Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

Bachelorarbeit

Analyse und Dokumentation des Datenbanksystems des Forschungsportal Sachsen-Anhalt

Autor:

Jan Wedding

13. Februar, 2013

Betreuer:

Prof. Dr. rer. nat. habil. Gunter Saake
M.Sc. Sebastian Breß

Institut für Technische und Betriebliche Informationssysteme
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Wedding, Jan:
Analyse und Dokumentation des Datenbanksystems des Forschungsportal Sachsen-Anhalt
Bachelorarbeit, Otto-von-Guericke-Universität Magdeburg, 2013.

Abstract

Die Ladezeit einer Website ist in der heutigen Zeit sehr wichtig. Nielsen [Nie00] stellt in seinem Buch fest, dass die Performance einer Website und damit auch die Ladezeiten zum Erfolg/Misserfolg eine Website beitragen können. Die Performance einer Website kann nicht nur durch eine langsame Anbindung, sondern unter anderem auch durch eine dahinter liegende Datenbank beeinflusst werden. In dieser Arbeit wird das Datenbanksystem des Forschungsportals Sachsen-Anhalt (ein Webportal) auf Probleme hin analysiert. Zu diesem Zweck werden zunächst Analysen und Messungen des Datenbanksystems durchgeführt, um daraus Vorschläge zur Verbesserung der Performance des Datenbanksystems und der Konsistenz der Daten zu entwickeln. Ein Teil der gemachten Vorschläge wird bereits in dieser Arbeit umgesetzt.

Danksagungen

Ich möchte mich zuerst bei meiner Familie bedanke, die mich in der Zeit des Schreibens dieser Arbeit voll unterstützt hat. Des Weiteren möchte ich mich bei Frau Dr. Sylvia Springer bedanken, die mir das Praktikum und diese Arbeit am Forschungsportal Sachsen-Anhalt ermöglicht hat. Auch dem restlichen Team des TTZ gilt mein Dank für die Unterstützung. Schließlich möchte ich mich auch noch bei meinem Betreuer Sebastian Breß bedanken, dessen Hinweise und Vorschläge ein wertvoller Beitrag für die Verbesserung dieser Arbeit darstellen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

DBMS	Datenbankmanagementsystem
DOI	Digital Object Identifier
FK	Foreign Key
FPSA	Forschungsportal Sachsen-Anhalt
FTS	Full table scan
ID	Identifikationsnummer

1. Einführung

In diesem Kapitel werden Informationen zum Hintergrund und zur Motivation für diese Arbeit angegeben. Außerdem werden konkrete Ziele für die Arbeit benannt. Das Kapitel schließt mit der Erläuterung des Aufbaus dieser Arbeit.

1.1 Hintergrund

Datenbanksysteme finden bei dynamischen Internet-Applikationen eine große Anwendung. Insbesondere bei historisch gewachsenen Portalen kann es jedoch mit der Zeit zu Problemen mit dem Datenbanksystem kommen: Über die Zeit hinweg kann es passieren, dass das Datenbanksystem durch fehlerhafte Wartung oder fehlendes Datenbank-Tuning verlangsamt wird und die Konsistenz der Daten abnimmt. Aus diesem Grund wird sich in dieser Arbeit mit dem Tuning von Datenbanken zur Beschleunigung von Webanwendungen beschäftigt. Dazu wird das Forschungsportal Sachsen-Anhalt (FPSA) untersucht.

1.2 Motivation

Niedrige Ladezeiten einer Website sind in der heutigen Zeit sehr wichtig. Eine gute Performance wird von vielen Nutzern erwartet. Insbesondere bei Informationsportalen, wie dem Forschungsportal Sachsen-Anhalt sollte beispielsweise die Suche vergleichsweise schnell ausgeführt werden. Bereits in [Nie00] wurde festgestellt, dass die Performance einer Website maßgeblich zum Erfolg/Misserfolg dieser beitragen kann. Dort wird geschrieben, Nutzer hätten den Autor bei jeder Web-Usability-Studie gebeten, die Performance der Website zu verbessern. Unter Performance versteht sich hierbei die Ladezeit einer Website, d.h. wie lange es dauert, bis die Website vollständig aufgebaut ist. Es zählt aber auch, wie lange es dauert, bis die ersten Informationen auf einer Website zu sehen sind. Die Ladezeit wird von verschiedenen Faktoren beeinflusst, wie beispielsweise vom Laden von einzelnen Elementen auf der Website, z.B. Bilder, CSS-Stylesheets und JavaScript-Dateien. Aber auch die Zeit zur Ermittlung von Daten aus einer Datenbank zur Anzeige auf der Website beeinflusst die Ladezeit. Durch Optimierung des Datenbanksystems bzw. von einzelnen

Abfragen sind Performance-Gewinne möglich und somit kann die Geschwindigkeit für das Seitenladen erhöht werden. Die Performance einer Website ist zum einen für den Nutzer, zum anderen aber auch für die Suchmaschinenoptimierung wichtig. Dass für Nutzer kurze Ladezeiten wichtig sind, zeigt auch eine Studie von Google [Bru09]: Müssen Nutzer mehr als 400ms auf ein Suchresultat warten, so ist dies für viele bereits zu lang. Die Anzahl der Suchanfragen durch betroffene Nutzer sank in dem Mess-Zeitraum von 6 Wochen um rund 0,6%. Benötigt eine Website im Vergleich zu einem Wettbewerber mehr als 200ms länger zum Laden, dann führt dies auch im Allgemeinen zu weniger Besuchen. Je länger eine Anfrage dauerte, desto weniger Suchanfragen wurden durch die Nutzer ausgeführt. Diese Erkenntnisse sind sicherlich auch auf andere Websites neben Google, also auch auf das Forschungsportal Sachsen-Anhalt übertragbar, da auch dieses unter anderem eine Suchfunktion bietet und beide im weitesten Sinne zum Durchsuchen und Anzeigen von großen Datenbeständen dienen. Aber nicht nur für die Nutzer, sondern auch für die Suchmaschinenoptimierung sind kurze Ladezeiten ein wichtiger Faktor: Ausgehend von den Untersuchungen in der referenzierten Google-Studie bewertet Google Websites nicht nur nach Relevanz, sondern auch nach Ladezeiten [GWC12]. Dementsprechend kann auch die Performance der Datenbank einen Einfluss auf das Ranking innerhalb des Suchindexes haben, wenn diese die Ladezeiten signifikant erhöht. Somit kann auch unter diesem Gesichtspunkt die Performance des Datenbanksystems zum Erfolg einer Website beitragen. Eine Webanwendung, bei der das zugrunde liegende Datenbankmanagement-System die Performance-Spezifikationen nicht erfüllt, kann also signifikant verlangsamt werden. Dies ist beispielsweise beim FPSA der Fall, bei welchem es durch Probleme mit der Wartung des Datenbankmanagementsystems und der Anwendungsprogrammierung zu Performanceverlusten und Inkonsistenzen gekommen ist. Aus diesem Grund beschäftigt sich diese Arbeit mit dem Datenbankmanagementsystem des FPSA.

1.3 Zielstellung

Das Ziel dieser Arbeit ist es, die Datenbank des FPSA hinsichtlich Performance und Performanceverbesserungsmöglichkeiten zu analysieren und somit dazu beizutragen, die Geschwindigkeit des Datenbanksystems und Konsistenz der Daten des Forschungsportals im Allgemeinen zu verbessern.

Die Arbeit hat folgende Ziele:

1. Analyse des Datenbankschemas, um Quellen von Redundanzen und Inkonsistenzen zu identifizieren
2. Analyse der Datenbankanfragen
3. Analyse der Web-Anwendung
4. Erarbeitung von Verbesserungsvorschlägen zur Optimierung der in Punkt 1 bis 3 gefundenen Probleme

Zusammengefasst bedeutet dies, dass Vorschläge gemacht werden sollen, wie die Performance verbessert, die Datenhaltung vereinheitlicht, Inkonsistenzen bei der

aktuellen und zukünftigen Entwicklung reduziert werden können und wie generell das Arbeiten auf den Datenbeständen vereinfacht werden kann. Insbesondere ist der Aspekt der zukünftigen Entwicklung wichtig, um langfristig den Wartungsaufwand und damit auch die Wartungskosten zu senken.

1.4 Gliederung der Arbeit

In Kapitel 2 wird zunächst eine kurze Einführung in grundlegende Konzepte gegeben, die für das Verständnis der Arbeit wichtig sind. Anschließend findet in Kapitel 3 die Analyse des Datenbanksystems statt, wobei in diesem Kapitel zunächst die Vorgehensweise und dann die eigentlich Analyse inklusive der Verbesserungsvorschläge beschrieben wird. Die Arbeit schließt mit einer Bewertung der Ergebnisse, der Zusammenfassung und dem Ausblick auf zukünftige Arbeiten in Kapitel 4.

2. Grundlagen

In diesem Kapitel werden für das Verständnis der Arbeit wichtige Grundlagen, Begriffe und verwendete Tools näher erläutert. Außerdem wird kurz auf das Untersuchungsobjekt, das Forschungsportal Sachsen-Anhalt, eingegangen.

2.1 Datenbankmanagementsystem

In diesem Abschnitt wird der Aufbau und die allgemeine Funktionsweise eines Datenbankmanagementsystems (DBMSs) erklärt. Außerdem wird auf einige für die Arbeit relevante und mit Datenbankmanagementsystemen in Bezug stehende Begriffe eingegangen. Alle Informationen aus diesem Abschnitt wurden aus [SSH11] entnommen, sofern sie nicht anders gekennzeichnet wurden.

Ein DBMS ist eine Software, die zur Verwaltung von Datenbanken dient. Abbildung 2.1 verdeutlicht die Architektur eines DBMSs. Es folgt eine Beschreibung der einzelnen Komponenten.

Zu den *Benutzerkomponenten* gehören zum einen die **Anfragen und Änderungen**. Diese bieten einen interaktiven Zugriff auf die Datenbestände über Datenanwendungsprogramme: Daten können abgerufen und geändert werden. Die einzelnen Nutzer werden in der Abbildung durch $\mathbf{P_1}$-$\mathbf{P_n}$ dargestellt.

Zu den sogenannten *Programmierkomponenten* gehören die Komponenten zur Definition von **Masken**, die im Prinzip Möglichkeiten zur Definition von User-Interfaces für die Anwendungsprogramme darstellen. Die **Einbettung** bildet die Schnittstelle der Datenbank zur Anwendungsprogrammierung, sodass mittels höherer Programmiersprachen auf die Datenbestände der Datenbank zugegriffen werden kann. Die **Datenbank-Operationen** realisieren die nötigen Operationen für Anfragen und Änderungen, die von Anwendungen genutzt werden.

Die *Definitionskomponenten* bieten den Datenbank- bzw. Anwendungsadministratoren die Möglichkeit, ihrer administrativen Arbeit nachzukommen. Zu diesem Zweck gibt es die **Sichtdefinitionen**, welche die Benutzersichten definieren. Diese stellen also eine Deklaration der Datendarstellung auf der externen Ebene dar, also wie die

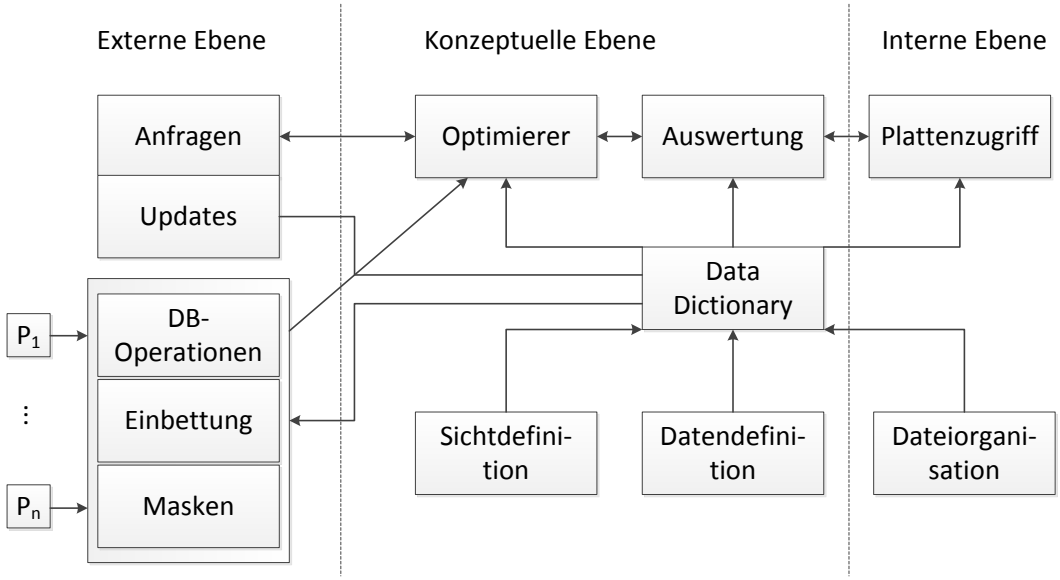

Abbildung 2.1: vereinfachte Architektur eines DBMSs (in Anlehnung an [SSH11])

Daten nach außen hin, beispielsweise für Anwendungsprogramme, sichtbar sind. Des Weiteren gibt es die **Datendefinition**, welche das konzeptuellen Schema festlegt. Das konzeptuelle Schema ist eine implementierungsunabhängige Modellierung der gesamten Datenbank in einem systemunabhängigen Datenmodell. Die **Dateiorganisation** beschreibt zum einen die Zugriffspfade auf die interne Ebene, zum anderen die Art und Weise, wie die Dateien der Datenbank organisiert sind.

Zu den sogenannten *Transformationskomponenten* zählt der **Optimierer**. Der Optimierer verbessert die Datenbankzugriffe, um somit möglichst ressourcenschonend und schnell auf die Daten in der Datenbank anhand der durch die Anwendung gestellten Anfragen zugreifen zu können. Auf den Optimierer bzw. die Phasen des Optimierens wird in Abschnitt 2.1.5 noch näher eingegangen. Der **Plattenzugriff** realisiert die Plattenzugriffssteuerung, um die Daten der Datenbank von der Festplatte zu lesen und geänderte Daten zurückzuschreiben. Außerdem gibt es eine Komponente zur **Auswertung** der Ergebnisse von Anfragen und Änderungen.

Das **Data Dictionary** (oft auch Katalog genannt) ist zentraler Speicherort aller für die Datenhaltung relevanten Informationen (Meta-Daten) und versorgt die anderen Komponenten des DBMSs mit den nötigen Daten.

2.1.1 Indexe

Indexe dienen dazu, effizient und schnell auf Datenbestände zugreifen zu können. Sie dienen also als alternative Zugriffspfade auf Daten. Dadurch können unter anderem auch Verbundoperationen beschleunigt werden. Indexe sind auch für die Anfrageoptimierung wichtig, da diese unter Umständen Einfluss darauf haben, welcher Anfrageplan in der kostenbasierten Auswahl gewählt wird.

Indexstrukturtypen

Indexstrukturen lassen sich in die Typen **dünnbesetzt/dichtbesetzt** und **geclustert/nicht geclustert** einordnen [SSH11]:

- Ein **dünnbesetzter Index** ist ein Index, bei dem nicht für jeden Zugriffsattributwert ein Eintrag in der Indexdatei gespeichert wird. Dies setzt voraus, dass die interne Relation nach den Zugriffsattributwerten sortiert ist.
- Bei einem **dichtbesetzten Index** wird für jeden Datensatz der internen Relation ein Eintrag in der Indexdatei hinterlegt.
- Ein **geclusterter Index** ist in der gleich Form sortiert, wie die interne Relation. Ist die interne Relation nach einem bestimmten Attributwert sortiert, dann ist auch der geclusterte Index nach diesem Attributwert sortiert.
- Ein **nicht geclusterter Index** ist anders organisiert als die interne Relation. Gibt es beispielsweise einen Sekundärindex, der nach einem Attribut A sortiert ist, aber die Datei selbst ist nach einem anderen Attribut B sortiert, dann handelt es sich um einen geclusterten Index.

konkrete Indexstrukturen

Beispiele für häufig eingesetzte Indexstrukturen sind **B-Bäume** bzw. **B+-Bäume**, **Hash-Indexe** sowie **Bitmap-Indexe**. Daneben gibt es noch eine Vielzahl anderer Indexstrukturen, die aber für diese Arbeit nicht relevant sind, da das zu untersuchende System diese nicht unterstützt. **B-Bäume** lassen sich auf den Artikel [BM72] zurückführen und sind im Prinzip dynamische, balancierte Indexbäume, bei denen jeder Indexeintrag auf eine Seite in der Hauptdatei der Datenbank verweist. **B+-Bäume** speichern im Unterschied zu B-Bäumen ihre Daten ausschließlich in den Blattknoten [SSH11]. **Hash-Indexe** basieren auf sogenannten Hash-Tables, welche auf verschiedene Autoren zurückzuführen sind [MS05]. In DBMSen sind den Hash-Werten in der Hash-Tabelle eines solchen Hash-Indexes die Speicherplätze der Datensätze zugeordnet. Es werden also Attributwerte mittels einer Hash-Funktion auf Speicherbereiche abgebildet. **Bitmap-Indexe** wurden zuerst in [SM85] diskutiert und werden durch eine zweidimensionale Matrix von Boole'schen Werten realisiert. Die eine Dimension sind die Identifier für die jeweiligen Datensätze und die andere Dimension bilden die Ausprägungen des zu indexierenden Attributs. Hat ein Datensatz eine Attributsausprägung a, dann wird für diese Ausprägung der Wert in der Matrix auf 1 gesetzt und für alle anderen Ausprägungen auf 0 [SSH11]. Weitere Details zu den Indexstrukturen sind in [SSH11] zu finden. Wichtig für diese Arbeit ist insbesondere die Information, für welche Anwendungszwecke bzw. unter welchen Bedingungen die Indexstrukturen eingesetzt werden sollten. Ein **B+-Baum-Index** unterstützt viele verschiedene Arten von Anfragen, beispielsweise Punktanfragen oder Anfragen, die Verbunde enthalten und wird deswegen als beste allgemeine Datenstruktur angesehen. Diese Indexe sollten insbesondere dann eingesetzt werden, wenn auf die Daten viele Bereichsanfragen ausgeführt werden, oder wenn Anfragen oft nach einem Minimal- oder Maximalwert in den Daten suchen [SB03]. Ein **Hash-Index** eignet sich besonders gut für Punktanfragen, da diese mit nur einem Festplattenzugriff beantwortet werden können. Außerdem eignen sie sich gut für Punktanfragen, wenn der Index Teil eines geclusterten Indexes ist [SB03]. Ein **Bitmap-Index** eignet sich speziell für Data-Warehouse-Anwendungen. Insbesondere Attribute mit geringer Kardinalität (wenigen Ausprägungen) werden durch Bitmap-Indexe unterstützt. Bei Attributen mit hoher Kardinalität steigt der Speicheraufwand für diese Indexstruktur und ist in diesem Fall weniger empfehlenswert [SSH11].

