# Diplomarbeit

Entwicklung eines Webinterfaces für das
Oracle-Datenbankpraktikum

von
Nils Weber

26. September 2004

Hochschule Wismar
Fachhochschule für Technik, Wirtschaft und Gestaltung

# 1 Abstrakt

Diese Diplomarbeit beschreibt die Konzeption, Implementierung und Testphase eines datenbankgestützten Webinterfaces zur Durchführung des Praktikums im Fach Datenbankensysteme an der Hochschule Wismar. Das Praktikum soll die Studenten beim Erlernen der Abfragesprache SQL unterstützen und den Hochschulbetreuern eine Möglichkeit zur Auswertung der studentischen Leistungen bereitstellen. Das im Rahmen dieser Arbeit entwickelte System basiert auf dem Zusammenspiel eines Apache-Webservers und einer Oracle Datenbank.

Der allgemeine Teil soll dem Leser den Einstieg in das Thema der Arbeit erleichtern. In diesem Kontext werden häufig verwendete Begriffe eingeführt und erörtert, wobei der Schwerpunkt auf dem Komplex Datenbanken liegt.

Im Anschluß erfolgt die Dokumentation der Installation und Konfiguration eines Oracle9i Datenbank-Managementsystems an einem konkreten Beispiel. Dabei werden einige Besonderheiten dieses Systems herausgestellt und der Oracle9i Enterprise Manager vorgestellt. Der Kommunikationsschnittstelle von Oracle9i ist in diesem Kapitel ein gesonderter Abschnitt gewidmet.

Der Hauptteil der Arbeit befaßt sich mit der Konzeption, der Funktionsweise und der Nutzung des Webinterfaces. In diesem Zusammenhang werden die Installation und Konfiguration eines Apache-Webservers mit PHP-Modul, der Entwurf und die Umsetzung der verwendeten Datenbank und wesentliche Details der PHP-Skripte des Webinterfaces dokumentiert.

Abgeschlossen wird die Diplomarbeit durch eine Analyse der Performance-Testergebnisse des Datenbank-Online-Praktikums, die im Sommersemester 2004 an der Hochschule Wismar ermittelt wurden.

# Inhaltsverzeichnis

# Abkürzungen und Begriffe

| | |
|---|---|
| ANSI | **A**merican **N**ational **S**tandards **I**nstitute - amerikanisches Institut, das sich mit der Normierung industrieller Verfahrensweisen befaßt und der ISO (**I**nternational **O**rganization for **S**tandardization) untergeordnet ist |
| API | **A**pplication **P**rogram **I**nterface - Schnittstelle für die Anwendungsprogrammierung |
| Case Tool | Entwurfswerkzeug zur Entwicklung und Analyse von Datenbankschemata |
| Client/Server-System | Netzwerk-System, in dem ein leistungsfähiger Rechner verschiedenen im Netz befindlichen Clients Rechenleistung und Speicherkapazität zur Verfügung stellt |
| cookie | im Browser gespeicherte Text-Dateien zur Zwischenspeicherung von Daten für den Webserver - beinhalten beispielsweise Informationen über Formulareingaben oder besuchte Seiten |
| DBA | **D**ata**b**ase **A**dministrator - Nutzer mit Administrationsrechten für die Datenbank |
| DNS | **D**omain **N**ame **S**ystem - Speicherung von Domainnamen und zugehöriger IP-Adresse zur effektiveren Suche |
| E-Learning | Lernen mit Hilfe von elektronischen Medien beziehungsweise Hilfsmitteln |
| EI | **E**lektrotechnik/**I**nformatik - Fachbereich der Hochschule Wismar |
| HTML | **H**yper**T**ext **M**arkup **L**anguage - Beschreibungssprache für die Struktur von Internetseiten oder Dokumenten mit Hypertextmöglichkeiten |

| | |
|---|---|
| IPC | **I**nter **P**rocess **C**ommuniction - Datenaustausch zwischen zwei Prozessen |
| KAPV | **K**unden-**A**uftrags-**P**rodukt-**V**erwaltung - ein fiktive Unternehmenssituation beziehungsweise Datenbankanwendung für das Praktikum |
| LDAP | **L**ightweight **D**irectory **A**ccess **P**rotocol - Verzeichnisdienst auf Basis von TCP/IP |
| MIME | **M**ultipurpose **I**nternet **M**ail **E**xtensions - Standard Multimedia-Dateiformate für E-Mails und Internetseiten zur eindeutigen Identifizierung von Datentypen |
| OCI | **O**racle **C**all **I**nterface - Bibliothek mit definierter Programmierschnittstelle |
| Oracle | Hersteller eines gleichnamigen sehr leistungsfähigen, objekt-relationalen Datenbank-Managementsystems |
| OSI | **O**pen **S**ource **I**nitiative - Vereinigung zur Standardisierung von Open Source Produkten |
| OUI | **O**racle **U**niversal **I**nstaller - Java-Programm, welches den Nutzer durch die Installation führt |
| PHP | **H**ypertext **P**reprocessor - serverseitige in HTLM eingebettete Skriptsprache zur Entwicklung dynamischer Internetseiten |
| PL/SQL | **P**rocedural **L**anguage extension to SQL - prozedurale Spracherweiterung des SQL-Standards |
| RDBMS | **R**elational **D**ata**b**ase-**M**anagement**s**ystem - deutsch auch RDBVS: **R**elationales **D**aten**b**ank**v**erwaltung**s**ystem |
| SGA | (Oracle) **S**ystem **G**lobal **A**rea - strukturierter Speicherbereich zur Optimierung des Datentransfers zwischen Clients und dem Oracle Datenbank-Managementsystem |
| shell | Kommando-Interpreter, mit dem Befehle an das Betriebssystem abgegeben werden |
| SID | **S**ervice **Id**entifier - Bezeichner zur Verbindung mit der Datenbank bei Oracle |

| | |
|---|---|
| SQL | **S**tructured **Q**uery **L**anguage - nichtprozedurale, mengenorientierte Datenbankanfragesprache für die Bedienung, Aktualisierung und Abfrage von relationalen Datenbanken |
| SSL | **S**ecure **S**ockets **L**ayer - offenes Protokoll zur Verschlüsselung des Hyper Text Transfer Protokolls |
| Stored Procedures | Auf dem Datenbankserver gespeicherte und kompilierte Prozedur, die durch einen einfachen Befehl vom Client ausgeführt werden kann. Die Ausführung erfolgt auf dem Server. |
| tablespace | logisches Strukturelement von Datenbanken zur Gruppierung von Datenbankobjekten |
| Trigger | Funktionaltät zur Reaktion auf bestimmte Vorgänge in Datenbanken - Bei einer bestimmten Art der Änderungen (z.B. INSERT, UPDATE, DELETE bei SQL) von Daten in einer Tabelle wird ein gespeichertes Programm aufgerufen, daß diese Änderung erlaubt, verhindert und/oder weitere Tätigkeiten vornimmt. Trigger werden unter anderem zur Wahrung der Datenkonsistenz und zum Einfügen/Löschen von Referenzdaten eingesetzt. |
| URL | **U**niform **R**esource **L**ocator - Globale Adresse von Dokumenten und anderen Ressourcen im Internet |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 2 Einleitung

## 2.1 Motivation

Die Lehrveranstaltungen an der Hochschule Wismar gliedern sich im allgemeinen in einen theoretischen Vorlesungsteil und einen Praktikumsteil, in dem das in der Vorlesung erlangte Wissen angewandt werden soll. Das Praktikum im Fach Datenbanksysteme [2], der Studiengänge Mutimediatechnik und Elektrotechnik/DIT soll den Studenten den Umgang mit verschiedenen Datenbanksystemen, wie *DB2* und *Oracle*, vermitteln. In diesem Rahmen werden verschiedene Themen, wie Datenbankentwurf, Relationales Modell, Datenbankabfragen und WebDB (Java-JDC, PHP), behandelt. Ein wesentlicher themenübergreifender Bestandteil des Praktikums ist das Erlernen der Datenbank-Abfragesprache SQL (vgl. Abschnitt 3.4).

Bisher stehen zur Durchführung des Praktikums feste Zeiten in einem Rechnerpool des Fachbereiches EI zur Verfügung, in denen die Praktikumsaufgaben bearbeitet werden müssen. Mit dem erfolgreichen Abschluß des Praktikums haben die Studenten die Zulassung zur Fachprüfung erreicht. Aus diesem Grunde ist es notwendig, die Leistungen der Praktikumsteilnehmer zu überprüfen und zu bewerten, was bisher im wesentlichen anhand einer schriftlichen Darlegung der Lösungen und durch Anwesenheitskontrollen erfolgte. Die relativ große Anzahl der Praktikumsteilnehmer und die begrenzten Kapazitäten der Rechnerpools erfordern eine gruppenweise Abarbeitung des Praktikums, wodurch sich der Arbeitsaufwand für den Praktikumsbetreuer natürlich entsprechend erhöht. Aufgrund der weiterhin stetig ansteigenden Anzahl der Studenten, ist die Vergabe der Praktikumszeiten mit organisatorischen Schwierigkeiten verbunden, die dazu führen, daß zwischen den Lehrveranstaltungen und den Praktika häufig längere Zeiträume liegen.

Um der im vorangegangenen Absatz beschriebenen Situation zu begegnen, wäre es wünschenswert, eine flexiblere Gestaltung des Praktikums zu ermöglichen und den damit verbundenen Arbeitsaufwand für das Hochschulpersonal zu senken, sowie den Studenten eine flexiblere Gestaltung des Studienablaufs zu ermöglichen. Es muß jedoch weiterhin eine Kontrolle über die im Praktikum erbrachten Leistungen gewährleistet sein, da der erfolgreiche Abschluß des Praktikums die erforderliche Prüfungsvorleistung im Fach Datenbanksysteme darstellt. Die zu entwickelnde

Praktikumsanwendung muß also über eine Nutzerauthentifizierung und eine Loggingfunktion der Nutzeraktivitäten verfügen, um später eine Zuordnung der Praktikumsleistungen gewährleisten zu können. Weiterhin soll eine Abarbeitung der Aufgaben des Praktikums, unabhängig von den Öffnungszeiten der Hochschulpools, ermöglicht werden. Diese Anforderungen können am ehesten durch eine Online-Umsetzung des Praktikums erfüllt werden.

Direkte Effekte einer solchen Art der Praktikumsabarbeitung wären eine erhebliche Reduzierung des Arbeitsaufwandes für die Betreuer und eine deutliche Entlastung der Rechnerpools des Fachbereichs. Zusätzlich könnten die Studenten die Kommunikation mit Datenbanken über das Internet ausprobieren, was ein in der Praxis häufig vorkommendes Szenario darstellt.

## 2.2 Aufbau der Arbeit

Diese Diplomarbeit gliedert sich in neun Kapitel. In Kapitel 1 und Kapitel 2 soll ein schneller Einstieg in das Thema der Arbeit ermöglicht werden. Dazu wird dem Leser ein kurzer Überblick über die momentan vorherrschenden Bedingungen an der Hochschule Wismar gegeben und die Motivation zu diesem Projekt veranschaulicht.

Kapitel 3 behandelt einige grundlegende Fragestellungen, die im Zusammenhang mit dem Thema der Arbeit von Interesse sind. Dabei richtet sich das Hauptaugenmerk auf den Komplex Datenbanken.

Im Kapitel 4 wird das Aufsetzen eines Oracle-DBMS von der Vorbereitung der Installation bis zur Einrichtung des Webzuganges beschrieben. Weiterhin enthält das Kapitel eine Übersicht über die Werkzeuge des *Oracle Enterprise Managers*.

Das Kapitel 5 beschreibt die den Praktikumsaufgaben zugrunde liegende Unternehmenssituation, die dazugehörige Datenbank sowie deren Implementierung.

Im Kapitel 6 werden die einzelnen Anwendungs-Module und deren Funktionsweise vorgestellt. Der Leser wird über den Ablauf und die Aufgabenstellungen des Praktikums informiert und soll eine allgemeine Vorstellung über den Funktionsumfang der Anwendung bekommen.

Kapitel 7 dokumentiert die Arbeitsschritte zur Umsetzung der konzeptionellen Lösung und geht dabei auf die Besonderheiten der einzelnen Module ein. Ein wesentlicher Schwerpunkt dieses Kapitels ist die Funktionsweise des Webinterface über das die Interaktion mit der Anwendung erfolgt.

Das Kapitel 8 beschreibt die Testphase der Anwendung. Dabei steht die Einschätzung der Leistungsfähigkeit des Systems im Mittelpunkt.

### Hinweise zum Layout

In diesem Dokument sind spezielle Bezeichner wie Parameter-, Tabellen-, Spalten- und Dateinamen sowie Beispiele für technische Ausdrücke *kursiv* dargestellt. Diese Schreibweise wird auch für Begriffe verwendet, die im Abkürzungs- und Begriffsverzeichnis der Arbeit aufgeführt sind.

Quellcode, Kommandos, Fehlermeldungen des Systems und Auszüge aus Dateien werden durch die Schriftart `Courier` gekennzeichnet.

# 3 Vorbetrachtungen

Im folgenden werden einige in dieser Diplomarbeit häufig verwendete und mit dem Thema eng verknüpfte Begriffe besprochen. Die einzelnen Abschnitte dieses Kapitels sollen den Einstieg in das Thema der Arbeit erleichtern und eventuell aufkommende Fragen im voraus klären. Die Ausführungen sind an dieser Stelle allgemein gehalten und werden durch Querverweise auf die vertiefenden Abschnitte der einzelnen Themen ergänzt.

## 3.1 E-Learning

Das Medium Internet ist mittlerweile aus dem Hochschulalltag nicht mehr wegzudenken. Ob zur Recherche oder zur Kommunikation mit Professoren und Komillitonen, die Nutzung des Internets vereinfacht verschiedenste Aufgaben und bringt häufig erhebliche Zeitersparnis mit sich. Auch im Bereich E-Learning gibt es bereits eine breite Palette von Angeboten unterschiedlicher Qualität. Eine nähere Untersuchung der Angebote wird in dieser Arbeit nicht vorgenommen. Viele der bestehenden Projekte bieten im Rahmen ihrer virtuellen Lernangebote im Minimum zwei Lernressourcen an. Häufig wird ein gewisser Bestand von Grundlagenwissen exponiert und die Möglichkeit angeboten, in Foren bestimmte Themen gemeinsam zu bearbeiten oder Erfahrungen bei speziellen Problemstellungen auszutauschen. Die effektive Betreuung solcher Foren ist jedoch mit hohem zeitlichen Aufwand verbunden.

Andere Projekte präsentieren sich in Form von Multiple-Choice-Anwendungen, in denen der Nutzer eine bestimmte Anzahl von Fragen durchläuft und im Anschluß eine Einschätzung der erbrachten Leistung erhält. Systeme dieser Art erweisen sich allerdings häufig als sehr statisch und bieten keine detaillierte und direkt verwertbare Fehlerauswertung. Die meisten Anwendungen verfügen über eine nur unzureichende Interaktion mit dem Nutzer und gehen auf eventuell auftretende Fehler überhaupt nicht oder nicht kontextgebunden ein. Die angebotenen Fehlerauswertungen beschränken sich in der Regel auf Prozentangaben der richtigen und falschen Lösungen, die zur Auswertung durch einen Standardtext ergänzt werden.

E-Learning-Systeme können die klassische Form der Lehrveranstaltungen nicht ersetzen, bieten jedoch ein deutlich größeres Potenzial. Durch Interaktion mit dem Nutzer,

direkte Hilfe bei Fehlern beziehungsweise aussagekräftiger Fehlermeldungen und das Angebot von Vorschlägen zu weiterführender Literatur, könnte sich der Lerneffekt solcher Anwendungen weiter steigern.

## 3.2 Webfähige relationale Datenbank-Managementsysteme

Ein relationales Datenbank-Managementsystem (RDBMS) ist eine Sammlung von Programmen, die die anwendungsunabhängige dauerhafte Speicherung von Daten in einer Datenbank ermöglicht und die damit zusammenhängende Verwaltung und Autorisierungsprüfung übernimmt. Mit einem *RDBMS* wird die Sicherheit, Integrität und Konsistenz der Daten bei minimaler, kontrollierter Redundanz gewährleistet. Für die Datendefinition, -manipulation und -selektion steht die Abfragesprache *SQL*, auf die im Abschnitt 3.4 näher eingegangen wird, zur Verfügung. Bekannte relationale Datenbank-Managementsysteme sind beispielsweise:

- IBM DB/2
- Oracle
- PostgreSQL
- MySQL

In dieser Arbeit wird das *Oracle9i Datenbank-Managementsystem* verwendet. Das Kapitel 4 enthält eine ausführliche Beschreibung dieser Version des *Oracle-RDBMS*.

Webfähige relationale Datenbank-Managementsysteme bieten zusätzlich zu den oben genannten Merkmalen auch die Fähigkeit zum Datenaustausch mit bestimmten Anwendungen über ein Netzwerk. Über das Netz können Datenbankinhalte abgefragt oder administrative Aufgaben durchgeführt werden. Weiterhin ist es möglich, verteilte Datenbanksysteme zu betreiben und einheitlich zu verwalten.

Zur Kommunikation mit einem solchen Datenbank-Managementsystem über ein Netzwerk ist jedoch ein zusätzliches *API* nötig. Bei *Oracle* übernimmt diese Aufgabe das *OCI* (**O**racle **C**all **I**nterface), über das die Verbindung zum *Oracle-Server* von einer beliebigen Programmiersprache aus realisiert wird. Auf die Funktionsweise dieser Schnittstelle wird in den Abschnitten 4.7 und 7.5 genauer eingegangen.

## 3.3 Das Datenbank-Managementsystem Oracle

Oracle ist ein sehr leistungsfähiges, objekt-relationales Datenbank-Managementsystem. Es gilt unangefochten als Marktführer und ist für eine

Vielzahl von Betriebssystemen erhältlich. Die aktuelle Version (August 2004) trägt die Bezeichnung 10g. Neben dem Datenbanksystem stellt Oracle auch sehr viele Zusatzkomponenten, bis hin zur Business-Software zur Verfügung. Da in Oracle-Datenbanksystemen schon vor der Definition des *SQL-92-Standards* viele der darin standardisierten Leistungsmerkmale implementiert waren, gibt es gewisse Defizite in der Standardkonformität. Diese Abweichungen erschweren den Umstieg von oder zu Oracle. Hier einige Beispiele:

- Der Datentyp *date* umfaßt bei Oracle neben einem Datum auch eine Uhrzeit. Dafür gibt es im SQL-Standard den *timestamp*, den Oracle in Version 9 ebenfalls eingeführt hat. Leider widerspricht der Type *date* immer noch diesem Standard, was in der Praxis zu vielen nicht gerade trivialen Problemen führt.
- Das aktuelle Datum heißt *SYSDATE* und nicht *CURRENT_DATE*.
- Es gibt keinen reinen Zeit-Datentyp *time* (ohne Datum).
- Innerhalb von *CHECK-Bedingungsprüfungen* kann nicht direkt mit einem aktuellen Datum verglichen werden.
- Erst ab Version 9 findet die seit 1992 genormte Formulierung *NATURAL JOIN* Verwendung.
- In *UNIONs* ist der Wert *NULL* nur mit Zeichenketten, statt mit allen Datentypen, kompatibel.

Zusätzlich bietet Oracle mit *SQL*Plus* (vgl. Abschnitt 3.5.1) eine Erweiterung des SQL-Standards zur Administration des Datenbanksystem und unterstützt die Erstellung von *Stored Procedures* in *PL/SQL* (vgl. Abschnitt 3.5.2) beziehungsweise ab der Version 8i auch in *Java*. Neben *Stored Procedures* bietet Oracle auch *Trigger* an. Zur Sicherstellung der referentiellen Integrität unterstützt Oracle die Einrichtung von *Constraints*. Weiterhin lassen sich sogenannte *Functional Indices* realisieren, bei denen mit Hilfe von PL/SQL oder Java berechnete Indizes auf Tabellenspalten angelegt werden können.

## 3.4 Strukturierte Abfragesprache - SQL

*SQL* (**S**tructured **Q**uery **L**anguage) bildet die sprachliche Basis für die Kommunikation mit einer Datenbank. Die Abfragesprache wurde in den siebziger Jahren von *IBM* für die relationale Datenbank *DB2* entwickelt. Das *ANSI* (**A**merican **N**ational

**S**tandards **I**nstitute) beauftragte 1982 die Standardisierung einer relationalen Sprache, die 1986 ratifiziert wurde und zum größten Teil aus dem IBM-Dialekt von *SQL* bestand. Der *SQL*-Standard teilt sich grob in drei Bereiche auf:

- DDL (Data Definition Language) - Anweisungen zur Definition von Datenstrukturen. In diesem Bereich sind Befehle, wie CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE VIEW, CREATE INDEX und viele mehr anzusiedeln.
- DML (Data Manipulation Language) - Anweisungen zum Suchen und zur Veränderung von Daten. Hierzu gehören Befehle, wie INSERT, INTO, UPDATE, DELETE und SELECT
- DCL (Data Control Language) - weitere Anweisungen, um das DBMS bedienen und den ordnungsgemäßen Betrieb des DBMS gewährleisten zu können. In diesen Bereich gehören insbesondere Anweisungen zur Definition von Zugriffsrechten (GRANT, REVOKE) beziehungsweise zur Transaktionskontrolle (SET TRANSACTION, COMMIT, ROLLBACK).

*SQL* ist keine Programmiersprache im herkömmlichen Sinne, da sie keine prozeduralen Funktionen bietet, mit denen der Ablauf eines Programms gesteuert werden kann. Oracle hat mit *SQL*Plus* diesen Standard um viele nützliche Funktionen erweitert, um prozedurale Codeabschnitte direkt in der Datenbank speichern zu können und in *SQL*-Abfragen zu integrieren. *SQL* bei Oracle teilt sich in zwei Sprachen auf:

- SQL*Plus - Oracles Erweiterung des SQL - Standards
- PL/SQL - die prozedurale Abfragesprache von Oracle

Diese Erweiterungen erlauben eine sehr viel effektivere und benutzerfreundlichere Arbeit mit der Datenbank, beispielsweise wenn Abfragen direkt in Programmabläufe eingebettet werden und vorher definierte Bedingungen zu erfüllen sind.

## 3.5 Die Oracle SQL-Erweiterungen

Dieser Abschnitt gibt einen allgemeinen Überblick über die Funktionalität der Oracle SQL-Erweiterungen *SQL*Plus* und *PL/SQL*. Der praktische Einsatz dieser Werkzeuge wird teilweise im Kapitel 4 an Beispielen erläutert.

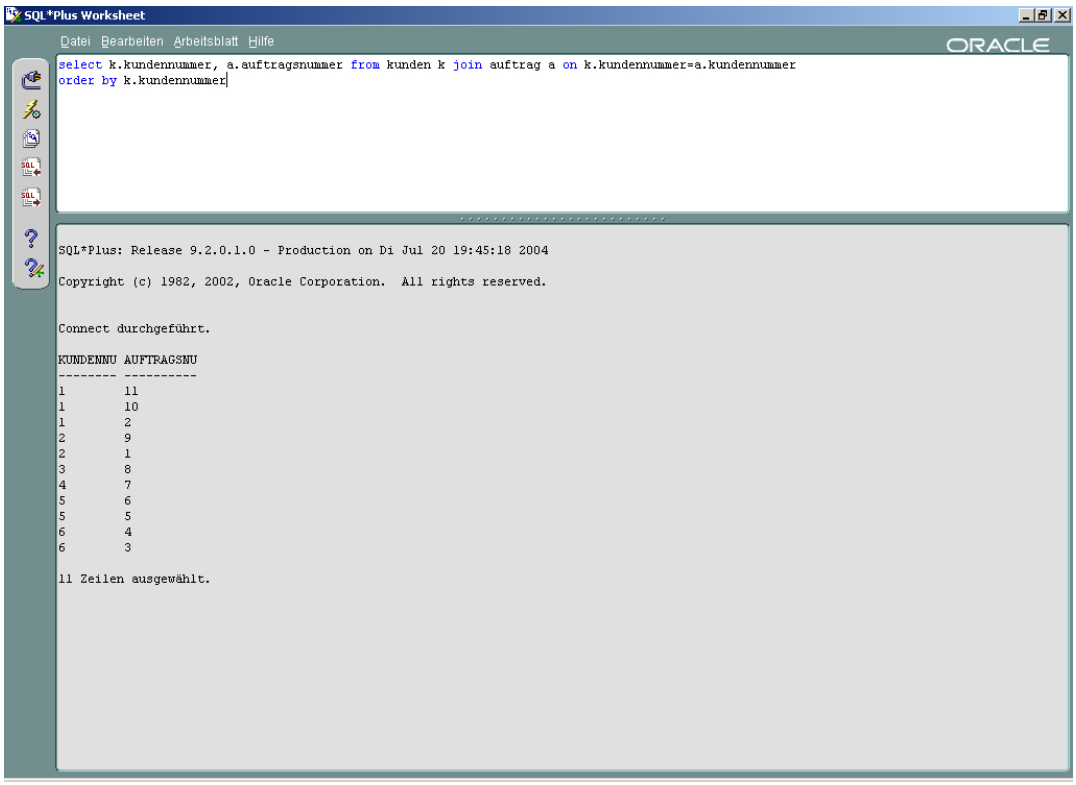



**Abbildung 3.1:** Oberfläche des SQL*Plus Worksheet von Oracle9i

### 3.5.1 Die Erweiterung des SQL-Standards zu SQL*Plus

*SQL*Plus*, die primäre Schnittstelle zum Oracle Database Server, ist ein mächtiges leicht zu handhabendes Kommandozeilen-Werkzeug zur Realisierung von Abfragen, zur Datendefinition und Datenkontrolle. Die Bezeichnung '**Plus*' deutet auf die Erweiterungen des Standard-SQL hin, die seit der Version Oracle9i auch das Anpassen von Berichten, das Editieren und Speichern von Daten, das Definieren von Variablen und sogar administrative Aufgaben, wie das Anlegen von Benutzern ermöglichen. *SQL*Plus* kann entweder direkt aus der *shell* oder aber mit dem *SQL*Plus Worksheet* des *OEM* (vgl. Abschnitt 4.6) gestartet werden. Mit dem in Abbildung 3.1 dargestellten *SQL*Plus Worksheet* steht dem Nutzer eine grafische Version des SQL-Interpreters zur Verfügung. Einige nützliche Hilfsmittel, wie die Befehlshistorie oder das Speichern von Ausgaben in Dateien, werden durch die Nutzung dieses Werkzeuges wesentlich bequemer.

