbasis verwendet werden kann (siehe Abschnitt 2.6).

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.ra2sql.RAExerciseBean*, das das Interface *java.io.Serializable* implementiert. In diesem Objekt werden Informationen zu allen Datenbankschemata gespeichert, die grundsätzlich für Abfragen im Modul Relationale Algebra verfügbar sind. Aus den verfügbaren Schemata ist in der Benutzerschnittstelle ein Schema zu wählen, auf dem die durch einen Studenten formulierte Abfrage in Relationaler Algebra, bzw. die daraus generierte SQL-Abfrage, ausführbar ist.

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 Modul Relationale Algebra verwalteten Datenbank fehlschlägt.

2.4.4. Systemfehler

Systemfehler werden in der Log-Datei des Moduls Relationale Algebra 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.

Die Abfrage eines existierenden Übungsbeispiels dient im eTutor-System dazu, Assistenten den Zustand eines Übungsbeispiels anzeigen zu können, bzw. die Änderung der Informationen zu einem Übungsbeispiel zu ermöglichen. Diese Funktionalität wird allerdings in der derzeitigen Implementierung des Moduls Relationale Algebra noch nicht unterstützt.

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 im Zuge der Interaktionen des Benutzers mit der vom Modul Relationale Algebra 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 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 Modul Relationale Algebra 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 sind für das Erzeugen eines neuen Übungsbeispiels 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. Fehler können auftreten, wenn ein Objekt übergeben wird, das *null* ist, oder wenn der Zugriff auf die vom Modul Relationale Algebra verwaltete Datenbank fehlschlägt.