2.1.2 Fremdschlüssel

Ein Fremdschlüssel oder auch Foreign Key (FK) stellt einen Verweis eines Attributs in einer Tabelle (der referenzierenden Tabelle) auf einen Schlüssel in einer anderen Tabelle (der referenzierten Tabelle) dar. Dieses Konzept wird in Abbildung 2.2 verdeutlicht. In dieser Abbildung sind zwei Tabellen zu sehen. Die *FK_ID*-Werte der referenzierenden Tabelle verweisen auf ID-Werte in der referenzierten Tabelle.

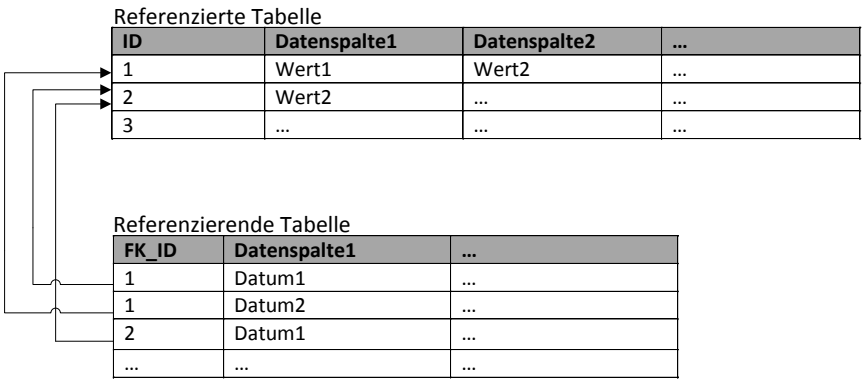

Abbildung 2.2: Fremdschlüsselbeziehung

Fremdschlüssel dienen in DBMSen zur Wahrung der referenziellen Integrität. Referenzielle Integrität beschreibt in DBMSen die Beziehung zwischen Objekten (dargestellt durch Tabellen). Die Integritätsbedingung besagt, dass in der referenzierten Tabelle der Attributwert des Fremdschlüssels der referenzierenden Tabelle vorhanden sein muss. Gibt es keinen solchen Datensatz in der referenzierten Tabelle, dann ist die referenzielle Integrität verletzt und die Daten in der Datenbank sind unvollständig und somit inkonsistent [SSH10]. Um diese referenzielle Integrität zu wahren, gibt es die Möglichkeit, dem Fremdschlüssel ein bestimmtes **ON DELETE** bzw. **ON UPDATE**-Ereignis zuzuweisen. Dies bestimmt, was passiert, wenn der Datensatz in der referenzierten Tabelle gelöscht bzw. der Schlüssel dort aktualisiert wird [SSH10]:

Ist die **CASCADE**-Option aktiv, dann wird der Datensatz in der referenzierten Datenbanktabelle gelöscht, dann wird auch der Datensatz in der referenzierenden Tabelle gelöscht.

Wenn der Fremdschlüssel mit **SET NULL/DEFAULT** definiert wurde, dann wird der Datensatz in der referenzierten Datenbanktabelle gelöscht und der Wert für den Fremdschlüssel in der Fremdschlüsseltabelle auf **NULL** bzw. den **DEFAULT**-Wert gesetzt.

Wurde **NO ACTION** definiert, dann wird nichts ausgeführt, sollte in der Haupttabelle noch ein referenzierter Datensatz bestehen. Dies bedeutet, dass die **DELETE/UPDATE**-Aktion nicht durchgeführt wird.

2.1.3 Prepared Statements

Bei **Prepared Statements** handelt es sich um vorkompilierte SQL-Anfragen. In diesen Anfragen können sogenannte **bind-Variablen** eingesetzt sein. Dies sind im Prinzip Platzhalter für Daten. Um ein solches Prepared Statement auszuführen, muss dieses zunächst kompiliert werden. Dann müssen an alle bind-Variablen Daten gebunden werden und erst dann kann die Anfrage ausgeführt werden. Ein Prepared Statement eignet sich insbesondere für den Schutz vor SQL-Injections, bei denen von Außenstehenden versucht wird, den Anfragetext so zu manipulieren, sodass Anfragen ausgeführt werden, die vom Anwendungsprogramm nicht ausgeführt werden sollten und möglicherweise schädliche Auswirkungen haben [TW07].

2.1.4 Stored Procedures und Stored Functions

Bei **Stored Procedures** und **Stored Functions** handelt es sich um vom DBMS verwaltete Programme, die die Möglichkeit besitzen Ablaufkonstrukte, wie zum Beispiel Sequenzen, bedingte Ausführungen oder Schleifen zu nutzen. Da **Stored Procedures** und **Stored Functions** vom DBMS verwaltet werden, müssen diese nur ein einziges mal kompiliert werden, um sie ausführen zu können. **Stored Procedures** bieten die Möglichkeit mehrere Ein- und Ausgabeparameter zu definieren, **Stored Functions** hingegen können nur einen Eingabeparameter haben und einen Wert zurückliefern [SSH10].

2.1.5 Optimierung von Anfragen

Bei der Optimierung von Anfragen geht es vorrangig darum, dass die Anfragebearbeitung möglichst schnell und ressourcenschonend durchgeführt wird. In diesem Abschnitt werden kurz die Phasen der Optimierung in einem DBMS vorgestellt.

Die Phasen der Optimierung sind [SSH11]:

Logische Optimierung Der Anfrageplan wird unabhängig von der Form der Speicherung der Relation umgeformt, beispielsweise indem Selektionen möglichst früh während der Anfragebearbeitung stattfinden, um kleine Zwischenergebnisse zu erhalten. Auch das Entfernen redundanter Operationen gehört in diese Phase.

Physische Optimierung Hier werden ausgehend von den Informationen über vorhandene Indexstrukturen im Katalog ein oder mehrere Zugriffspläne erzeugt.

Kostenbasierte Auswahl Anhand von im Katalog vorhandenen Statistiken findet eine Auswahl des voraussichtlich kostengünstigsten Anfrageplanes statt.

Diese Phasen können nicht unabhängig voneinander arbeiten, weswegen die Phasen üblicherweise als Optimierung zusammengefasst werden.

2.1.6 Statistiken

Unter Statistiken versteht sich im allgemeinen eine Sammlung von Daten über die (statistische) Verteilung von Attributen. Diese findet in DBMSen häufig über sogenannte Histogramme statt. Anhand der Statistiken kann ein Optimierer beispielsweise entscheiden, welcher erzeugte Plan kostengünstiger ist, also ob es beispielsweise günstiger ist, einen Zugriff auf die Daten über einen Index oder einen Full table scan durchzuführen. [SSH11]

2.1.7 Full table scan

Findet ein sogenannter Full table scan (FTS) statt, dann wird beim Suchen von Datensätzen die gesamte Relation durchlaufen. Da diese Art von Scan häufig dazu führt, dass viele, langsame Festplattenzugriffe durchgeführt werden müssen, ist diese Art bei Punktanfragen bzw. Anfragen, die weniger als eine bestimmte Menge an Daten zurückliefern, zu vermeiden, da sonst zu viel Aufwand betrieben werden muss. Burleson gibt in seinem Buch für das Oracle DBMS Werte von 40% für sortierte Tabellen und 7% für unsortierte Tabellen an, ab denen das DBMS statt einer Index-Suche einen FTS durchführt [Bur10]. Diese Werte können je nach Konfiguration des Systems jedoch abweichen. Um einen FTS zu vermeiden, um eine Anfrage zu beschleunigen, bietet es sich an, einen passenden Index anzulegen. Wird dieser dann vom Datenbank-Optimierer in den Anfrage-Plan aufgenommen, reduzieren sich die Ausführungszeit und die I/O-Kosten der Anfrage teils drastisch, da statt die gesamte Relation durchlaufen zu müssen, über den Index nach dem Datensatz gesucht wird.

2.2 Datenbank-Tuning

Unter Datenbank-Tuning verstehen sich alle Aktionen, um eine Datenbank-Anwendung zu beschleunigen. Dies bedeutet, dass beispielsweise der Datendurchsatz erhöht oder die Antwortzeit des Systems verringert wird [SB03].

2.2.1 Tuning-Prinzipien

Für das Datenbank-Tuning gibt es fünf allgemeine Prinzipien [SB03]:

1. **Think globally, fix locally.** Hier ist gemeint, dass die Auswirkungen des Tunens global bedacht werden sollten. Als Beispiel wird genannt, dass oftmals beim Tunen hauptsächlich die Auslastung der Hardware berücksichtigt wird und bei Bedarf die Hardware-Ressourcen erhöht werden. Dies ist jedoch nicht immer sinnvoll, denn es gibt viele Fälle, in denen die Ursache der hohen Hardware-Auslastung beispielsweise FTSs für Punktanfragen sind. Wird dann für die Relation ein Index angelegt, dann ist das Problem zum einen schneller und zum anderen aber auch kostengünstiger behoben worden. Ein anderes Beispiel für das Prinzip ist, dass Tuner oftmals versuchen, die Zeit für eine bestimmte Anfrage zu reduzieren. Dieser Aufwand lohnt sich allerdings kaum, wenn die Anfrage nur sehr selten ausgeführt wird. Dementsprechend sollten vorrangig die Anfragen verbessert werden, die in irgendeiner Form kritisch für das System sind, beispielsweise Anfragen, die oft ausgeführt werden.

2. **Partitioning breaks bottlenecks.** Oftmals gibt es in Datenbanksystemen eine bestimmte Komponente, die das Gesamtsystem in der Performance limitiert. Durch Aufteilen des Aufwandes über eine bestimmte Zeit oder über vergrößerte Ressourcen kann der Flaschenhals verringert werden. Allerdings sollte zunächst versucht werden, die einzelnen Komponenten zu beschleunigen. Erst, wenn dies nicht ausreichen sollte, sollte versucht werden, zu partitionieren.

3. **Start-up costs are high; running costs are low.** Hiermit ist beispielsweise das Lesen von Daten von einer Festplatte gemeint. Während das Lesen des ersten angeforderten Bytes sehr langsam ausgeführt wird, spielt es danach kaum noch eine Rolle, weitere Daten zu lesen. Daher wird vorgeschlagen, dass häufig verwendete Tabellen nacheinander auf der Festplatte abgelegt werden. Außerdem sollte vertikale Partitionierung verwendet werden, wenn Tabellen mehrere hundert Spalten haben, von denen nur wenige Spalten häufig in Anfragen verwendet werden.

4. **Render unto server what is due unto server.** Mit diesem Prinzip ist gemeint, dass die Aufgaben zwischen Datenbanksystem und Anwendung (dem Client) gut verteilt sind. Beispielsweise können rechenintensive Aufgaben vom Server auf den Client ausgelagert und somit Server-Rechenleistung eingespart werden.

5. **Be prepared to trade-offs.** Oftmals ist es nötig Zielkonflikte zu berücksichtigen: Beispielsweise ist es möglich, die Puffergröße des Datenbank-Systems zu vergrößern, indem dem System mehr RAM hinzugefügt wird. Allerdings ist RAM nicht kostenfrei, sodass sich ein Zielkonflikt zwischen den zusätzlichen Kosten und dem Nutzen, der sich daraus ergibt, entwickelt. Auch das Anlegen eines Indexes kann beispielsweise für einzelne Anfragen von Vorteil sein, aber kann für das Gesamtsystem Nachteile bedeuten: Ein Index muss bei jedem **INSERT**, **UPDATE** oder **DELETE** aktualisiert werden, was zusätzlich aufzuwendende Prozessorleistung, zusätzliche I/O-Operationen und für den Index selbst zusätzliche Speicherkapazität bedeutet.

2.2.2 Tuning-Methoden

Beim Tuning von Datenbanken gibt es verschiedene Möglichkeiten, um das Datenbanksystem zu beschleunigen. In vielen Tuning-Büchern (beispielsweise [SB03] oder [Mit03]) wird in der einen oder anderen Form auf die folgenden Möglichkeiten näher eingegangen. Zu den Möglichkeiten zählt das Anfrage-Tuning, das Index-Tuning, das Tunen der Parameter des DBMSs und das Hardware-Tuning.

Beim **Anfrage-Tuning** wird versucht, das vorhandene Datenbankschema bestmöglich auszunutzen, indem Anfragen so formuliert werden, dass beispielsweise vorhandene Indexe ausgenutzt werden.

Das **Index-Tuning** beschäftgt sich damit, durch Anlegen passender Indexe den Zugriff auf die Daten für bestimmte Anfragen zu beschleunigen bzw. damit, unnötige Indexe zu identifizieren und somit löschen zu können, um den Aufwand zur Wartung des Indexes einsparen zu können.

Außerdem kann ein Datenbank-Tuner verschiedene Parameter des Datenbanksystems anpassen (**Tuning der Parameter des Datenbanksystems**), beispielsweise die Größe des Puffers, um somit häufige, sehr langsame Festplattenzugriffe zu vermeiden (indem der Puffer vergrößert wird).

Zudem gibt es auch die Möglichkeit, das Datenbanksystem durch Änderungen an Hardware zu beschleunigen (**Hardware-Tuning**), indem schnellere oder größere Komponenten eingesetzt werden. Oftmals ist diese Form des Tunings unangebracht: Beispielsweise können häufige Festplattenzugriffe durch Anfragen, die FTSs statt einer Index-Suche durchführen, ausgelöst werden. In solchen Fällen ist es meist günstiger, einen Index anzulegen, statt neue Hardware zu kaufen [SB03].

2.3 Oracle

Oracle ist eine Softwarefirma die unter anderem DBMSe vertreibt, darunter auch das Oracle DBMS, welches im FPSA eingesetzt wird.

In diesem Abschnitt wird auf die von Oracle zum Tuning des Oracle DBMS bereitgestellten Tools eingegangen.

2.3.1 Oracle Tools zur Unterstützung beim Datenbank-Tuning

Oracle bietet zu seinem DBMS verschiedene Tools zur Unterstützung beim Datenbank-Tuning an. Diese unterstützen auch teilweise implizit die in Abschnitt 2.2 vorgestellten Methoden und Prinzipien. Die zwei für diese Arbeit verwendeten Programme werden hier kurz vorgestellt. Informationen für diesen Abschnitt wurden, sofern nicht anders gekennzeichnet, aus [Cor10] entnommen.

Enterprise Manager

Der Oracle Enterprise Manager bietet eine Vielzahl verschiedener Tools, um das Analysieren und Tunen von Oracle Datenbanken zu unterstützten. Die Grundlage für die Darstellungen im Oracle Enterprise Manager sind folgende Archive [Cor08]:

- Das **Automatic Workload Repository (AWR)** sammelt, verarbeitet und wartet Performance-Statistiken zur Problem-Erkennung und zum Self-Tuning.
- Der **Automatic Database Diagnostic Monitor (ADDM)** analysiert die von dem AWR gesammelten Informationen, um mögliche Performance-Probleme in der Oracle Datenbank zu erkennen.
- Der **SQL Tuning Advisor** erlaubt es, die Performance zu steigern, ohne die Anfragen selbst zu modifizieren
- Der **SQL Access Advisor** bietet Hinweise zu Materialized Views, Indexen, und Materialized Views Logs.
- Das **End to End Application tracing** findet und listet hohe Auslastungen des Servers nach Nutzer, Service oder Anwendungskomponenten auf.

- Außerdem gibt es Server-generierte, automatische Benachrichtigungen über bevorstehende Probleme mit der Datenbank.

- Zusätzlich gibt es noch weitere Advisors, z.B. den **Memory Advisor** zur Analyse der Speicherauslastung. Andere Advisors werden dazu verwendet, um beispielsweise die Mean Time to Recovery, also die mittlere Zeit zur Behebung eines Fehlers, zu optimieren. Diese zusätzlichen Advisors können bei Bedarf aus dem Oracle Enterprise Manager heraus aufgerufen werden.

- **V$ Performance Views** bieten allen Oracle Performance Tuning Tools die nötigen Informationen. Diese Views werden automatisch von Oracle verwaltet.

Diese Daten können zum einen textuell ausgewertet werden, indem beispielsweise vordefinierte SQL-Scripts ausgeführt werden, zum anderen können diese aber auch mit der Weboberfläche des Oracle Enterprise Managers grafisch dargestellt werden. Auf die zweite Möglichkeit wird hier insbesondere eingegangen, da sie teilweise die Grundlage der Arbeit darstellt.