### 3.5.2 Prozedurales SQL - PL/SQL

Oracle bietet mit *PL/SQL* [1] eine prozedurale Spracherweiterung des SQL-Standards, mit der es möglich ist, Anwendungslogik, die sich mit Datenbankzugriffen

und Datenverarbeitung beschäftigt, in der Datenbank zu belassen und von außen mit unterschiedlichen Mechanismen zuzugreifen. *PL/SQL* kann in der relationalen Datenbank *Oracle*, im *Oracle Server* und in clientseitigen Werkzeugen zur Anwendungsentwicklung komplexer Datenbank-Konzeptionen verwendet werden. Folgende Konzepte können hierbei genutzt werden:

- Prozeduren, Funktionen und Pakete:

  Mit PL/SQL besteht die Möglichkeit, Unterprogramme in Form von Prozeduren, welche auf Funktionen oder andere Prozeduren zurückgreifen können, abzuspeichern und dadurch wiederkehrende Routinen in einmalige Strukturen auszulagern.

- Ausnahmebehandlung:

  Ausnahmen sind Programmabläufe, die unerwünschte Zustände in einer Datenbank zur Folge haben. Ein solcher Programmablauf kann zu Inkonsistenzen im Datenbestand oder in der Datenstruktur führen. Derartige Situationen erfordern eine geeignete Behandlung, die entweder zu einem Abbruch der Transaktion oder zu einem gewünschten Datenbank-Zustand führen.

- Trigger-Programmierung:

  Trigger ähneln den Ausnahmebehandlungen, da auch sie verhindern sollen, daß Inkonsistenzen in Datenstruktur oder Dateninhalt entstehen. Allerdings lassen sie sich nicht explizit aufrufen und in den Programmablauf einbauen. Vielmehr stehen sie ständig bereit, um bei beliebigen Situationen, in denen unerwünschte Ereignisse eintreten können, durch geeignete Routinen konsistente Datenbank-Zustände herzustellen.

- Cursor-Behandlung:

  Über das Cursor-Konzept lassen sich auch größere Datenmengen wesentlich angenehmer und eleganter verarbeiten oder ausgeben. Es wurde entwickelt, um die zeilenweise Abarbeitung von Abfrageergebnissen zu ermöglichen.

- XML-Unterstützung:

  Mit PL/SQL ist es möglich, XML-Dokumente zu parsen, zu untersuchen oder in unterschiedliche Strukturen umzuwandeln. Dies können sowohl weitere XML-Dokumente, als auch andere Dokumentformate sein.

PL/SQL-Programme sind in logische Codeblöcke unterteilt. Ein Block wiederum besteht in der Regel aus drei Abschnitten. Eingeleitet wird ein solches Programm von dem optionalen *Deklarations-Abschnitt* zur Definition von Variablen, Cursorn oder

Konstanten. Der obligatorische Teil eines solchen Blocks ist der *Prozedur-Abschnitt*, der Bedingungsbefehle sowie SQL-Anweisungen enthält. In einem weiteren optionalen Block, dem *Exception-Abschnitt* wird festgelegt, wie bestimmte Fehler und benutzerdefinierte Ausnahmen (Exceptions) zu behandeln sind.

# 4 Oracle9i

Das Oracle9i Datenbank-Managementsystem wurde speziell für das Internet konzipiert und ist in drei verschiedenen Editionen erhältlich:

- Enterprise
- Standard
- Personal

Die Versionen unterscheiden sich in ihrem Funktionsumfang, wobei die Enterprise-Edition die leistungsfähigste Version, mit allen Features, darstellt. Dieses Kapitel beschreibt die notwendigen Arbeitsschritte zur Installation von Oracle9i in der Enterprise-Edition unter SuSE-Linux 8.0 und die Konfiguration zu einem webfähigen *RDBMS*. In dem Abschnitt 4.3.2 finden sich unter anderem detaillierte Informationen zur Namensauflösung bei Oracle. Im Abschnitt 4.6 soll dem Leser ein Überblick über die Funktionsvielfalt des Oracle9i-Datenbank-Systems vermittelt werden.

## 4.1 Installation

### 4.1.1 Systemvoraussetzungen

Vor der Installation von Oracle9i müssen die Systemvoraussetzungen überprüft und gegebenenfalls einige Anpassungen im Betriebssystem vorgenommen werden. Eine Oracle9i-Installation unter Linux setzt ein funktionierendes X-Windows-System voraus. Der *OUI* enthält eine eigene *JRE* (**J**ava **R**untime **E**nvironment), die eventuell nicht mit einer, bereits auf dem System installierten, *JRE*, harmoniert. In diesem Fall ist die vorhandene Version zu deinstallieren, da ansonsten der *OUI* nicht gestartet werden kann.

Oracle ist ein Schwergewicht unter den Datenbank-Managementsystemen und erfordert für eine typische Installation ungefähr 3.5 GB Speicher und zusätzlich bis zu 400MB temporären Speicherplatz, die der *OUI* während der Installation benötigt. Weitere entscheidende Kriterien für eine erfolgreiche Installation sind unter anderem

die Größe des temporären Plattenspeicherplatzes und die Kernelparameter *SHMMAX* für Shared Memory, sowie *SEMMSL* und *SEMMNI* für das Semaphorenmanagement. Oracle stellt umfassende Dokumentationen [3] zur Abarbeitung dieser Schritte bereit. Besonders hilfreich ist in diesem Zusammenhang der *Oracle9i Installation Guide*.

| **Hardware** | **Mindestanforderungen** |
|---|---|
| RAM | 512 MB (256 MB für den Client) |
| Swap Space | mindestens wie RAM, besser 1 GB |
| Festplattenspeicher | 3.5 GB für die typische Installation |

**Tabelle 4.1:** Mindestanforderungen für die Installation

Oracle empfiehlt den Kernelparameter *SHMMAX* dauerhaft zu erhöhen, um so die *IPC* (**I**nter **P**rocess **C**ommunication) zu optimieren und damit eine gute Performance der Datenbank zu gewährleisten. Shared-Memory-Segmente werden für das *SGA* (**S**hared **G**lobal **A**rea) benötigt, einem Speicherbereich, der bei Oracle von allen Vorder- und Hintergrundprozessen zur Prozeßkommunikation genutzt wird. Um alle Shared-Memory-Settings zu überprüfen, kann der Befehl `ipcs -lm` genutzt werden. Der Default-Wert für *SHMMAX* kann wie folgt geändert werden:

```
su - root
sysctl -w kernel.shmmax=2147483648
```

Semaphore werden für die Prozeß-Synchronisation genutzt. Mit dem Befehl `ipcs -ls` können die aktuellen Semaphor-Einstellungen des Systems abgefragt werden. Das Semaphorenmanagement erfolgt mit Hilfe von sogenannten Sets - also in logischen Gruppen zusammengefasste Semaphore. Der Kernel-Parameter *SEMMNI* definiert systemweit die maximale Anzahl der Semaphorsets. Oracle erfordert einen Mindestwert von *100*. Jedes Set wiederum kann eine bestimmte Anzahl von Semaphoren enthalten, die im Parameter *SEMMSL* festgelegt wird. Auch hier sollte die Mindestanzahl von *100* Semaphoren pro Set nicht unterschritten werden.

In dem hier beschriebenen Beispiel waren keine weiteren Systemanpassungen mehr nötig und die Installation konnte problemlos durchgeführt werden. Weiterführende hilfreiche Informationen zu dem Thema finden sie im Webseitenverzeichnis unter 'Systemanpassungen vor der Installation' [4].

### 4.1.2 Einrichten von Gruppen und Nutzern

Zur Installation und zum Betrieb von Oracle9i sind zusätzlich zwei Gruppen und ein Benutzer anzulegen.

1. Gruppe OSDBA (dba) für Datenbankadministratoren
2. Gruppe OSOPER (oper) für normale Nutzer
3. Benutzer oracle für die Installation

Zuvor sollte allerdings überprüft werden, ob die oben genannten Gruppen nicht bereits vorhanden sind, da neuere SuSE-Distributionen diese schon standardmässig anlegen. Im nächsten Schritt muß dafür gesorgt werden, daß *umask* für den Benutzer *oracle* auf *022* gesetzt ist, damit die Gruppe des Benutzers und die Gruppe *others* grundsätzlich Lese- und Ausführungsrechte erhalten. Die schon existierenden Gruppen und Nutzer sind in den Dateien */etc/group* bzw. */etc/passwd* abgelegt. Im vorliegenden System sind unter anderem nachfolgende Einträge vorhanden:

```
dba:x:500:
oper:x:501:oracle
```
und
```
oracle:x:500:500::/home/oracle:/bin/bash
```

Der Benutzer *oracle* hat die User-ID *500* und gehört primär der Gruppe *500* (*dba*) an, weiterhin gehört er auch noch der Gruppe *501* (*oper*) an.

### 4.1.3 Setzen der Systemvariablen

Während der Installation werden verschiedene relevante Systemvariablen abgefragt, die in der Vorbereitung wenigstens für den user *oracle* gesetzt werden müssen.

1. ORACLE_BASE - Wurzelverzeichnis, in dem Oracle9i installiert wird
2. ORACLE_HOME - hier befindet sich die Software der jeweiligen Version von Oracle9i - so wird gewährleistet, daß Applikationen aus unterschiedlichen Distributionen zum Einsatz kommen
3. ORACLE_SID - **O**racle **S**ervice **I**dentifier - der Name, der bei der Installation angelegten Datenbank
4. NLS_LANG - Variable zur Spracheinstellung (zum Beispiel für die Nutzung deutscher Umlaute)

Es bietet sich außerdem an, *$PATH* um *$ORACLE_HOME/bin* zu erweitern, damit die Applikationen direkt ausgeführt werden können.

### 4.1.4 Installation

Seit *Oracle8i* ist die Installation nur mehr grafisch mit dem *OUI* möglich. Der *OUI* kann nur von dem für diesen Zweck eingerichteten Benutzer *oracle* gestartet werden. Der Nutzer wird nun interaktiv durch die Installation geführt und muß in deren Verlauf einige Angaben zur Grundkonfiguration machen. Durch die im Vorfeld gesetzten Systemvariablen *$ORACLE_BASE*, *$ORACLE_HOME* und *$ORACLE_SID* erhält der Installer die Informationen, in welche Verzeichnisse kopiert und unter welchem Namen, die bei der Installation angelegte Datenbankinstanz, gespeichert werden soll. Außerdem ist die Festlegung des Installationsumfangs notwendig. Im weiteren Verlauf der Installation wird man durch den Installer aufgefordert, ein Shellskript auszuführen, um ein Bestandsverzeichnis unter */etc* anzulegen. Um dieses Skript auszuführen, sind *root*-Rechte notwendig. Eine detaillierte Beschreibung der Installation wird von Oracle als PDF-Dokument (Oracle9i Installation Guide) auf der Oracle Download-Seite [3] angeboten.

### 4.1.5 Fehlerkorrektur während der Installation

In dem hier beschriebenen Beispiel kam es trotz Einhaltung aller aufgeführten Regeln zu Problemen während der Installation. Der erste Fehler tritt aufgrund eines unvollständigen *makefiles* auf. Die laufende Installation wird angehalten und folgende Fehlermeldung ausgegeben:

```
Error in invoking target install of makefile
$ORACLE_HOME/ctx/lib/ins_ctx.mk
```

In der Datei *$ORACLE_HOME/ctx/lib/env_ctx.mk* fehlt unter dem Punkt *INSO_LINK* der Eintrag *$(LDLIBFLAG)dl*.

Der vollständige Eintrag lautet:

```
INSO_LINK = -L$(CTXLIB) $(LDLIBFLAG)m $(LDLIBFLAG)dl
$(LDLIBFLAG)sc_ca $(LDLIBFLAG)sc_fa $(LDLIBFLAG)sc_ex $(LDLIBFLAG)sc_da
$(LDLIBFLAG)sc_ut $(LDLIBFLAG)sc_ch $(LDLIBFLAG)sc_fi $(LLIBCTXHX)
$(LDLIBFLAG)c -Wl,-rpath,$(CTXHOME)lib $(CORELIBS) $(COMPEOBJS)
```

Wird die Installation anschließend mit `retry` fortgesetzt, kommt es zu keinem Abbruch mehr und die Installation wird erfolgreich abgeschlossen.

### 4.1.6 Fehlerkorrektur nach der Installation

Nach der Installation werden verschiedene Datenbanktools gestartet, die die Konfiguration der Datenbank vereinfachen sollen. So wird in dieser Phase erfolglos ver-

sucht, den *dbca* (**D**ata**b**ase **C**reation **A**ssistant) zu starten. Abhilfe schafft eine kleine Änderung der Datei *$ORACLE_HOME/bin/dbca*. In dem hier beschriebenen Beispiel reichte es aus, die letzten Zeilen der Datei, wie im folgenden auszukommentieren, um das Problem zu beheben.

```
#if [-f /etc/rac_on]; then
# Run DBCA $JRE_DIR/bin/jre -native -DORACLE_HOME=$OH
-DJDBC_PROTOCOL=thin -mx64m -classpath $CLASSPATH
oracle.sysman.assistants.dbca.Dbc a $ARGUMENTS
#else
# Run DBCA
#$JRE_DIR/bin/jre -DORACLE_HOME=$OH -DJDBC_PROTOCOL=thin -mx64m
-classpath $CLASSPATH oracle.sysman.assistants.dbca.Dbca $ARGUMENTS
#fi
```

Eine wesentliche Voraussetzung für die Arbeit mit den Konfigurationstools ist die Größe des Kernelparameters *SHMMAX*. Dieser sollte mindestens 1GB betragen, da es anderenfalls während der Ausführung der Tools zu Oracle-Fehlermeldungen kommt.

## 4.2 Änderung der Standard-Paßwörter

Der *dbca* installiert nun die Datenbank mit der, vor der Installation, festgelegten *SID* und richtet während dessen automatisch die Standardaccounts für den Zugang zum Datenbanksystem ein. Die Paßwörter dieser Accounts sollten aus Sicherheitsgründen direkt nach der Installation geändert werden. Es handelt sich hierbei um folgende Nutzer:

- sys (Paßwort: change on install)
- system (Paßwort: manager)
- scott (Paßwort: tiger)

Die Nutzer *sys* und *system* sind DBA-Accounts, *scott* ist normaler Datenbankbenutzer mit Connect- und Ressource-Rechten. Detaillierte Informationen zur Nutzerverwaltung und Rechtevergabe finden sie im Abschnitt 4.4.

## 4.3 Netzkonfiguration

### 4.3.1 Netzwerkkonfiguration - lsnrctrl

Für die Anbindung des Oracle-Datenbank-Servers ist das Zusatzpaket *Oracle Net Services* (vgl. Abschnitt 4.7) zuständig, welches zur Oracle9i-Standardinstallation gehört. Damit der Oracle9i Enterprise Server auch im Netz durch Clients ansprechbar ist, muß ein sogenannter *Listener* aktiviert werden, der dann an einem bestimmten *tcp-Port* auf Anfragen wartet und die Kommunikation zwischen dem Datenbank-Server und den Clients ermöglicht. Der Standardport ist *1521/tcp*. Der *OUI* hat während der Installation in der Datei */etc/services* eingetragen, auf welchem Port der *Listener* tatsächlich arbeitet. Der *Listener* wird über das Programm *lsnrctl* (Listener Control) aktiviert und konfiguriert. *lsnrctl* ist ein textbasiertes Werkzeug zur Netzwerk-Administration und wird aus der *shell* gestartet. Es erfolgt folgende Ausgabe:

```
LSNRCTL for Linux: Version 9.2.0.1.0 - Production on 17-
   JUL-2004 10:01:58

Copyright (c) 1991, 2002, Oracle Corporation.  All rights
   reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL>
```

Ist *Oracle Net Services* im Zuge der Installation durch den *OUI* automatisch eingerichtet worden, erfolgt zum Start des Listeners lediglich die Eingabe des Befehls '*start*'. Folgende Ausgabe erscheint:

```
Starting /opt/oracle/9i/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 9.2.0.1.0 - Production
System parameter file is /opt/oracle/9i/network/admin/
   listener.ora
Log messages written to /opt/oracle/9i/network/log/
   listener.log
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=
   EXTPROC)))
```

```
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=
   athos.pool1.et.hs-wismar
.de)(PORT=1521)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=
   EXTPROC)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version
   9.2.0.1.0 - Production
Start Date                17-JUL-2004 10:02:11
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  OFF
SNMP                      OFF
Listener Parameter File   /opt/oracle/9i/network/admin/
   listener.ora
Listener Log File         /opt/oracle/9i/network/log/
   listener.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=athos.pool1.et
     .hs-wismar.de)(PORT=15
21)))
Services Summary...
Service "OEMREP" has 1 instance(s).
  Instance "OEMREP", status UNKNOWN, has 1 handler(s) for
     this service...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s)
     for this service...
Service "dbprakt" has 1 instance(s).
  Instance "dbprakt", status UNKNOWN, has 1 handler(s) for
      this service...
The command completed successfully
```

Damit ist der *Listener* gestartet und der Oracle9i-Datenbankserver kann von entsprechend konfigurierten Clients aus dem Netz angesprochen werden. Die

Ausgabe beim Start des Listeners erzeugt eine Statusanzeige, in der diverse Informationen über das interne Verhalten der Datenbank bei Anfragen von Clients gelistet werden. Dieses Verhalten wird in der Netzwerkkonfigurationsdatei *$ORACLE_HOME/network/admin/listener.ora* festgelegt, die bei der Namensauflösung bei Oracle eine maßgebliche Rolle spielt.

### 4.3.2 Namensauflösung unter Oracle

Damit ein Client sich mit einer Datenbank verbinden kann, muß dieser den Namen der Datenbank auflösen können. Oracle stellt hierfür verschiedene Methoden zur Verfügung:

- Host Naming (HOSTNAME)
- Local Naming (TNSNAMES)
- Oracle Name Server (ORANAMES)
- Directory Naming
- Externe Benennung

In kleineren Anwendungen wird meist das Local Naming oder Host Naming verwendet. Bei komplexeren Organisationsformen wird dagegen eher Directory Naming mittels *LDAP*-kompatiblen Directory Server eingesetzt, worauf an dieser Stelle jedoch nicht weiter eingegangen werden soll.

Welche dieser Methoden eingesetzt wird, ermittelt Oracle anhand der Datei *$ORACLE_HOME/network/admin/sqlnet.ora*, in der sich folgende Angaben finden:

```
# SQLNET.ORA Network Configuration File: /opt/oracle/9i/
   network/admin/sqlnet.ora
# Generated by Oracle configuration tools.

NAMES.DEFAULT_DOMAIN = pool1.et.hs-wismar.de

NAMES.DIRECTORY_PATH= (TNSNAMES, ONAMES, HOSTNAME)
```

Gemäß den Einstellungen für `NAMES.DIRECTORY_PATH` wird in dem hier besprochenen Beispiel als erstes die Namensauflösung mit Hilfe von Local Naming, anschließend mit Hilfe eines Oracle Name Servers und als letztes durch Hostnaming durchgeführt werden.

*Host Naming* ermöglicht Benutzern in einer TCP/IP-Umgebung die Auflösung von Host-Namen mit Hilfe bestehender Dienste der Namensauflösung (*DNS, /etc/hosts*). Clients stellen die Verbindung mittels Host-Name zu einem Oracle Datenbank-Server über die Software *Oracle Net Service Client* her. Wenn mehrere *SID* pro Maschine angesprochen werden sollen, müssen jeweils eigene Host-Namen (Alias) für jede *SID* definiert werden.

*Local Naming* ist die einfachste und wohl am häufigsten verbreitete Methode der Namensauflösung eines Clients und wird auch in dem hier beschriebenen Beispiel genutzt. Die für die Namensauflösung notwendigen Informationen befinden sich in der Datei *$ORACLE_HOME/network/admin/tnsnames.ora* - hier ein Beispiel:

```
# TNSNAMES.ORA Network Configuration File: /opt/oracle/9i/
   network/admin/tnsnames.ora
# Generated by Oracle configuration tools.

DBPRAKT.POOL1.ET.HS-WISMAR.DE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = athos.pool1.et.hs
         -wismar.de)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = dbprakt)
    )
  )
)
```

Die Datenbank befindet sich auf einem Server namens `athos.pool1.et.hs-wismar.de`. Anstelle des Namens kann auch eine IP-Adresse angegeben werden. Die Verbindung zur Datenbank wird über den Port `1521` hergestellt. Über `CONNECT_DATA` werden die Verbindungsdaten festgelegt. In unserem Beispiel handelt es sich um eine *dedizierte* Verbindung und der Service-Name der Datenbank auf dem Server ist `dbprakt`. Der lokale NetServiceName ist in diesem Beispiel `DBPRAKT.POOL1.ET.HS-WISMAR.DE`. Das bedeutet, daß der Client bei der Verbindung dies als Service-Name angeben muß. Die korrekte Namensauflösung kann mit dem Programm *tnsping* getestet werden. Hier wird nun überprüft, ob der Service-Name aufgelöst werden kann und ob der entsprechende Port bereit ist.

```
oracle@athos:~> tnsping dbprakt

TNS Ping Utility for Linux: Version 9.2.0.1.0 - Production
    on 17-JUL-2004 12:17:14

Copyright (c) 1997 Oracle Corporation.  All rights
   reserved.

Used parameter files:
/opt/oracle/9i/network/admin/sqlnet.ora


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (
   ADDRESS = (PROTOCOL = TCP)(HOST = athos.pool1.et.hs-
   wismar.de)(PORT = 1521))) (CONNECT_DATA = (SERVER =
   DEDICATED) (SERVICE_NAME = dbprakt)))
OK (20 msec)
```

*Oracle Name Server* ist eine zentralisierte Methode der Namensauflösung eines Clients. Bei der Namensauflösung mit Hilfe von Oracle Name Service wird neben dem `NAMES.DIRECTORY_PATH` zusätzlich ein bevorzugter Nameserver angegeben, der speziell für diesen Zweck gestartet und konfiguriert werden muß. Bei Auflösungsaufträgen wird dann dieser Server gefragt.

*Directory Naming* löst den Namen eines Datenbankdienstes oder Net Service in einem Connect-Descriptor auf, der in einem zentralen *LDAP*-kompatiblen Directory-Server gespeichert ist.

*Externe Benennung* verwendet einen unterstützten Benennungsdienst Dritter.

## 4.4 Anlegen zusätzlicher Nutzer

Um weiteren Benutzern den Zugang zur Datenbank zu ermöglichen, müssen zuvor entsprechende Accounts eingerichtet werden. Das Einrichten dieser Accounts muß vom *DBA* oder einem Benutzer mit *DBA-Rechten* vorgenommen werden. Oracle bietet eine Vielzahl von grafischen und textbasierten Administrationswerkzeugen (vgl. Abschnitt 4.6), die auch Einsteigern das Durchführen solcher administrativen Aufgaben ermöglichen beziehungsweise vereinfachen sollen. In dem nun folgenden Beispiel

wird das Anlegen eines neuen Datenbankbenutzers unter Verwendung von *SQL*Plus* gezeigt:

```
oracle@athos:/> sqlplus

SQL*Plus: Release 9.2.0.1.0 - Production on Sat Jul 17
   14:38:22 2004

Copyright (c) 1982, 2002, Oracle Corporation.  All rights
   reserved.

Enter user-name: sys as sysdba
Enter password:

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production

SQL> create user ora identified by orapass;
User created
```

Versucht sich der so angelegte Nutzer `ora` nun an der Datenbank anzumelden, wird er jedoch, wie im folgenden Beispiel dargestellt, abgewiesen:

```
ERROR:
ORA-01045: user ORA lacks CREATE SESSION privilege;
logon denied
```

Um eine erfolgreiche Anmeldung zu ermöglichen, müssen dem Nutzer noch entsprechende Rechte zugewiesen werden. Die Rechtevergabe ist ein wesentlicher und sicherheitsrelevanter Bestandteil der Nutzerverwaltung.

## 4.5 Rollen und Berechtigungen

Oracle faßt die verschiedenen charakteristischen Berechtigungen typischer Nutzerprofile in drei Gruppen - so genannten *Rollen* - zusammen: *CONNECT*, *RESOURCE*

und *DBA*. Diese drei Rollen beinhalten eine Reihe von Privilegien, welche für die sinnvolle Arbeit mit der Datenbank in vordefinierten Paketen zusammengefaßt wurden. Bei den Privilegien unterscheidet Oracle zusätzlich zwischen System- und Objektberechtigungen. Jede Systemberechtigung ermöglicht einem Benutzer eine bestimmte Datenbank-Operation auszuführen. Operationen können Aktionen, wie das Erstellen, Löschen und Ändern von Tabellen, Views, Rollback Segmenten und Prozeduren umfassen. Durch Objektberechtigungen kann ein Benutzer eine bestimmte Aktion auf ein spezielles Objekt, wie eine Tabelle, View, Sequenz, Prozedur, Funktion oder Package ausführen.