2.6.4. Systemfehler

Systemfehler werden in der Log-Datei des 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)
```

Das Speichern einer geänderten Version zu einem bereits existierenden Übungsbeispiel wird allerdings in der derzeitigen Implementierung des Moduls Relationale Algebra noch nicht unterstützt.

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 Modul Relationale Algebra 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 Modul verwaltete Datenbank fehlschlägt.

2.8.4. Systemfehler

Systemfehler werden in der Log-Datei des 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 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 Moduls Relationale Algebra beschrieben. Behandelt wird zum einen die Systemarchitektur hinsichtlich der Verteilung der Komponenten des Moduls und der Integration in das eTutor-System, und zum anderen die interne Struktur des Moduls in Form von Java-Packages. Ein weiterer Aspekt ist dabei die Abhängigkeit des Moduls Relationale Algebra vom SQL-Modul.

3.1. Architektur

In diesem Abschnitt werden die Komponenten, aus denen sich das Modul zusammensetzt, ihr Zusammenwirken untereinander, sowie das Zusammenwirken mit dem eTutor Core beschrieben. Abbildung 3.1 zeigt die Verteilung der Komponenten des Moduls Relationale Algebra im Kontext des eTutor-Systems. Zu beachten ist hierbei, dass das 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 Abfragen ausgeführt werden (*Referenced DBs*).

Die Informationen zu einem Übungsbeispiel setzen sich aus einer Abfrage, die die Musterlösung repräsentiert, sowie aus Parametern für den Aufbau von Datenbankverbindung zu Datenbankschemata zusammen, auf denen die Musterlösungen und Abfragen von Studenten ausgeführt werden. Für die Ausführung muss eine Abfrage in Relationale Algebra vorerst in eine SQL-Abfrage transformiert werden. Die Ausführung, Analyse und Bewertung dieser SQL-Abfragen wird über das SQL-Modul bewerkstelligt.

Nähere Informationen zur Datenverwaltung im Modul Relationale Algebra sind in Kapitel 4 zu finden.

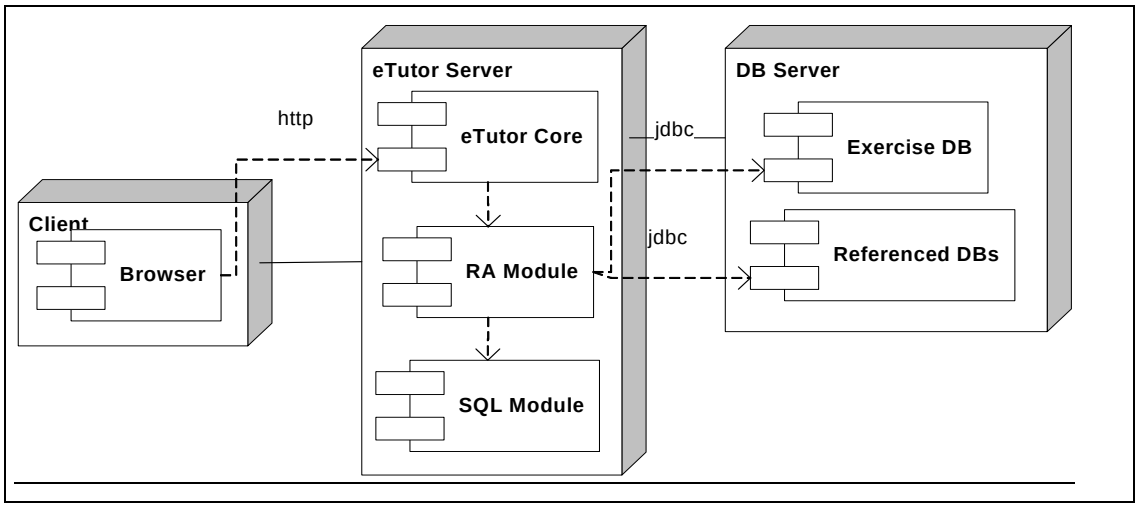

Abbildung 3.1: Verteilungsdiagramm des Moduls Relationale Algebra

3.1.1. Klassenbibliotheken (Libraries)

Die in Tabelle 3.1 aufgelisteten Java-Klassenbibliotheken werden vom Modul Relationale Algebra 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.
antlr.jar	Bibliothek, die das Einlesen von Abfragen in Relationaler Algebra unterstützt [ANTL06]. Die Bibliothek ermöglicht es, syntaktische Fehler in Studentenlösungen zu erkennen, sowie Java-Objekte zu generieren, die die eingelesene Abfrage repräsentieren. Mithilfe dieser Objekte wird im Modul Relationale Algebra wiederum eine SQL-Abfrage generiert, die in einem Datenbankschema ausgeführt

	werden kann.

Tabelle 3.1: Klassenbibliotheken

3.1.2. Packages

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

Packages	**Zweck**
etutor.modules.ra2sql	Dieses Package enthält mit den Klassen *RAEvaluator* und *RAExerciseManager* vor allem die Schnittstellen für die Auswertung und Spezifikation von Übungen. Eine zentrale Rolle nimmt die Klasse *RAConstants* ein, die die wichtigsten Konstanten für Java-Klassen und JSP-Seiten des Moduls definiert.