Top-Aktivität

So ist es mit dem Oracle Enterprise Manager unter anderem möglich, eine Liste von Anfragen mit Top-Aktivität innerhalb einer definierten Zeitspanne einzusehen. Es ist somit direkt einsehbar, welche Anfrage wieviele Ressourcen wann verbraucht hat. Dies wird in Abbildung 2.3 verdeutlicht: Im oberen Bereich des Screenshots ist eine Grafik zu sehen, die die Auslastung in der letzten Stunde anzeigt. Über diese kann eine 5-Minuten-Zeitspanne ausgewählt werden. Für die ausgewählte Zeitspanne wird dann im unteren Teil eine Rangfolge der Anfragen angezeigt, die den größten Teil der Auslastung in der Zeitspanne verursacht haben. Durch diese Form der Anzeige wird unter anderen das Anfrage-Tuning und auch das Prinzip **Think globally; Fix locally** unterstützt. Auch das Prinzip von **Start-up costs are high; running costs are low** kann hiermit untersucht werden, da für jede Anfrage die Aufteilung der Ressourcen angezeigt werden kann, sodass beispielsweise Anfragen mit häufigen Festplattenzugriffen identifiziert werden können.

SQL Tuning Advisor

Auch das Analysieren einzelner/mehrerer Anfragen ist möglich. Mit dem Oracle SQL Tuning Advisor, welcher aus dem Enterprise Manager heraus gestartet werden kann, werden dann Vorschläge für die Optimierung von Anfragen gemacht. Unter anderem werden somit Probleme mit dem Datenbankschema identifiziert, wenn beispielsweise ein FTS durchgeführt wird, weil ein Index fehlt. Auch fehlende Optimizer-Statistiken werden ermittelt. Das Umstrukturieren des SQL-Statements kann ebenfalls zu den Vorschlägen gehören. Die vorgeschlagenen Änderungen können begutachtet und dann mit einem Klick eingepflegt werden, sofern es sich dabei nicht um die Umstrukturierung der Anfrage (Anfrage-Rewriting) handelt.

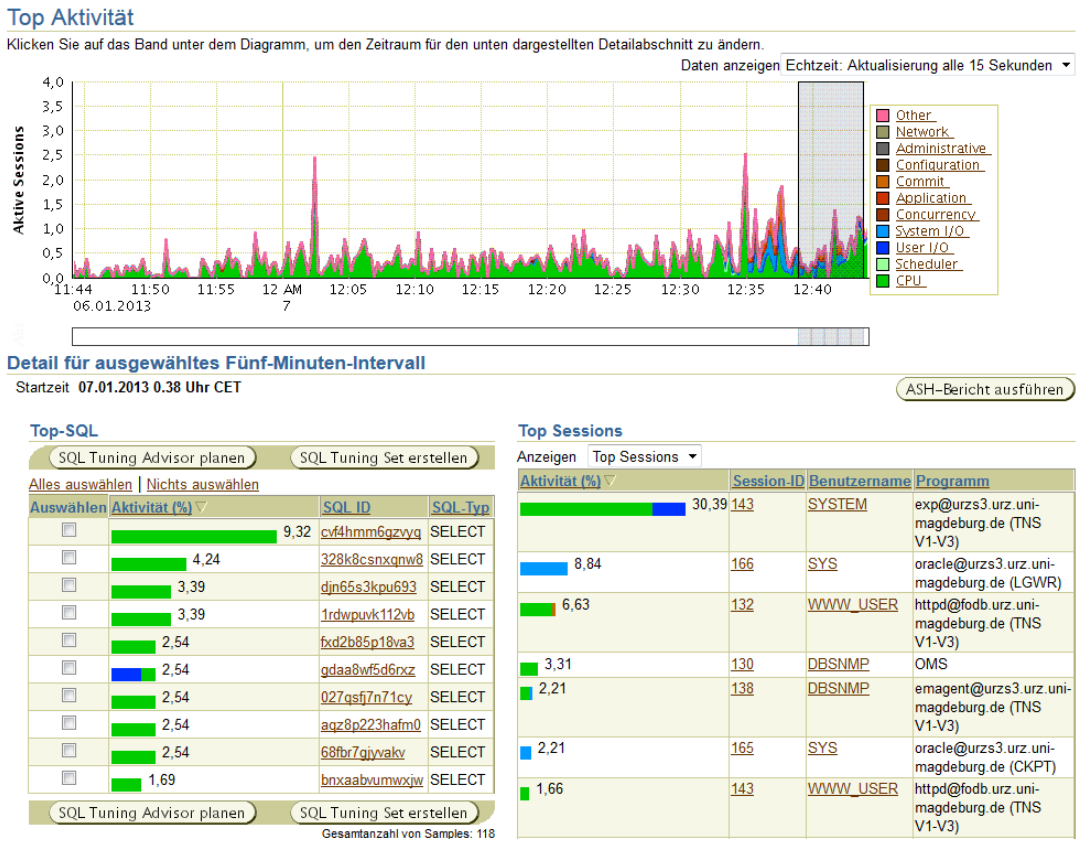

Abbildung 2.3: Anzeige der Top-Aktivität im Oracle Enterprise Manager

Anzeige von Hardware-Auslastungen

Aus dem Oracle Enterprise Manager heraus ist es möglich, sich die aktuelle und vergangene Hardware-Auslastung detailliert anzeigen zu lassen. In Abbildung 2.4 ist beispielsweise eine Übersicht für die aktuelle CPU-Auslastung, die Speicherauslastung und die Menge der Festplattenzugriffe zu sehen. Für jede der drei Kategorien gibt es weitere, detailliertere Informationen zur Begutachtung und Auswertung der Auslastung des DBMSs. Aus den Werten kann dann abgeleitet werden, inwiefern

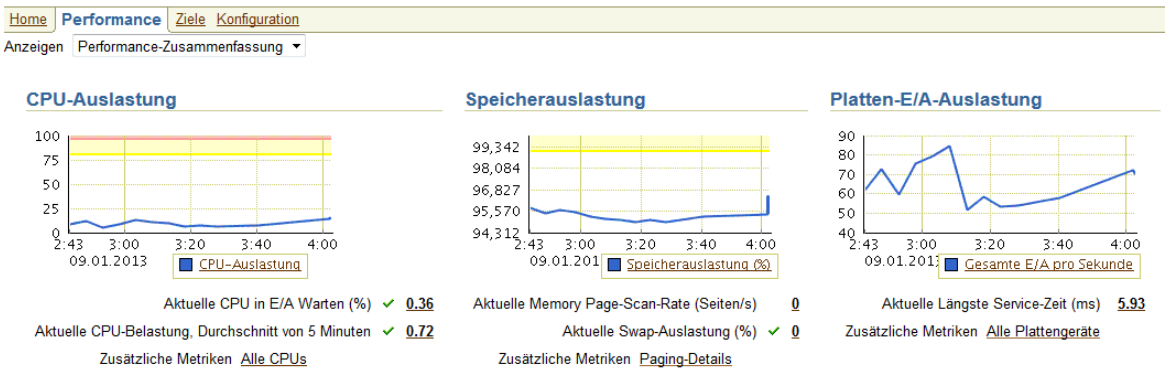

Abbildung 2.4: Anzeige der Hardware-Auslastung im Oracle Enterprise Manager

Handlungsbedarf in der einen oder anderen Form nötig ist, beispielsweise das Vergrö-

ßern der Hardware-Ressourcen oder das Analysieren der Anwendung zur generellen Reduzierung der Auslastung.

Automatic Database Diagnostic Monitor

Dieses je nach Einstellung in regelmäßigen Abständen laufende Programm liefert Auswertungen zu allen Aktivitäten innerhalb der eingestellten Zeitpanne. Somit können beispielsweise Anfragen erkannt, die häufig (im Vergleich zu allen anderen Anfagen) ausgeführt werden und bei denen es somit lohnenswert ist, diese Anfragen zu optimieren. Auch sogenannte **Doppelte SQL** werden erkannt. Dies sind Anfragen, die aufgrund von Literalen im Anfragetext bei jeder Ausführung neu geparst werden müssen und somit unnötige Rechenzeit für das Erstellen des Anfrageplanes verbrauchen. Zu diesen Anfragen gibt es eine zusätzliche Rangfolge, anhand der Schritt für Schritt die Anzahl solcher Anfragen reduziert werden kann.

2.4 Das Forschungsportal Sachsen-Anhalt

Das Forschungsportal Sachsen-Anhalt FPSA (http://forschung-sachsen-anhalt.de) ist eine historisch gewachsene Webplattform, welche seit mehr als 10 Jahren besteht. Das auf PHP mit unterliegender Oracle-Datenbank basierende System hatte bereits im Jahr 2009 täglich mehr als 200.000 Zugriffe aus 100 verschiedenen Ländern [Spr09]. Das FPSA dient verschiedenen Einsatzgebieten, hauptsächlich:

- Eintragung von veröffentlichten Publikationen
- Veröffentlichung von Projekten
- Eintragung von Veranstaltungen
- Generierung von Forschungsberichten

Ein Großteil der Datenbestände kann außerdem durchsucht werden. Eine Übersicht über den Aufbau des FPSA ist in Abbildung 2.5 zu sehen. In der Grafik wird gezeigt, dass das FPSA eine Vielzahl an Datenquellen besitzt, welche die Oracle Datenbank über die PHP-Anwendung mit Daten speist. Diese Daten werden dann dazu genutzt, sie zum einen auf der Seite direkt anzuzeigen, zum anderen aber auch für externe Portale nutzbar zu machen. Beispielsweise wird die XML-Schnittstelle des FPSA dazu genutzt, die Profilseiten von Mitarbeitern der Otto-von-Guericke Universität mit Informationen zu füllen.

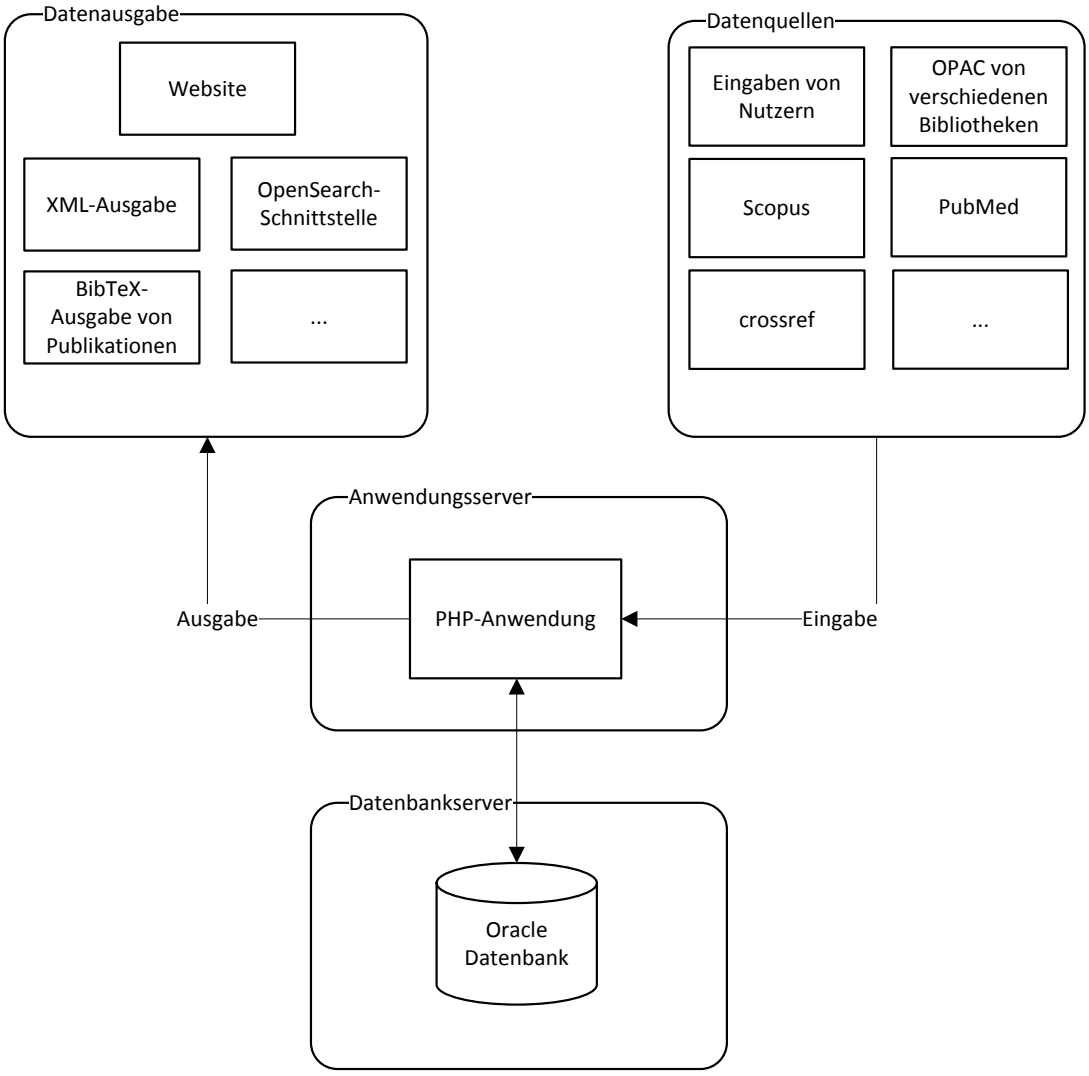

Abbildung 2.5: Aufbau des Forschungsportal Sachsen-Anhalt

Das FPSA hat allerdings eine Reihe von Problemen:

Daten-Konsistenzprobleme Durch eine teilweise fehlerhaft programmierte Anwendung und durch fehlende Fremschlüsselbeziehungen passiert es, dass Daten inkonsistent gespeichert werden.

Geschwindigkeitsprobleme Durch langsame Datenbank-Anfragen und fehlende Indexe verlangsamt sich die Website. Dies führt beispielsweise dazu, dass bei bestimmten Suchanfragen die Anwendung in einen Timeout läuft und statt der Suchresultate eine weiße Seite angezeigt wird.

Zeichenkodierungsprobleme Das FPSA wurde im Zeichensatz ISO-8859-1 programmiert und auch die Datenbank-Einstellungen basieren darauf. Allerdings gibt es eine Reihe von Zeichen, die nicht mittels ISO-8859-1 dargestellt werden können, sodass solche Zeichen nicht nativ in der Datenbank gespeichert werden können, was unter anderem für das Speichern von Publikationen mit ausländischem Titel ein Problem darstellt.

Dokumentation Sowohl der Anwendungcode als auch das Datenbankschema des FPSA ist nur sehr rudimentär dokumentiert, was es schwierig macht, Neu-Entwicklungen zu implementieren oder Fehler schnell zu beheben.

Die ersten beiden Probleme werden in dieser Arbeit analysiert.

3. Analyse des Datenbanksystems

In diesem Kapitel werden die Analysen des Datenbanksystems des Forschungsportals Sachsen-Anhalt und deren Ergebnisse vorgestellt. Um das Datenbanksystem des Forschungsportals zu analysieren, wurde mehrschichtig anhand der typischen Architektur eines Datenbanksystems vorgegangen, wie es auch den Tuning-Prinzipien von Shasha et al. [SB03] (siehe Abschnitt 2.2.1) entspricht:

1. Zum einen wurde direkt das Data Dictionary des Oracle DBMS zur Analyse verwendet (Abschnitt 3.1), um somit Probleme mit dem Datenbankschema identifizieren zu können.

2. Zum anderen wurde sich mit Hilfe des Oracle Enterprise Managers eine Liste von Anfragen mit Top-Aktivität, d.h. Anfragen, die einen Großteil der Auslastung des DBMS ausmachen, angesehen. Es wurden Verbesserungsvorschläge für die Optimierung dieser einzelnen Anfragen erarbeitet (Abschnitt 3.2).

3. Als dritte Quelle von Informationen über Probleme diente der Anwendungscode selbst. Dieser wurde nach Auffälligkeiten, wie zum Beispiel die Generierung von Identifikationsnummern (IDs) im Quellcode, durchsucht (siehe Abschnitt 3.3).

In Abschnitt 3.4 wird abschließend ein Überblick über die Ergebnisse der Analyse gegeben.

3.1 Analyse mittels Data Dictionary

Zunächst wurde das Datenbankschema über das Data Dictionary analysiert. Zu diesem Zweck wurden verschiedene SQL-Anfragen formuliert und mittels SQL-Developer von Oracle ausgeführt. Somit wurden unter anderem leere/fast leere Tabellen und „Test“-Tabellen gefunden. Zudem wurden mit ID benannte Spalten ohne Index, Fremdschlüsselspalten ohne Fremdschlüssel-Constraint und alte Tabellen ermittelt. Auch zusammengesetzte Indexe und Lösch-Trigger wurden gefunden. Des Weiteren

wurden Spalten, mit nur einem einzigen Wert in allen Feldern sowie Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel-Constraint gefunden. Alle Resultatstabellen befinden sich in Abschnitt A.1 im Anhang auf Seite 47. Die Dokumentation zu den verwendeten Data Dictionary Tabellen und Views kann in [Cor09] gefunden werden.

3.1.1 Überblick

Im Schema der Datenbank des Forschungsportals befinden sich derzeit 412 Tabellen, wovon 235 Tabellen auf das Forschungsportal entfallen. Tabellen, die direkt zum Forschungsportal gehören, werden durch ein Präfix *FODB_* gekennzeichnet, alle anderen Tabellen sind entweder veraltet oder gehören zu anderen Programmen.