Die Rollen-Pakete sind jedoch nicht statisch. Es können jederzeit Einzelrechte aus anderen Rollen hinzugefügt werden oder aus der Rolle entfernt werden. Die nachfolgende Tabelle beinhaltet einige wichtige von Oracle vordefinierte Rollen und Privilegien.

| **Rolle** | **Rechte/Privilegien** |
|---|---|
| CONNECT | ALTER SESSION<br>CREATE TABLE<br>CREATE CLUSTER<br>CREATE DATABASE LINK<br>CREATE SEQUENCE<br>CREATE SESSION<br>CREATE SYNONYM<br>CREATE VIEW |
| RECOURCE | CREATE TABLE<br>CREATE PROCEDURE<br>CREATE SEQUENCE<br>CREATE TRIGGER<br>CREATE CLUSTER |
| DBA | Alle Systemrechte mit Adminoption<br>Rolle EXP_FULL_DATABASE<br>Rolle IMP_FULL_DATABASE |
| EXP_FULL_DATABASE | SELECT ANY TABLE<br>BACKUP ANY TABLE<br>INSERT, UPDATE, DELETE ON<br>(Systemtabelle) |
| IMP_FULL_DATABASE | BECOME USER<br>WRITE DOWN |

**Tabelle 4.2:** Oracle's Standardrollen und Privilegien

Aktives Arbeiten mit der Datenbank ist nur dem Benutzer möglich, der die Rollen *CONNECT* und *RESOURCE* besitzt, mindestens muß jedoch die *CONNECT*-Berechtigung vorliegen, damit eine Verbindung zur Datenbank zustande kommt. Andernfalls kommt es zu einer Fehlermeldung. Selbstverständlich ist der Besitzer von *CREATE*-Rechten auch zum Löschen der von ihm angelegten Objekte berechtigt. Die Rollen *IMP_FULL_DATABASE* und *EXP_FULL_DATABASE* berechtigen zum Im- und Export ganzer Datenbankinhalte. Diese beiden Rollen werden in der Regel nur Nutzern erteilt, die besondere Aufgaben zur Verwaltung und Sicherung der Datenbank wahrnehmen sollen.

Die Rechtevergabe mit SQL*Plus erfolgt mit dem grant - Befehl:

```
SQL> grant connect, resource to ora;
Benutzerzugriff (Grant) wurde erteilt.
```

Zum Löschen von Nutzern wird der drop - Befehl benutzt:

```
SQL> drop user ora cascade;
User dropped.
```

Durch das Schlüsselwort *cascade* werden sämtliche Datenbankobjekte, die dem Nutzer zugeordnet sind, gelöscht.

Grundsätzlich werden Berechtigungen mit dem grant-Befehl vergeben und mit dem revoke-Befehl entzogen. Wird die Option `with admin option` an den Befehl angehängt, erhält der Nutzer die Möglichkeit, seine Berechtigungen an einen anderen Benutzer oder an eine Rolle weiterzugeben.

## 4.6 Oracle Enterprise Manager

Mit dem *OEM* beziehungsweise der ***O**racle **E**nterprise **M**anager-Konsole* steht dem Nutzer ein umfangreiches grafisches Werkzeug (siehe auch Abbildung 4.6) zur Verfügung, mit dem die Verwaltung einer oder mehrerer Datenbanken und alle damit verbundenen Aufgaben vereinfacht werden sollen. Die Komponenten des Enterprise Managers sind thematisch zu Paketen zusammengefaßt. Jedes dieser Pakete gruppiert wiederum verschiedene Werkzeuge zur Arbeit mit dem Datenbank-Managementsystem, deren Aufgaben und Bestandteile im folgenden beschrieben werden:

1. Diagnostic Pack
2. Tuning Pack

3. Configuration Management Pack
4. Change Management Pack

Das **Oracle Diagnostics Pack** bietet eine umfassende Leistungsüberwachung und Fehlerbenachrichtigung inklusive automatischer diagnostischer Datenbanküberwachung, Rechnerlast-Repository und Blackoutplanung. Zum *Diagnostic Pack* gehören:

- Lock Monitor
- Performance Manager & Performance Überblick
- Top Sessions
- Top SQL
- Trace Data Viewer

Die Werkzeuge des **Oracle Tuning Pack** bieten dynamische Optimierungsempfehlungen zur effizienteren Ressourcennutzung, zum Erreichen eines höheren Transaktionsdurchsatz und einer besseren Abfragenperformance (SQL-Analyse, automatische Reorganisierung, Indexoptimierung). Bestandteile des *Tuning Packs* sind:

- Oracle Expert
- Outline Management
- SQL Analyse
- Tablespace Map

Das **Oracle Database Configuration Pack** ermöglicht Datenerfassung, Analyse und die Erstellung von Berichten zu Systemdaten. Dazu kommen Vergleiche mit früheren Systemkonfigurationen, ein Patch-Assistent zur Installation erforderlicher Patches und Policy-Management. Folgende Tools übernehmen diese Aufgaben:

- Backup-Verwaltung
- Datenverwaltung
- Analyse

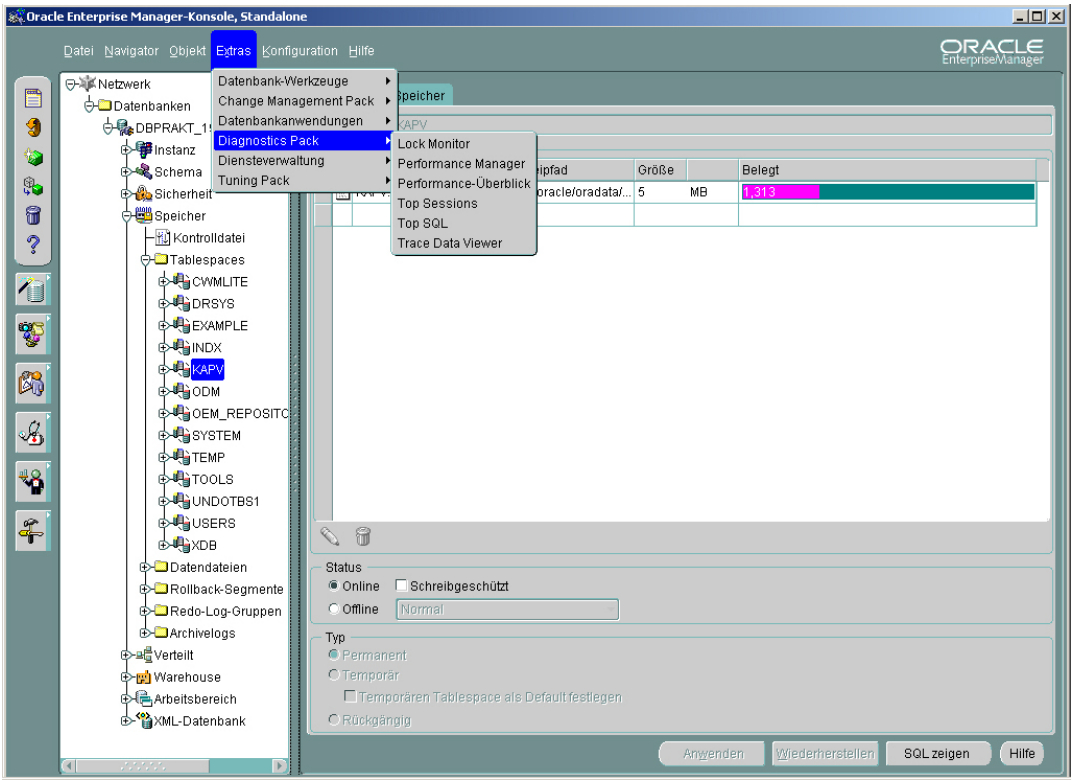



**Abbildung 4.1:** Oracle Enterprise Manager-Konsole

Mit dem **Oracle Change Management Pack** bietet Oracle Unterstützung beim Testen, Planen und Implementieren von Datenbankschemenänderungen. Mit diesem Paket können Fehler und Datenverluste bei Änderungen im Datenmodell minimiert und so die Ausfallzeiten so kurz wie möglich gehalten werden. Alle Aufgaben des Pakets werden mit dem **Change Manager** bewältigt.

Unter dem Punkt **Diensteverwaltung** des OEM-Menüs findet sich der **Oracle Net Manager**, der für die Benennung von Diensten, die Änderung von Benennungsmethoden (siehe auch Abschnitt 4.3.2) und zur Konfiguration der Listener (vgl. Abschnitt 4.3.1) zuständig ist. Bei der Installation eines Oracle9i-Datenbanksystems wird der *Oracle Enterprise Manager* standardmäßig mit installiert. Es besteht jedoch auch die Möglichkeit, die Oracle9i als **Client Release** auf einem entfernten Rechner zur Administration übers Netz zu benutzen. Die entsprechende Software ist auf der Oracle Download-Page [3] frei erhältlich.

Zusätzlich zu den im vorangegangenen Abschnitt besprochenen Paketen bietet Oracle mit dem **Client Release** auch einige Entwicklungstools zur vereinfachten und schnellen Anwendungsprogrammierung an. Das Paket des sogenannten **Integrated Management** Tools beinhaltet Werkzeuge, wie den *Wallet Manager*, zur Verwaltung von Sicherheits-ID-Daten mit öffentlichen Schlüsseln auf *Client/Server-Systemen* oder

den *Policy Manager*, mit dem *Label Security Policys*, *Feinzugriffs-Policys* sowie *Anwendungskontexte* erstellt und verwaltet werden können.

Eine ausführliche Erläuterung der einzelnen Werkzeuge kann im Rahmen dieser Arbeit aufgrund der großen Vielfalt nicht stattfinden.

## 4.7 Der Kommunikationsstack bei Oracle

Zur Gewährleistung einer reibungslosen Kommunikation zwischen beliebigen Client-Applikationen und dem Oracle Server müssen die verschiedenen Stufen des *OSI-Modells* durchlaufen werden. Das Hauptanliegen ist dabei das Erreichen von Plattformunabhängigkeit, Unabhängigkeit vom Netzwerkprotokoll und Verbindungstransparenz. Eine typischer Oracle-Kommunikationsstack ist in Abbildung 4.2 dargestellt.

In der Anwendungsschicht müssen alle Nutzeraktivitäten, die im Zusammenhang mit der Datenbank stehen, erkannt und weitergeleitet werden. Die Schnittstelle hierfür bildet clientseitig das *OCI*, mit dem sich der Abschnitt 7.5 intensiv am Beispiel der *OCI*-Implementation von *PHP* beschäftigt. Durch das *OCI* werden alle notwendigen Informationen für einen SQL-Dialog zwischen einem Client und dem Server an *Net8* weitergereicht. Dabei müssen für eine sinnvolle Kommunikation bestimmte Konvertierungen zwischen Datenformaten und unterschiedlichen Charactersets vorgenommen werden. Oracle hat mit *Two-Task-Common* eine eigene Implementation der *OSI-Präsentationsschicht* zur Umsetzung dieser Aufgaben entworfen.

Oracle *Net8* stellt bei Netzwerkverbindungen alle Funktionalitäten der *Session-Schicht* des *OSI-Referenz-Modells* zur Verfügung und wird in weitere Komponenten unterteilt:

| **Net8-Unterschicht** | **Beschreibung** |
|---|---|
| Network Interface (NI) | generische Schnittstelle für Oracle-Clients, -Server oder externe Prozesse für den Zugang zu Net8-Funktionen - NI behandelt die *break-* und *reset-Anfragen* für eine Verbindung |
| Network Routing (NR) | Routing von Netzwerksessions |
| Network Naming (NN) | Auflösung der Net Service Names zu den Net8-Adressen |
| Network Authentication (NA) | Behandlung von Authentifizierungsanfragen |

| | |
|---|---|
| Transparent Network Substrate (TNS) | Schnittstelle für Standardprotokolle zum Oracle Protokoll |

**Tabelle 4.3:** Interne Struktur einer Net8-Schicht

*Net8* ist eine *Weiterleitungsschicht*, welche über den *Netzwerkprotokollen* angesiedelt ist und den eigentlichen Verbindungsaufbau zum Server übernimmt. Folgende kritische Funktionen werden durch diese Schnittstelle durchgeführt:

- Weiterleitung von Aufrufen (Calls) vom Client zum Server und zurück
- Abgleichen der Zeichensätze von Client und Server
- Umformung von Datentypen

Das Bindeglied zwischen *Net8* und der *Netzwerkschicht* stellt *Oracle Protocols* dar. Diese Oracle-Implementation der *OSI-Transportschicht* ist verantwortlich für das Mapping zwischen *TNS* und dem spezifischen Protokoll. *Oracle Protocols* unterstützt:

- LU6.2
- Named Pipes
- SPX
- TCP/IP
- TCP/IP with SSL

Die, auf diese Weise aufbereiteten, Informationen können nun über ein Netzwerk zum Server übertragen werden und auf der Gegenseite entgegengesetzt aufgelöst werden. Die Aufgaben des *OCI* übernimmt auf der Serverseite das *OPI (Oracle Program Interface)*.

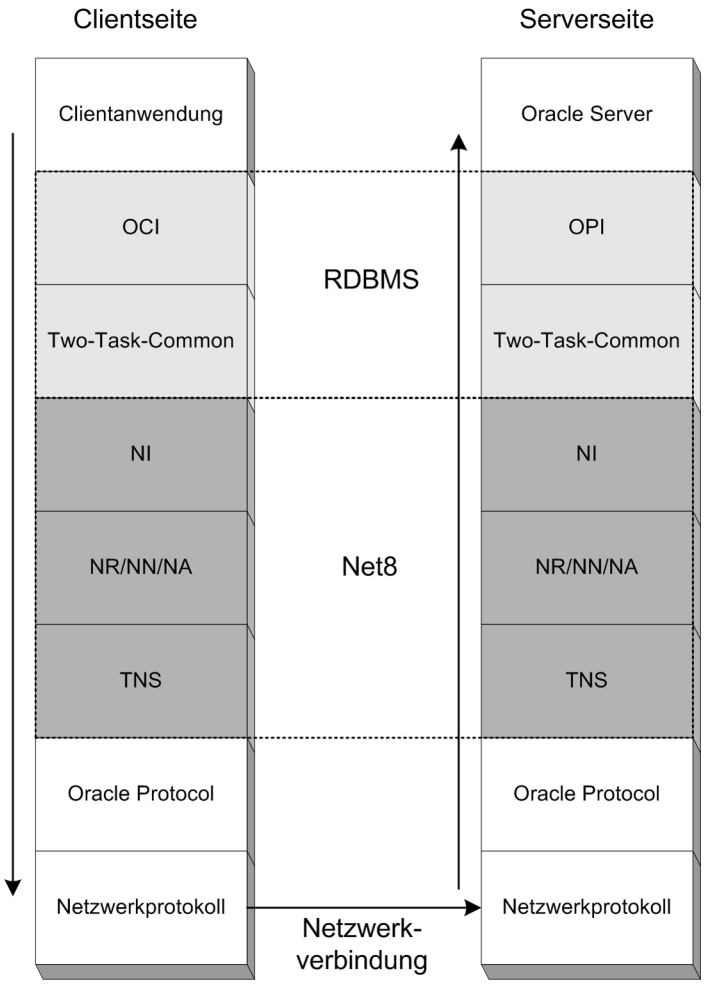



**Abbildung 4.2:** Kommunikationsstack einer Netzwerkverbindung bei Oracle

# 5 Kunden-Auftrag-Produkt-Verwaltung (KAPV)

Die Aufgaben des Praktikums beziehen sich auf eine im Vorfeld definierte Rahmenapplikation - die KAPV. In diesem Kapitel wird die Struktur des Unternehmens und der dazugehörigen Anwendung erläutert und deren Umsetzung in einem bestimmten Bereich der Datenbank dokumentiert.

## 5.1 Die Unternehmenssituation

Die KAPV ist eine Datenbank-Anwendung, die in einem fiktiven Unternehmen zur Lagerverwaltung eingesetzt wird. In dieser Datenbank werden Kundendaten, ihre jeweiligen Aufträge, Produktdaten und Lagerbestände gehalten und verwaltet. **Kundendaten** bestehen aus folgenden Inhalten:

- Kundennummer
- Name
- Vorname
- Geburtsdatum
- Geschlecht
- Adreßdaten
- Kundenprofil
- Kundenkonto

Das **Kundenprofil** enthält Angaben über das bisherige Auftragsvolumen, die Zahlungsbilanz, das Zahlungsverhalten, die Frequenz der Bestellungen und die Vorlieben der jeweiligen Kunden, dokumentiert in Produktkategorien. In dem sogenannten Kundenkonto werden alle Auftragsbeträge mit Bestelldatum, alle eingegangenen Zahlungen, sowie die Anzahl von eventuell erstellten Mahnungen pro Auftrag gespeichert.

Bei nicht fristgemäßer Zahlung wird 21 Tage nach Rechnungsdatum ein Mahnungsschreiben erstellt. Zu jeder Kundenbestellung wird ein **Auftrag** angelegt, der aus folgenden Angaben besteht:

- Auftragsnummer
- Kundennummer
- Auftragsdatum
- Auftragsstatus

Zu jedem Auftrag gehört eine **Liste** mit den bestellten Produkten oder Positionen, die mit dem Produktlager abgeglichen wird. Diese Liste besteht aus:

- Auftragsnummer
- Produktnummer
- bestellte Stückzahl

Sind nicht genügend Produkte im Lager vorhanden, wird dies auf der Rechnung vermerkt. Ansonsten wird eine **Rechnung** ausgestellt und zusammen mit dem Produktpaket verschickt. Eine Rechnung bezieht sich immer auf einen Auftrag und einen Kunden und besteht aus:

- Rechnungsdatum
- Rechnungsbetrag
- Rechnungsbemerkung

Der Auftragsstatus wird nach dem Abschicken des Produktpaketes in 'ausgeliefert' geändert. Nach Eingang der Zahlung des Kunden ändert sich der Status in 'bezahlt'.

Das **Produktlager** besteht aus den folgenden Informationen:

- Produktnummer
- Bezeichnung
- Produktionsdatum
- Material
- Größe

- Stückzahl
- Preis

Nach jeder Lieferung muß die verfügbare Stückzahl im Produktlager entsprechend aktualisiert werden. Eine Datenbank gemäß dieser Unternehmenssituation, mit ihren speziellen Anforderungen an die Datenbank, wurde in ein Relationenschema (siehe Abbildung 5.1) übertragen.

## 5.2 Entwurf der KAPV-Datenbank

Ein wichtiger Themenbereich der Vorlesung befaßt sich mit dem Datenbankentwurf. Im Praktikum wird darauf aufbauend von den Studenten mit dem CASE Tool *DB-Main* ein konzeptioneller Entwurf der Datenbank, anhand der, im Abschnitt 5.1 beschriebenen Vorgaben, angefertigt. Ein solcher Entwurf bildete die Grundlage für die Datenbank, die im Hintergrund des im Rahmen dieser Arbeit entwickelten Online-Praktikums genutzt wird. Mit dem Entwurfs-Programm *DB-Main* wurde dazu aus einem vorliegenden Schema ein SQL-Skript (siehe Appendix A) generiert, mit dessen Hilfe eine, dem Schema entsprechende, Datenbank erstellt wird. Das in Abbildung 5.1 dargestellte Schema wird als statisches Modell in einem speziell dafür angelegten *Tablespace* der Oracle-Datenbank abgebildet und mit Beispieldaten gefüllt. Auf eine dynamische Umsetzung der KAPV-Datenbank wurde aus Sicherheitsgründen bewußt verzichtet. Die Einzelheiten zu diesem Sicherheitskonzept werden in den Abschnitten 7.8 beziehungsweise Abschnitt 7.3.2 erörtet.

Um dennoch das Füllen der Datenbank zu simulieren, wird jedem Nutzer temporär ein individueller Tablespace zugeteilt, in dem der jeweilige Nutzer zum Anlegen von Tabellen sowie zum Einfügen und Manipulieren von Daten berechtigt ist.

## 5.3 Anlegen der KAPV-Datenbank

Das mit *DB-Main* generierte SQL-Skript kann nun mit *SQL*Plus* ausgeführt werden. Vorher sind jedoch noch einige anwendungsspezifische Änderungen vorzunehmen, die dem DB-Main-Schema nicht entnommen werden können. So wird beispielsweise ein spezieller Tablespace festgelegt, in dem der statische Teil der Datenbank angelegt werden soll.

Nachdem diese Anpassungen erfolgt sind, kann das Skript gestartet werden. Zum Ausführen von Dateien unter *SQL*Plus* wird dem Dateinamen einfach ein @ vorangestellt:

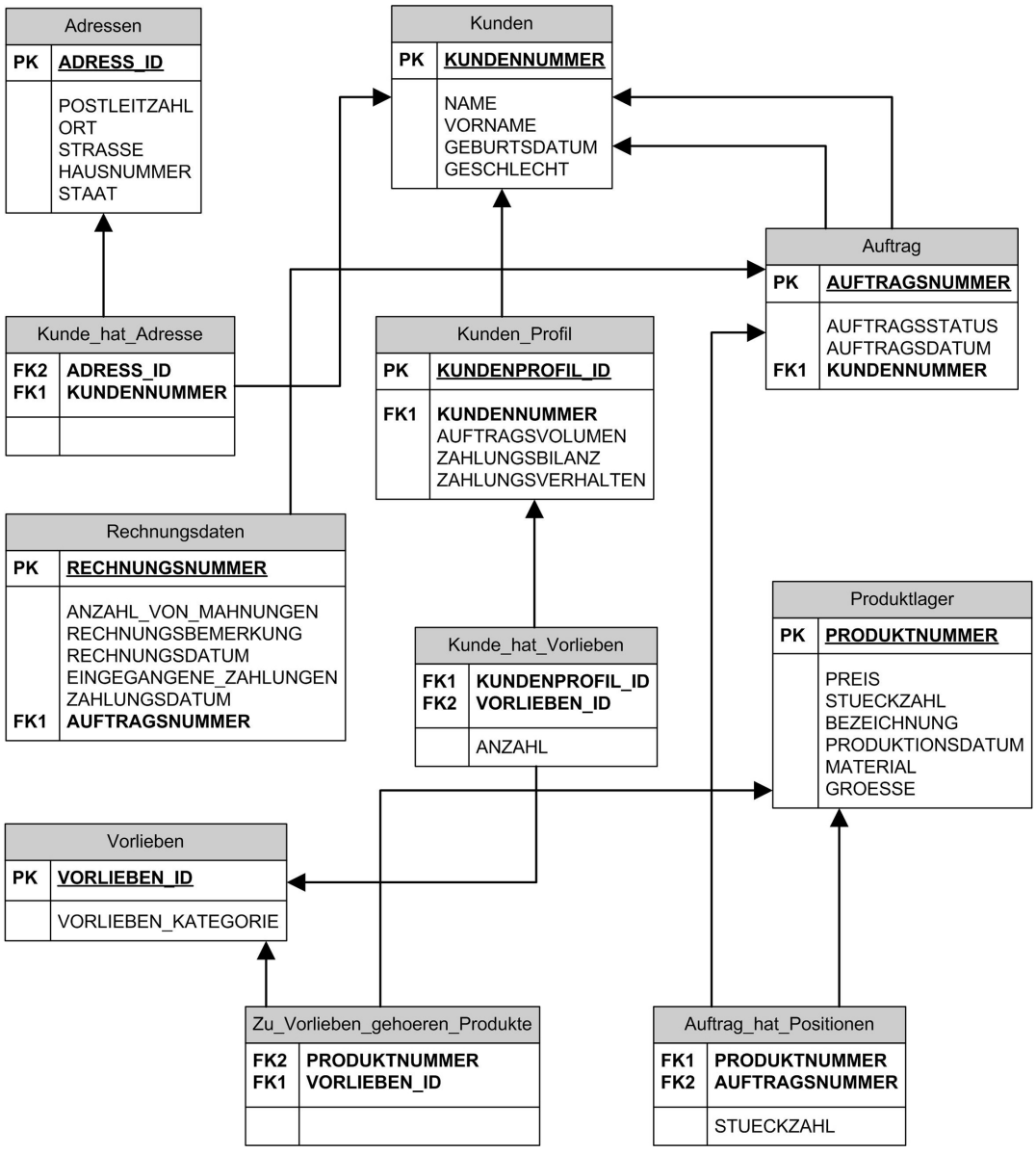



**Abbildung 5.1:** Die Struktur der KAPV-Datenbank

```
SQL> @KAPV
```

Wird die Dateiendung beim Aufruf nicht explizit angegeben, erwartet *SQL*Plus* eine Datei mit der Endung `.sql`. Die im Skript enthaltenen Befehle werden der Reihe nach abgearbeitet und deren Ausgaben auf dem Monitor ausgegeben. Nachdem das Skript *KAPV.sql* erfolgreich ausgeführt wurde, kann die angelegte Struktur mit Musterdaten gefüllt und genutzt werden.

# 6 Konzept

Im Ergebnis dieser Arbeit soll eine Möglichkeit geschaffen werden, den Studenten der Hochschule Wismar die Abarbeitung des Praktikums im Fach Datenbanken über das Internet zu ermöglichen. Zur Umsetzung des Online-Praktikums werden verschiedene Module benötigt, die sicher und effizient zusammenarbeiten müssen, um eine dauerhafte und dynamische Nutzung des Systems zu gewährleisten. Dieses Kapitel gibt einen Überblick über die einzelnen Komponenten und soll deren konzeptionelle Funktionsweise verdeutlichen, sowie das Zusammenspiel der einzelnen Elemente erklären.

## 6.1 Überblick

Den Mittelpunkt des Systems bildet die Oracle-Datenbank. Sie liefert die fiktiven Inhalte der KAPV-Datenbank, enthält Nutzerdaten für den Zugang zum Praktikum, sichert die dynamische Bereitstellung der Inhalte des Praktikums und speichert die Log-Daten, die durch die Nutzung des Praktikums entstehen. Diese Datenbank besteht im Wesentlichen aus drei Bereichen:

1. KAPV-Bereich (statisch)
2. Verwaltungs- und Logbereich (dynamisch)
3. Nutzerbereich (dynamisch)

Der **statische KAPV-Bereich** ist ein Tablespace, in dem sich die mit Musterdaten gefüllten Tabellen der KAPV-Datenbank befinden. Die Inhalte dieser Tabellen sollen in den Aufgaben des Praktikums abgefragt werden, dürfen jedoch von den Nutzern nicht verändert werden. Die Struktur dieses Bereiches wird den Studenten in der Aufgabenstellung zur Verfügung gestellt.

In dem **dynamischen Verwaltungs- und Logbereich** sind Daten gespeichert, die zur Nutzerauthentifizierung benötigt werden. Weiterhin befinden sich in diesem Bereich die Aufgabenstellungen, die durch den Praktikumsadministrator bearbeitet

und erweitert werden können. Zur Auswertung der Nutzeraktivitäten werden deren Eingaben für eine spätere Auswertung ebenfalls in diesem Bereich gespeichert.