etutor.modules.ra2sql.model	Die in diesem Package enthaltenen Klassen bilden ein Java-Modell, das die Operationen und Konstrukte der Relationalen Algebra repräsentiert. Mithilfe dieser Klassen lassen sich Objekte erzeugen, die einer Abfrage in Relationaler Algebra entsprechen.

Tabelle 3.2: Java-Packages

3.1.3. RMI

Das Modul Relationale Algebra unterstützt die Kommunikation über *Remote Method Invocation* (RMI) derzeit noch nicht, weshalb es mit dem eTutor Core im Web-Container installiert sein und dort ausgeführt werden muss (siehe dazu auch Modulbeschreibung des SQL-Moduls).

3.2. Ordnerstruktur

In Abbildung 3.2 wird die Struktur des Moduls Relationale Algebra 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 Moduls befinden. Analog dazu befinden sich in *etutor.resources* alle modulspezifischen Ressourcen und in einem Ordner *lib* alle benötigten Klassenbibliotheken. Die Klassenbibliotheken des Ordners *lib* werden in Abschnitt 3.1.1 beschrieben, während auf die Packages des Moduls in Abschnitt 3.1.2 eingegangen wird.

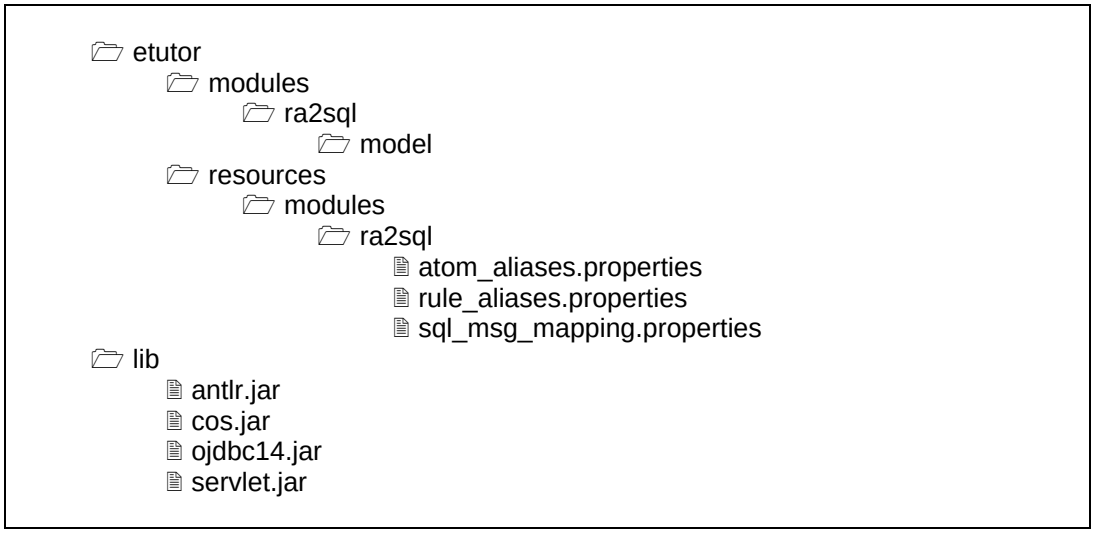

Abbildung 3.2: Ordnerstruktur des Moduls Relationale Algebra

4. Definition von Übungsaufgaben

Im folgenden werden das Datenbankschema und die Bedeutung der Tabellen erläutert, die für die Definition von Übungsaufgaben verwendet werden. Struktur und Verwendungszweck des Datenbankschemas entspricht weitgehend dem Datenbankschema des SQL-Moduls (siehe Modulbeschreibung des SQL-Moduls).

Für jedes Übungsbeispiel wird im wesentlichen die 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 Abfragen im Modul Relationale Algebra 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.

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

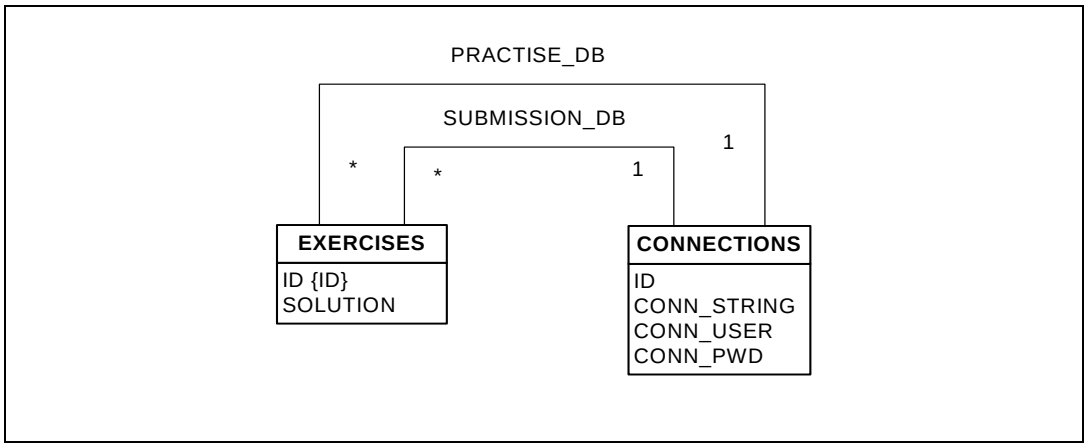

Abbildung 4.1: Datenbankschema für Übungsbeispiele

4.1.1. Übungsbeispiele

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

Spaltenbezeichnung	**Erklärung**
ID	Eindeutiger Schlüssel für ein Übungsbeispiel.
SOLUTION	Abfrage, die die Musterlösung darstellt. Diese Abfrage wird dabei nicht in Relationaler Algebra gespeichert, sondern als SQL-Abfrage. Während eine zu analysierende Studentenlösung vor der Ausführung in eine SQL-Abfrage transformiert werden muss, ist dies bei der Musterlösung damit nicht notwendig.
PRACTISE_DB	Referenz auf das Datenbankschema, auf dem 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 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 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 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 des Moduls Relationale Algebra werden in einer Java-Klasse definiert (*etutor.modules.sql.SQLConstants*). Für eine bessere Konfigurierbarkeit empfiehlt sich im Zuge der weiteren Entwicklung des Moduls eine Auslagerung von Konfigurationsparametern in eine Datei.

Literaturverzeichnis

[ANTL06] ANTLR. http://www.antlr.org/, March 2006.

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