3.1.2 Leere/fast leere Tabellen

Um leere Tabellen zu finden, wurde das Data Dictionary von Oracle abgefragt. Dazu wurden der View *dba_tables* des Data Dictionarys vom Oracle DBMS genutzt, in der unter anderem auch die Anzahl der Zeilen in der Spalte num_rows verzeichnet ist.

```
1 SELECT dba_tables.table_name, dba_tables.num_rows
2 FROM dba_tables
3 WHERE tablespace_name = 'FORSCHUNG'
4 AND dba_tables.num_rows <= 20
5 ORDER BY dba_tables.num_rows ASC;
```

Listing 3.1: SQL-Quellcode zum Finden leerer bzw. fast leerer Tabellen

Die Ergebnisse befinden sich in Tabelle A.1. Insgesamt sind somit 55 Tabellen vollständig leer und werden die fast leeren Tabellen ebenfalls hinzu addiert, dann sind es 146 Tabellen, die potenziell nicht benötigt werden.

Ein Teil der Ergebnis-Tabellen muss jedoch leer sein, da es sich um Tabellen zu Zwischenspeicherung handelt. Ein Großteil der vollständig leeren Tabellen ist jedoch in der Tat ungenutzt, dies ist auch im Anwendungscode prüfbar, indem eine Quelltextsuche über den Namen der Tabelle durchgeführt wird. Bei den nichtleeren Tabellen ist zu prüfen, ob diese Tabellen tatsächlich wenige Einträge haben müssen oder ob auch diese Tabellen überflüssig sind. Alle ausgewählten Tabellen können dann gelöscht werden. Damit wird zum Einen für das DBMS der Aufwand für die Verwaltung der Tabellen reduziert und zum anderen steigt die Übersichtlichkeit für zukünftige Entwicklungen.

3.1.3 Test-Tabellen

Bei den „Test“-Tabellen handelt es sich um Tabellen, die im Tabellennamen einen „Test“-Substring enthalten, also extra gekennzeichnet wurden, dass diese Tabellen nicht produktiv verwendet werden und somit möglicherweise vergessen wurden, nach dem Test zu löschen. Die Anfrage 3.2 erzeugt eine Liste mit diesen Tabellen und verwendet dazu die Tabelle *table_name*.

```
1 SELECT table_name, num_rows FROM dba_tables
2 WHERE tablespace_name = 'FORSCHUNG'
3 AND table_name LIKE '%TEST%';
```

Listing 3.2: SQL-Quellcode zum Finden von Test-Tabellen

Das Resultat befindet sich in Tabelle A.6. Zur Behebung des Problems muss im Quelltext und in den Stored-Procedures zunächst noch einmal nach den erhaltenen Tabellen gesucht werden. Werden diese dort nicht verwendet, können die Tabellen ohne weiteres gelöscht oder zumindest temporär umbenannt werden, um sie dann nach Ablauf einer gewissen Frist zu löschen. Wird sich dafür entschieden, die Tabellen nach einer Frist zu löschen, dann sollte dies auch tatsächlich gemacht werden. Dies wurde zuvor jedoch nicht konsequent durchgeführt, sodass alte Tabellen auch nach längerer Zeit weiterhin im Datenbankschema zu finden sind.

3.1.4 ID-Spalten ohne Index

Bei diesen Spalten handelt es sich um Spalten, die einen „ID“-Substring im Namen haben, jedoch keinen Index. Da Identifikationsnummer (ID)-Spalten in der Regel für das Joinen von Tabellen verwendet werden, bzw. ein Zugriff auf die Zeile über eine ID erfolgt, ist unter Umständen ein Anlegen eines Indexes (bei FK-IDs) oder generell das Anlegen eines Primärschlüssel-Constraints (wenn die Tabelle so etwas noch nicht besitzt) sinnvoll. Anfrage 3.3 identifiziert diese Spalten. Es wurde für die Anfrage die Tabellen bzw. Views *dba_tables*, *all_tab_columns* und *dba_ind_columns* des Data Dictionarys verwendet. In *all_tab_columns* befinden sich Einträge zu allen Spalten aller Tabellen der Datenbank. In *dba_ind_columns* werden wiederum alle Spalten aufgeführt, die einen Index definiert haben. Werden diese Tabellen mittels **LEFT JOIN** verbunden, so ergibt dies im Falle, dass eine Spalte keinen Index hat, einen **NULL**-Wert. Durch das Eingrenzen auf Spalten mit ID im Namen wird das gewünschte Resultat geliefert.

```
1  SELECT t.table_name, a.column_name, t.num_rows
2  FROM dba_tables t
3  LEFT JOIN all_tab_columns a
4   ON t.table_name = a.table_name
5  LEFT JOIN dba_ind_columns dic
6   ON a.column_name = dic.column_name
7  WHERE tablespace_name = 'FORSCHUNG'
8   AND a.column_name LIKE '%ID%'
9   AND dic.column_name IS NULL
10 ORDER BY t.num_rows DESC;
```

Listing 3.3: SQL-Quellcode zum Ermitteln von ID-Spalten ohne Index

Zur Behebung des Problems sollte die Anwendung nach Verwendungen der Tabellen (vergleiche Tabelle A.7) und deren Spalten durchsucht werden. Wird dort über den jeweiligen Index verbunden oder wird dort über die ID ein Zugriff auf die Daten vorgenommen und ist die Tabelle groß genug, so sollte für diese Spalte ein Index bzw. einen Primärschlüssel angelegt werden.

Viele Systemhandbücher empfehlen über Tabellen mit weniger als 200 Einträgen keinen Index zu legen [SB03]. Dabei wird allerdings noch über die Größe der Tabelleneinträge differenziert: Nimmt jeder Eintrag eine Seite ein, dann würde für die Suche nach einem Datensatz die Anzahl der Festplattenzugriffe der Anzahl der Einträge entsprechen. Wäre über die Relation ein Index, könnte dies wiederum reduziert werden auf wesentlich weniger Festplattenzugriffe (bei 200 Zeilen, die jeweils eine gesamte Seite belegen, würden statt 200 Diskzugriffe nur 2 bis 3 erfolgen).

Das Anlegen eines Primärschlüssels erfordert jedoch eine Einzigartigkeit der Spaltenwerte, was im Forschungsportal nicht immer gegeben ist. Deshalb muss in diesem Fall eine Bereinigung der Datenbank-Inhalte vorgenommen werden.

3.1.5 Fremdschlüsselspalten ohne Fremdschlüssel-Constraint

Um Fremdschlüsselspalten zu identifizieren, die kein entsprechendes Constraint gesetzt haben, wurde im Data Dictionary nach Spalten gesucht, die einen „FK"-Substring im Spaltennamen haben, aber nicht das entsprechende Constraint gesetzt haben. Zu diesem Zweck wurde die Anfrage 3.4 ausgeführt. Dazu wurden die Tabellen / Views *dba_tables*, *all_tab_columns*, *all_cons_columns* und *all_constraints* verwendet. In der Data Dictionary-Tabelle *all_cons_columns* befindet sich unter anderem eine Auflistung von Tabellennamen, Spaltenname und Constraint-Namen und in *all_constraints* wiederum die Auflistung der Constraints selbst mit u.a. dem Constrainttyp. Werden diese über **LEFT JOIN**s verbunden und dann nach FK im Namen gefiltert und werden Spalten ausgeschlossen, die bereits einen R-Constraint (also ein Constraint für die **r**eferentielle Intergrität) besitzen, so wird das gewünschte Ergebnis geliefert.

```
1  SELECT DISTINCT dba_tables.table_name, dba_tables.num_rows,
2   all_tab_columns.column_name
3  FROM dba_tables t
4  LEFT JOIN all_tab_columns a
5   ON t.table_name = a.table_name
6  LEFT JOIN all_cons_columns c
7   ON (a.column_name=c.column_name
8    AND a.table_name=c.table_name)
9  LEFT JOIN all_constraints ac
10  ON cs.constraint_name = ac.constraint_name
11 WHERE tablespace_name = 'FORSCHUNG'
12  AND (constraint_type IS NULL
13  OR constraint_type <> 'R')
14  AND all_tab_columns.column_name LIKE '%FK%'
15 ORDER BY table_name;
```

Listing 3.4: SQL-Quellcode zur Ermittlung von Fremdschlüssel-Spalten ohne Fremdschlüssel-Constraint

Für alle diese Spalten (siehe Tabelle A.9) sollte nun überprüft werden, ob ein Fremdschlüssel-Constraint sinnvoll ist, denn unter Umständen ist die Bezeichnung FK fälschlicherweise verwendet worden. Für manche der Spalten muss vermutlich außerdem eine manuelle Bereinigung der Daten vorgenommen werden, da das Constraint ansonsten nicht angelegt werden kann, da die Fremdschlüssel-Bedingung bereits verletzt ist.

3.1.6 Alte Tabellen

Bei den sogenannten alten Tabellen handelt es sich um Tabellen, die einen Substring „OLD" im Namen tragen. Das soll bedeuten, dass die Tabellen nicht mehr aktiv aus der Anwendung heraus aufgerufen werden sollen. Dies muss jedoch geprüft werden. Die Anfrage 3.5, welche die Data Dictionary-View *dba_tables* verwendet, erzeugt eine Liste mit diesen Tabellen, wobei anzumerken ist, dass es auch alte Tabellen geben

kann, die kein „OLD“ im Namen tragen. Beispielsweise sind alle Tabellen zunächst einmal als alt anzusehen, die kein fodb-Präfix enthalten und keine Systemtabellen sind oder zu einer anderen Anwendung gehören. Eine andere Anwendungen, die in der Datenbank des Forschungsportals eigene Tabellen besitzt, ist z.B. der Oracle Enterprise Manager.

```
1 SELECT table_name, num_rows FROM dba_tables
2 WHERE tablespace_name = 'FORSCHUNG'
3  AND table_name LIKE '%OLD%';
```

Listing 3.5: SQL-Quellcode zum Ermitteln alter Tabellen

Die Resultate befinden sich in Tabelle A.12. Zur Behebung des Problems sollte zunächst mit einem Verantwortlichen gesprochen werden, ob die darin enthaltenen Daten noch benötigt werden. Alternativ ist zu prüfen, ob die Daten an anderer Stelle, also in einer anderen Tabelle redundant vorhanden sind. Außerdem sollte im Anwendungscode geprüft werden, dass die Tabellen dort nicht mehr referenziert werden. Falls dies alles der Fall sein sollte, können die Tabellen gelöscht werden. Werden diese Tabellen noch weiterhin im Quellcode verwendet, dann sollte versucht werden, die Anwendung so umzustellen, dass diese Tabellen nicht mehr benötigt werden und dann auch tatsächlich gelöscht werden können.

3.1.7 Zusammengesetzte Indexe

Zusammengesetzte Indexe werden mit Anfrage 3.6 identifiziert. Hierfür wurden die Data Dictionary-Views *dba_tables* und *dba_ind_columns* verwendet. Grund für die Suche war das Entdecken des Indexes *SEARCH_FODB_PUB_LM_TEST* in der Tabelle *FODB_PUBLIKATION*, welcher nur testweise angelegt wurde ohne ihn später wieder zu entfernen oder zu ändern, sodass der Index beim Ausführen der Anfragen der Anwendung auch tatsächlich verwendet wird.

```
1  SELECT DISTINCT idx1.table_name, idx1.index_name, num_cols, num_rows
2  FROM dba_tables t
3  LEFT JOIN dba_ind_columns idx1
4   ON t.table_name = idx1.table_name
5  LEFT JOIN
6    (SELECT dic.index_name, count(*) as num_cols
7    FROM dba_tables dt
8    LEFT JOIN dba_ind_columns dic
9    ON dt.table_name = dic.table_name
10   AND index_name IS NOT NULL
11   GROUP BY dic.index_name
12   HAVING COUNT(*) > 1 ) idx2
13  ON idx1.index_name = idx2.index_name
14 WHERE num_cols IS NOT NULL
15  AND tablespace_name = 'FORSCHUNG'
16  AND idx1.table_name NOT LIKE 'EVT_%'
17  AND idx1.table_name NOT LIKE 'SM_%'
18 ORDER BY  num_cols DESC;
```

Listing 3.6: SQL-Quellcode zur Ermittlung zusammengesetzter Indexe

Tabelle A.13 illustriert die Resultate. Es ist hierbei zu prüfen, ob diese Indexe von der Anwendung überhaupt verwendet werden. Ansonsten ist der Overhead zur Pflege der

Indexstruktur unnötiger Aufwand. Wird der jeweilige Index nicht verwendet, kann er ohne Probleme gelöscht werden.

3.1.8 Lösch-Trigger

Bei den Lösch-Triggern handelt es sich um Trigger, die bei einer **DELETE**-Operation aktiviert werden. Diese werden in der Datenbank des Forschungsportals meist zur Sicherung der referenziellen Integrität eingesetzt, könnten in diesem Fall aber auch genauso gut durch Fremdschlüssel mit der **DELETE CASCADING**-Option ersetzt werden, um eine gewisse Einheitlichkeit zu wahren. Das Verwenden von Triggern zur Erhaltung der referenziellen Intergrität ist generell eine unsaubere Vorgehensweise. Die **DELETE CASCADING** Option sorgt dafür, dass von dem Tupel referenzierte Einträge in der anderen Tabelle automatisch mit gelöscht werden, wenn eine **DELETE**-Anfrage ausgeführt wird. Anfrage 3.7 gibt Auskunft über die vorhandenen Lösch-Trigger. Verwendet wurde dazu die Data Dictionary-Tabelle *all_triggers*, aus der sich alle Trigger direkt auslesen lassen.

```
1 SELECT *
2 FROM all_triggers
3 WHERE triggering_event LIKE '%DELETE%'
4  AND owner NOT LIKE '%SYS%'
5  AND owner <> 'XDB';
```

Listing 3.7: SQL-Quellcode zur Ermittlung von Lösch-Triggern

Es sollte hierbei überprüft werden, welche der Trigger tatsächlich für den beschriebenen Zweck eingesetzt werden. Diese sollten dann gelöscht und durch ein äquivalentes Fremdschlüssel-Constraint mit **DELETE CASCADING**-Option ersetzt werden.

3.1.9 Spalten mit nur einem einzigen Wert

Spalten, die nur einen einzigen Wert in allen Feldern enthalten, sind ein Kennzeichen für Redundanz bzw. Speicherplatzverschwendung. Hat eine Spalte immer einen bestimmten Wert, dann muss dies nicht in der Datenbank abgespeichert werden. Für die Anfrage zur Ermittlung solcher Spalten müssen **CLOB**-Spalten ausgeblendet werden, da bei diesen die Anzahl der unterschiedlichen Werte in der Spalte nicht gezählt werden und somit keine Aussage über die unterschiedlichen Werte möglich ist. Anfrage 3.8 liefert Auskunft über solche Spalten, wobei hier anzumerken ist, dass in der Spalte *num_distinct* eine 0 steht, wenn nur ein einziger Wert in der Spalte steht. Daher erfolgt darüber die Selektion in der Anfrage.

```
1  SELECT t.table_name, a.column_name, t.num_rows, a.num_distinct
2  FROM dba_tables t
3  LEFT JOIN all_tab_columns a
4   ON t.table_name = a.table_name
5  WHERE t.tablespace_name ='FORSCHUNG'
6   AND a.owner = 'WWW_USER'
7   AND a.num_distinct = 0
8   AND a.table_name NOT LIKE 'BIN%'
9   AND a.data_type <> 'CLOB'
10  AND num_rows > 1
11 ORDER BY num_rows DESC;
```

Listing 3.8: SQL-Quellcode zur Ermittlung von Spalten mit nur einem Wert

Das Resultat der Ausführung, welches zusätzlich manuell ausgewertet wurde, befindet sich in Tabelle A.16. Bei diesen Spalten ist dann zu prüfen, ob die Spalten in der Anwendung verwendet werden. Werden die Spalten nicht verwendet, können sie gelöscht werden. Werden sie dagegen verwendet, ist zu prüfen, ob die Verwendung notwendig ist. Bei Bedarf können sie dann aus der Anwendung entfernt oder ersetzt werden und erst dann können die Spalten auch in den Tabellen gelöscht werden.

3.1.10 Spalten mit Schlüsseleigenschaft aber ohne Primärschlüssel-Constraint

Spalten, die bei n Zeilen n verschiedene Werte gespeichert haben, sind in der Regel Kandidaten für Primärschlüssel. In der Forschungsdatenbank finden sich jedoch einige Tabellen, die keinen Primärschlüssel auf diesen Spalten definiert haben. Es ist also nötig Tabellen zu identifizieren, die keinen Primärschlüssel haben, was ebenfalls in der Datenbank auftritt. Anfrage 3.9 erzeugt eine Liste mit diesen Spalten.

```
1  SELECT t.table_name, c.column_name, t.num_rows, c.num_distinct,
2  co.constraint_type
3  FROM all_tables t
4  LEFT JOIN all_tab_columns c
5   ON t.table_name = c.table_name
6  LEFT JOIN all_cons_columns cc
7   ON (c.column_name=cc.column_name
8   AND c.table_name=cc.table_name)
9  LEFT JOIN all_constraints co
10  ON cc.constraint_name = co.constraint_name
11 WHERE t.num_rows = c.num_distinct
12  AND t.tablespace_name = 'FORSCHUNG'
13  AND t.table_name NOT LIKE 'EVT_%'
14  AND co.constraint_type <> 'P'
15  AND t.num_rows > 0
16 ORDER BY t.num_rows DESC;
```

Listing 3.9: SQL-Quellcode zum Finden von Spalten mit Schlüsseleigenschaft ohne Primärschlüssel-Constraint

Hierbei ist nun manuell zu prüfen, ob für diese Spalten ein Primärschlüssel definiert werden sollte, oder zumindest ein **UNIQUE**-Constraint, wenn dieses nicht sogar bereits vorhanden ist. Das Ergebnis der Ausführung dieser Anfrage befindet sich in Tabelle A.17.