Der **dynamische Nutzerbereich** ist ein Bereich der Datenbank, in dem die Teilnehmer des Praktikums selbst gezielt Objekte anlegen und verändern dürfen. Diese Tatsache bedarf einer besonderen Abgrenzung von den anderen Teilen der Datenbank.

Die logische Trennung der einzelnen Bereiche erfolgt durch eine Aufteilung in verschiedene Tablespaces, für die den Anforderungen entsprechende Berechtigungen gelten.

Die Bereitstellung der Inhalte des Praktikums und die Abarbeitung der Aufgaben sowie die Administration erfolgen über ein **Benutzerinterface**, welches durch einen *Apache2-Server* in Form einer interaktiven Website zur Verfügung gestellt wird. Über diese vorrangig in *PHP* erstellte Website erfolgt die gesamte Kommunikation (vgl. siehe Abschnitt 7.5) mit dem Datenbank-Managementsystem.

## 6.2 Interne Kommunikation

Die Kommunikation mit der Datenbank findet aus Nutzersicht ausschließlich über das Webinterface statt. Ein Großteil der Funktionen der Praktikumsanwendung basiert auf einem Datenaustausch zwischen dem Datenbank-Managementsystem und den Anwendern. Beim Transfer der Inhalte kommen verschiedene Techniken zum Einsatz, die im folgenden Überblick kurz zusammengefaßt werden.

Die Praktikumsanwendung ist in verschiedene inhaltlich strukturierte Bereiche aufgeteilt, deren spezielle Formulare und Funktionen die Abfragen der Nutzer an den Apache2-Server weitergeben.

Der Datenaustausch zwischen Browser und Server erfolgt über eine *SSL* verschlüsselte *TCP/IP-Verbindung*. Der Webserver kommuniziert über die *OCI8-Schnittstelle* des *PHP-Moduls* mit der Netzschnittstelle des Datenbank-Managementsystems. Der Nutzer bekommt die Ergebnisse seiner Abfragen durch den Webserver als reines *HTML* auf den Bildschirm geliefert.

Die Abbildung 6.1 soll den Ablauf der Kommunikation allgemein verdeutlichen. Detaillierte Beschreibungen der einzelnen Techniken werden in den Abschnitten 7.2.4, 7.5 und 4.7 gegeben.

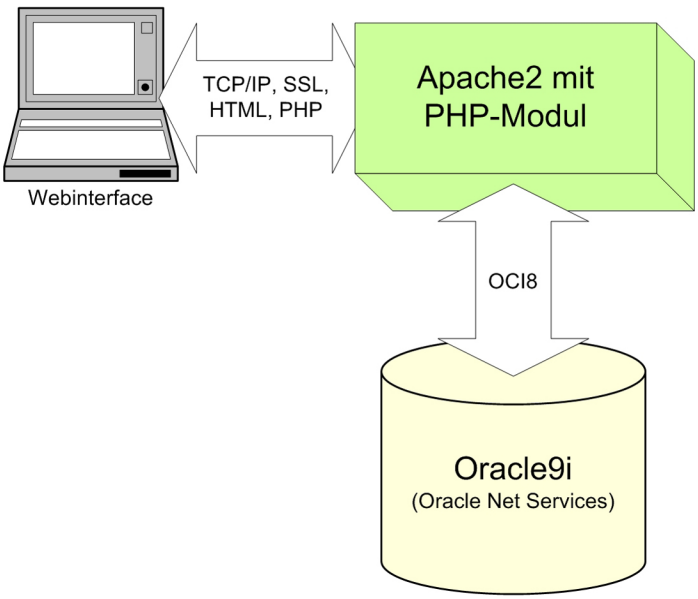



**Abbildung 6.1:** Mechanismen der internen Kommunikation

## 6.3 Nutzer

Der Zugang zum Online-Praktikum erfordert ein gültiges Login, über das entschieden wird, welche Inhalte dem Nutzer zur Verfügung gestellt werden und welche Berechtigungen er erhält. Die Nutzer werden grundsätzlich in zwei Gruppen unterteilt:

- Praktikumsteilnehmer
- Administratoren

**Praktikumsteilnehmer** sind Studenten, die für das Praktikum zugelassen sind. Ihnen werden über das Webinterface die Inhalte der Aufgaben angeboten und eine Möglichkeit zur Lösung der Aufgaben bereitgestellt. Sie erhalten die Rechte, den statischen *KAPV-Bereich* der Datenbank entsprechend den Aufgabenstellungen abfragen zu dürfen und können in dem *dynamischen Nutzerbereich* unter bestimmten Vorgaben Objekte anlegen und manipulieren. Individuelle Daten aus dem *Logbereich* werden den Praktikumsteilnehmern in Form einer persönlichen Auswertung zur Verfügung gestellt.

**Administratoren** sind verantwortlich für die Pflege und Wartung des Systems. Sie bekommen über das Interface Zugang zum *Verwaltungs- und Logbereich* der Datenbank. Zu den Administrationsaufgaben gehören die Nutzerverwaltung, sowie die Aktualisierung und Erweiterung der Aufgaben. Die Logdaten der Studenten werden zur Auswertung der Praktikumsarbeit detailliert aufbereitet. In

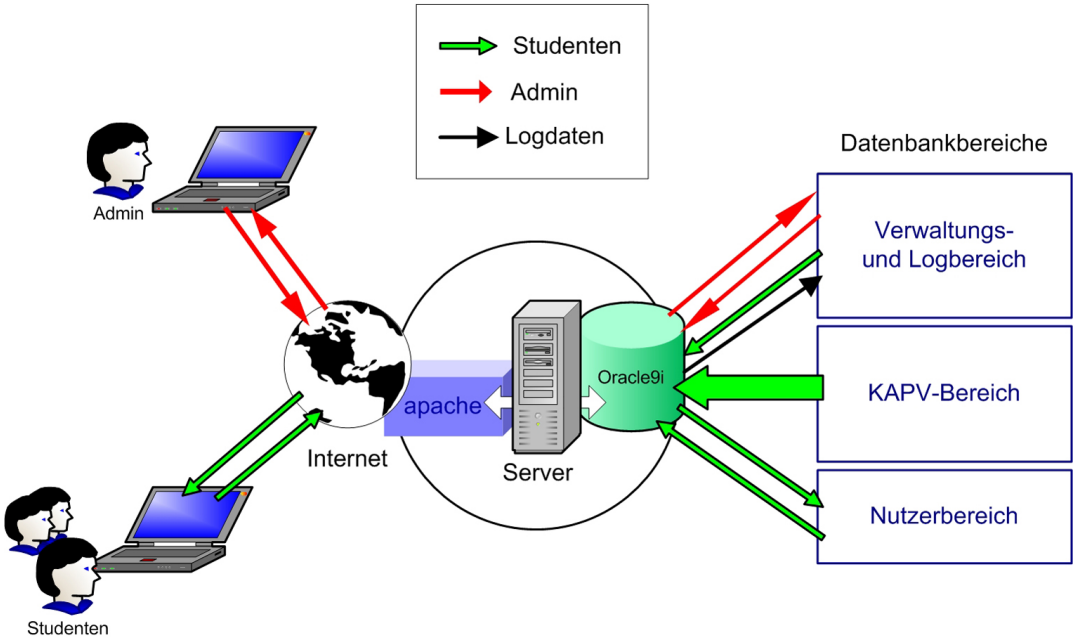



**Abbildung 6.2:** Kommunikation der Nutzer mit den einzelnen Datenbankbereichen

Abbildung 6.2 ist dargestellt, welche Bereiche der Datenbank den jeweiligen Nutzern zur Verfügung stehen.

## 6.4 Allgemeiner Ablauf des Praktikums

Das Hauptanliegen der Online-Anwendung ist es, die Studenten bei der Bewältigung der Praktikumsanforderungen zu unterstützen. Da in der Regel keine direkte Betreuung der Praktikanten stattfindet, sind alle zur Lösung der Aufgaben notwendigen Informationen über das Webinterface verfügbar. Von zentraler Bedeutung ist hierbei die Beschreibung der Unternehmenssituation, die im Kapitel 5 ausführlich erläutert wurde. Ergänzt werden diese Angaben durch hilfreiche Dokumente und Verweise auf thematisch verwandte Internetseiten.

Die zu lösenden Aufgaben können in beliebig vielen Sitzungen abgearbeitet werden. Bei jeder Anmeldung stehen den Teilnehmern die Ergebnisse ihrer Arbeit aus zurückliegenden Sitzungen jederzeit wieder zur Verfügung. Das schließt sowohl die abgeschlossenen Aufgaben, als auch die fehlerhaften Lösungsversuche ein.

Beim Bearbeiten der Aufgaben werden die Eingaben der Studenten an die Datenbank weitergereicht und das Ergebnis der Ausführung mit einer Referenzlösung verglichen. Anhand dieses Vergleichs erfolgt die Bewertung der Eingaben. Diese Bewertung erfolgt durch die Anwendungslogik. Die Ergebnisse der Aktionen werden in der Daten-

bank gespeichert und den Praktikanten zur Kontrolle angezeigt. Die Auswertungen enthalten aussagekräftige Fehlermeldungen, die je nach Art des Fehlers von der Anwendung oder der Fehlerbehandlung des Datenbanksystems stammen. Der Nutzer wird dabei mit Hilfe von dynamisch generierten Links durch die Anwendung navigiert.

Zur Einschätzung der Praktikumsleistung bietet die Anwendung den Praktikumsbetreuern eine Möglichkeit, die Leistungen der Studenten jederzeit detailliert einzusehen.

### 6.4.1 Praktikumsaufgaben

Beim Login wird überprüft, welche Aufgaben der Nutzer bereits gelöst hat und welche noch zu bearbeiten sind. Abgeschlossene Aufgaben werden nicht mehr aufgeführt. Die Ergebnisse der erbrachten Leistungen stehen den Studenten jedoch in der persönlichen Auswertung in jeder Sitzung zur Verfügung. Zum besseren Verständnis des Ablaufes des Online-Praktikums soll an dieser Stelle die Struktur der Aufgabenstellungen näher beleuchtet werden.

Grundsätzlich besteht das Praktikum aus zwei Teilen, die sich aus Sicht der Anwender nur unwesentlich voneinander unterscheiden. Bei näherer Betrachtung sind jedoch einige gravierende Unterschiede zu erkennen, die sich auf die interne Abarbeitung beziehen.

#### Vorbereitungsteil

In der Vorbereitung des Praktikums werden die Teilnehmer aufgefordert, eine beliebige Tabelle anzulegen, diese mit selbst gewählten Daten zu füllen und das Ergebnis dieser Aktionen zu überprüfen. Dabei sollen möglichst wenige Einschränkungen gelten und trotzdem die maximale Sicherheit des Systems gewährleistet bleiben. Außerdem muß jedes angelegte Objekt später immer eindeutig einem Nutzer zugeordnet werden können. Welche Mechanismen zur Realisierung dieser Forderungen angewandt werden, wird im Abschnitt 7.3.2 erörtert.

Beispielaufgabe des Vorbereitungsteils:

"*Legen Sie eine beliebige Tabelle an. Wählen Sie bitte einen Tabellennamen, der sich beispielsweise aus Ihrem Namen herleitet. Bei Namen, wie "test","tabelle" etc. könnte es zu der Fehlermeldung kommen, daß das Objekt bereits existiert. Bitte versuchen Sie es in dem Fall mit einem anderen Tabellennamen.*"

Die weiteren Aufgaben des Vorbereitungsteils beziehen sich anschließend auf die Tabelle, die von dem Studenten angelegt wurde. In den Aufgabenstellungen der Folgefragen wird dem Nutzer stets die Struktur der von ihm angelegten Tabelle angezeigt.

Bei der Abarbeitung der Praktikumsvorbereitung ist entsprechend dieses chronologischen Aufbaus eine logische Reihenfolge einzuhalten. Wird die Abarbeitungsreihenfolge nicht eingehalten, erfolgt ein entsprechender Hinweis.

#### Abfragenteil

Die gewöhnlichen Abfragen beziehen sich ausschließlich auf den statischen Bereich der Datenbank. Die Aufgabenstellungen in diesem Teil beschreiben umgangssprachlich eine bestimmte Datenmenge aus der Struktur der KAPV-Datenbank. Diese Datenmenge ist mit einem SQL-Statement zu erfassen. Jede syntaktisch richtige Abfrage liefert eine Ausgabe, die das Ergebnis der Abfrage enthält, sowie einen Hinweis darauf, ob es sich dabei um die gesuchte Datenmenge handelt oder nicht. Ist dies der Fall, gilt die Aufgabe als abgeschlossen und der Praktikant wird aufgefordert, die nächste Aufgabe zu bearbeiten. Andernfalls muß die Aufgabe wiederholt werden. Für die Abarbeitung der normalen Abfragen ist keine Reihenfolge vorgeschrieben. Dieser Teil kann sogar vor oder während der Abarbeitung der Vorbereitung bearbeitet werden, da er sich auf einen vorgefertigten Bereich der Datenbank bezieht, der den Nutzern auch ohne Abschluß der Vorbereitungen zur Verfügung steht.

Beispielaufgabe des Abfragenteils:

"*Gesucht: Alle Kunden mit Kundennummer & Auftragsnummer, die irgend etwas in 2003 oder vor 2000 bestellt haben.*"

Voraussetzung zur erfolgreichen Lösung des Abfragenteils ist natürlich die Kenntnis der KAPV-Datenbank-Struktur. Diese Struktur wird den Studenten in Form einer textuellen Beschreibung gegeben, welche durch ein Entity-Relationship-Modell (siehe Abbildung 5.1) ergänzt wird. Beide Information werden ebenfalls über das Webinterface bereitgestellt.

# 7 Implementierung

## 7.1 Technische Rahmenbedingungen

Zur Umsetzung der Aufgabenstellung stellte die Hochschule Wismar einen Rechner mit einem 2.0 GHz Intel Pentium 4-Prozessor, einem Linuxkernel (SuSE Linux Version 2.4.20) und 512KB RAM zur Verfügung. Der Rechner verfügt über eine Netzwerkkarte (Realtek RT8139), durch die das System von außen über das Internet erreichbar ist. Bei der verwendeten Software handelt es sich ausschließlich um frei verfügbare kostenlose Software.

## 7.2 Der Webserver

Bevor die Entwicklung der eigentlichen Anwendung beginnen kann, muß ein Webserver eingerichtet werden, der das Online-Praktikum öffentlich als HTML-Seiten im Internet zur Verfügung stellt. Im folgenden Kapitel wird die Installation eines *Apache2-Servers* und dessen Konfiguration dokumentiert.

### 7.2.1 Installation eines Apache2-Servers mit PHP-Modul

Als Webserver fungiert ein *Apache2-Server* mit *PHP-Modul*. Die Beschaffung der notwendigen Software erfolgt am besten auf den offiziellen Downloadseiten von *Apache*[6] beziehungsweise *PHP*[5]. Die entpackten Dateien *httpd-2.0.48.tar.gz* und *php-4.3.4.tar.bz2* enthalten jeweils ein *configure-Skript*, mit dem die Optionen zum kompilieren gesetzt werden. Im Ergebnis der Ausführung von *configure* wird ein *make-file* erzeugt, welches nun ausgeführt wird, um die Anwendung gemäß den vorher festgelegten Optionen zu kompilieren. Der Apache-Server wird wie folgt konfiguriert:

```
athos:~ # ./configure --enable-so --prefix=/usr/local/
   apache2
```

Die Option `--enable-so` steht für ***s**hared **o**bjects* und ermöglicht das dynamische Einbinden von Modulen. Das bietet den Vorteil, daß sich der http-Server nachträglich erweitern läßt und Module dynamisch eingebunden werden können. Außerdem können

die einzelnen Module separat neu konfiguriert werden, ohne daß der Webserver ebenfalls neu übersetzt werden muß. Mit `make && make install` wird die Installation abgeschlossen. Zur Konfiguration des PHP-Moduls sind folgende Optionen zu setzen:

```
athos:~ # ./configure --with-apxs2=/usr/local/apache/bin/
   apxs
--with-oci8=/opt/oracle/9i --prefix=/usr/local/php
```

Dabei wird die Option `--with-apxs2=/usr/local/apache/bin/apxs` benötigt, um das dynamische Einbinden des Moduls zu ermöglichen. Mit `--with-oci8=/opt/oracle/9i` werden die *OCI8-Bibliotheken* eingebunden, die zur Kommunikation von PHP mit Oracle genutzt werden. Die Installation wird mit `make && make install` abgeschlossen.

### 7.2.2 Konfiguration

Um dem Webservers den Umgang von PHP-Dateien zu ermöglichen, muß die Konfigurationsdatei *httpd.conf* noch um zwei Einträge erweitert werden. Unter der Rubrik *Dynamic Shared Object (DSO) Support* wird angegeben, welche zusätzlichen Module dynamisch eingebunden werden sollen. Der Eintrag lautet:

```
LoadModule php4_module modules/libphp4.so
```

Der zweite Eintrag erweitert die *MIME-Konfiguration* des Webservers und ermöglicht die Nutzung spezieller Dateitypen. Um den Webserver zum Umgang mit PHP-Dateien zu befähigen, muß die *httpd.conf* um die folgende Zeile erweitert werden:

```
AddType application/x-httpd-php .php
```

Mit einem einfachen Test kann anschließend die syntaktische Richtigkeit der Änderungen überprüft werden:

```
athos:~ # /usr/local/apache/bin/apachectl configtest
Syntax OK
```

Vorausschauend sollte nun noch eine Änderung in der Konfigurationsdatei *php.ini* vorgenommen werden. Das PHP-Modul kennzeichnet Sonderzeichen, wie ' *(single quote)*, " *(double quote)* oder \ *(backslash)* standardmäßig automatisch mit einem sogenannten *Escape-Character*, ebenfalls einem vorangestellten *backslash*. Verantwortlich für dieses Verhalten ist die Funktion *magic_quotes*. Dabei beeinflußt *magic_quotes_gpc* Daten, die per GET, POST oder COOKIE übergeben werden und *magic_quotes_runtime* Daten, die aus Datenbanken, Dateien oder anderen externen Quellen kommen. Der Standardwert des Parameters *magic_quotes_gpc* ist *ON*. In

der Praktikumsanwendung ist jedoch davon auszugehen, daß die SQL-Statements häufig solche Sonderzeichen beinhalten und per *COOKIE* beziehungsweise *POST* übergeben werden. Die Abfragen sollen jedoch unverändert an die Datenbank weitergegeben und gegebenenfalls dort ausgeführt werden. Um das zu garantieren, muß die Datei *php.ini* um mindestens den folgenden Eintrag erweitert werden:

```
magic_quotes_gpc = Off
```

Um nun die Funktion des Servers zu testen, empfiehlt es sich, ein einfaches info.php mit der *phpinfo()-Funktion* zu schreiben. Die Ausgabe dieser Funktion eignet sich hervorragend, um die Konfiguration des PHP-Moduls und des Apache-Servers noch einmal gründlich zu überprüfen. Sämtliche Informationen über die Konfigurationsparameter, den Pfad zur *php.ini* und alle weiteren Einstellungen werden bei einem Aufruf der Datei info.php mit einem Browser ausgegeben.

### 7.2.3 Umgebungsvariablen

Zur erfolgreichen Kommunikation zwischen dem Webserver und dem Oracle Datenbank-Managementsystems müssen jetzt nur noch die entsprechenden Umgebungsvariablen und Pfade des *Apache2-Servers* gesetzt werden. Dies erfolgt am besten vor dem Start des Servers in der Datei */usr/local/apache/bin/envvars*. Der Inhalt der Datei ist im Appendix B dargestellt.

Als erstes wird das Home-Verzeichnis von Oracle angegeben. Dazu wird eine Variable *$ORACLE_HOME* angelegt. Die Variablen *$NLS_LANG* und *$ORA_NLS33* steuern das Sprachverhalten, wobei *$NLS_LANG* für die Steuerung der Meldungen des Clients (in diesem Fall das PHP-Modul) zuständig ist und mit dem Sprachpaket der Datenbank abgestimmt werden muß. Die Einstellung von *$ORA_NLS33* dient dazu, dem Client mitzuteilen, wo sich die Dateien mit den verschiedensprachigen Prompts befinden. Außerdem muß sichergestellt werden, daß der *$LD_LIBRARY_PATH* die Angabe *$ORACLE_HOME/lib* enthält. Diese Variable definiert die Speicherorte für Oracle's *shared libraries*. Die Pfadangaben der Variablenwerte sind natürlich bei jeder Installation individuell und werden aus diesem Grunde an dieser Stelle nicht angeben.

Im Appendix B ist die in diesem Beispiel verwendete Datei *envvars* abgedruckt. Für die Verbindung zu einer konkreten Datenbank muß dem Webserver nun nur noch die *SID* bekannt gemacht werden. Dazu wird die Umgebungsvariable *$TWO_TASK* gesetzt. Deren Deklaration erfolgt jedoch direkt in dem PHP-Skript, durch das die Verbindung hergestellt wird.

### 7.2.4 SSL-Konfiguration

Aus Sicherheitsgründen wurde für den Datenaustausch zwischen dem Browser des Nutzers und dem Apacheserver eine SSL verschlüsselte TCP-Verbindung gewählt. Das SSL-Protokoll ist eine weit verbreitete Technik, um das Abhören und die Manipulation des Datenverkehrs zwischen den beiden Instanzen zu verhindern. Dabei wird ein *Handshake-Verfahren* angewandt, bei dem der Browser ein Zertifikat mit dem öffentlichen Schlüssel des angesprochenen Webservers abfragt. Anschließend wird anhand der übermittelten Daten die Identität des Servers überprüft und gegebenenfalls die synchron verschlüsselte Verbindung aufgebaut.

Zur Nutzung von SSL muß der Apache2-Server nun noch entsprechend konfiguriert werden. Um mit dem Apache Webserver sichere SSL Verbindungen aufbauen zu können, muß das SSL-Modul *mod_ssl* beim Start eingebunden werden. Dafür wird in der Datei *httpd.conf* folgender Eintrag hinzugefügt:

```
<IfModule mod_ssl.c>
Include conf/ssl.conf
</IfModule>
```

Die eigentliche SSL-Konfigurationsdatei ist die Datei *ssl.conf*. In dieser Datei befinden sich die Informationen zum Speicherort der Zertifikate und der Schlüssel, sowie alle Angaben über weitere Verhaltensvorschriften.

### 7.2.5 Der geschützte Bereich des Webservers

Eine zusätzliche Maßnahme zur Verbesserung der Sicherheit der Anwendung ist die Nutzung eines geschützten Bereichs (engl. restricted area). Dabei wird der Webserver so konfiguriert, daß bestimmte Verzeichnisse nur für autorisierte Benutzer erreichbar sind. In der *httpd.conf* oder der *ssl.conf* erfolgt die Definition der zu schützenden Verzeichnisse. Hier erfolgen auch die Angaben über den Speicherort der Paßwortdatei und den Typ der Autorisierung.

```
<Directory "/usr/local/apache/htdocs_ssl/dipl">
AuthType Basic
AuthName "Restricted Area"
AuthUserFile /usr/local/apache/passwords/passwords
require valid-user
</Directory>
```

Bei Anfragen auf Dateien innerhalb dieser Ordner werden die Nutzer nun aufgefordert, den Nutzernamen und ein Paßwort anzugeben. In der Datei */usr/local/apa-*

*che/passwords/passwords* sind sämtliche berechtigte Nutzer aufgeführt. Zur Nutzerverwaltung dient das Werkzeug *htpasswd*.

## 7.3 Die Datenbank

Die wesentliche Funktionalität der Praktikumsanwendung liefert die Hintergrunddatenbank, die Datenbank inhaltlich und logisch, wie bereits im Abschnitt 6.1 beschrieben wurde, in drei Bereiche unterteilt ist. Diese Aufteilung ist in der Abbildung 7.1 skizziert.

Der statische Teil, der die Struktur der KAPV-Datenbank abbildet, enthält die Daten, die in den Aufgaben des Praktikums von den Studenten abgefragt werden sollen und kann nicht manipuliert werden. Dieser Bereich kann als unveränderliches Modell angenommen werden. Die Studenten bekommen eine Beschreibung der Tabellenstruktur dieses Tablespaces und können ihre SQL-Statements auf die gegebene Struktur abgestimmt formulieren. Für die Verbindung mit diesem Bereich existiert ein eigens für diese Zwecke eingerichteter Datenbank-Nutzer. Im Abschnitt 7.5.1 werden die Verbindungsarten mit der Datenbank genauer beschrieben.

Im dynamischen Verwaltungs- und Logbereich werden alle Daten, die durch die Nutzung des Systems anfallen, gespeichert. Außerdem werden in diesem Bereich sowohl die Nutzereigenschaften als auch die Inhalte der Aufgaben eingepflegt. Dieser Bereich dient zur internen Verwaltung des gesamten Systems und bildet damit den funktionalen Kern der Anwendung.

Der dritte Bereich der Datenbank wird genutzt, wenn die Aufgabenstellungen des Praktikums über das normale Abfragen des statischen Bereiches hinausgehen. Hier dürfen die Studenten in einem fest definierten Rahmen selbst Objekte anlegen und manipulieren. Die logische Trennung von den anderen beiden Bereichen dient der Sicherheit der gesamten Anwendung. Aus diesem Grund wird dem dynamischen Teil der Datenbank im Abschnitt 7.3.2 besondere Aufmerksamkeit geschenkt.

Alle Aktionen, die ein Nutzer während der Abarbeitung des Online-Praktikums ausführt, sind direkt mit der Datenbank verknüpft. Schon beim Login wird eine Datenbankverbindung aufgebaut und geprüft, ob der Nutzer bereits im System erfaßt ist und ob er für das Praktikum zugelassen ist. Ist das Login gültig, werden abhängig von der bisherigen Arbeit des Nutzers, die noch zu lösenden Aufgaben zur Verfügung gestellt. Während der Bearbeitung der Aufgaben werden richtige und falsche Lösungen zur späteren Auswertung in der Datenbank gespeichert. Es existiert jeweils eine Tabelle zur Sicherung der richtigen beziehungsweise der falschen Lösungen.