3.1.11 Zusammenfassung

In diesem Abschnitt wurden verschiedene Anfragen an das Data Dictionary gestellt, um Fehler bzw. Probleme im Datenbankschema zu identifizieren. Es wurde dazu unter anderem nach leeren bzw. fast leeren Tabellen, ID-Spalten ohne Index, Fremdschlüssel-Spalten ohne Fremdschlüssel-Constraint, alten Tabellen, Zusammengesetzten Indexen, Lösch-Triggern, Spalten mit nur einem einzigen Wert und Spalten mit Schlüsseleigenschaft ohne Primärschlüssel-Constraint gesucht. Zu allen Problemen wurden Lösungsansätze diskutiert.

3.2 Analyse von Anfragen mit Top-Aktivität

Zu Beginn der Analyse wurde über 1 Woche hinweg die Top-Aktivität im Oracle Enterprise Manager beobachtet und die für hohe Auslastung verantwortlichen Anfragen aufgezeichnet. Insgesamt werden hierbei 13 Anfragen identifiziert. Für jede dieser Anfragen werden ein oder mehrere Verbesserungsvorschläge erarbeitet, beispielsweise die Anpassung des Datenbankschemas, das Anlegen von Indexen oder die Anpassung der Anfrage bzw. des Ändern des Anwendungscodes. Das Umschreiben der Anfragen zur Beschleunigung dieser ist dabei die am wenigsten aufwändige Variante: Wie in [SB03] beschrieben, kann von dieser Methode nur Vorteile erwartet werden. Im Gegensatz dazu kann das Anlegen von Indexen oder gar das Ändern des Datenbankschemas zu nichtgewünschten Seiteneffekten führen und ist dadurch auch in der Regel aufwändiger zu implementieren. In [SB03] wird beschrieben, wie erkennbar ist, dass eine Anfrage zu lange läuft:

1. Es werden zuviele Zugriffe auf die Festplatte getätigt, z.B. wenn für eine Punktanfrage die gesamte Relation durchlaufen werden muss.
2. Es wird der Ausführungsplan analysiert, und herausgefunden, dass relevante Indexe nicht verwendet werden.

Diese Merkmale wurden auch teilweise in den betrachteten Anfragen beobachtet, z.B. in Abschnitt 3.2.5, wo der zweite Punkt zutrifft.

Ein Teil der im Folgenden aufgezeigten Probleme wurde bereits gelöst und Messwerte wurden für diese Anfragen aufgezeichnet. Alle Messwerte wurden durch folgende Vorgehensweise gewonnen: Es wurden jeweils 120 einzelne Messungen aufgezeichnet, wovon dann die 10 höchsten und die 10 niedrigsten Werte herausgefiltert wurden, sodass Ausreißer in den Daten nicht berücksichtigt werden. Von den restlichen 100 Messwerten wurde dann der Mittelwert gebildet, was das Endresultat darstellt. Dieser Wert wird auch **gestutztes Mittel** genannt [Pre05].

3.2.1 Anfrage zur Ermittlung von Projekten eines bestimmten Nutzers

Die Anfrage (die vollständige Anfrage befindet sich im Anhang A.1) dient zur Ermittlung von Projekten eines bestimmten Nutzers, inklusive referenzierter Informationen, wie zum Beispiel Projektbearbeiter. Dazu werden Daten aus den Tabellen *fodb_projekte*, *fodb_projektbearbeiter* und *fodb_user* abgerufen und verknüpft. Diese Anfrage (bzw. ähnliche Anfragen - Literale nicht miteinbezogen) wurde mit Abstand am häufigsten als Verursacher für Top-Aktivität identifiziert. Bei der Anfrage fällt auf, dass über die *fodb_projekte.from_id* und über die *fodb_projektbearbeiter.fp_nr* jeweils ein Zugriff auf die Datensätze erfolgt. Da es auf diesen Spalten keine Indexe gibt, erfolgt ein FTS, was bei der der großen Anzahl an Einträgen in den verwendeten Tabellen sehr aufwändig ist. Zur Verbesserung des Ausführungsplans der Anfrage wird nun vorgeschlagen, auf beide Spalten einen Index zu definieren. Die eigentlichen hohen Kosten werden jedoch durch die Selektionsbedingungen (siehe 3.10) ausgelöst, was auch im Ausführungsplan (Abbildung 3.1) dunkelblau markiert wurde. Um dieses Problem zu beheben, empfiehlt es sich, beispielsweise das Projektendejahr einzeln

abzuspeichern und darüber einen Index zu legen. Alternativ bietet sich ein als Funktion definierter Index an, um über den Substring 1 bis 4 einen Index zu definieren. Dies würde bedeuten, dass für jeden *substr(fodb_projekte.projektende, 1, 4)*-Wert ein Eintrag im Index vorhanden wäre, sodass im Index nach den Datensätzen gesucht werden kann. Dies sollte die Suche beschleunigen.

```
1 fodb_projekte.status < 3 AND
2 substr(fodb_projekte.projektende, 1, 4) >= '1990'
```

Listing 3.10: SQL-Quellcode

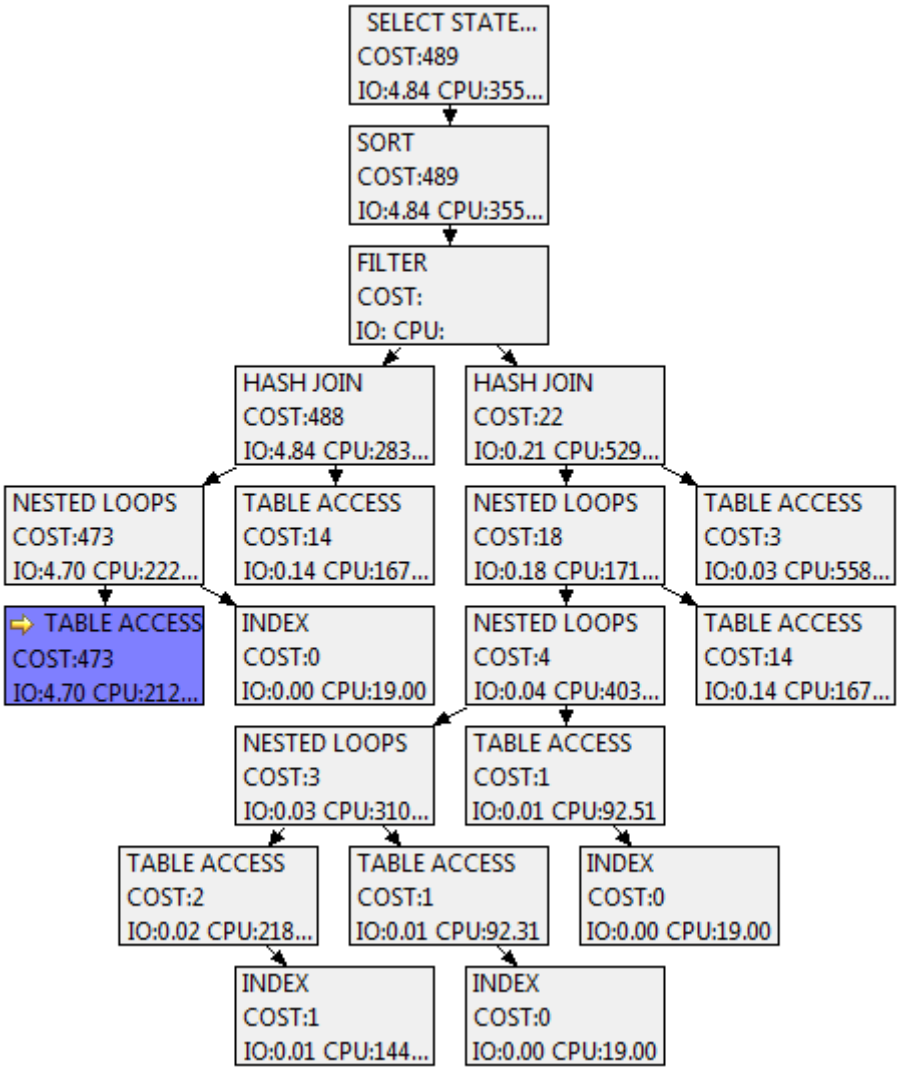

Abbildung 3.1: Ausführungsplan zu Anfrage A.1

3.2.2 Anfrage zur Ermittlung des letzten Updates der Projekte eines Nutzers

Anfrage 3.11 dient zur Ermittlung von Aktualisierungszeiten der Projekte eines bestimmten Nutzers, wozu Daten aus der Tabelle *fodb_projekte* abgerufen werden. Auch

bei Anfrage 3.11 erfolgt der Zugriff über die Spalte *fodb_projekte.from_id*, sodass auch hier derzeit ein FTS durchgeführt werden muss. Dementsprechend wurde das Problem behoben, indem wie bereits für Abschnitt 3.2.1 vorgeschlagen ein Index auf die Spalte *from_id* gesetzt wurde. Die Ausführungszeit der Anfrage reduzierte sich dadurch von 9ms auf 0.5ms.

```
1  SELECT substr(last_update,1,4)||substr(last_update,6,2)
2  ||substr(last_update,9,2) AS datum FROM fodb_projekte
3  WHERE from_id = :pid
4  ORDER BY last_update DESC
```

Listing 3.11: SQL-Quellcode zur Ermittlung des letzten Updates eines Projektes

3.2.3 Anfrage zur Ermittlung von Publikationen mit mindestens einem Zitat und einer DOI

Anfrage 3.12 dient zur Ermittlung von Publikationen mit mindestens einem Zitat und einem Digital Object Identifier (DOI). Das DOI-System bietet eine Infrastruktur zur persistenten und eindeutigen Identifikation von Objekten jeden Typs [Fou]. Das Forschungsportal Sachsen-Anhalt nutzt unter anderem den DOI, um Informationen zu Publikationen abzurufen (vgl. Abschnitt 2.4). Um die Daten für die Anfrage zu ermitteln, werden aus den Tabellen *fodb_publikation*, *fodb_publikation_user* und *fodb_publikation_extern* Daten abgerufen, miteinander verknüpft und nach den genannten Bedingungen gefiltert. Bei dieser Anfrage ist bei Blick auf den Ausführungsplan zu sehen, dass über die *fk_user* in der Tabelle *fodb_publikation_user* ein FTS erfolgt. Insgesamt liefert die Subquery in dem Fall 89 Zeilen zurück. Da die Tabelle *fodb_publikation_user* insgesamt mehr als 50.000 Einträge besitzt, hat es sich empfohlen, auf diese Spalte einen Index zu erstellen. Dabei wurde ein B-Baum-Index bzw. ein Hash-Index in Betracht gezogen, da die Anzahl unterschiedlicher Werte mehr als 1.000 beträgt und somit ein Bitmap-Index zu viel Speicher-Aufwand bedeutet hätte. Letztendlich wurde zunächst für die Tabelle ein Primärschlüssel auf die Spalten *fk_user* und *fk_catid* definiert, welcher bis dahin noch nicht existierte. Dadurch wurde auch ein **UNIQUE**-Index auf *fk_user* und *fk_catid* angelegt. Dadurch sank die Ausführungsgeschwindigkeit zunächst von 11ms auf 4ms. Zudem wurde ein Index auf *fk_user* angelegt. Durch die erneute Änderung konnte die durchschnittliche Ausführungszeit der Anfrage dann auf 1ms reduziert werden.

```
1  SELECT fodb_publikation.jahr as year,
2  fodb_publikation.titel,fodb_publikation.quelle,
3  fodb_publikation.autoren,fodb_publikation.catid,
4  fodb_publikation.impfact,fodb_publikation.import,
5  fodb_publikation_extern.*
6  FROM (Select fk_catid as fk_publ
7  FROM fodb_publikation_user WHERE fk_user=:1)
8  LEFT JOIN fodb_publikation
9  ON fk_publ=catid
10 LEFT JOIN fodb_publikation_extern
11 ON fodb_publikation_extern.fk_catid=catid
12 WHERE doi IS NOT NULL AND cite_count > 0
13 ORDER BY cite_count desc, id asc
```

Listing 3.12: SQL-Quellcode zur Ermittlung von Publikationen mit mindestens einem Zitat und einem DOI

3.2.4 Anfrage zur Ermittlung aller Publikationsinformationen aller Publikationen eines Autors

Anfrage 3.13 ermittelt Informationen zu allen Publikationen eines Autors, wobei der Autor über dessen ID bestimmt ist. Dazu werden insgesamt vier Tabellen miteinander verbunden: *fodb_publikation*, *fodb_publikation_extern*, *fodb_publikationen_link* und *fodb_publikation_user*. Für die Anfrage 3.13 wurde wie bereits für 3.12 vorgeschlagen, auf die Spalte *fodb_publikation_user.fk_user* ein Index gesetzt, da auch hier ein Zugriff auf die Daten über diese Spalte erfolgt. Außerdem wurde in der Zeile 14 statt des **LIKE**-Operators ein Gleichheitszeichen gesetzt, da ansonsten ein String-Vergleich ausgeführt werden würde, was wesentlich aufwändiger ist, als der Vergleich von zwei Integer-Werten und außerdem dazu führen würde, dass ein Index auf der Spalte nicht ausgenutzt wird. Zudem fehlte auf der Tabelle *fodb_publikationen_link* ein Primärschlüssel, sodass mit Anlegen des Primärschlüssels auf der Spalte *id* ein Index angelegt wurde, welcher für den **LEFT JOIN** zwischen *fodb_publikationen_link* und *fodb_publikation* in der Anfrage verwendet wird. Durch die durchgeführten Änderungen konnte die Ausführungszeit dieser Anfrage von 28ms auf 3ms reduziert werden.

```
1  SELECT
2  fodb_publikation.*,
3  fodb_publikation_extern.*,
4  fodb_publikationen_link.abstract,
5  fodb_publikationen_link.volltext,
6  fodb_publikation_user.*
7  FROM fodb_publikation
8  LEFT JOIN fodb_publikation_extern
9   ON fodb_publikation_extern.fk_catid = fodb_publikation.catid
10 LEFT JOIN fodb_publikationen_link
11  ON fodb_publikationen_link.id = fodb_publikation.catid
12 LEFT JOIN fodb_publikation_user
13  ON fodb_publikation_user.fk_catid = fodb_publikation.catid
14 WHERE fodb_publikation_user.fk_user LIKE :author
15 ORDER BY fodb_publikation.jahr DESC,
16 fodb_publikation.titel ASC
```

Listing 3.13: SQL-Quellcode zur Ermittlung aller Publikationsinformationen aller Publikationen eines Autors

3.2.5 Anfrage zur Ermittlung von Publikationen inklusive Zusatzinformationen

Anfrage 3.14 dient zur Ermittlung von Publikationen (aus der Tabelle *fodb_publikation*) inklusive referenzierter Informationen aus der *fodb_publikation_extern*-Tabelle. Auch in dieser Anfrage wird durch Verwenden des **LIKE** Operators der Index *catid* nicht ausgenutzt. Daher sollte in 3.14 statt des **LIKE**-Operators ein Gleichheitszeichen verwendet werden, um einen aufwändigen Stringvergleich zu vermeiden und den bereits vorhandenen Index auf der Spalte *catid* ausnutzen zu können. Durch Ändern der Anfrage konnte die Ausführungszeit von 50ms auf 0.6ms reduziert werden.

```
1 SELECT *
2 FROM
3 fodb_publikation
4 LEFT JOIN
5 fodb_publikation_extern
6 ON fodb_publikation_extern.fk_catid = fodb_publikation.catid
7 WHERE catid LIKE :catid
```

Listing 3.14: SQL-Quellcode zur Ermittlung von Publikationen inklusive Zusatzinformationen

3.2.6 Anfrage zur Ermittlung der ID eines bestimmten Nutzers

Die Anfrage 3.15 dient zur Ermittlung von Nutzern, die einer bestimmten Struktur zugeordnet und nicht in der Tabelle *fodb_user_status* enthalten sind. Diese Anfrage wird im Oracle Enterprise Manager ADDM als Anfrage aufgeführt, die signifikante Zeit für das Parsen der Statements verursacht. Das bedeutet, dass nicht das Ausführen die aufwendigste Aktion ist, sondern das wiederholte Parsen und Erzeugen eines Ausführungsplans. Um dieses Problem zu beheben, könnten in der Anfrage bind-Variablen (siehe Abschnitt 2.1.3) statt der Literale verwendet werden. Alternativ könnte diese Anfrage in einer Stored Function gespeichert werden. Bei beiden Möglichkeiten würde das neuerliche Parsen und Erzeugen eines Ausführungsplanes bei mehrfacher Ausführung mit verschiedenen Werten wegfallen. Die Anfrage befindet sich im Anwendungscode in der Datei ./functions/pub_findlinks.php auf Zeile 59 und 71. Da diese Anfrage an verschiedenen Stellen ausgeführt wird, empfiehlt es sich letztendlich, eine Stored Function dafür anzulegen, um somit redundanten Code zu vermeiden.

```
1 SELECT id FROM
2 fodb_user
3 WHERE
4 (struktur1_nr='430' OR
5 struktur2_nr='430' OR
6 struktur3_nr='430') AND
7 REGEXP_LIKE(trim(replace(nachname,' '))||
8 substr(trim(vorname), 1, 2), 'H(oe|ö)ffkenKl')
9 AND id NOT IN (SELECT id FROM fodb_user_status)
```

Listing 3.15: SQL-Quellcode zur Ermittlung der ID eines bestimmten Nutzers

3.2.7 Anfrage zur Ermittlung von Kooperationen bei bestimmten Projekten

Anfrage 3.16 dient zur Ermittlung von Kooperationen bei bestimmten Projekten. Dazu werden die Tabellen *fodb_kooperation* und *fodb_koop* jeweils mit der Tabelle *fodb_kooperation_projekt* vereinigt und nach bestimmten Projekt-IDs durchsucht und die zwei Resultatstabellen mittels **UNION** vereinigt. Diese Anfrage zu beschleunigen erfordert manuellen Aufwand: Wenn es möglich ist, sollten die beiden Tabellen *fodb_kooperation* und *fodb_koop* zusammengeführt werden, denn danach entfällt die **UNION**-Operation. Alternativ könnte

ein **MATERIALIZED VIEW** eingesetzt werden, um das Ergebnis des **UNION**s nicht bei der Anfrage-Ausführung neu berechnen zu müssen. Beide Möglichkeiten wären auch für die Anfrage 3.17 hilfreich. Außerdem fehlen Indexe auf *fodb_kooperation_projekt.projekt_id*, *fodb_kooperation_projekt.kooperation_id* und auf *fodb_kooperation.id* sowie Fremdschlüsselbeziehungen in der Tabelle *fodb_kooperation_projekt*. Die Anfrage wird im Anwendungscode in der Datei ./admin/admin_forschungsbericht.adm auf Zeile 1882 ausgeführt.

```
1  SELECT kooperation as partner
2  FROM fodb_kooperation_projekt, fodb_kooperation
3  WHERE projekt_id=:1
4  AND id=kooperation_id
5  AND kooperation_id<100000
6  UNION
7  SELECT name as partner
8  FROM fodb_kooperation_projekt, fodb_koop
9  WHERE projekt_id=:1
10 AND kooperation_id>=100000
11 AND id=kooperation_id-100000
12 ORDER BY partner
```

Listing 3.16: SQL-Quellcode zur Ermittlung von Kooperationen bei bestimmten Projekten

3.2.8 Anfrage zum Ermitteln von Kooperationen und ihrem Status

In Anfrage 3.17 werden die zwei Tabellen *fodb_kooperation* und *fodb_koop* nach bestimmten IDs durchsucht und mittels UNION vereinigt, um daraus dann Kooperationen mit ihrem Status auslesen zu können. Auch bei Anfrage 3.17 hilft es, die beiden Tabellen *fodb_kooperation* und *fodb_koop* zusammenzuführen, wie bereits in Abschnitt 3.2.7 beschrieben wurde. Es ist außerdem zu sehen, dass das zweite **SELECT** bei ID-Werten von kleiner 100000 kein Resultat liefern wird, sodass es unnötig ist. Die Anfrage wird in ./classes/kooperation.class.inc.php auf Zeile 78 ausgeführt.

```
1  SELECT kooperation, status
2  FROM fodb_kooperation
3  WHERE id='4223'
4  UNION
5  SELECT name as kooperation, 1 as status
6  FROM fodb_koop
7  WHERE id='4223'-100000
```

Listing 3.17: SQL-Quellcode zum Ermitteln von Kooperationen

3.2.9 Anfrage zum Ermitteln von Transfers

Anfrage 3.18 durchsucht die *fodb_user_transfer*-Tabelle nach Transfers eines bestimmten Nutzers, welcher durch die *from_id* bestimmt ist. Für die Anfrage 3.18 empfiehlt es sich, *from_id* in der Tabelle *fodb_user_transfer* als Fremdschlüssel zur Spalte *user_id* in der Tabelle *fodb_user* zu kennzeichnen und auf diese Spalte einen Index zu setzen. Außerdem könnten hierfür eine bind-Variable statt eines Literals

verwenden werden, sodass der SQL-Parser das Statement nicht bei jedem neuen Literal neu parsen und einen neuen Ausführungsplan dafür erstellen muss. Eine Alternative dazu wäre es, für diese Anfrage eine Stored Function anzulegen und diese dann stattdessen aus dem Anwendungscode heraus aufzurufen.

```
1 SELECT transfer
2 FROM fodb_user_transfer
3 WHERE from_id='81950'
```

Listing 3.18: SQL-Quellcode zum Ermitteln von Transfers

Die selbe Anfrage wird an 3 unterschiedlichen Stellen im Quelltext ausgeführt, was in Tabelle 3.1 sichtbar ist.

Datei	Zeile
./content/eigene_daten_reminder.cnt	30
./content/gen_start.cnt	36
./functions/progressBar.inc.php	28

Tabelle 3.1: Fundstellen für Anfrage 9 in der Webanwendung

Da diese Anfrage an verschiedenen Stelle im Code benutzt wird, würde sich hier eine Stored Function empfehlen, da Veränderungen so nur an einer Stelle eingepflegt werden müssen.

3.2.10 Anfrage zur Ermittlung der Anzahl der für den Forschungsbericht relevanten Publikationen eines Autors

Anfrage 3.19 dient zur Ermittlung der Anzahl der Publikationen eines Autors. Zunächst werden alle Publikationen einer bestimmten Struktur gesucht, wobei dort auch Publikationen herausgefiltert werden, die in einer Ausnahmen-Tabelle stehen. Diesem Resultat werden weitere Informationen mittels **LEFT JOIN** aus der Tabelle *fodb_publikation* hinzugefügt, sodass abschließend nach dem Namen des Autors gefiltert werden kann.

```
1  SELECT count(*) as anzahl
2  FROM (SELECT distinct fk_publ_catid
3   FROM fodb_publikation_strukt
4   WHERE (fk_strukt = '124' OR fk_strukt = '6')
5   AND fk_publ_catid NOT IN
6   (SELECT pub_id AS catid
7    FROM fodb_fo_bericht_pub_ausnahmen
8    WHERE user_id='84537' AND year=jahr)
9   ) LEFT JOIN fodb_publikation ON fk_publ_catid=catid
10 WHERE REGEXP_LIKE(autoren,
11 '(J.{1,2}ttner[^[:lower:]]+S|S[^;]{1,2}J[^;]{1,2}ttner)')
```

Listing 3.19: SQL-Quellcode zur Ermittlung der Anzahl der für den Forschungsbericht relevanten Publikationen eines Autors

In dieser Anfrage erfolgt ein **SELECT** über die Spalten *year* und *pub_id*, auf welchen jedoch kein Index vorhanden ist, sodass ein FTS ausgeführt wird. Um den Ausführungsplan der Anfrage 3.19 zu verbessern, empfiehlt es sich einen Index auf die Spalten *year* und *pub_id* anzulegen, da über diese beiden Werte der Zugriff im Subselect erfolgt, sodass ein FTS vermieden wird.

3.2.11 Anfrage zur Suche nach Projekten

Anfrage A.2 (der Quellcode befindet sich aus Platzgründen im Anhang) dient zur Suche von Projekten, die bestimmte Wörter enthalten. Dazu werden der Tabelle *fodb_projekte* mittels **LEFT JOIN** die Tabellen *fodb_projekte_suche*, *fodb_user*, *fodb_projektbearbeiter* und *fodb_user_status* hinzugefügt. Danach werden Informationen aus den Tabellen nach dem Suchwort (im Beispiel „mobilisierung“) durchsucht. Das Ausführen dieser Anfrage dauert im Mittel mehr als 1,5s, um dann 28 Resultate zurückzuliefern. Um das Ausführen der Anfrage zu beschleunigen, wurde das Oracle Tool SQL Tuning Advisor ADDM verwendet. Damit wurde ermittelt, dass es günstig wäre, Statistiken für den Optimierer zu sammeln. Wie anhand der Ausgabe des neuen Ausführungsplans (es ist 3.20 und 3.21 zu vergleichen) zu sehen ist, kann durch das Sammeln der Statistiken vom Optimierer ein Filter-Prädikat weiter nach unten geschoben werden (in der Ausgabe des Ausführungsplan von Zeile 5 nach Zeile 9), wodurch sich der Ausführungsplan verbessert. Das Verbesserungspotenzial wird dabei vom Advisor mit 44,28% angegeben.

```
| Id  | Operation                    | Name                     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                          |  9864 |  3641K|       |  2014   (1)| 00:00:25 |
|   1 |  SORT ORDER BY               |                          |  9864 |  3641K|  7528K|  2014   (1)| 00:00:25 |
|*  2 |   FILTER                     |                          |       |       |       |            |          |
|*  3 |    HASH JOIN RIGHT OUTER     |                          |  9864 |  3641K|       |  1215   (1)| 00:00:15 |
|   4 |     TABLE ACCESS FULL        | FODB_USER                |  2608 | 65200 |       |    21   (0)| 00:00:01 |
|*  5 |     FILTER                   |                          |       |       |       |            |          |
|*  6 |      HASH JOIN RIGHT OUTER   |                          |  9864 |  3400K|       |  1193   (1)| 00:00:15 |
|   7 |       TABLE ACCESS FULL      | FODB_USER_STATUS         |     1 |     8 |       |     3   (0)| 00:00:01 |
|*  8 |       HASH JOIN RIGHT OUTER  |                          |  9864 |  3323K|       |  1190   (1)| 00:00:15 |
|   9 |        TABLE ACCESS FULL     | FODB_PROJEKTBEARBEITER   |  7450 |   327K|       |    14   (0)| 00:00:01 |
|* 10 |        HASH JOIN RIGHT OUTER |                          |  9864 |  2889K|  1120K|  1175   (1)| 00:00:15 |
|  11 |         TABLE ACCESS FULL    | FODB_PROJEKTE_SUCHE      | 11015 |   989K|       |   540   (1)| 00:00:07 |
|* 12 |         HASH JOIN            |                          |  9864 |  2003K|       |   477   (1)| 00:00:06 |
|* 13 |          TABLE ACCESS FULL   | FODB_EINRICHTUNG         |    55 |   330 |       |     3   (0)| 00:00:01 |
|* 14 |          TABLE ACCESS FULL   | FODB_PROJEKTE            | 10582 |  2087K|       |   473   (1)| 00:00:06 |
```

Listing 3.20: Ausgabe des Oracle SQL Tuning Advisors für Anfrage A.2 vor Aktivieren der Statistiken

```
| Id  | Operation                    | Name                     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                          |  9917 |  3660K|       |  2023   (1)| 00:00:25 |
|   1 |  SORT ORDER BY               |                          |  9917 |  3660K|  7576K|  2023   (1)| 00:00:25 |
|*  2 |   FILTER                     |                          |       |       |       |            |          |
|*  3 |    HASH JOIN RIGHT OUTER     |                          |  9917 |  3660K|       |  1219   (1)| 00:00:15 |
|   4 |     TABLE ACCESS FULL        | FODB_PROJEKTBEARBEITER   |  7450 |   327K|       |    14   (0)| 00:00:01 |
|*  5 |     HASH JOIN RIGHT OUTER    |                          |  9865 |  3208K|       |  1204   (1)| 00:00:15 |
|   6 |      TABLE ACCESS FULL       | FODB_USER                |  2608 | 65200 |       |    21   (0)| 00:00:01 |
|*  7 |      HASH JOIN RIGHT OUTER   |                          |  9864 |  2966K|  1120K|  1182   (1)| 00:00:15 |
|   8 |       TABLE ACCESS FULL      | FODB_PROJEKTE_SUCHE      | 11015 |   989K|       |   540   (1)| 00:00:07 |
|*  9 |       FILTER                 |                          |       |       |       |            |          |
|* 10 |        HASH JOIN RIGHT OUTER |                          |  9864 |  2080K|       |   480   (1)| 00:00:06 |
|  11 |         TABLE ACCESS FULL    | FODB_USER_STATUS         |     1 |     8 |       |     3   (0)| 00:00:01 |
|* 12 |         HASH JOIN            |                          |  9864 |  2003K|       |   477   (1)| 00:00:06 |
|* 13 |          TABLE ACCESS FULL   | FODB_EINRICHTUNG         |    55 |   330 |       |     3   (0)| 00:00:01 |
|* 14 |          TABLE ACCESS FULL   | FODB_PROJEKTE            | 10582 |  2087K|       |   473   (1)| 00:00:06 |
```

Listing 3.21: voraussichtlicher Ausführungsplan für Anfrage A.2 nach Aktivieren der Statistiken, ausgegeben durch den Oracle SQL Tuning Advisor

3.2.12 Anfrage zur Ermittlung einer Strukur-ID, welche zu einer bestimmen Kooperation gehört

Die Anfrage 3.22 durchsucht die Tabelle *fodb_kooperation_struk* nach einer bestimmten *kooperation_id*. Für die Anfrage hat es geholfen, einen Index auf die Spalte *kooperation_id* anzulegen, da hierüber ein Zugriff auf die Daten erfolgt und die Menge der zurückgelieferten Zahlen im Vergleich zur Größe der Tabelle als relativ klein zu erwarten ist: Bei derzeit knapp über 10.000 Einträgen in der Tabelle gibt es knapp 8.000 unterschiedliche *kooperation_id*-Werte, sodass die Wahrscheinlichkeit hoch ist, wenige Wert mit der Anfrage zurückzuerhalten. Da auf dieser Tabelle ein Primärschlüssel bislang fehlte, wurde durch das Anlegen des Primärschlüssels auch der nötige Index mit angelegt, sodass die Ausführungszeit der Anfrage von 0.8ms auf 0.4ms reduziert wurde.

```
1 SELECT struk_id
2 FROM fodb_kooperation_struk
3 WHERE kooperation_id='2034'
```

Listing 3.22: SQL-Quellcode zur Ermittlung einer Strukur-ID, welche zu einer bestimmen Kooperation gehört

3.2.13 Anfrage zum Zählen aller Publikationen

Anfrage 3.23 zählt alle Publikationen. Dazu werden Duplikate durch gruppieren nach Titel, Quelle und Autoren entfernt. Die Auswirkungen der Anfrage 3.23 sind insofern zu spüren, dass diese bei jedem Aufruf der Startseite des Forschungsportals ausgeführt wird. Bei dieser Anfrage ist das Gruppieren die aufwändigste Operation, da sich bereits mehr als 60.000 verschiedene Publikationen in der Tabelle befinden. Für alle diese mehr als 60.000 Publikationen Titel, Quelle und Autoren zu konkatenieren und danach zu gruppieren ist für den Zweck der Anzeige einer einzigen, nur informativen, Zahl zu aufwändig.

```
1 SELECT count(*) as anzahl
2 FROM
3  (SELECT titel
4   FROM fodb_publikation
5   GROUP BY (titel||quelle||autoren)
6  )
```

Listing 3.23: SQL-Quellcode zum Zählen aller Publikationen

Demzufolge sollte versucht werden, die Gruppierung obsolet werden zu lassen, indem beispielsweise Duplikate von $titel||quelle||autoren$ entfernt werden und dafür gesorgt wird, dass diese nicht wieder in die Tabelle gelangen. Danach wird der Gruppierungsoperator bzw. die gesamte Subquery nicht mehr benötigt. Die angepasste Anfrage wird mehr als 50x schneller ausgeführt (4ms statt 217ms), wie eine entsprechende Messung ergab. Eine Alternative dazu wäre es, einen **MATERIALIZED VIEW** anzulegen, der in bestimmten Zeitabständen automatisch aktualisiert wird, sodass die Zahlen direkt ausgelesen werden können. Das Problem redundanter Datensätze würde so jedoch nicht gelöst werden, da die Original-Tabellen weiterhin mit doppelten Datensätzen gefüllt sind. Es würde also nur das Symptom behoben werden,

aber nicht die eigentliche Ursache des Problems. Daher sollte der Weg der Datenbereinigung und der Verhinderung des Auftretens von Duplikaten gegangen werden. Duplikate könnten beispielsweise durch das Anlegen eines **UNIQUE**-Constraints über die drei betroffenen Spalten verhindert werden.

3.2.14 Zusammenfassung

In diesem Abschnitt wurde anhand einzelner Anfragen gezeigt, dass die Probleme des Forschungsportals vielschichtig sind. Teilweise sind Anfragen inperformant geschrieben, teilweise ist aber auch das Datenbankschema nicht dafür ausgelegt, die Anfragen performant auszuwerten. Auch Konsistenzprobleme lassen sich über die Auswertung von Anfragen finden, wie z.B. in Abschnitt 3.2.7 erkennbar. Diese Form der Analyse ist mit beheben der Probleme der aufgelisteten Anfragen nicht abgeschlossen. Es sollte vielmehr ein fortwährender Prozess sein, in welchem regelmäßig die Anfragen mit Top-Aktivität geprüft werden und somit bei Bedarf die Ursachen für eine langsame Ausführung solcher Anfragen behoben werden können [SB03]. Grund für diese Vorgehensweise ist, dass womöglich weniger langsame Anfragen von wesentlich langsameren maskiert werden und diese weniger langsamen Anfragen sich dann nach beheben der Ursachen in der Liste der Top-Aktivitäten nach oben schieben. Durch die Behebung von einigen der aufgezeigten Fehlerursachen konnten bereits einige Performanceprobleme behoben werden. Dies ist in Abbildung 3.2, Abbildung 3.3 und Abbildung 3.4 sichtbar. Dort werden zum einen die Ausführungszeiten, zum anderen die I/O-Kosten und auch die CPU-Kosten der Anfragen vor und nach der Optimierung gezeigt. Die I/O-Kosten-Werte und die CPU-Kosten-Werte wurden mittels Oracle SQL Tuning Advisor ermittelt. Durch die Abbildungen wird deutlich, wie groß das Optimierungspotenzial für die noch nicht optimierten Anfragen ist. Für die bereits optimierten Anfragen konnte die Ausführungszeit um einen Faktor von bis zu 81 (Abschnitt 3.2.5) reduziert werden. Auch bei den I/O-Kosten war die Anfrage aus Abschnitt 3.2.5 die Anfrage, mit den größten Gewinnen: Die I/O-Kosten der optimierten Anfrage konnte um das 281-fache gesenkt werden. Aber auch bei den meisten anderen Anfragen konnten die I/O-Kosten durch das Implementieren von Indexen oder das Ändern des Anfragetextes teilweise drastisch reduziert werden. Gleiches gilt für die CPU-Kosten, welche beispielsweise für die Anfrage aus Abschnitt 3.2.5 um das 2159-fache reduziert werden konnten. Auch für andere Anfragen konnte die generelle Last gesenkt werden. Bei der Anfrage aus Abschnitt 3.2.3 konnten zwar die I/O-Kosten kaum reduziert werden, dafür konnte allerdings die CPU-Last deutlich gesenkt werden, sodass die Ausführungszeit dieser Anfrage um das 10-fache verringert wurde.