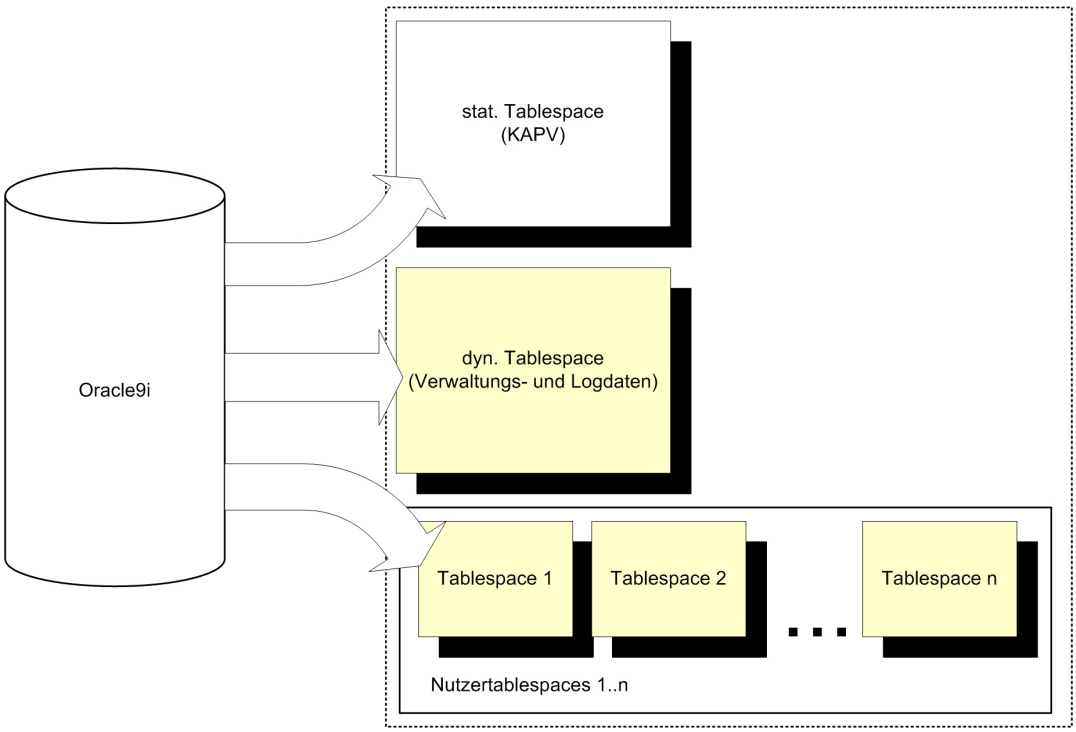



**Abbildung 7.1:** Die Aufteilung der Datenbank in Tablespaces

Die Entscheidungskriterien, ob eine Aufgabe als richtig gewertet werden kann, liefert eine Referenzlösung, die ebenfalls in der Datenbank abgelegt ist. Bei der Speicherung der Lösungsversuche wird zu jedem SQL-Statement die jeweilige Aufgaben-ID, der aktuelle Nutzer und das Datum der Eingabe festgehalten. Aus diesen Angaben kann jederzeit der momentane Stand der Praktikumsarbeit der Studenten ermittelt werden und in Form einer persönlichen Auswertung abgerufen werden. Administratoren bekommen über das Webinterface eine detaillierte Auswertung über den Status aller Studenten. Die folgende Liste enthält alle Anforderungen, die beim Entwurf der Datenbankstruktur berücksichtigt werden müssen:

- Bereitstellung der Inhalte der KAPV-Datenbank
- Bereitstellung der Nutzerdaten und der Aufgabenstellungen
- Bereitstellung einer Referenzlösung
- Speicherung der Lösungsversuche getrennt nach richtigen und falschen Lösungen
- jede Referenzlösung wird eindeutig einer Aufgabe zugeordnet
- jeder Nutzer kann genau einer Gruppe zugeordnet werden

- jeder Lösungsversuch kann eindeutig einem Nutzer und einer Aufgabe zugeordnet werden

### 7.3.1 Struktur der Datenbank

Der statische Bereich der Datenbank wurde bereits in Kapitel 5 ausführlich erläutert. Dieser Abschnitt beschreibt die Tabellen des Verwaltungs- und Logbereiches der Datenbank und erklärt deren Funktion innerhalb der Praktikumsanwendung. Die folgende Übersicht gibt Aufschluß über den allgemeinen Verwendungszweck der einzelnen Datenbankrelationen. Die Abbildung 7.2 enthält das dazugehörige Entity-Relationship-Modell für diesen Datenbank-Bereich.

| **Relation** | **Daten** |
|---|---|
| STUDENTS | Nutzerdaten |
| GROUPS | Nutzergruppen |
| LESSONS | Aufgabenformulierungen und deren Referenzlösungen |
| LESSON_ORDER | Reihenfolge der Aufgaben |
| LOGON | richtige Lösungen |
| ERROR_LOG | fehlerhafte Lösungsversuche |

**Tabelle 7.1:** Relationen und ihre Verwendung

#### STUDENTS

Jeder aktuelle oder ehemalige Nutzer besitzt genau einen Datensatz in der Tabelle *STUDENTS*. Diese Relation enthält eine eindeutige Nutzer-ID, den vollständigen Namen, Angaben zum Nutzertyp, ein Datum, welches angibt, seit wann der Student für das Praktikum zugelassen ist, ein Feld mit dem das Login aktiviert beziehungsweise deaktiviert wird, den Login-String und einen Eintrag zur Gruppenzugehörigkeit des Nutzers. Die beiden möglichen Nutzertypen sind *Administrator* und *normaler Nutzer*.

#### GROUPS

Um die Benutzerverwaltung übersichtlicher zu gestalten, werden Studenten bestimmten Gruppen zugeordnet. Diese Gruppenaufteilung ermöglicht es später, innerhalb der Nutzerverwaltung mehrere Nutzerdatensätze gleichzeitig zu bearbeiten.

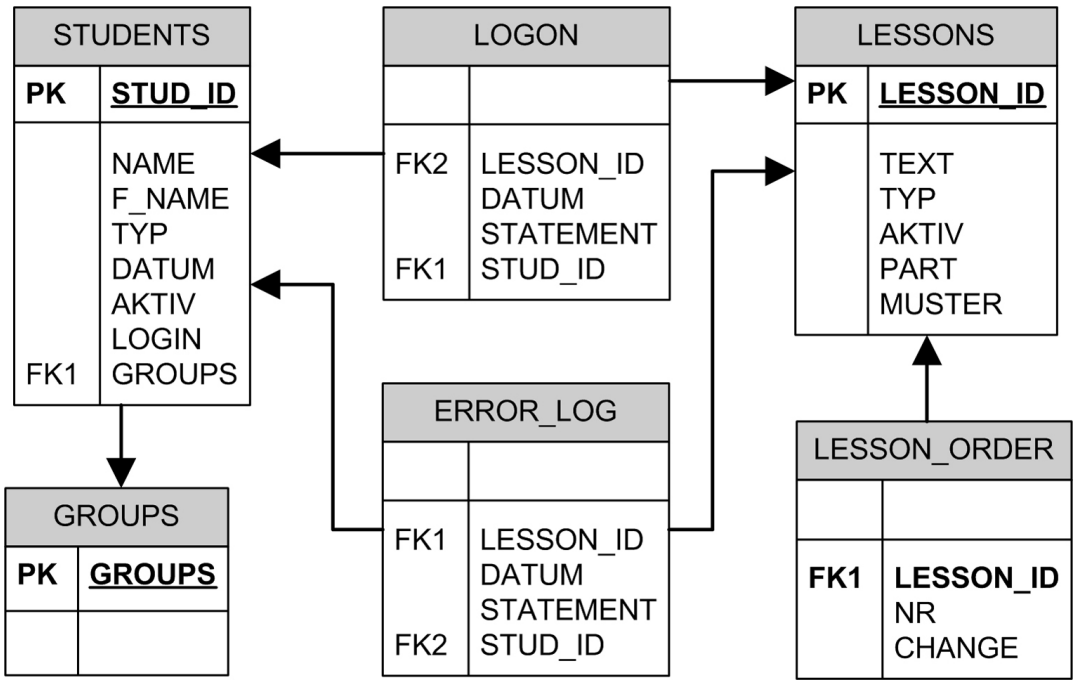



**Abbildung 7.2:** Relationales Modell des dynamischen Verwaltungs- und Log-Bereiches

### LESSONS

Die Relation *LESSONS* speichert alle Informationen zu den Aufgaben. Sie enthält eine eindeutige ID, den Aufgabentext der Fragestellung, den Aufgabentyp, einen Aktivierungsschalter, ein Attribut, mit dem die Aufgaben dem Vorbereitungsteil oder dem Abfragenteil des Praktikums zugeordnet werden und die Referenzlösung der jeweiligen Aufgabe. Aufgabentypen sind die verschiedenen SQL-Schlüsselworte, die in den Aufgabenstellungen gesucht werden. Die möglichen Werte sind *SELECT*, *INSERT* oder *CREATE*. Anhand dieser Angaben findet später eine Vorentscheidung bei der Prüfung der Richtigkeit der SQL-Statements statt.

### LESSON_ORDER

In der Tabelle *LESSON_ORDER* wird die Reihenfolge der vorhandenen Aufgaben festgelegt. Jede Aufgabe hat genau einen Eintrag in dieser Relation. Durch diese Relation wird einer bestimmten Aufgabe, ein fester Platz in der Abarbeitungsreihenfolge der Aufgaben zugewiesen.

### LOGON

Die Relation *LOGON* dient zur Speicherung der richtigen Lösungen. Die Entscheidung, ob eine Aufgabe richtig oder falsch gelöst wurde, wird mit Hilfe der Referenzlösung (siehe Relation *LESSONS*) durch die Logik der Praktikumsanwendung

getroffen. Wird sie als richtig gewertet, wird das SQL-Statement, die zugehörige Aufgaben-ID, die Nutzer-ID des Studenten und das Lösungsdatum in der Tabelle *LOGON* gespeichert. Die hier abgelegten Informationen werden zur Auswertung der Praktikumsleistungen der Studenten benötigt.

#### ERROR_LOG

Fehlerhafte Lösungsversuche werden ebenfalls zur Auswertung benötigt. Einerseits kann der Student jederzeit überprüfen, welche SQL-Statements bisher nicht den gewünschten Erfolg gebracht haben, andererseits lassen sich aus dem Inhalt dieser Tabelle Rückschlüsse auf den Schwierigkeitsgrad der einzelnen Aufgaben ziehen und die Administratoren können dementsprechend darauf reagieren. Gespeichert werden in der Relation *ERROR_LOG*, genauso wie in der Tabelle *LOGON*, das SQL-Statement, die Aufgaben-ID, die Nutzer-ID und das Datum des Versuches.

### 7.3.2 Nutzertablespaces

Dieser Bereich der Datenbank muß gesondert betrachtet werden, da die Praktikumsteilnehmer hier aktiv Objekte anlegen dürfen und daher eine besondere Sicherheitsstrategie zur Anwendung kommt.

Bevor die Studenten mit der Formulierung der Abfragen an den statischen Bereich der Datenbank beginnen, werden sie im Vorbereitungsteil (vgl. Abschnitt 6.4.1) des Praktikums aufgefordert, eine Tabelle anzulegen und diese mit selbst gewählten Daten zu füllen. Diese Aktionen stellen natürlich ein hohes Risiko für die gesamte Anwendung dar. Es ist davon auszugehen, daß von den Studenten fehlerhafte Statements eingegeben werden, die unerwünschte Ergebnisse liefern oder sogar Schaden in der Datenbank anrichten. Aus diesem Grund wurde der Nutzerbereich der Datenbank entworfen. Dieser Bereich ist genau genommen eine lose Zusammenfassung voneinander unabhängiger Tablespaces, die jeweils genau einem Praktikumsteilnehmer zugeordnet werden können.

Bevor eine Tabelle erstellt werden kann, wird durch die Anwendungslogik ein individueller Tablespace angelegt, der automatisch eine Bezeichnung erhält, über die er eindeutig mit einem speziellen Nutzer verknüpft wird. In diesem "Container" hat der zugehörige Nutzer die Möglichkeit, eine eigene Tabelle zu erstellen und zu bearbeiten. Die Bezeichnung des Nutzertablesspaces ist eine Kombination aus dem Loginnamen und der Nutzer-ID des Studenten und somit eindeutig. Der individuelle Tablespace für einen Praktikumsteilnehmer mit dem Login-Namen *Schulz* und der Nutzer-ID *73* würde demnach mit *Schulz73* bezeichnet werden. Der Anwender hat keinen Einfluß auf die Benennung dieses Bereiches und bemerkt von der internen Strukturierung

nichts.

Die logische Trennung des Nutzerbereiches von den anderen Datenbank-Bereichen erfolgt durch verschiedene Hilfsmittel, die auf der einen Seite die Praktikumsanwendung selbst hergibt, auf der anderen Seite aber von der Datenbank bereitgestellt werden.

Die Verbindung zum dynamischen Bereich der Datenbank erfolgt durch einen speziellen Benutzer, der nur zur Abhandlung dieser kritischen Aufgaben erstellt wurde. Er ist mit wesentlich mehr Rechten ausgestattet, als der normale Datenbank-Benutzer, welcher zur Verbindung mit dem statischen Teil genutzt wird. Dieser besonders privilegierte Benutzer besitzt neben den Standard-Rollen *CONNECT* und *RESOURCE* auch die Systemberechtigungen *CREATE TABLESPACE* sowie *SELECT ANY DICTIONARY* und verfügt über *UNLIMITED TABLESPACE* (vgl. Abschnitt 4.5). Innerhalb der Praktikums-Anwendung wird entschieden, für welche Aufgaben, welcher Benutzer zum Datenbank-Logon verwendet wird. Im Abschnitt 7.5.1 wird die Verbindung zur Datenbank ausführlich beschrieben.

Um sicherzustellen, daß der Nutzerbereich der Datenbank auch über einen längeren Zeitraum hinweg übersichtlich bleibt und seine Funktion zuverlässig erfüllt, werden die Objekte der Teilnehmer nach Beendigung des Praktikums aus der Datenbank gelöscht. Diese Aktion wird automatisch ausgeführt, wenn ein Student das Praktikum erfolgreich beendet hat. Zusätzlich verfügt der Administrator über eine Funktion, mit der sämtliche Tablespaces gelöscht werden, die zu Teilnehmern gehören, welche nicht mehr im Praktikum aktiv sind.

## 7.4 Das Webinterface

Das Webinterface stellt die interaktive Schnittstelle zu allen Komponenten des Online-Praktikums dar. Die Implementierung erfolgte vorrangig in der Skriptsprache *PHP*. Für bestimmte Funktionen finden zusätzlich *Perl* und *Java-Funktionen* Verwendung. Weiterführende Informationen zur Nutzung von *Perl* werden im Abschnitt 7.7 gegeben. *Java* wird in der Anwendung genutzt, um die Kontextinformationen der Menüpunkte des Webinterface dynamisch einzublenden und soll in dieser Arbeit nicht weiter besprochen werden.

Die Nutzer kommunizieren über einen beliebigen Internetbrowser mit der Anwendung. Um größtmögliche Sicherheit zu gewährleisten, ist die Praktikumsanwendung in einem privaten Bereich des Apache-Webserver abgelegt. Autorisierte Praktikumsteilnehmer werden nach Eingabe der *URL* [1] aufgefordert, einen Benutzernamen und ein Kennwort einzugeben und erhalten bei Verwendung der richtigen Kombination

Zugang zur Startseite des Praktikums.

Nun erfolgt die eigentliche Anmeldung zum Online-Praktikum. Versucht sich ein Teilnehmer für das Praktikum anzumelden, wird eine Datenbankverbindung aufgebaut und die Eingaben des Anfragenden mit den Nutzer-Daten aus der Datenbank verglichen. Dabei werden drei beziehungsweise vier verschiedene Kriterien überprüft und entsprechend darauf reagiert:

1. Login/Passwort-Prüfung
2. Aktivierungs-Prüfung
3. Prüfung des Nutzertyps
4. Status-Abfrage (nur für normale Nutzer)

#### Login/Passwort-Prüfung

Das Login erfolgt auf der Startseite (*praktikum.php*) der Praktikumsanwendung. Als erstes werden der Loginname und das Paßwort überprüft. Diese Kombination ist in einer bestimmtem Tabelle der Hintergrund-Datenbank (vgl. Abschnitt 7.3.1) abgelegt. Ist das nicht der Fall, wird der Nutzer abgewiesen und es erfolgt eine aussagekräftige Fehlermeldung.

#### Aktivierungs-Prüfung

Der Administrator ist in der Lage, Nutzer für das Praktikum zu sperren. Deren Datensätze bleiben dabei weiterhin in der Datenbank gespeichert und können jederzeit wieder für die Nutzung des Praktikums aktiviert werden. Handelt es sich bei dem Loginversuch um einen deaktivierten Teilnehmer, erfolgt ein Hinweis auf diesen Status. Der Nutzer erhält jedoch keinen Zutritt zum Praktikum.

#### Prüfung des Nutzertyps

Der Nutzertyp ist ebenfalls in der Datenbank definiert und entscheidet darüber, ob ein Teilnehmer in den Administrationsbereich oder in den normalen Nutzerbereich geleitet wird. Sind die ersten beiden Stufen der Prüfung erfolgreich absolviert worden, steht Administratoren bereits nach dieser Prüfung der volle Funktionsumfang der Anwendung zur Verfügung.

#### Status-Abfrage

Die Studenten können die Abarbeitung des Praktikums in beliebig vielen Sitzungen durchführen. Während jeder Sitzung wird der Aufgabenstand der Teilnehmer in der Datenbank abgespeichert und steht bei erneutem Login stets aktuell zur

Verfügung. Diese Prüfung ermöglicht es den Studenten, die bisher unternommenen Lösungsversuche, sowie die bereits abgeschlossenen Aufgaben, noch einmal einzusehen und gegebenfalls daran anzuknüpfen. Wie diese Status-Abfrage im Einzelnen realisiert wird, beschreibt der Abschnitt 7.4.1.

Hat der Nutzer die Anmeldungsprozedur erfolgreich durchlaufen, wird eine PHP-Session gestartet, die über die gesamte Sitzungsdauer hinweg Informationen über den Nutzer und seine Aktivitäten bereitstellt (vgl. Abschnitt 7.4.4) und darüber hinaus zusätzliche Sicherheit bietet. Über ein Menü sind alle Funktionen des Webinterfaces erreichbar, wobei Administratoren und Studenten nur bestimmte, auf die speziellen Anforderungen der Nutzertypen abgestimmte, Bereiche der Anwendung nutzen können.

### 7.4.1 Nutzerinterface

Der prinzipielle Aufbau der Seite ist für beide Nutzertypen gleich. Sie besteht aus vier Bereichen - einem *Header*, dem *Navigationsmenü*, einem *Abstrakt* und dem *Inhaltsbereich* der Anwendung. In Abbildung 7.3 ist ein Formular zur Eingabe der SQL-Statements und das Menü des Nutzerinterfaces dargestellt.

Der Header (*header.php*) bildet den optischen und inhaltlichen Rahmen des Seitenlayouts. Diese Datei wird bei jedem Seitenaufruf includiert und enthält neben einer Überschrift und dem Logo der Hochschule Wismar, das *Navigationsmenü*, den *Abstrakt* und einen Button zum Logout des Nutzers. Über diesen Button wird die *PHP-Session* geschlossen und die Sitzung ordnungsgemäß beendet. Um zu verhindern, daß Nutzer ohne gültiges Login in die Anwendung eindringen können, wird durch den Header ebenfalls überprüft, ob ein Seitenaufruf aus einer gültigen Session heraus erfolgte oder nicht. Ist das nicht der Fall, wird der Nutzer abgewiesen und aufgefordert, sich erneut über die Startseite anzumelden. Außerdem werden durch den Header Funktionen includiert, die in jedem Bereich der Anwendung zur Verfügung stehen müssen und zur Verbindung mit der Datenbank genutzt werden.

Das Menü des Nutzerinterfaces bietet Zugang zu allen Funktionen, die mit der Abarbeitung des Praktikums zusammenhängen. Fährt der Nutzer mit der Maus über einen Menüpunkt, wird er über eine Texteinblendung im *Abstrakt-Bereich* darüber informiert, welche Funktion sich hinter dem Menüpunkt verbirgt. Das Menü setzt sich aus folgenden Punkten zusammen:

1. Aufgaben
2. persönliche Auswertung

3. Hilfe
4. links
5. Prof. Dr.-Ing. A.Düsterhöft
6. KAPV-Situation

Beim Klick auf einen Menüpunkt, wird im *Inhalts-Bereich* der Anwendung die entsprechende Funktion bereitgestellt. Im folgenden werden die einzelnen Punkte des Menüs und deren Funktionen detailliert besprochen.

### Aufgaben

Dieser Menüpunkt führt in den individuellen Aufgabenbereich (*lessonstart.php*) des Nutzers. Die Aufgaben werden zunächst in Form einer übersichtlichen Linkliste, die für jeden Studenten persönlich erstellt wird, bereitgestellt. Dazu wird beim Aufruf dieses Bereiches eine Datenbankverbindung aufgebaut und geprüft, welche Aufgaben der Nutzer bereits abgeschlossen hat. In die zu erstellende Liste werden nur Aufgaben aufgenommen, die noch nicht gelöst wurden.

Jede noch zu bearbeitende Aufgabe bekommt einen Eintrag in der Liste und wird mit der entsprechenden *Aufgabennummer* und der *Aufgabenkategorie* bezeichnet. Hinter jedem dieser Einträge verbirgt sich ein Link, der zu einem Eingabeformular führt, über das die zugehörige Aufgabe bearbeitet werden kann. In der Abbildung 7.3 ist anhand einer Beispielaufgabe zu erkennen, wie solch ein Formular aufgebaut ist und aus welchen Informationen die *URL* des Links zusammensetzt ist. Die vollständige Adresse in diesem Beispiel lautet:

```
https://athos.et.hs-wismar.de/dipl/lessons/lessoni.php?
   lessNr=53&Nr=15&SQLtype=SELECT&Part=Abfragen&
```

Im ersten Teil der Adresse ist die relative Lage der Datei *lessoni.php* auf dem Apache-Server definiert. In diesem *PHP-Skript* ist das Layout des Formulars und dessen Funktion definiert. Beim Aufruf dieses Skripts werden die Session-Variablen *$_GET['lessNr']*, *$_GET['Nr']*, *$_GET['SQLtype']* und *$_GET['Part']* benötigt, über die die Inhalte der Aufgaben aus der Datenbank ermittelt werden.

Die Variable *$_GET['lessNr']* entspricht der eindeutigen ID, mit der jede Aufgabe in der Datenbank assoziiert wird. Diese Angabe wird genutzt, um die Abfrageversuche zur Auswertung zu speichern und später wieder genau einer Aufgabe zuordnen zu können. Außerdem wird über diese Variable der Aufgabentext und die Musterlösung ermittelt. Der Aufgabentext entspricht der Fragestellung, die der Student über das

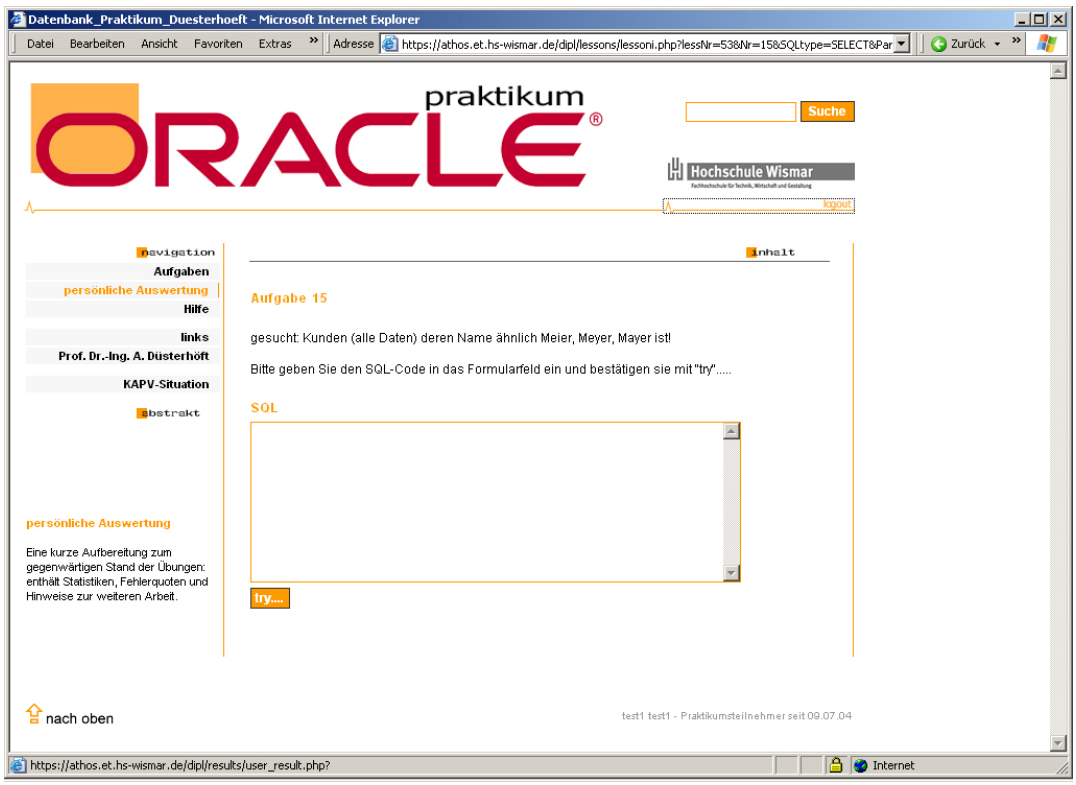



**Abbildung 7.3:** Eingabeformular des Nutzerbereiches der Online-Anwendung

Webinterface bekommt. Die Musterlösung wird mit den Lösungsvorschlägen der Studenten verglichen und stellt das ausschlaggebende Kriterium bei deren Bewertung dar.