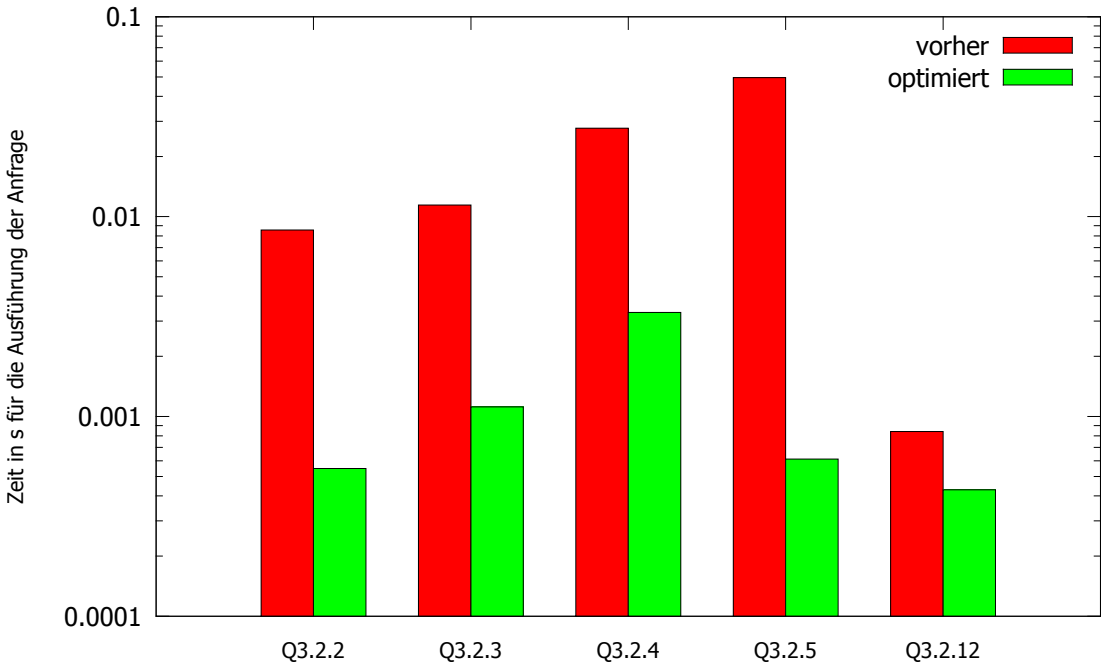

Abbildung 3.2: Ausführungszeiten der Anfragen bei Aufzeichnung der Anfrage und nach Beheben der Problemursache

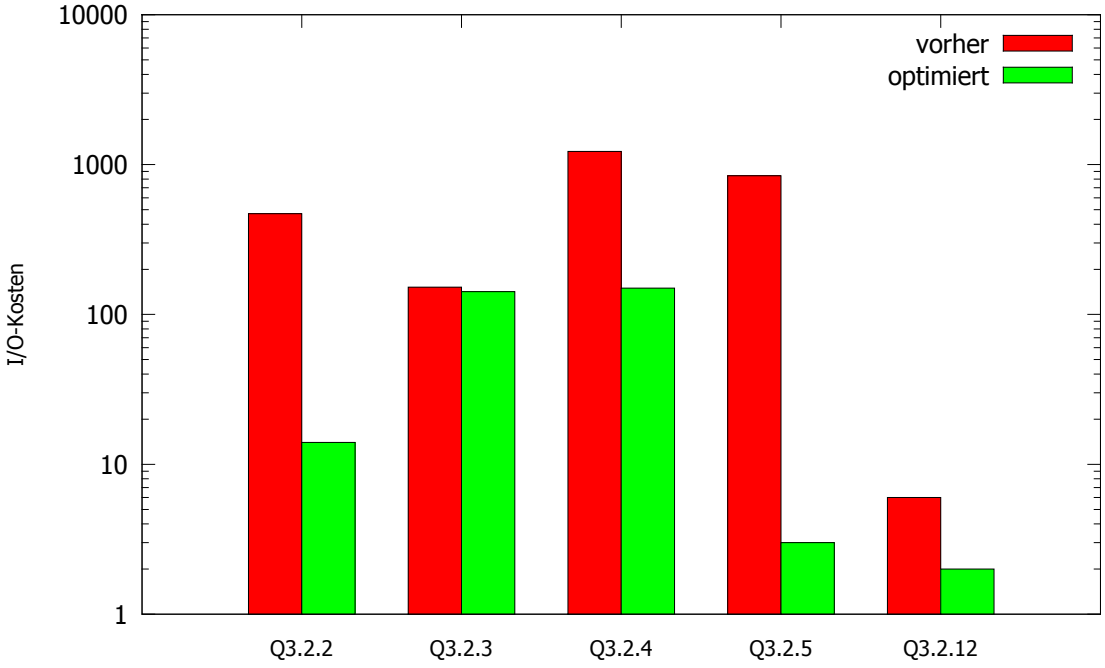

Abbildung 3.3: I/O-Kosten der Anfragen bei Aufzeichnung der Anfrage und nach Beheben der Problemursache

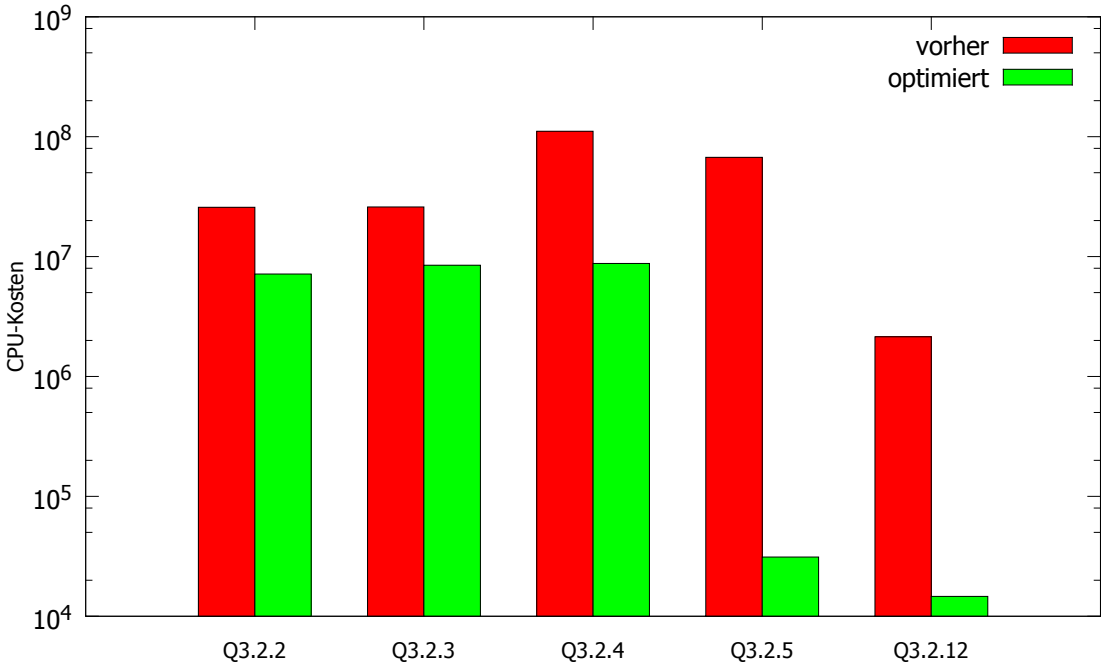

Abbildung 3.4: CPU-Kosten der Anfragen bei Aufzeichnung der Anfrage und nach Beheben der Problemursache

3.3 Analyse der Anwendung

In diesem Abschnitt werden Probleme aufgezeigt, die indirekt oder direkt durch die Art und Weise, wie der Anwendungscode geschrieben ist, verursacht werden.

3.3.1 Probleme im Anwendungscode

Es wurden zunächst nach Anfragen gesucht, die Inkonsistenzen auslösen könnten. Damit sind insbesondere Anfragen gemeint, die eine neue ID für einen Primärschlüssel im Programmcode selbst berechnen. Dies geschieht dann meistens in der Form:

$$new\_id = max(id) + 1 \tag{3.1}$$

Da jedoch aus Tabellen auch wieder Einträge gelöscht werden, kann es unter Umständen dazu kommen, dass hiermit bereits einmal vergebene IDs (die dann gelöscht worden sind), noch einmal verwendet werden, was dann zu fehlerhaften Zuordnungen führen kann. Dies hängt dann auch insbesondere damit zusammen, dass wie in Abschnitt 3.1.5 beschrieben, ein Großteil der Tabellen nicht über Fremdschlüsselbeziehungen verknüpft sind und die Einhaltung der referenziellen Integrität, wenn überhaupt, im Anwendungscode stattfindet. Folgendes Problem könnte dadurch auftreten, wenn in der Haupttabelle Daten gelöscht werden: Wenn die referenzierten Daten nicht mit gelöscht bzw. angepasst werden und die gelöschte ID in der Haupttabelle neu vergeben wird, dann gehören die referenzierten Daten des gelöschten Eintrags nun zum neuen Eintrag, was zu einer inkonsistenten Datenbank führt.

Um solche Probleme zu finden, wurde der gesamte Quelltext mithilfe des regulären Ausdrucks 3.24 case-insensitiv durchsucht.

```
1 INSERT.+?INTO.+?VALUES.+?id|INSERT.+?INTO.+?id.+?VALUES
```

Listing 3.24: Regulärer Ausdruck zum Finden von Quellcode, in dem IDs direkt generiert werden

Hiermit werden Anfragen gefunden, die in ein Datenbankfeld, welches im Namen eine ID enthält, schreiben bzw. als Wert eine ID übergeben. Anfragen, die nicht diese Namenskonventionen beachten, werden nicht gefunden. Nach dem Suchen wurde mit den gefundenen Codestellen ein Code-Review durchgeführt, wobei daraus die Fundstellen als Probleme einzustufen sind. Tabelle 3.2 gibt einen Überblick über die Fundstellen im Anwendungscode.

Pfad und Datei	Fundstelle auf Zeile
./admin/admin_einrichtung_unterstruktur_kooperation.adm	94
./admin/admin_infoportal.adm	417
./admin/admin_news_cred.adm	103
./admin/admin_projekt_kooperationen2.adm	31, 36
./admin/admin_sponsoren_thema_neu.adm	68, 93
./admin/admin.uebersicht.adm	123, 133
./ajax/admin/cmd/admin_koop.php	17, 60
./ajax/user/cmd/user_koop.php	17, 60
./classes/flyer.class.php	72
./classes/Session.class.php	73
./content/eigenes_projekt_kooperationen2.cnt	60
./content/eigenes_thema_erstellen.cnt	66, 91, 108
./content/kooperationen_verwalten.cnt	30, 36
./content/user_registration.cnt	519
./functions/project.php	633, 708-725, 2128
./functions/wordbook.php	8

Tabelle 3.2: ID-Generierung ohne Sequenzen im Anwendungscode des Forschungsportals

Zur Behebung empfiehlt es sich, da Oracle Auto Increment-Spalten nicht unterstützt, Sequenzen für die Spalten anzulegen und für die ID-Generierung zu verwenden. Dies kann beispielsweise mit einer Anfrage wie 3.25 geschehen. In dem Beispielcode wird die Sequenz *sequence_name* mit den Einstellungen Minimalwert gleich 1, Maximalwert gleich 10^{27}-1 angelegt. Außerdem erhöht sich der Sequenzwert um 1, wenn er abgerufen wird. Des Weiteren beginnt die Zählung bei 1 und es werden 20 Werte gecached. Zudem beginnt die Zählung durch das Setzen von **NOCYCLE** bei Erreichen des Maximalwertes nicht wieder bei 1, sondern es wird stattdessen ein Fehler geworfen. Das Schlüsselwort **NOORDER** legt fest, dass nicht garantiert werden muss, dass die Sequenznummern in der Reihenfolge der Anfrage generiert werden müssen [Cor05].

```
1 CREATE SEQUENCE sequence_name
2 MINVALUE 1
3 MAXVALUE 999999999999999999999999999
4 INCREMENT BY 1
5 START WITH 1
6 CACHE 20
7 NOORDER
8 NOCYCLE;
```

Listing 3.25: SQL-Quellcode zur Erzeugung einer Sequenz

Nachdem die Sequenzen angelegt wurden, kann über *sequence_name.nextval* eine neue ID abgefragt werden. Außerdem muss der Anwendungscode dann entsprechend angepasst werden. Durch Verwendung von Sequenzen wird nicht nur die Konsistenz verbessert, sondern es wird auch die Performance gesteigert, da das Abrufen einer neuen ID somit weniger Rechenzeit als vorher beansprucht. Zudem sollten überall, wo notwendig, Fremdschlüsselbeziehungen hinzugefügt werden.

3.3.2 Probleme, welche indirekt durch den Anwendungscode verursacht werden

Als Ausgangspunkt wurde eine Stored Function entdeckt, die zum Erzeugen einer neuen ID für die Tabelle *fodb_woerterbuch* verwendet wurde. Das Ausführen dieser Funktion ist sehr aufwändig: Eine Messung ergab, dass die Funktion zum Ausführen im Mittel 280ms benötigt, was auch nachvollziehbar ist, da diese Tabelle mehr als 180.000 Einträge besitzt. Um die neue ID zu generieren, wird zunächst eine Liste von IDs aus der *fodb_woerterbuch* abgerufen. Danach wird über alle Einträge iteriert, um bis zu einer ID zu gelangen, die es in der Liste noch nicht gibt. Der SQL-Quellcode 3.26 zeigt die Funktion.

Dieses Problem wurde durch eine Sequenz zur ID-Generierung gelöst, wodurch sehr viel Rechenzeit eingespart werden konnte, da die ID-Generierung im Schnitt nun nur noch 0.55ms benötigt. Dies entspricht einem Speedup von rund 500. Es könnte sich demzufolge lohnen, nach weiteren solchen Problemstellen zu suchen, indem alle Stored Procedures und Stored Functions manuell analysiert und derartige Probleme identifiziert und behoben werden.

```
1  create or replace
2  FUNCTION next_wordbook_id
3   RETURN NUMBER
4   IS
5    CURSOR cTableLoop IS SELECT id FROM fodb_woerterbuch ORDER BY id ASC;
6    nextID     INT := 1;
7   BEGIN
8    FOR tableLoop IN cTableLoop LOOP
9     EXIT WHEN tableLoop.id <> nextID;
10    nextID := nextID+1;
11   END LOOP;
12   RETURN nextID;
13  END;
```

Listing 3.26: SQL-Funktion zum Ermitteln einer neuen ID für das Wörterbuch

3.4 Ergebnisse

In diesem Abschnitt werden die Ergebnisse dieser Arbeit diskutiert. Ziel der Arbeit war es, das Datenbanksystem des FPSA zu analysieren und Performance- und Konsistenzprobleme zu finden, zu dokumentieren und für einen Teil dieser Probleme Lösungsvorschläge zu erarbeiten. Die folgenden Tabellen fassen die Ergebnisse der Analyse zusammen, wobei zum einen gezeigt wird, welche Probleme gelöst wurden und zum anderen welche Probleme noch bestehen.

Analyse des Datenbankschemas

In Abschnitt 3.1 wurden Anfragen an das Data Dictionary gestellt, aus deren Ergebnissen (siehe Abschnitt A.1) weitere Aufgaben entstehen. Die Maßnahmen fasst Tabelle 3.3 zusammen.

Analyse von Anfragen mit Top-Aktivität

In Abschnitt 3.2 wurden verschiedene Anfragen aufgezeichnet, dafür Verbesserungsvorschläge erarbeitet und teilweise umgesetzt. Tabelle 3.4 zeigt die Ergebnisse dieser Analyse. Wie dort zu sehen ist, bedarf es für einige der Anfragen noch einer Umsetzung der Verbesserungsvorschläge. Außerdem sollte, wie bereits in dem entsprechenden Abschnitt erwähnt, diese Form der Analyse regelmäßig stattfinden, um auch zukünftig aufwändige Anfragen identifizieren zu können.