Durch die Variable *$_GET['Nr']* wird die Reihenfolge festgelegt, in der die Aufgaben in der Liste auftauchen. Außerdem wird diese Nummer für eine Überschrift im Eingabeformular benötigt. Auch sie wird mit Hilfe der ID, aus einer Tabelle der Datenbank (vgl. Abschnitt 7.3.1), ermittelt.

Die *Aufgabenkategorie* wird mit *$_GET['SQLtype']* beschrieben. Diese Information ist notwendig, um eine Vorentscheidung zur Richtigkeit der Lösungsversuche zu fällen. Die Kategorie ist das Schlüsselwort, des in der Aufgabenstellung gesuchten SQL-Statements und kann die Werte *SELECT*, *CREATE* oder *INSERT* annehmen. Wird also nach einem *SELECT-Statement* gesucht, erfolgt mit Hilfe einer speziellen *OCI8-Funktion* (vgl. Abschnitt 7.4.2) eine Prüfung, ob die Eingabe des Nutzers dieses Schlüsselwort enthält oder nicht. Abhängig vom Ergebnis dieser Untersuchung wird eine entsprechende Meldung über das Webinterface ausgegeben und die Bearbeitung fortgesetzt oder unterbrochen.

Die Aufgaben sind in die zwei Bereiche *Vorbereitung* und *Abfragen* gegliedert. Auch

die Information, zu welchem *Part* eine Aufgabe gehört, ist in der Datenbank gespeichert. Die Variable *$_GET['Part']* wird zur internen Trennung der beiden Bereiche benötigt.

Will der Student eine Aufgabe bearbeiten, klickt er einen Link in der Aufgabenliste an. Daraufhin wird durch das Skript *lessoni.php* im *Inhaltsbereich* des Webinterfaces - wie in Abbildung 7.3 zu sehen - die Aufgabenstellung und ein Textfeld zur Eingabe der Lösungen bereitgestellt. Die Lösungsversuche werden in das Eingabefeld des Fomulars geschrieben und mit einem Button abgeschickt. Das eingegebene *SQL-Statement* wird nun in der Datenbank ausgeführt und mit dem Ergebnis der *Musterlösung* verglichen. Daraufhin erfolgt eine Auswertung der Eingabe, wobei die Ergebnisse sinnvoller Abfragen dem Nutzer zur Kontrolle angezeigt werden und Hinweise auf eventuelle Abweichungen von der Musterlösung als Ausgabe von Fehlermeldungen (vgl. Abschnitt 7.4.2) erfolgen. Wird die Eingabe als richtig gewertet, erfolgt ein Eintrag in der Datenbank und der Nutzer wird zur nächsten Aufgabe weitergeleitet.

Eine Erweiterung dieses Bereiches stellt das Skript *showlessons.php* dar, welches von der Aufgabenliste (*lessonstart.php*) aus zugänglich ist. Über einen Link im unteren Bereich der Seite wird der Nutzer auf eine Seite geleitet, in der alle Aufgabenstellungen im Überblick aufbereitet sind. Dieser Zusatz soll es dem Nutzer ermöglichen, alle Fragestellungen schnell und bequem einsehen zu können, ohne jede Aufgabe einzeln öffnen zu müssen. In dieser Sektion werden auch die bereits abgeschlossenen Aufgaben, die dazugehörigen Fragestellungen, deren Musterlösungen, sowie die Lösungen des Studenten zur Kontrolle mit angezeigt.

#### persönliche Auswertung

Diese Auswertungsfunktion ermöglicht es den Studenten, die persönlich eingegebenen SQL-Statements, aller bisher bearbeiteten Aufgaben, noch einmal einzusehen. Dabei werden auch alle erfolglosen Versuche angezeigt, die in der Datenbank gespeichert wurden. Besonders nützlich ist diese Funktion, wenn ein Teilnehmer eine Aufgabe bereits bearbeitet, jedoch noch keine richtige Lösung gefunden hat. Setzt der Teilnehmer seine Arbeit in einer späteren Sitzung fort, kann er durch die persönliche Auswertung alle bisher eingegebenen Lösungsversuche schnell und nach den Aufgaben-Nummern geordnet nachvollziehen. Zusätzlich wird dem Studenten eine Information über die bisherige Dauer des Praktikums gegeben, die aus den Datumsangaben seiner Eingabeversuche ermittelt wird.

#### Hilfe

Die Hilfe des Online-Praktikums soll den Anwendern Tips zum Umgang mit dem Programm geben. Auf der einen Seite beziehen sich diese Hinweise auf die Funktion der Webanwendung selbst, auf der anderen Seite aber enthält dieser Bereich auch eine umfassende Beschreibung von *Oracle SQL* und seinen Besonderheiten. Im Verlauf der Nutzung des Praktikum könnten die Inhalte des Hilfebereiches erweitert und aktualisiert werden.

#### links

Hinter diesem Punkt verbirgt sich eine Linksammlung, die dem interessierten Praktikumsnutzer weiterführende Informationen zum Umgang mit Oracle und SQL geben soll. Themengebiete, die in der Vorlesung und im Praktikum nicht angesprochen werden können, sowie spezielle Dokumentationen und Publikationen, sind von hier aus zugänglich.

#### Prof. Dr.-Ing. A.Düsterhöft

Dieser Menüpunkt führt direkt zur Seite von Prof. Dr.-Ing. A.Düsterhöft. Die Studenten können sich auf diesem Wege schnell über Termine, Adressen oder Projekte informieren.

#### KAPV-Situation

Das Verständnis der KAPV-Datenbank ist die Grundlage zur Abarbeitung des Praktikums. Ähnlich wie in Kapitel 5 dieser Arbeit, wird den Studenten die Unternehmens-Situation und die Datenbankstruktur der Kunden-Auftrag-Produkt-Verwaltung in Worten beschrieben und in Form eines *Entity-Relationship-Models* als PDF-Datei zur Verfügung gestellt.

### 7.4.2 Kontrolle der Eingaben

Eine wesentliche Funktion der Anwendung ist die Überprüfung der Nutzereingaben auf ihre Richtigkeit. Die Bewertung ergibt sich aus einer Kombination von Programmschleifen der PHP-Skripte und den Rückgabewerten der Datenbank. Es müssen hierzu Kriterien gefunden werden, anhand derer zweifelsfrei und effektiv über das Ergebnis des Lösungsvorschlages geurteilt werden kann. Die Kontrolle erfolgt in mehreren Stufen.

Bevor jedoch mit der eigentlichen Prüfung der Eingaben begonnen wird, muß entschieden werden, ob eine Aufgabe zum Vorbereitungsteil oder zum Abfragenteil des Praktikums gehört. Im Vorbereitungsteil wird der Nutzer aufgefordert, selbst Datenbankobjekte anzulegen, beziehungsweise zu manipulieren. Diese Tatsache erfordert

eine spezielle Prüfung, auf die in diesem Abschnitt noch gesondert eingegangen wird. Wird eine Aufgabe aus dem Abfragenteil bearbeitet, läuft die Kontrolle wie folgt ab:

#### Abfragenteil

#### Stufe 1

Das SQL-Statement wird von den Nutzern in das Eingabefeld des Formulars eingetragen und per Klick auf den Formular-Button zum Webserver gesandt. Dieser interpretiert die Eingabe als *String*, der nun als Abfrage an die Datenbank weitergeleitet werden kann. Bevor das vermeintliche SQL-Statement jedoch ausgeführt wird, erfolgt ein Vergleich mit der *Aufgabenkategorie* (vgl. Abschnitt 7.4.1). Dabei wird das Statement vorerst nur an eine OCI8-Funktion übergeben, mit der der *Statementtyp* ermittelt wird. Stimmt dieser Typ mit dem in der Aufgabe gesuchten Typ überein, wird das Statement ausgeführt. Anderenfalls erfolgt eine Fehlermeldung, in der der Nutzer darauf hingewiesen wird, daß seine Eingabe nicht mit dem gesuchten Typen harmoniert.

Diese Prüfung hat den Vorteil, daß unsinnige Eingaben nicht ausgeführt und nicht in der Datenbank gespeichert werden.

#### Stufe 2

Ist die erste Stufe der Kontrolle erfolgreich durchlaufen worden, muß die Eingabe genauer überprüft werden. Dazu wird das Statement an eine OCI8-Funktion übergeben und in der Datenbank ausgeführt. Diese Funktion liefert einen Rückgabewert, über den entschieden werden kann, ob das SQL-Statement korrekt ausgeführt wurde oder nicht. Wird der Wert *-1* zurückgegeben, wurde die Abfrage nicht korrekt beendet. Eine weitere OCI8-Funktion zur Anzeige des Datenbankfehlers übernimmt die Ausgabe der Oracle-Fehlermeldung. Diese Meldung gibt dem Nutzer bestmöglichen Aufschluß über die Art des Fehlers. Neben der Fehlermeldung wird ein *link* eingeblendet, der den Nutzer zurück zum Eingabeformular führt, welches daraufhin das zuletzt geprüfte SQL-Statement zur Korrektur bereitstellt.

In der Regel werden in dieser Stufe der Kontrolle Abfragen auf nicht existierende Datenbankobjekte beziehungsweise syntaktisch falsche Abfragen gefiltert. Dabei bietet Oracle eine relativ detaillierte Fehlermeldung, inklusive der Oracle Fehlercode-Nummer, die bei der weiteren Bearbeitung der entsprechenden Aufgabe von Nutzen ist. Über die Fehlercode-Nummer erhält er Nutzer, auf einer Webseite zur Suche der Oracle-Fehler [7], Hilfe zur Lösung dieser Probleme. Die Ausgabe der Fehlermeldung erfolgt in deutscher Sprache.

Eingaben, die in der 2. Stufe der Kontrolle als fehlerhaft bewertet werden, gehen mit


**Abbildung 7.4:** Beispiel für eine Ausgabe der Fehlermeldung der Stufe 2

in die Auswertungen der Nutzeraktivitäten ein und werden in der dafür vorgesehenen Tabelle der Hintergrund-Datenbank gespeichert.

### Stufe 3

Hat eine Eingabe auch die 2. Stufe der Überprüfung erfolgreich durchlaufen, muß überprüft werden, inwieweit das Statement mit der gesuchten Lösung der Aufgabe übereinstimmt. Für diesen Teil der Kontrolle besitzt jede Aufgabe des Abfragenteils eine Referenzlösung, die in einer Tabelle der Hintergrund-Datenbank (vgl. Abschnitt 7.3.1) abgelegt ist.

Der Vergleich erfolgt über die Ergebnismengen der Referenzlösung und der Abfrage des Nutzers. Dazu werden beide Statements in der Datenbank ausgeführt und anschließend überprüft, ob die Anzahl der zurückgegebenen Spalten und Reihen beider Lösungen übereinstimmen. Ist dies nicht der Fall, wird die angebotene Lösung der Aufgabe als nicht gelöst gewertet. Zur Kontrolle wird dem Nutzer jedoch das Resultat seiner Lösung ausgegeben. Zusätzlich erhält er eine Information über die Differenz zur Ergebnismenge der Referenzlösung. Auch die, in diesem Teil der Prüfung als falsch gewerteten Lösungsversuche, werden zur Auswertung gespeichert.

#### Stufe 4

In die abschließende Kontrolle gelangen nur Lösungsangebote, deren Ergebnismengen quantitativ mit der der Referenzlösung übereinstimmen.

Die Inhalte der Resultate der vom Praktikanten eingegebenen Abfrage sowie die der gesuchten Musterlösung liegen nach der Ausführung in der Datenbank in Form von Arrays vor, die nun Feld für Feld miteinander verglichen werden können. Aus diesem Grunde ist es notwendig, die Formulierung der Aufgabenstellung strikt zu beachten, da bereits das Vertauschen von Spalten zu einer negativen Bewertung führen würde. Diese strenge Form der Auswertung hat sich bereits in der Praxis als sinnvoll erwiesen und entspricht den Zielen des Praktikums. Im Kapitel 8 ist dokumentiert, welche Tests und Überlegungen zur Anwendung dieser Form der Auswertung geführt haben.

Erst wenn eine Nutzereingabe alle Stufen der Kontrolle mit einem positiven Ergebnis durchlaufen hat, wird eine Aufgabe als gelöst gewertet und das SQL-Statement in der dafür vorgesehenen Tabelle der Hintergrund-Datenbank gespeichert. Der Nutzer wird durch das Webinterface über die erfolgreiche Lösung der Aufgabe informiert und das Ergebnis der richtigen Lösung in Form einer Tabelle ausgegeben. Zusätzlich enthält die Auswertung einen *link*, der auf das Eingabeformular mit der nachfolgenden Aufgabenstellung verweist. Die Aufgaben des Abfragenteils müssen jedoch nicht sequentiell abgearbeitet werden. Über das Navigationsmenü kann zu jedem beliebigen Zeitpunkt die Aufgabenliste und eine darin enthaltene Aufgabe des Abfragenteils zur Bearbeitung geöffnet werden. Abgeschlossene Aufgaben werden dem Studenten von nun an nicht mehr in der Aufgabenliste angeboten, können aber über die persönliche Auswertung beziehungsweise die Aufgabenübersicht (vgl. Abschnitt 7.4.1) jederzeit nachvollzogen werden.

### Vorbereitungsteil

Die Aufgaben des Vorbereitungsteils können nicht nach dem oben beschriebenen Konzept bewertet werden. In diesem Praktikumsteil sollen die Studenten individuelle Lösungen finden, für die möglichst wenig Einschränkungen gelten und gleichzeitig alle Aspekte der Sicherheit beachtet werden sollen.

Wie schon im Kapitel 6 deutlich wurde, wird der Nutzer in den Aufgabenstellungen der Vorbereitung aufgefordert, Tabellen anzulegen, diese mit selbst gewählten Daten zu füllen und deren Inhalte anschließend abzufragen. Diese Anforderungen setzen eine sequentielle Abarbeitung der Aufgaben des Vorbereitungsteils voraus. Weiterhin muß sichergestellt werden können, daß die angelegten Objekte den zugehörigen Besitzern stets eindeutig zugewiesen werden können. Referenzlösungen zur Kontrolle der Richtigkeit der Lösungen, wie sie im Abfragenteil des Praktikums genutzt werden, kommen hier aufgrund dieser Bedingungen nicht zum Einsatz.

Die Mechanismen zur Überprüfung der Syntax der Eingaben stellt im wesentlichen die Datenbank zur Verfügung, während die Ablaufkontrolle und Zuordnung der Objekte zu den Nutzern von der Anwendungslogik übernommen werden.

Beim Aufruf einer beliebigen Aufgabe durch einen Nutzer, muß eingangs eine Überprüfung stattfinden, welchem Teil des Praktikums diese Aufgabe zuzuordnen ist. Die Kontrolle der Aufgaben des Abfragenteils ist im ersten Teil dieses Absatzes bereits ausführlich erläutert worden. Soll eine Aufgabe des Vorbereitungsteils bearbeitet werden, muß eine Möglichkeit gefunden werden, die individuellen Lösungen der Studenten möglichst sinnvoll zu bewerten. Diese Bewertung läuft ebenfalls stufenweise ab.

#### Stufe 1

Die Aufgabenkategorie kann im Vorbereitungsteil die Werte *CREATE*, *INSERT* oder *SELECT* annehmen. Ähnlich wie in der ersten Kontrollstufe des Abfragenteils wird die Eingabe des Praktikumsteilnehmers vorerst nur betreffend dieser Kategorie mit der geforderten Lösung verglichen. In dieser Stufe werden vorerst nur Eingaben gefiltert, die nicht als sinnvolles SQL-Statement erkannt werden und Antworten, die nicht der Fragestellung entsprechen. Lösungsvorschläge, die in dieser Stufe negativ bewertet werden, fließen nicht mit in die spätere Auswertung ein.

#### Stufe 2

Die Charakteristik der Aufgabenstellungen der Vorbereitung erfordert die Einhaltung einer bestimmten Reihenfolge bei der Abarbeitung der einzelnen Aufgaben. Natürlich muß eine Tabelle erst angelegt werden, bevor sie mit Daten gefüllt oder abgefragt werden kann. Wird eine Aufgabe des Vorbereitungsteils bearbeitet, gelangt der Nutzer in einen Programmzweig, in dem überprüft wird, ob bereits ein Nutzertablespace angelegt wurde oder nicht.

Dazu wird in dem Oracle System-View *dba_tables* nach der individuellen Tabelle des entsprechenden Nutzers gesucht und der Name dieser Tabelle als Variable, für den weiteren Verlauf der Kontrolle, zur Verfügung gestellt. *dba_tables* enthält alle Tabellennamen des Systems, einschließlich Angaben über den Datenbanknutzer, den Tablespacenamen und eine Vielzahl weiterer datenbankinterner Angaben, die in dieser Arbeit nicht näher betrachtet werden sollen.

Hat der Nutzer bereits eine Tabelle angelegt, kann ihm in diesem System-View genau ein Tablespace zugeordnet werden. Welche Strategie bei der Zuordnung der Nutzertablespaces verfolgt wird, beschreibt der Abschnitt 7.3.2.

Mit dem Ergebnis dieser Abfrage können zwei für den weiteren Verlauf der Kontrolle wesentliche Aussagen sicher getroffen werden:

1. Hat der Nutzer bereits die Aufgabe der Kategorie *CREATE* erfolgreich abgeschlossen?

2. Welche Bezeichnung wurde für die dabei angelegte Tabelle gewählt?

Ist das erste Kriterium erfüllt, wird in dem System-View *dba_tab_columns* die Tabellenstruktur ermittelt und wiederum als String-Variable bereitgestellt. In *dba_tab_columns* kann anhand des im ersten Schritt ermittelten Tabellennamens festgestellt werden, welche Spaltenbezeichnungen die vom Nutzer angelegte Tabelle besitzt und welchen Typ die einzelnen Spalten haben. Die Tabellenstruktur wird dem Nutzer anschließend zur Bearbeitung der folgenden Aufgaben in der Aufgabenstellung zur Erinnerung angezeigt.

Versucht ein Teilnehmer nun eine Tabelle mit Daten zu füllen, obwohl im Vorfeld noch keine individuelle Tabelle angelegt wurde, wird er durch eine entsprechende Ausgabe darauf hingewiesen, die logische Abarbeitungsreihenfolge einzuhalten. Durch die Anwendungslogik wird ebenfalls sichergestellt, daß die Praktikanten nur Objekte innerhalb ihres persönlichen Tablespaces manipulieren können.

Eingaben, die in dieser Stufe der Kontrolle abgewiesen werden, werden ebenfalls nicht zur Auswertung genutzt.

#### Stufe 3

Ist durch die ersten beiden Stufen der Kontrolle sichergestellt, daß ein Nutzer die geforderte Abarbeitungsreihenfolge einhält und die nötigen Berechtigungen zur Bearbeitung eines Objektes besitzt, greift die Fehlerbehandlung der Datenbank. Hierzu wird das zu bewertende Statement in der Datenbank ausgeführt und der Rückgabewert, der dafür genutzten OCI8-Funktion, ausgewertet. Kommt es bei der Ausführung des SQL-Statements zu Fehlern, erfolgt in der Auswertung die Ausgabe der Datenbank-Fehlermeldung. Der Nutzer wird aufgefordert, die Aufgabe erneut zu bearbeiten und der gescheiterte Lösungsversuch zur Auswertung in der dafür vorgesehenen Tabelle der Hintergrunddatenbank gespeichert. Läuft die Ausführung problemlos ab, wird die SQL-Lösung in der Datenbank festgehalten, das Ergebnis der Eingabe angezeigt und der Nutzer aufgefordert die nachfolgende Aufgabe zu bearbeiten.

Die Tablespaces der einzelnen Nutzer werden nach erfolgreichem Absolvieren des Vorbereitungsteils automatisch gelöscht, da sie im weiteren Verlauf des Praktikums nicht mehr benötigt werden. Zusätzlich zu dieser automatischen Löschfunktion hat der Administrator jedoch auch die Möglichkeit, Tablespaces von Hand zu löschen.

### 7.4.3 Administrationsbereich

Die Administration der Praktikumsanwendung umfaßt eine Reihe von Aufgaben, die zum einen der Auswertung der studentischen Praktikumsleistungen, zum anderen aber der Wartung, Aktualisierung und Pflege der Anwendung dienen. Beim Login werden Nutzer mit entsprechenden Rechten erkannt und in den Administrationsbereich weitergeleitet.

Der optische Rahmen dieses Bereiches ist mit dem des Nutzerbereiches (vgl. Abbildung 7.3) vergleichbar. Über ein Menü stehen den Verantwortlichen nun folgende Funktionen zur Verfügung:

- Aufgabenverwaltung
- Auswertungen
- Benutzerverwaltung
- Statistik

Über das Webinterface hat der Administrator die Möglichkeit, Inhalte der Hintergrund-Datenbank direkt zu manipulieren und so die Inhalte und den Zugang zum Praktikum zu beeinflussen. Bei der Nutzung dieser Funktionen ist dementsprechend Sorgfalt geboten, da bestimmte Änderungen die Funktion des Praktikums einschränken können. So kann zum Beispiel nur durch den verantwortlichen Hochschulangestellten gewährleistet werden, daß die Musterlösung, die in der Aufgabenstellung gesuchten Inhalte, abbildet.

#### Aufgabenverwaltung

Ziel der Aufgabenverwaltung ist es, die Einflußnahme auf die Formulierungen der Fragestellungen und deren Musterlösungen, die Aktivierung beziehungsweise Deaktivierung, sowie das Hinzufügen von Aufgaben zu erleichtern.

Wird dieser Menüpunkt ausgewählt, erreicht der Administrator ein Formular (*lesson_admin.php*), über das neue Aufgaben angelegt oder vorhandene zur Bearbeitung vorselektiert werden können.

Das Anlegen neuer Aufgaben erfolgt über ein Formular, in das alle relevanten Inhalte, wie Aufgabentext, Musterlösung und Aufgabenkategorie (vgl. Abschnitt 7.3.1) eingegeben werden und per Buttonklick in die Datenbank hochgeladen werden. Dabei findet keine inhaltliche Prüfung statt.

Die Selektierung zur Bearbeitung wird anhand der Aufgabenkategorie vorgenommen. Dazu wird über eine *Listbox* eine Kategorie bestimmt und die Auswahl mit einem

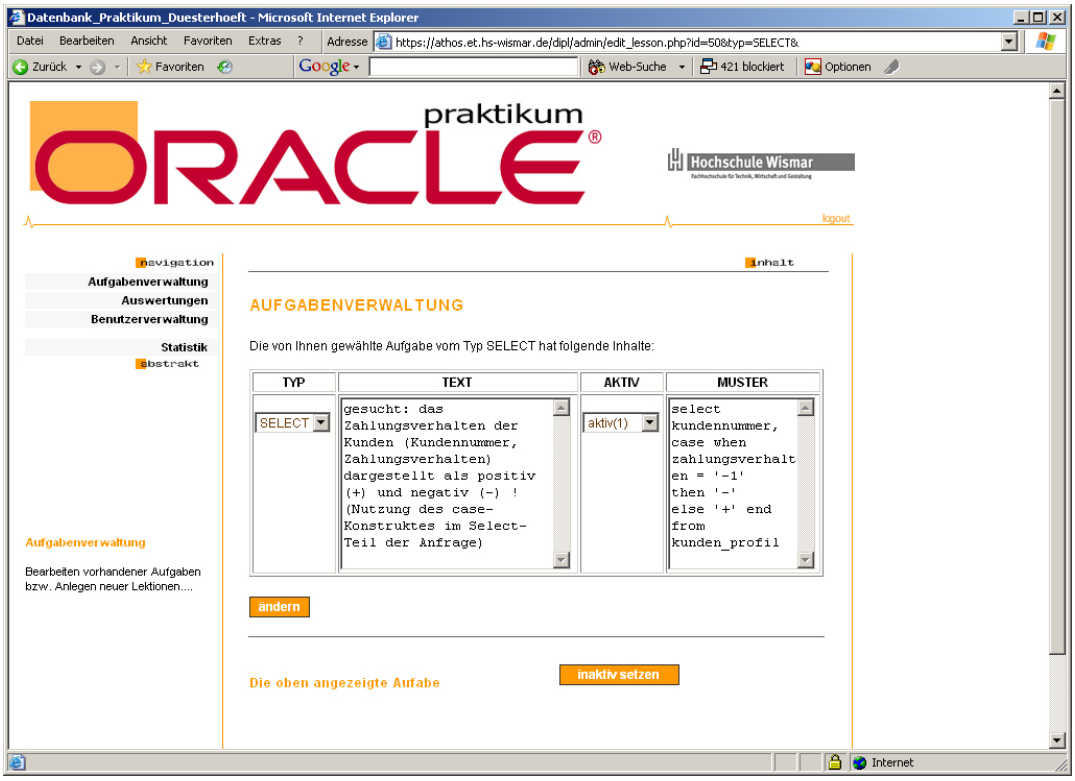



**Abbildung 7.5:** Eingabeformular des Administrationsbereichs zur Bearbeitung der Inhalte vorhandener Aufgaben

Button bestätigt. Die Darstellung aller relevanten Aufgaben erfolgt in Form einer HTML-Tabelle, die die Aufgaben-ID, die Kategorie und den Wortlaut der Fragestellung enthält. Die ID entspricht einem Link, welcher den Nutzer zum eigentlichen Bearbeitungsformular führt. Ein Klick auf diese Kennung führt den Nutzer zu einem weiteren Formular (*edit_lesson.php*), mit dem alle Inhalte direkt bearbeitet werden können. Zusätzlich steht ein Button zu Verfügung, durch den eine selektierte Aufgabe per Klick aktiviert oder deaktiviert werden kann. In Abbildung 7.5 ist ein solches Eingabeformular dargestellt.

Durch die Funktionen der Aufgabenverwaltung werden nur Aufgaben des Abfragenteils des Praktikums erfaßt. Die Aufgaben des Vorbereitungsteils sind aus Sicherheitsgründen vorerst statisch angelegt, können aber mit den Werkzeugen des *Oracle Enterprise Managers* (vgl. Abschnitt 4.6) bei Bedarf ebenfalls direkt bearbeitet werden.

### Auswertungen

Eine wesentliche Funktion der Praktikumsanwendung ist die Fähigkeit zur Speicherung und Darstellung der Aktivitäten der Praktikumsteilnehmer. Anhand der Auswertungsfunktion kann der Praktikumsbetreuer entscheiden, ob ein Student das

Praktikum erfolgreich absolviert hat oder nicht.

Die Grundlage der Auswertung bilden verschiedene SQL-Abfragen, an die Hintergrund-Datenbank, deren Ergebnisse grafisch übersichtlich und aussagekräftig aufbereitet werden. In den Abfragen werden die Inhalte der verschiedenen Tabellen zweckmäßig kombiniert.

Über den Menüpunkt Auswertungen gelangt der Administrator zunächst zu einer Übersicht, in der alle momentan im Praktikum aktiven Teilnehmer und deren Status angezeigt werden. Die Statusangabe erfolgt in Form einer Prozentzahl, die angibt, wieviele Aufgaben bereits erfolgreich abgeschlossen wurden. Grafisch wird diese Ausgabe durch eine Darstellung als Balkendiagramm untermauert, wie in Abbildung 7.6 deutlich wird.

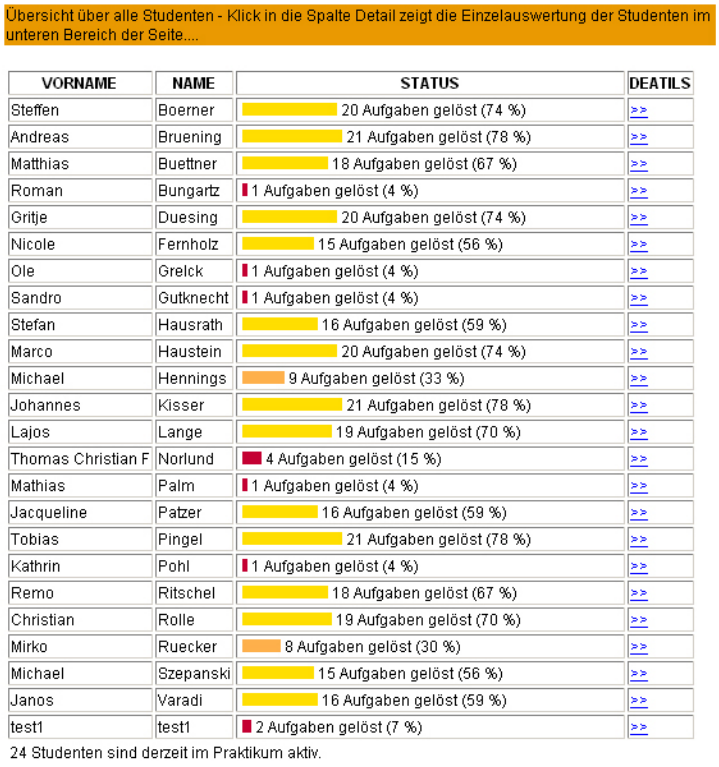
Teilnehmer, die zum Zeitpunkt des Aufrufs noch keine Aufgabe abgeschlossen haben, werden gesondert ausgegeben. In dieser Übersicht werden der vollständige Name, die Gruppe, das Datum der Zulassung und die Anzahl der erfolglosen Versuche der betreffenden Studenten ausgegeben.

Die Informationen, die durch die Übersicht der aktiven Praktikumsteilnehmer geliefert werden, sind für eine detaillierte Auswertung jedoch unzureichend. Aus diesem Grund kann die Auswertung der Arbeit der Teilnehmer erweitert werden, indem ein entsprechender Link in der Übersicht angeklickt wird. Daraufhin werden im unteren Bereich der Auswertungsseite eine Vielzahl von weiteren Informationen eingeblendet. Der Administrator kann hier die Lösungen der einzelnen Aufgaben einsehen, bekommt eine Auswertung, wieviele Versuche zur Lösung der jeweiligen Aufgaben nötig waren und welche Aufgaben beim ersten Lösungsversuch gelöst wurden. Abgeschlossen wird die Ausgabe durch die fehlerhaften Eingaben des Studenten, die anhand der Aufgabennummer geordnet sind.

Der Administrator kann sich im Auswertungsbereich detailliert über die Praktikumsarbeit der Teilnehmer informieren, Vergleiche der Lösungen vornehmen und Rückschlüsse über die Formulierungen der Aufgaben ziehen. Wird beispielsweise bei einem Kurs deutlich, daß eine bestimmte Aufgabe von keinem oder nur sehr wenigen Studenten erfolgreich bearbeitet wurde, kann eventuell mit einer Umstellung der Inhalte darauf reagiert werden.

#### Nutzerverwaltung

Jeder Teilnehmer muß vom Administrator für das Praktikum autorisiert werden. Zur Erleichterung dieser Prozedur stellt die Nutzerverwaltung diverse Funktionen zur Verfügung, die über das Formular *user_admin.php* erreicht werden. Dieses Formular gliedert sich in mehrere Funktionsbereiche:



Übersicht über alle Studenten - Klick in die Spalte Detail zeigt die Einzelauswertung der Studenten im unteren Bereich der Seite....

| VORNAME | NAME | STATUS | DEATILS |
|---|---|---|---|
| Steffen | Boerner | 20 Aufgaben gelöst (74 %) | >> |
| Andreas | Bruening | 21 Aufgaben gelöst (78 %) | >> |
| Matthias | Buettner | 18 Aufgaben gelöst (67 %) | >> |
| Roman | Bungartz | 1 Aufgaben gelöst (4 %) | >> |
| Gritje | Duesing | 20 Aufgaben gelöst (74 %) | >> |
| Nicole | Fernholz | 15 Aufgaben gelöst (56 %) | >> |
| Ole | Grelck | 1 Aufgaben gelöst (4 %) | >> |
| Sandro | Gutknecht | 1 Aufgaben gelöst (4 %) | >> |
| Stefan | Hausrath | 16 Aufgaben gelöst (59 %) | >> |
| Marco | Haustein | 20 Aufgaben gelöst (74 %) | >> |
| Michael | Hennings | 9 Aufgaben gelöst (33 %) | >> |
| Johannes | Kisser | 21 Aufgaben gelöst (78 %) | >> |
| Lajos | Lange | 19 Aufgaben gelöst (70 %) | >> |
| Thomas Christian F | Norlund | 4 Aufgaben gelöst (15 %) | >> |
| Mathias | Palm | 1 Aufgaben gelöst (4 %) | >> |
| Jacqueline | Patzer | 16 Aufgaben gelöst (59 %) | >> |
| Tobias | Pingel | 21 Aufgaben gelöst (78 %) | >> |
| Kathrin | Pohl | 1 Aufgaben gelöst (4 %) | >> |
| Remo | Ritschel | 18 Aufgaben gelöst (67 %) | >> |
| Christian | Rolle | 19 Aufgaben gelöst (70 %) | >> |
| Mirko | Ruecker | 8 Aufgaben gelöst (30 %) | >> |
| Michael | Szepanski | 15 Aufgaben gelöst (56 %) | >> |
| Janos | Varadi | 16 Aufgaben gelöst (59 %) | >> |
| test1 | test1 | 2 Aufgaben gelöst (7 %) | >> |

24 Studenten sind derzeit im Praktikum aktiv.

**Abbildung 7.6:** Ausschnitt der Darstellung der Auswertungen des Administrationsbereichs

- einzelne Nutzer aktivieren/deaktivieren
- Gruppen aktivieren/deaktivieren
- Hinzufügen einzelner Nutzer zu bestehenden Gruppen
- Anlegen einer neuen Gruppe
- Hochladen von Nutzerlisten

Die Daten der Nutzer, die für das System zugelassen sind oder waren, verbleiben auch nach Abschluß des Praktikums in der Datenbank. Dies erfolgt zum einen aus Integritätsgründen, da die gespeicherten Daten der Nutzer in statistische Auswertungen eingehen und daher nicht gelöscht werden. Zum anderen soll sichergestellt werden, daß die Praktikumsergebnisse der Nutzer im Bedarfsfall auch nach Beendigung des Praktikums abrufbar sind.

#### Verwaltung einzelner Nutzer

Das Deaktivieren einzelner Nutzer erfolgt mit Hilfe einer Auswahlliste, in der die vollständigen Namen aller momentan aktiven Teilnehmer angezeigt werden. Die Studenten können per Mausklick ausgewählt werden und mit einem weiteren Klick auf einen Button deaktiviert werden. Damit wird in der Tabelle *STUDENTEN* der Hintergrund-Datenbank das Feld zur Aktivierung/Deaktivierung auf *0* gesetzt, wodurch der Zugang zum Praktikum gesperrt ist.

Das Aktivieren einzelner Nutzer bedarf einer Vorselektierung, die über die Gruppenzugehörigkeit der Studenten vorgenommen wird. Nach der Auswahl einer bestimmten Studentengruppe werden in einer zusätzlichen Liste alle deaktivierten Studenten dieser Gruppe angezeigt. Diese Studenten können jetzt je nach Bedarf für die Nutzung der Anwendung aktiviert werden.

Soll ein einzelner Nutzer zu einer bestehenden Gruppe hinzugefügt werden, steht hierfür ebenfalls ein Extrabereich in diesem Formular zur Verfügung. Das Anlegen einzelner Nutzer erfordert natürlich die Angabe der Gruppenzugehörigkeit, des Namens, des Vornamens, des Logins und des Nutzertypen des Teilnehmers.

#### Verwaltung von Nutzergruppen

Eine weitere Rubrik der Nutzerverwaltung ermöglicht solche Aktionen für gesamte Studentengruppen. In der zuständigen Auswahlliste werden nur Gruppen angezeigt, die bereits Nutzer enthalten, da zur Deaktivierung die Datensätze jedes einzelnen Studenten der Gruppe geändert werden. Per Mausklick kann eine ausgewählte Gruppe aktiviert oder deaktiviert werden.

Die sich anschließende Rubrik zum Anlegen neuer Gruppen ist notwendig, um eine übersichtliche Aufteilung der Nutzerdaten zu gewährleisten. Diese Aufteilung erleichtert dem Administrator das systematische Arbeiten mit der Nutzerverwaltung.

Im letzten Bereich der Nutzerverwaltung kann eine im Vorfeld erstellte XLS-Liste, mit den Daten der Studenten, in die Datenbank übernommen werden. Diese Liste wird durch die verantwortlichen Mitarbeiter der Hochschule aus einer Textdatei generiert, die die erforderlichen Nutzerdaten, wie Namen, Vornamen und Matrikel-Nummer enthält.

Zur Übernahme in die Datenbank muß die Liste nun überprüft und für die weitere Verarbeitung auf den Server transferiert werden. Zu diesem Zweck enthält das Formular eine Upload-Routine, durch die ebenfalls eine Formatüberprüfung realisiert wird. Dateien, die diese Prüfung bestehen, werden in einen speziellen Ordner des Servers kopiert.

Die in der XLS-Datei vorliegenden Informationen müssen im nächsten Schritt extrahiert und sinnvoll aufbereitet werden. Diese Aufgabe übernimmt das Perlskript *parseexel.pl*. Im Abschnitt 7.7 wird der Ablauf dieses Vorgangs ausführlich erläutert. Im Ergebnis des Parsing-Vorgangs stehen die Information der EXCEL-Tabelle in Form einer HTML-Auswahlliste zur Verfügung, aus der schließlich einzelne oder mehrere Datensätze ausgewählt und anschließend in die Datenbank übernommen werden können.

Um auszuschließen, daß Datensätze mehrfach übernommen werden, überprüft die zuständige Insert-Funktion, ob bereits ein Datensatz existiert, der den gleichen Namen, Vornamen und Matrikelnummer enthält. Ist das der Fall, erfolgt eine entsprechende Ausgabe und der Datensatz wird verworfen. Anderenfalls wird der Datensatz in die Tabelle *STUDENTEN* der Hintergrund-Datenbank eingefügt.

#### Statistische Auswertung

Auch hinter diesem Menüpunkt verbirgt sich eine Reihe, miteinander kombinierter, SQL-Abfragen, deren Ergebnisse in eine übersichtliche grafische Form gebracht werden. Die statistische Auswertung beschränkt sich dabei auf Aussagen, die über die Gesamtheit der gespeicherten Datensätze getroffen werden und soll Aufschluß über das allgemeine Nutzerverhalten geben.

Diese Aussagen sind unter Umständen für den Administrator interessant, um Problemen bei der Abarbeitung entgegenzuwirken, indem schwierige Aufgaben überarbeitet werden. Außerdem kann er mit den Informationen spezielle Stärken und Schwächen von Studenten erkennen und gezielt Hilfe anbieten.

### 7.4.4 Die PHP-Session

Zu Beginn des Abschnitts 7.4 wird der Loginvorgang allgemein beschrieben. Ein erfolgreiches Login endet mit der Initiierung einer PHP-Session, die von nun an einen Großteil der Funktionalität der Anwendung bereitstellt. An erster Stelle steht dabei der Transport von Daten durch die Struktur der Anwendung. Bei der Verwendung von Session-Variablen wird dem seit PHP-Version 4.1.0 gültigen Sicherheitskonzept von PHP entsprochen, bei dem der Wert der Konfigurations-Variable *register_globals=off* gesetzt wird. Diese Konfiguration verhindert die Manipulation von Session-Variablen über die URL und erhöht die Sicherheit der Anwendung damit erheblich.

Beim Login werden bestimmte Nutzerdaten aus der Datenbank in Session-Variablen gespeichert. Die Variablen *$_SESSION['type']* für den Nutzertyp, *$_SESSION['user']* für die User-ID, *$_SESSION['since']* für das Aktivierungsdatum, sowie *$_SESSION['name']* und *$_SESSION['fname']* für den vollständigen Namen, werden in vielen Bereichen der Anwendung benötigt und stehen bis zum Beenden der Session jederzeit zur Verfügung. Über eine Session-ID werden die serverseitig gespeicherten Variablen dem entsprechenden Nutzer beziehungsweise der Sitzung zugeordnet. Die Session-ID wird clientseitig je nach Browsereinstellung in einem Cookie abgelegt oder in der URL übermittelt.

Formulardaten werden mit der POST-Methode verarbeitet. Nach dem Abschicken des Formulars stehen dabei alle Inhalte der Formularfelder in Variablen, die nach dem Schema *$_POST['Feldname']* benannt werden. Sollen nun bestimmte Variablen beim Wechsel auf eine andere Seite zur Verfügung gestellt werden, kann diese mit der GET-Methode, wie im folgenden Beispiel übergeben werden:

*https://athos.et.hs-wismar.de/dipl/lessons/lessoni.php?lessNr=43&Nr=6&SQLtype=SELECT&Part=Abfragen?*

Die URL wird zum Aufruf des Eingabeformulars zur Lösung der Aufgaben (vgl. Abbildung 7.3) verwendet und enthält die zur Bereitstellung der Aufgabeninhalte nötigen Informationen, die in Form von GET-Variablen übergeben werden. Das erste Fragezeichen schließt die eigentliche Adreßangabe der aufgerufenen Datei ab. Mit dem Ausdruck *lessNr=43* wird die Variable $_GET['lessNr'] innerhalb der URL übergeben. Diese Variable hat im Beispiel den Wert *43*. Diese Form der Übergabe hat unter anderem den Vorteil, daß die Variablen, abhängig von bestimmten Formulareingaben, in dynamisch generierte Links eingebaut werden können. Nach diesem Schema können beliebig viele GET-Variablen übergeben werden, die durch ein *&* voneinander getrennt werden und mit dem Konstrukt $_GET['Variablenname'] aufgerufen werden. Das letzte Fragezeichen in der URL schließt die Definition der GET-Variablen ab.

Zusätzlich zu dem Bereitstellen der Variablen wird die Session genutzt, um sicherzustellen, daß kein Benutzer ohne gültiges Login in die Webanwendung eindringen kann. Dazu wird bei jedem Aufruf einer Seite stets überprüft, ob die Session-Variable *$_SESSION['user']* für den anfragenden Nutzer gesetzt ist und ob der Nutzer die nötigen Berechtigungen für den angefragten Bereich hat. Diese Prüfung erfolgt innerhalb der Datei *header.php*, die bei jedem Seitenaufruf eingeblendet wird. Verläuft diese Prüfung negativ, wird die Abfrage abgewiesen.

## 7.5 Die OCI8-Bibliothek von PHP

Der Abschnitt 4.7 hat bereits das *Oracle Call Interface - OCI* und dessen Bedeutung bei der Datenbank-Verbindung eingeführt. PHP bietet mit den Funktionen der OCI8-Bibliothek eine Möglichkeit, die volle Leistungsfähigkeit des *OCI* zu nutzen. Im folgenden soll nun die Verwendung der OCI8-Funktionen am Beispiel der Praktikumsanwendung erläutert werden.

Nahezu jede Aktion, die ein Nutzer bei der Abarbeitung der Praktikumsaufgaben ausführt, bezieht sich auf die Inhalte der Datenbank. Schon beim Login wird eine Verbindung aufgebaut und die eingegebenen Logindaten mit Inhalten der gespeicherten Nutzerdaten verglichen. Voraussetzung für eine erfolgreiche Verbindung ist dabei ein korrekt konfigurierter Webserver und die Kenntnis der SID sowie die Login/Paßwort-Kombination eines Datenbank-Benutzers.

Die SID der zu verbindenden Datenbank wird innerhalb des PHP-Skripts als Umgebungsvariable durch folgende Zeile deklariert:

```
PutEnv(‘‘TWO_TASK=dbprakt’’);
```

Damit erfolgt jeder nun folgende Verbindungsversuch auf die Datenbank mit der SID *dbprakt*, falls nicht explizit eine andere SID angegeben wird.

### 7.5.1 Häufig verwendete OCI8-Funktionen

Die folgenden Beschreibungen ausgewählter OCI8-Funktionen soll einen Einblick in die Arbeitsweise der Schnittstelle zwischen PHP und der Datenbank geben. Eine vollständige Dokumentation der Schnittstelle wird auf der PHP-Website [8] angeboten.

#### Die Verbindung mit OCILogon()

Diese Funktion baut eine OCI-Verbindung auf und gibt bei Erfolg einen Verbindungsindex zurück. Im Fehlerfall wird *FALSE* zurückgegeben. Die Funktion wird folgendermaßen beschrieben:

*int OCILogon (string Benutzername, string Passwort [, string Datenbankname])*

Die Parameter *Benutzername* und *Paßwort* definieren den Datenbankbenutzer der zur Verbindung genutzt wird. Der dritte optionale Parameter gibt die lokale Oracle-Instanz an, falls nicht mit der in *TWO_TASK* definierten Instanz verbunden werden soll.

Über die Angabe verschiedener Benutzernamen/Paßwort-Kombinationen wird in der Anwendung die Berechtigung für den Zugriff auf bestimmte Datenbankobjekte realisiert. So sind beispielsweise zur Abarbeitung der Aufgaben des Vorbereitungsteils Berechtigungen zum Anlegen von Tabellen und Views nötig, die für andere Datenbankzugriffe unerwünscht sind.

#### Das Vorbereiten von SQL-Statements zur Ausführung mit OCIParse()

Bevor ein Statement in der Datenbank ausgeführt wird, erfolgt eine Analyse des SQL-Strings. Die Funktion *int OCIParse (int conn, string query)* verwendet den Verbindungsindex *conn*, welcher durch *OCILogon()* zurückgegeben wurde. Der zweite Parameter *query* ist eine String-Variable, welche die vermeintliche SQL-Abfrage enthält. Dabei liefert jede gültige Abfrage den Rückgabewert *TRUE*, der nun zur Ausführung an die zuständige Funktion übergeben wird oder von anderen Funktionen weiterverarbeitet werden kann.

#### Das Ausführen von SQL-Statements mit OCIExecute()

Statements, die mit *OCIParse()* erfolgreich zur Ausführung vorbereitet wurden, können anschließend mit der Funktion *OCIExecute()* ausgeführt werden. Die Funktion ist wie folgt definiert:

*int OCIExecute (int statement [, int mode])*

Als Paramenter *statement* wird der Rückgabewert der Funktion *OCIParse()* angegeben. Der Optionale Parameter *mode* ermöglicht die Wahl des Ausführungsmodus. Dabei wird zwischen *OCI_COMMIT_ON_SUCCESS* und *OCI_DEFAULT* unterschieden. Die Einstellung OCI_DEFAULT erfordert einen expliziten Aufruf der Funktion *OCICommit()* zum Ausführen des Statements, was bei *OCI_COMMIT_ON_SUCCESS* nicht notwendig ist. In der Praktikumsanwendung wird die Einstellung *OCI_COMMIT_ON_SUCCESS* verwendet.

Der Rückgabewert der ausführenden Funktion kann auf verschiedene Weise ausgewertet werden. In der Regel werden die Ergebnisse nach dem Ausführen eines Statements mit *int OCIFetchStatement (int statement, array)* in ein Array geschrieben oder das Result-Set mit *int OCIFetch (int statement)* in einer Schleife durchlaufen und verarbeitet. Der Parameter *statement* ist auch hier der Rückgabewert der Funktion *OCIParse()*.

Bei der Bewertung der studentischen Eingaben kommt die Funktion string *OCIStatementType (int statement)* zum Einsatz. Diese Funktion verarbeitet ein durch *OCIParse()* präpariertes Statement und gibt den Typ eines Statements als Zeichenkette zurück, ohne es auszuführen. Der Rückgabewert der Funktion wird in der Anwendung unter anderem dazu genutzt, um nach verschiedenen Kriterien zu prüfen, ob ein Statement ausgeführt werden soll oder nicht. Unsinnige Eingaben werden so schon früh erkannt und müssen nicht mehr mit *OCIExecute()* ausgeführt werden, was sich günstig auf die Performance und die Sicherheit der Anwendung auswirkt.

Häufig ist es interessant, die Fehlermeldungen der Datenbank abzufangen. In der Praktikumsanwendung wird die Oracle-Meldung beispielsweise für die Auswertung (vgl. Abschnitt 7.4.2) der studentischen Eingaben verwendet. Die Ausgabe der Fehlermeldungen wird durch die Funktion *OCIErrors()* realisiert.

Im Appendix C ist zum Vergleich der Quelltext der Datei connect.php dargestellt. Die darin definierte Prozedur wird zum Aufbau einer Standard-Datenbankverbindung genutzt und kapselt mehrere der oben beschriebenen OCI8-Funktionen.

## 7.6 Verwendung von Triggern und Sequenzen

Trigger dienen zur Überwachung bestimmter Datenbankobjekte. Mit ihrer Hilfe ist es möglich, bestimmte vorhersehbare Situationen zu definieren und gezielte Reaktionen darauf zu veranlassen. Sequenzen sind mit einem Generator erzeugte Zahlenfolgen, die beispielsweise zur Generierung von Indexfolgen genutzt werden.

Bei der Erstellung neuer Aufgaben oder dem Anlegen neuer Nutzer über das Webinterface werden per Formulareingabe Datensätze definiert und in die entsprechenden Tabellen der Datenbank eingefügt. Bei Oracle wird eine Kombination aus Triggern und Sequenzen genutzt, um das allgemein bekannte *auto_increment* für Indizes zu simulieren. Dabei wird die Vergabe des Primärschlüssels durch einen Trigger übernommen. Am Beispiel der Tabelle Nutzer soll dieser Vorgang verdeutlicht werden.

Zunächst muß eine Sequenz mit eindeutigen Werten für den Primärschlüssel erstellt werden, welche später in die Tabelle eingefügt werden sollen. Folgende SQL-Anweisung kann dazu verwendet werden:

```
create sequence STUDENT_INDEX_SEQ start with 1 increment by 1
nomaxvalue;
```

Der Parameter *start with 1* gibt den Startwert der Sequenz an, mit *increment by 1* wird die Schrittweite festgelegt. Der Zusatz *nomaxvalue* gibt an, daß die Sequenz stets inkrementiert werden soll und kein Reset vorgenommen wird.

Nun muß ein Trigger erstellt werden, der bei Bedarf automatisch die nächste Nummer aus der Sequenz in die Index-Spalte einfügt. Dazu wird die folgende SQL-Anweisung verwendet:

```
create trigger INSERT_TRIGGER before insert on test for each row begin
select STUDENT_INDEX_SEQ into :new.STUDENT_ID from dual; end;
```

Ein solches Trigger/Sequenz-Paar existiert auch für das Erstellen neuer Aufgaben. Hierbei muß jedoch noch ein zusätzlicher Trigger erstellt werden, der automatisch einen Eintrag in der Tabelle *LESSON_ORDER* erzeugt. Diese Tabelle wird verwendet, um die Aufgaben in aufsteigender Reihenfolge von 1 bis n durchzunumerieren. Der Trigger sorgt dafür, daß jeder Aufgaben-ID genau eine Aufgabennummer zugeordnet wird. Dazu wird bei Eintritt eines solchen Falls, die derzeitig höchste Aufgabennummer ermittelt und die darauffolgende Ziffer an die neu erstellte Aufgabe vergeben.

Trigger und Sequenzen können auch bequem mit dem Oracle Enterprice Manager erstellt und bearbeitet werden.

## 7.7 Verwendung von Perl

Perl wird ausschließlich für das Parsen der XLS-Liste verwendet, die zur Verwaltung der Nutzergruppen (vgl. Abschnitt 7.4.3) durch den Administrator genutzt wird. Diese Liste enthält die Namen, Vornamen und Matrikel-Nummern der Studenten eines gesamten Matrikels. Die Daten müssen nun zur Speicherung in der Datenbank aus der Liste extrahiert werden und in ein nutzbares Format transferiert werden. Diese Aufgaben übernimmt das CPAN-Modul (**C**omprehensive **P**erl **A**rchive **N**etwork) *Spreadsheet::ParseExcel*, das als Zusatzmodul installiert werden muß. Dieses Modul befähigt Perl Informationen aus Excel95, Excel97 oder Excel2000-Dateien zu extrahieren. Die Installation erfolgt durch die Eingabe des folgenden Kommandos in die *shell* des Servers:

```
perl -MCPAN -e 'install Spreadsheet::ParseExcel'
```

Nun kann das Perl-Skript *parse-excel.pl* (vgl. Appendix D) genutzt werden, welches auf die Funktionen des *Spreadsheet::ParseExcel-Moduls* zurückgreift.

Beim Aufruf des Skripts wird als Argument der Name der, zu parsenden Excel-Datei, übergeben. In einer Schleife wird nun die Datei zeilenweise durchlaufen und die Daten in eine Datei geschrieben. Dabei werden die einzelnen Zeilen der XLS-Datei durch '>>>' und die Zellen durch '>' voneinander getrennt.

Anschließend kann die Datei mit PHP anhand dieser Trennzeichen wieder in einer entsprechend strukturierten Array-Variablen zur Verfügung gestellt werden.

## 7.8 Sicherheitsaspekte

Die Sicherheitsstrategie der Praktikumsanwendung ist im Rahmen der einzelnen Kapitel bereits ausführlich beschrieben worden. Die folgende Übersicht soll noch einmal verdeutlichen, welche Aspekte zur Erhöhung der Sicherheit berücksichtigt wurden.

| **Sicherheitsaspekt** | **Maßnahmen** |
|---|---|
| SSL-Verschlüsselung | abhörsichere Verbindung zur Verhinderung des Ausspähens von Paßwörtern etc. |
| privater Webserver-Bereich | Zugang in den Bereich der PHP-Struktur erfordert eine Autorisierung des Nutzers |
| Loginumgebung | Zugang zur Praktikumsanwendung erfordert ein gültiges Login - Nutzerdaten werden aus der Datenbank bezogen und mit den Login-Eingaben verglichen |
| Session | Gültigkeit der Session-ID wird bei jedem Navigationsschritt überprüft - Verhindern von Quereinstiegen |
| Datenbankstruktur | Nutzung der Datenbankmechanismen zur Sicherung der Datenintegrität |
| Datenbankrechte | Verwendung verschiedener Nutzer mit möglichst exakt auf die unterschiedlichen Aufgaben abgestimmten Rechten - Verhinderung der unberechtigten Manipulation von Datenbankinhalten |
| Nutzertablespace | abgegrenzter Bereich zum Anlegen von Nutzer-Objekten in der Datenbank |

**Tabelle 7.2:** Maßnahmen zur Erhöhung der Sicherheit der Praktikumsanwendung

# 8 Testphase

Um Aussagen über die Nutzbarkeit und die Performance der Anwendung treffen zu können, wurde das Datenbank-Praktikum des Matrikels *MM02* ab dem 07.06.2004 unter realen Bedingungen über das Netz abgewickelt und der Verlauf mit den verfügbaren Werkzeugen überwacht und ausgewertet. Dabei kamen das Unixprogramm *top* zur Überwachung der Prozessorlast des Servers und verschiedene Bestandteile des Oracle Diagnostic Packs zur Analyse der internen Abläufe des Oracle Datenbankmanagement-Systems zum Einsatz. Außerdem waren die Anmerkungen, Wünsche und Verbesserungsvorschläge der Studenten in dieser Phase von besonderem Interesse.

Beim ersten Testlauf zeichnete sich relativ schnell ab, daß die Methode zur Beurteilung der eingegebenen Lösungsvorschläge zu statisch war. Die Bewertung der Eingaben erfolgte zu diesem Zeitpunkt über einen Zeichenkettenvergleich mit der, in der Datenbank gespeicherten, Musterlösung. Dabei wurden bestimmte vorhersehbare Abweichungen, wie unterschiedliche Groß- und Kleinschreibung, mehrfache Leerzeichen oder das Setzen eines Semikolons zum Abschluß der Anweisung, vor dem Vergleich mit Hilfe von PHP-Stringfunktionen in eine Standardform überführt. Zur richtigen Lösung der Aufgaben mußte also die Eingabe der Nutzer syntaktisch mit der Musterlösung übereinstimmen. Die folgende Beispielaufgabe soll aufzeigen, aus welchem Grund diese Form der Eingabeprüfung für eine sinnvolle Auswertung ungeeignet ist.

**Beispiel:** Gesucht sind alle Kunden, die *Meier* heißen oder die Kundennummer *6* haben.

Die Aufgabenstellung setzt keine Reihenfolge voraus, mit der die beiden Bedingungen abgetestet werden müssen. Das folgende SQL-Statement wäre demnach als gültige Referenzlösung denkbar:

*select * from KUNDEN where NAME='Meier' or KUNDENNUMMER=6*

Bei einem strengen Zeichenkettenvergleich wird das folgende Statement allerdings als falsch abgewiesen, obwohl beide Anweisungen identische Ergebnisse haben:

*select * from KUNDEN where KUNDENNUMMER=6 or NAME='Meier'*

Eine derartig unflexible Bewertung könnte die Teilnehmer im ungünstigsten Fall sogar von richtigen Lösungen ablenken und damit den Lernerfolg ungünstig beeinflussen.

Bei komplexeren Aufgabenstellungen erhöht sich die Wahrscheinlichkeit einer solchen Fehlinterpretation enorm, was ein erfolgreiches Arbeiten mit der Praktikumsanwendung nahezu ausschließt.

Diese Erkenntnis führte zu der Entwicklung der aktuellen Methode zur Eingabenbewertung, welche in Abschnitt 7.4.2 ausführlich erläutert wird. Die Bewertung der Statements erfolgt bei diesem Ansatz anhand eines Ergebnismengen-Vergleichs von Musterlösung und Eingabe.

### Die Performance der Praktikumsanwendung

Aus technischer Sicht verliefen die ersten Tests vollkommen unproblematisch. Um das Verhalten der Anwendung unter maximaler Last zu beobachten, wurde die Einführungsveranstaltung mit einer Gruppe von 18 Studenten im Pool 1 der Hochschule Wismar durchgeführt. Da das Praktikum später vermutlich nicht mehr von einer so umfangreichen Nutzergruppe gleichzeitig genutzt werden wird, wurden während dieser Sitzung eine Reihe von Messungen durchgeführt, die Auskunft über die Kapazitäten des Systems geben sollten.

Eine komfortable Möglichkeit zur Überwachung bietet das Unix-Programm *top*. Mit Hilfe dieses Programms können die laufenden Prozesse des Systems dynamisch überwacht werden. Die Ausgabe von *top* ist in zwei Abschnitte geteilt. In der Kopfzeile finden sich die zuletzt vom System zugeteilte *PID*, eine Angabe zur Systemauslastung (*load average*), sowie die Systemlaufzeit seit dem letzten Reboot. Die weiteren Zeilen im ersten Teil der Ausgabe beschreiben die Auslastung des verfügbaren Speichers und Swapspaces.

Im Hauptteil der Ausgabe werden diverse Informationen über die derzeit laufenden Prozesse gegeben, die nach der prozentualen CPU-Belegung der Prozesse geordnet und standardmäßig alle zwei Sekunden aktualisiert werden. Aus der Vielfalt der Informationen sind die Werte *%CPU*, *%MEM* und *TIME+* von besonderem Interesse für die Auswertung. *%CPU* und *%MEM* geben die prozentuale Nutzung der CPU, beziehungsweise des Hauptspeichers seit der letzten Aktualisierung an. Mit *TIME+* wird die absolute CPU-Zeit angegeben, die ein Prozeß verbraucht hat, seit er gestartet wurde.

In Abbildung 8.1 ist die Ausgabe von *top* dargestellt, die während der Einführungsveranstaltung aufgezeichnet wurde. In dieser Auswahl finden sich die Prozesse, die im Zusammenhang mit der Nutzung der Praktikumsanwendung stehen. Das sind zum einen die Prozesse, die unter dem Nutzer *oracle* ausgeführt werden, also alle laufenden Oracle-Prozesse und der, für die Kommunikation verantwortliche, Listener *tnslsnr*. Dazu kommen die Child-Prozesse des Apache-Servers *httpd*, die beim Aufruf der Webseiten erzeugt werden. Die Werte, die beim Test der Anwen-

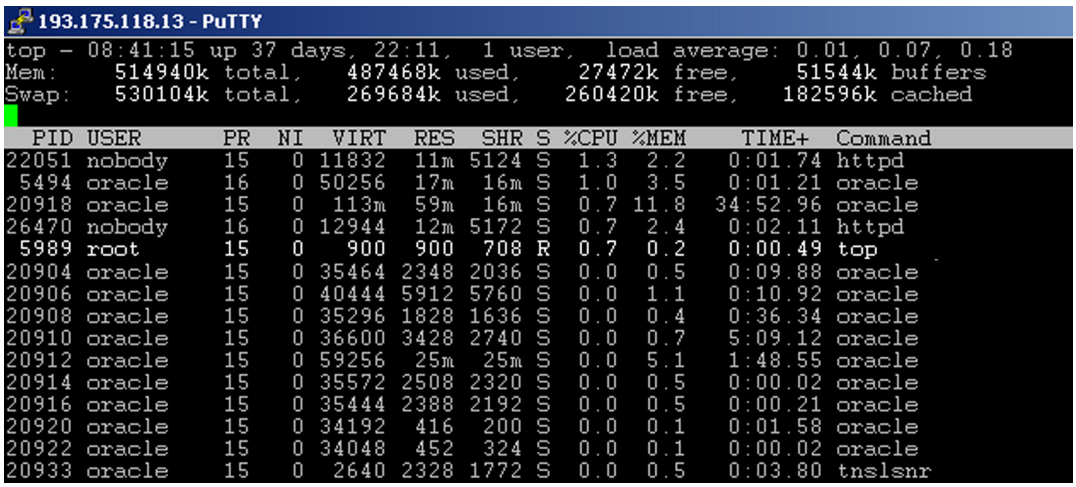



**Abbildung 8.1:** Prozeßüberwachung mit dem Unix-Programm *top*

dung auftraten, zeigten eine unbedeutend erhöhte Prozessorlast durch die Arbeit der Studenten auf dem System.

Zur Einschätzung der Datenbankbelastung wurde der *Performance-Überblick* aus dem *Oracle Diagnostics Pack* verwendet. Dieses Tool ermöglicht eine umfassende Analyse zahlreicher Zustandsgrößen des Oracle Datenbank-Managementsystems, welche in einem definierbaren Zeitintervall aktualisiert werden. Die Übersicht gibt Auskunft über die Datenbank-CPU-Aktivität, den Datenbankspeicher und die Datenbank-E/A-Aktivität. Die Verteilung der Ressourcen wird nach den einzelnen Sessions aufgegliedert und eine Übersicht zu den Warten-Ereignissen gemacht. Für jede dieser Kategorien sind über Untermenüs weitere Details abrufbar. Die Ausgabe dieses Analysetools während des Test ist in Abbildung 8.2 dargestellt. Die abgebildeten Werte stellen eine Momentaufnahme dar, bei der mit *10,4* Benutzerzugriffen pro Sekunde, eine verhältnismäßig hohe Aktivität registriert wurde. Mit diesem Wert werden die Login-, Parse- oder Executeaufrufe seit der letzten Aktualisierung angegeben.

Mit dem Wert *Executes ohne Parsevorgänge* unter der Rubrik *Anwendungseffizienz* wird der prozentuale Anteil von Statements angegeben, die keinen korrespondierenden Parsevorgang benötigen. Optimal sind hier Werte um 100%. Dabei muß bei der Auswertung natürlich die Charakteristik der Anwendung berücksichtigt werden. Die SQL-Statements der Studenten werden in der Regel nach dem Parsen nur einmalig ausgeführt, was sich natürlich negativ auf diesen statistischen Wert auswirkt. Die Beurteilung dieses Wertes muß daher im Kontext mit dem Anwendungskonzept vorgenommen werden. Der Wert von *73,68%* ist in diesem Zusammenhang unbedenklich.

In der Rubrik *Datenbankspeicher* befinden sich Angaben über die Effizienz der Spei-

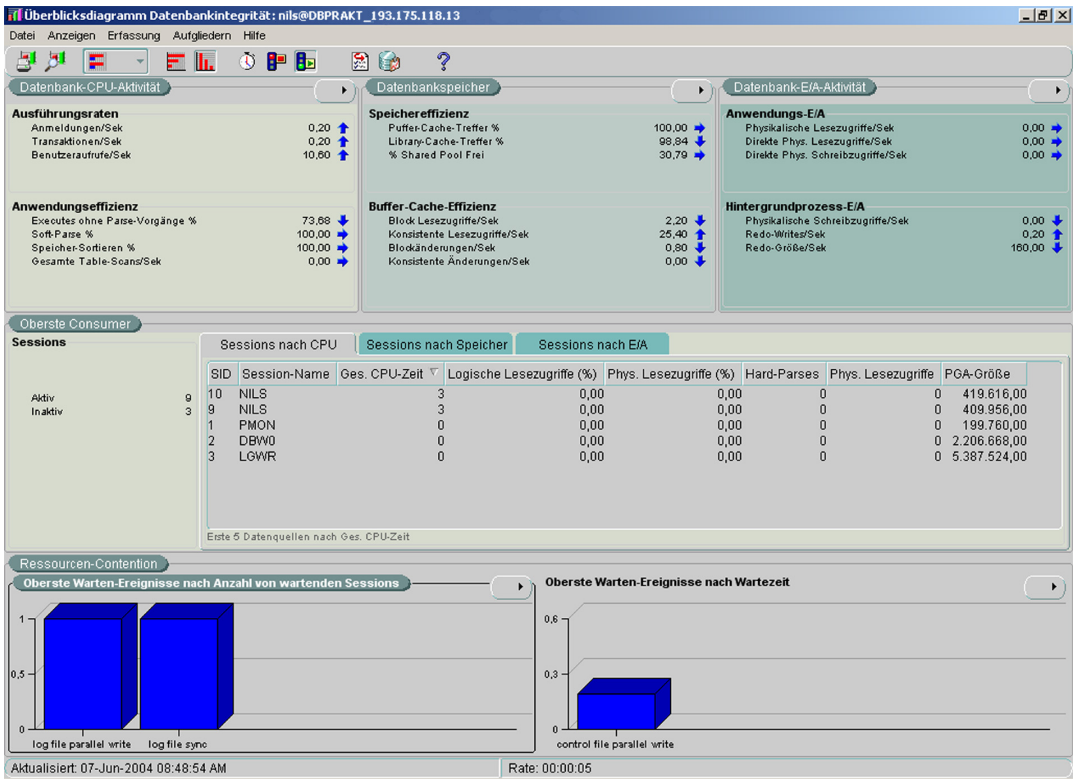



**Abbildung 8.2:** Der Oracle Performance-Überblick während des Testlaufs

chernutzung. Für eine optimale Konfiguration des *Oracle System Global Area (SGA)* ist der Wert für *% Shared Pool Frei* besonders interessant, da dieser Bereich alle anderen Segmente des *SGA* enthält. Übersteigt dieser Parameter den Wert 50%, ist das ein Indiz dafür, daß bei der Konfiguration des Datenbank-Managementsystems unnötig viel Speicher allokiert wurde, was sich ungünstig auf das Verhalten anderer Anwendungen auswirken könnte.

In der Sektion *Oberste Consumer* werden die kostenintensivsten Sessions nach der ingesamt verbrauchten CPU-Zeit angezeigt. Hier finden sich an erster Stelle zwei Sessions, die dem Nutzer *NILS* gehören. Hierbei handelt es sich um Vorgänge, die direkt auf die Aktivitäten der Studenten zurückzuführen sind, da dieser Datenbankbenutzer für meisten Datenbankvorgänge des Praktikums verwendet wird. Die Sessions *PMON*, *DBW0* und *LGWR* sind auf Hintergrundprozesse der Datenbank zurückzuführen, die gestartet werden, wenn eine Datenbankinstanz angelegt wird und für die interne Organisation des Systems benötigt werden.

Der Sessionname *PMON* steht für *Oracle Process Monitor*. Der *PMON*-Prozeß sorgt für die Freigabe von Ressourcen für den Fall, daß ein Nutzerprozeß fehlschlägt.

*DBW0 (Oracle DataBase Writer)* ist ein Oracle Hintergrundprozeß, der bei Bedarf Daten vom *SGA-Bereich* in die Oracle Datenbank-Dateien auf der Festplatte

schreibt.

Der *LGWR (Oracle Log Writer)* wird genutzt, um die Redo-Puffer im Bedarfsfall auf die Festplatte zu schreiben.

Ein umfangreiches Glossar über diverse Begriffe, die in diesem Zusammenhang auftauchen, bietet Oracle auf einer Webseite des Oracle-FAQ[10].

Da die während der Performance-Tests aufgezeichneten Werte keine Hinweise auf eventuelle Leistungsengpässe des Systems gaben, wurde von einer detaillierteren Analyse des Systems abgesehen. Die Praktikumsanwendung wird seit dem 07.06.2004 störungsfrei im normalen Hochschulbetrieb genutzt. Trotzdem ist es notwendig, die Analyse der Anwendung auch weiterhin zu betreiben, da die Werte infolge der langfristigen Nutzung variieren können.

# A KAPV.sql

```
-- ******************************************
-- * Standard SQL generation                *
-- *----------------------------------------*
-- * Generator date:  Apr 29 2004           *
-- * Generation date: Mon May 17 17:30:38 2004 *
-- ******************************************


-- Table Section
-- -------------

create table Adressen (
     Adress_ID numeric(10) not null,
     Postleitzahl numeric(5) not null,
     Ort varchar(1) not null,ß
     Strae varchar(1) not null,
     Hausnummer char(5) not null,
     Staat varchar(1) not null,
     primary key (Adress_ID))
     tablespace KAPV;

create table Auftrag (
     Auftragsstatus varchar(12) not null,
     Auftragsdatum date not null,
     Auftragsnummer char(10) not null,
     Kundennummer char(8) not null,
     primary key (Auftragsnummer))
     tablespace KAPV;

create table Auftrag_hat_Positionen (
     Produktnummer char(8) not null,
     Stueckzahl numeric(10) not null,
     Auftragsnummer char(10) not null)
     tablespace KAPV;

create table Kunden (
     Kundennummer char(8) not null,
     Name varchar(30) not null,
     Vorname varchar(30) not null,
     Geburtsdatum date not null,
     Geschlecht char(1) not null,
     primary key (Kundennummer))
     tablespace KAPV;

create table Kunden_Profil (
     KundenProfil_ID numeric(10) not null,
     Kundennummer char(8) not null,
     Auftragsvolumen numeric(10,2) not null,
     Zahlungsbilanz char(1) not null,
     Zahlungsverhalten numeric(1) not null,
     primary key (KundenProfil_ID),
     unique (Kundennummer))
     tablespace KAPV;

create table Kunde_hat_Adresse (
     Adress_ID numeric(10) not null,
     Kundennummer char(8) not null)
     tablespace KAPV;

create table Kunde_hat_Vorlieben (
```

```
     KundenProfil_ID numeric(10) not null,
     Vorlieben_ID numeric(5) not null,
     Anzahl numeric(6) not null)
     tablespace KAPV;

create table Produktlager (
     Preis numeric(10,2) not null,
     Stueckzahl numeric(10) not null,
     Produktnummer char(8) not null,
     Bezeichnung varchar(1) not null,
     Produktionsdatum date not null,
     Material varchar(1) not null,
     Groesse varchar(1) not null,
     primary key (Produktnummer))
     tablespace KAPV;

create table Rechnungsdaten (
     Anzahl_von_Mahnungen char(1) not null,
     Rechnungsbemerkung varchar(1) not null,
     Rechnungsnummer char(10) not null,
     Rechnungsdatum date not null,
     Eingegangene_Zahlungen numeric(10,2) not null,
     Zahlungsdatum date not null,
     Auftragsnummer char(10) not null,
     primary key (Rechnungsnummer))
     tablespace KAPV;

create table Vorlieben (
     Vorlieben_Kategorie varchar(30) not null,
     Vorlieben_ID numeric(5) not null,
     primary key (Vorlieben_ID))
     tablespace KAPV;

create table Zu_Vorlieben_gehoeren_Produkte (
     Produktnummer char(8) not null,
     Vorlieben_ID numeric(5) not null)
     tablespace KAPV;




alter table Auftrag add constraint FKKunde_hat_Auftrag
     foreign key (Kundennummer)
     references Kunden;

alter table Auftrag_hat_Positionen add constraint FKAuf_Pro
     foreign key (Produktnummer)
     references Produktlager;

alter table Auftrag_hat_Positionen add constraint FKAuf_Auf
     foreign key (Auftragsnummer)
     references Auftrag;

alter table Kunden_Profil add constraint FKKunde_hat_Profil
     foreign key (Kundennummer)
     references Kunden;

alter table Kunde_hat_Adresse add constraint FKKun_Kun_1
     foreign key (Kundennummer)
     references Kunden;

alter table Kunde_hat_Adresse add constraint FKKun_Adr
     foreign key (Adress_ID)
     references Adressen;

alter table Kunde_hat_Vorlieben add constraint FKKun_Vor
     foreign key (Vorlieben_ID)
     references Vorlieben;

alter table Kunde_hat_Vorlieben add constraint FKKun_Kun
     foreign key (KundenProfil_ID)
     references Kunden_Profil;

alter table Rechnungsdaten add constraint FKKundenkonto_hat_Auftrag
```

```
     foreign key (Auftragsnummer)
     references Auftrag;

alter table Zu_Vorlieben_gehoeren_Produkte add constraint FKZu_Vor
     foreign key (Vorlieben_ID)
     references Vorlieben;

alter table Zu_Vorlieben_gehoeren_Produkte add constraint FKZu_Pro
     foreign key (Produktnummer)
     references Produktlager;
```

# B envvars

```
# envvars-std - default environment variables for apachectl
#
#This file is generated from envvars-std.in
#

ORACLE_SID="dbprakt" export ORACLE_SID

ORACLE_HOME="/opt/oracle/9i" export ORACLE_HOME

ORA_NLS33="$ORACLE_HOME/ocommon/nls/admin/data" export ORA_NLS33

NLS_LANG="GERMAN_GERMANY.WE8ISO8859P1" export NLS_LANG

LD_LIBRARY_PATH="/usr/local/apache/lib:$ORACLE_HOME/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH
```

# C connect.php

```
<?php

$dbuser="nils";
$dbpass="dbprakt";
/* Datenbanklogin */


PutEnv("TWO_TASK=dbprakt");                    //SID der zu verbindenden Datenbankinstanz


function dbconnect($sql)
{

    global $dbuser, $dbpass;

    $con = OCILogon($dbuser, $dbpass);         //Verbindungsaufbau

    $stmt = OCIParse($con, $sql);              //Vorbereitung zum üAusfhren des Statements

    return $stmt;

    OCILogoff($con);                           //Trennen der Verbindung
}
?>
```

# D parse-excel.pl

```
#!/usr/bin/perl -w

use strict;
use Spreadsheet::ParseExcel;

my $oExcel = new Spreadsheet::ParseExcel;

die "Bitte eine EXCEL-DATEI als $0 angeben!" unless @ARGV;

my $oBook = $oExcel->Parse($ARGV[0]);
my($iR, $iC, $oWkS, $oWkC);


for(my $iSheet=0; $iSheet < $oBook->{SheetCount} ; $iSheet++)
{
 $oWkS = $oBook->{Worksheet}[$iSheet];
 #print "--------- SHEET:", $oWkS->{Name}, "\n";
 for(my $iR = $oWkS->{MinRow} ;
      defined $oWkS->{MaxRow} && $iR <= $oWkS->{MaxRow} ;
      $iR++)
 {
    print ">>>";
  for(my $iC = $oWkS->{MinCol} ;
       defined $oWkS->{MaxCol} && $iC <= $oWkS->{MaxCol} ;
       $iC++)
  {
   $oWkC = $oWkS->{Cells}[$iR][$iC];
   print ">", $oWkC->Value if($oWkC);
  }
 }
}
```

# E Literaturverzeichnis


[1] Oracle PL/SQL - Grundlagen

Steven Feuerstein - Deutsche Übersetzung von Matthias Kalle

O'Reilly Verlag 1. Auflage April 1999

ISBN 3-89721-180-7

[2] Oracle Networking

Hugo Toledo

Oracle Press 1996

ISBN 0-07-882165-7

[3] Datenbanken und SQL

Erwin Schicker

B.G. Teubner Stuttgard - Leipzig - Wiesbaden 3. Auflage 2000

ISBN 3-519-22991-9

[4] Linux Server-Tipps

George M. Doss - Deutsche Übersetzung von Wolfgang Jährling

Software & Support Verlag Franfurt 2000

ISBN 3-935042-03-5

[5] Oracle9i UNIX Administrationshandbuch

Donald K. Burleson

Carl Hanser Verlag München 2003

ISBN 3-446-22201-4

# F Webseitenverzeichnis


[1] Die Startseite der Praktikumsanwendung
https://athos.et.hs-wismar.de/dipl/praktikum.php

[2] Inhalte der Lehrveranstaltung Datenbanksysteme
http://www.et.hs-wismar.de/ duest/lv.html

[3] Dokumentationen - Oracle9i
http://otn.oracle.com/documentation/oracle9i.html

[4] Systemanpassungen vor der Installation
http://www.puschitz.com/TuningLinuxForOracle.shtml/SettingSharedMemory

[5] Downloadseite von PHP
http://de3.php.net/get/php-4.3.4.tar.bz2/from/a/mirror

[6] Downloadseite von Apache
http://www.apache.de/dist/httpd/

[7] Oracle Error Search
http://www.oracle.com/pls/db92/db92.homepage

[8] PHP OCI8-Funktionen
http://de3.php.net/manual/de/ref.oci8.php

[9] CPAN-Modul-Support (Spreadsheet::ParseExcel)
http://search.cpan.org/dist/Spreadsheet-ParseExcel/ParseExcel.pm#Functions

[10] Oracle Begriffserläuterungen
http://www.orafaq.com/glossary/index.htm

# G Selbständigkeitserklärung

Hiermit erkläre ich, daß ich die hier vorliegende Arbeit selbständig, ohne unerlaubte fremde Hilfe und nur unter Verwendung der in der Arbeit aufgeführten Hilfsmittel angefertigt habe.

Wismar, 26. September 2004

Nils Weber