Bezeichnung	Resultate in	Maßnahmen
Leere/fast leere Tabellen	Tabelle A.1	Prüfen der leeren Tabellen im Anwendungscode und Löschen der Tabellen
Test-Tabellen	Tabelle A.6	Löschen der Tabellen
ID-Spalten ohne Index	Tabelle A.7	Anlegen von Indexen bzw. Primärschlüsseln
Fremdschlüsselspalten ohne Fremdschlüssel-Constraint	Tabelle A.9	Anlegen der Fremdschlüssel
Alte Tabellen	Tabelle A.12	Löschen der Tabellen
Zusammengesetzte Indexe	Tabelle A.13	Prüfen anhand des Anwendungscodes, welche Indexe tatsächlich benötigt werden und unbenutzte Indexe löschen
Lösch-Trigger	Tabelle A.15	Anlegen von Fremdschlüsseln mit **DELETE CASCADE**-Option und Löschen der Trigger
Spalten mit nur einem einzigen Wert	Tabelle A.16	Prüfen des Anwendungscodes auf Verwendung der Spalten und anschließendes Löschen der Spalten aus den Tabellen
Spalten mit Schlüsseleigenschaft aber ohne Primärschlüssel-Constraint	Tabelle A.17	Anlegen von Primärschlüsseln

Tabelle 3.3: Zusammenfassung der Ergebnisse für Abschnitt 3.1

Anfragebezeichnung	Verbesserungsvorschläge	
	erarbeitet	umgesetzt
Anfrage zur Ermittlung von Projekten eines bestimmten Nutzers	✓	-
Anfrage zur Ermittlung des letzten Updates der Projekte eines Nutzers	✓	✓
Anfrage zur Ermittlung von Publikationen mit mindestens einem Zitat und einer DOI	✓	✓
Anfrage zur Ermittlung aller Publikationsinformationen aller Publikationen eines Autors	✓	✓
Anfrage zur Ermittlung von Publikationen inklusive Zusatzinformationen	✓	✓
Anfrage zur Ermittlung der ID eines bestimmten Nutzers	✓	-
Anfrage zur Ermittlung von Kooperationen bei bestimmten Projekten	✓	-
Anfrage zum Ermitteln von Kooperationen und ihrem Status	✓	-
Anfrage zum Ermitteln von Transfers	✓	-
Anfrage zur Ermittlung der Anzahl der für den Forschungsbericht relevanten Publikationen eines Autors	✓	-
Anfrage zur Suche nach Projekten	✓	-
Anfrage zur Ermittlung einer Strukur-ID, welche zu einer bestimmen Kooperation gehört	✓	✓
Anfrage zum Zählen aller Publikationen	✓	-

Tabelle 3.4: Zusammenfassung der Ergebnisse für Abschnitt 3.2

Analyse der Anwendung

In Abschnitt 3.3 wurde dann die Anwendung analysiert. Das Ergebnis dieser Analyse war zum einen eine Liste mit Fundstellen von Inkonsistenzen auslösenden Programmteilen, zum anderen wurde auch eine sehr langsame Datenbank-Funktion aufgeführt. Aus diesen 2 Punkten ergeben sich folgende Aufgaben für die Zukunft:

- Ersetzen der gefunden Codeteile durch semantisch äquivalente, aber nicht Inkonsistenzen auslösenden Code (z.B. Verwendung von Sequenzen)
- Bei Neu-Entwicklungen auf ID-Generierung innerhalb des Anwendungscode vollständig verzichten
- Alle Datenbank-Funktionen und -Prozeduren untersuchen, da an Hand eines Beispiels (Abschnitt 3.3.2) gezeigt wurde, dass dort ebenfalls Ursachen für Performanceverluste zu finden sind

3.5 Zusammenfassung

In diesem Kapitel wurden eingehende Analysen der Datenbank des FPSA vorgenommen. Für alle Punkte wurden Verbesserungsvorschläge erarbeitet. Dies betrifft sowohl den Anwendungscode, als auch die im Anwendungscode ausgeführten SQL-Anfragen, sowie das Datenbankschema selbst, welches derzeit nur beschränkt den Anforderungen der Anwendung entspricht.

4. Zusammenfassung und zukünftige Arbeiten

In diesem Kapitel werden zunächst die Ergebnisse der Arbeit bewertet. Außerdem wird die Arbeit zusammengefasst und es wird ein Ausblick gegeben, was zukünftige Arbeiten zum Datenbanksystem des FPSA behandeln könnten.

4.1 Bewertung der Ergebnisse

Mit dieser Arbeit wurden bereits einige Performanceprobleme des Forschungsportals behoben, indem die Geschwindigkeit einzelner Anfragen signifikant gesteigert werden konnte. Teilweise sind die optimierten Anfragen um mehr als das 80-fache schneller als die nicht-optimierten Varianten. Das Umsetzen der Vorschläge der verbleibenden Anfragen, sowie die Umsetzung der Vorschläge zur Änderung des Datenbankschemas mussten jedoch aus Zeitgründen offen bleiben. Auch das Ändern des Programmcodes zur Verringerung der Menge an Inkonsistenzen auslösendem Code liegt nicht im Fokus dieser Arbeit und kann in zukünftigen Arbeiten behandelt werden.

4.2 Zusammenfassung

In dieser Arbeit wurde das Datenbanksystem des Forschungsportals Sachsen-Anhalt analysiert und Probleme teilweise behoben. Dazu wurden vielschichtige Analysen erstellt, unter anderem wurde das Datenbankschema, einzelne Anfragen und der Anwendungscode analysiert. Zur Behebung der gefundenen Probleme wurde eine Reihe von Lösungsvorschlägen diskutiert und teilweise bereits umgesetzt. Dazu zählt unter anderem das Beheben von Inkonsistenzen durch Anlegen von Fremdschlüsselbeziehungen oder das Anlegen von Indexen sowie das Optimieren der an die Datenbank gestellten Anfragen. Diese Arbeit kann dazu genutzt werden, um die gefundenen Probleme Schritt für Schritt zu lösen. Zu diesem Zweck befinden sich im Anhang Tabellen mit den gefundenen Problemstellen, welche als Grundlage zur Behebung der Probleme anhand der gemachten Vorschläge dienen sollen.

4.3 Zukünftige Arbeiten

Arbeiten, die sich in Zukunft mit dem Datenbanksystem des Forschungsportals Sachsen-Anhalt beschäftigen, sollten insbesondere folgende Punkte behandeln: Zunächst sollten die in dieser Arbeit gemachten Verbesserungsvorschläge umgesetzt werden. Dies bedeutet, dass sowohl die Struktur des Datenbankschemas, als auch die Anwendung umgestellt werden müssen. Ein anderer, wesentlicher Punkt ist die mögliche Umstellung auf ein anderes DBMS, beispielsweise PostgreSQL oder MySQL, um insbesondere die Lizenzgebühren für die Oracle-Datenbanken in Zukunft einzusparen. Dabei ist dann zunächst eine Aufwandsabschätzung durchzuführen. Insbesondere der Anwendungscode kann voraussichtlich nur mit sehr hohem Aufwand umgestellt werden, da es keine Funktionssammlung für Datenbank-Zugriffe gibt und somit alle Datenbank-Zugriffe im Code verstreut stattfinden. Zudem unterscheidet sich die Syntax der Anfragen der verschiedenen Systeme leicht, was den Umstellungsaufwand weiter erhöhen sollte.

A. Anhang

A.1 Tabellen

TABLE_NAME	NUM_ROWS
FODB_CACHE_LSF_PERSON_STATUS	0
FODB_CACHE_LSF_INSTITUTE	0
FODB_CACHE_LSF_PERSON_ROLES	0
TOAD_PLAN_SQL	0
TRIP_DATA	0
TX_DATA	0
FODB_CACHE_LSF_PERSON	0
CUSTOMER_DATA	0
IMAGE_DATA	0
STATE_DATA	0
TRIP_DATA	0
TX_DATA	0

Tabelle A.1: Leere/fast leere Tabellen - Teil 1

TABLE_NAME	NUM_ROWS
AUSRUESTUNGEN	0
CHK_PROJ	0
CUSTOMER_DATA	0
EIN_TEST	0
FODB-VERANSTALTUNGEN_TYP	0
FODB_CACHE_EINR	0
FODB_CACHE_FREISCHALTEN	0
FODB_CACHE_GERAET	0
FODB_CACHE_KOOP	0
FODB_CACHE_PROFIL_EINR	0
FODB_CACHE_PROFIL_GERAET	0
FODB_CACHE_PROFIL_KOOP	0
FODB_CACHE_PROFIL_PROJEKTE	0
FODB_CACHE_PROFIL_P_EINR	0
FODB_CACHE_PROFIL_P_MITTEL	0
FODB_CACHE_PROFIL_P_SWORT	0
FODB_CACHE_PROFIL_P_TYP	0
FODB_CACHE_PROFIL_SWORT	0
FODB_CACHE_PROFIL_ZENTRUM	0
FODB_CACHE_SWORT	0
FODB_CACHE_UNKLAR	0
FODB_CACHE_ZENTRUM	0
FODB_EINRICHTUNG_UNTERSTR_ALT	0
FODB_FAQ_USER	0
FODB_FORSCHUNGSPARTNER	0
FODB_FO_BERICHT_ADMIN	0
FODB_KOOP_WORT	0
FODB_PUBLIKATION_FAV_OLD	0
FODB_SCHEDULED_JOBS	0
FODB_USER_STATUS	0
FODB_VERANSTALTUNG	0
FODB_WOERTERBUCH_REF	0

Tabelle A.2: Leere/fast leere Tabellen - Teil 2

TABLE_NAME	NUM_ROWS
FODB_ZENTRUM_WORT	0
FODB_ZENTRUM_WORT2	0
IMAGE_DATA	0
ANWENDUNG	1
TEST_WORDS	1
FODB_CONTACT_ANFRAGE	1
FODB_FO_BERICHT_ZEITRAUM	1
FODB_ONLINE	1
FODB_PROJEKT_TYP_WEITERE	1
FODB_SCHEDULED_JOB_TYPE	1
FODB_SPONSOREN_BILD	1
FODB_SPONSOREN_RESET	1
FODB_UEBERSETZUNG	1
FODB_VERANSTALTUNG_TYP	1
FODB_WOERTERBUCH_REFTYPE	1
MURX	1
TTZ_ENTW	1
CONV_TEST	1
FODB_CONFIG	2
FODB_FAQ_KATEGORIE	2
FODB_PROJEKT_TYP_DRITTMITTEL	2
FODB_TR_ARCHIV	2
SERVER	2
FODB_CACHE_PROFIL_USER	3
FODB_EINRICHTUNGEN_BED	3
FODB_GAESTEBUCH_NEW	3
FODB_KALENDER	3
FODB_SUPPORT_MESSAGE	3
FODB_TEST	3
FODB_TEST2	3
FODB_TR_ANTWORT	3
FORSCH_BERICHT	3

Tabelle A.3: Leere/fast leere Tabellen - Teil 3

TABLE_NAME	NUM_ROWS
APPLET_DATA	4
APPLET_DATA	4
FODB_CACHE_USER_ALT	4
FODB_CONTACT_BEWERTUNG2	4
FODB_EINRICHTUNGEN_TYP_BED	4
FODB_GAESTEBUCH2	4
FODB_KAT_ANSPRECHPARTNER	4
FODB_MAIL_GRUPPE	4
FODB_SESSION	4
FODB_SUPPORT_TYPE	4
FINANZEN	5
FODB_BEWERTUNG	5
FODB_CONTACT_KIND_OF_QUESTION	5
FODB_CONTACT_VOTE_OPTIONS	5
FODB_EINRICHTUNGS_TYP	5
FODB_GAESTEBUCH	5
FODB_KAT_ATLAS_EBENE1	5
LANGUAGE_DATA	6
NDW__BROWSERS	6
NDW__BROWSER_ATTRIBUTES	6
FODB_CONTACT_BEWERTUNG	6
FODB_INFOPORTAL_ART	6
LANGUAGE_DATA	6
DESTINATION_DATA	7
DATEI	7
DESTINATION_DATA	7
FODB_EMP_MAIL_MODULE	7
FODB_EMP_STATUS	7
FODB_FO_BERICHT_PUB_TYP_ALLG	7
THEMENTYP	7
VERBUNDTHEMEN	7
FODB_HILFE	8

Tabelle A.4: Leere/fast leere Tabellen - Teil 4

TABLE_NAME	NUM_ROWS
FODB_SUPPORT_SECTION	8
FODB_PROJEKT_TYP	9
FODB_USER_PORTRAIT_CATEGORY	9
FODB_ZENTRUM_SFB	9
LOGREPORT	9
VERTRAGSPARTNER	9
FODB_FO_BERICHT_PUBLIKATIONEN	10
NUMMER	10
FODB_FAQ	11
FODB_SPONSOREN_TEMP	11
FODB_TR_ANFRAGE	11
FODB_EINRICHTUNG_UNIVIS	12
FODB_EINRICHTUNG_UNTERSTR_MUA	13
EINRICHTUNGEN	14
FODB_EINRICHTUNG_UNTERSTR_ABK	14
FODB_GESCHICHTE	14
FODB_NTR_OPERATOR	14
FODB_THEMEN_ZUORDNUNG	14
FODB_USER_TOP_PR_SENDEN	14
ADMIN	15
STYLE	15
FODB_MAP	16
FODB_USER_MAHNUNG_SENDEN	16
GAESTEBUCH	16
FODB_FO_BERICHT_PUB_AUSNAHMEN	17
FODB_THEMEN_ERINNERUNG	17
FODB_USER_TOP_PL_SENDEN	17
FODB_RENTNER	20
FODB_SPONSOR_MAIL	20
MITTELGEBER	20

Tabelle A.5: Leere/fast leere Tabellen - Teil 5

TABLE_NAME	NUM_ROWS
CONV_TEST	1
EIN_TEST	0
FODB_TEST	3
FODB_TEST2	3
TESTDIRECTOR	0
TEST_WORDS	1

Tabelle A.6: Test-Tabellen

TABLE_NAME	COLUMN_NAME	NUM_ROWS
FODB_VPUBL_MINID	MIN(ID)	45702
FODB_PROJEKT_SCHLAGWORT	SCHLAGWORT_ID	35173
FODB_FO_BERICHT_PROJEKTE	IDBERICHT	26061
FODB_FO_BERICHT_PROJEKTE	IDPROJEKT	26061
FODB_MAIL_MAIL_LISTE	LISTID	12759
FODB_KOOPERATION_STRUK	KOOPERATION_ID	10019
FODB_KOOPERATION_STRUK	STRUK_ID	10019
FODB_KOOPERATION_PROJEKT	KOOPERATION_ID	9117
FODB_MAIL_USER	LISTID	4616
FODB_VERANSTALTUNG_GESEHEN	VERANSTALTUNG_ID	2031
FODB_MAIL_MAIL_EINR	EINRICHTID	1979
FODB_VERANSTALTUNGEN	UNIVIS_ID	1637
FODB_EMP_AUTHOR	EMP_USER_ID	1610
FODB_UMFRAGENIP	OPTIONID	1374
FODB_UMFRAGENIP	UMFRAGEID	1374
FODB_GERAET_STRUK	STRUK_ID	933
FODB_FO_BERICHT_BEREICH	UNTERSTRUKID	835
FODB_FO_BERICHT_BEREICH	ADMINID	835
FODB_FO_BERICHT_JOURNAL	STRUKID	592
FODB_MAIL_MAIL	FREISCHALT_ID	538
FODB_USER_PUBL_TOP5	ORDER_ID	442
FODB_SPONSOREN_STATISTIK	SPONSOR_ID	386
FODB_CONTACT_MAIN	FK_CO_VO_OP_ID	293
FODB_CONTACT_MAIN	FK_USER_ID	293
FODB_CONTACT_MAIN	PK_CO_MA_ID	293
FODB_CONTACT_MAIN	FK_KIND_QU_ID	293
FODB_BEREICH_ADMIN	ADMIN_ID	236
FODB_SPONSOREN	VIDEO_FORMAT	211
FODB_SPONSOREN	ACCOUNT_ID	211
FODB_SPONSOREN	VIDEO_ID	211
FODB_CONTACT_NO_REG_USER	FK_CO_MA_ID	205
FODB_CONTACT_NO_REG_USER	PK_CO_NO_REG_ID	205
FODB_USER_NEWS	FREISCHALT_ID	187
FODB_VERANSTALTUNG_FLYER	VERANSTALTUNG_ID	134
FODB_USER_PUBL_TOP5_SAVE_OLD	ORDER_ID	82
FODB_UMFRAGEOPTIONS	UMFRAGEID	72
FODB_EINRICHTUNG_EADMIN	EADMIN_ID	65
FODB_INFOPORTAL_LINK	ART_ID	44
FODB_USER_NEWS_IMAGE	FK_USER_NEWS_ID	42
FODB_USER_NEWS_IMAGE	IWIDTH	42
FODB_FO_BERICHT_PUB_AUSNAHMEN	PUB_ID	20

Tabelle A.7: ID-Spalten ohne Index - Teil 1

ADMIN	ADMIN_ID	15
FODB_NTR_OPERATOR	LIVEZILLA_ID	14
DESTINATION_DATA	DESC_ID	7
DESTINATION_DATA	DESC_ID	7
DATEI	DATEI_ID	7
FODB_EMP_MAIL_MODULE	SORT_ID	7
FODB_CONTACT_BEWERTUNG	PK_CO_MA_ID	6
FODB_CONTACT_BEWERTUNG	FK_CO_VO_OP_ID	6
FODB_CONTACT_BEWERTUNG	FK_USER_ID	6
FODB_CONTACT_BEWERTUNG	FK_KIND_QU_ID	6
FODB_CONTACT_KIND_OF_QUESTION	PK_KIND_OF_QU_ID	5
FODB_TR_ANTWORT	TR_ID	4
FODB_SESSION	SESS_ID	4
FODB_CONTACT_BEWERTUNG2	FK_KIND_QU_ID	4
FODB_CONTACT_BEWERTUNG2	FK_CO_VO_OP_ID	4
FODB_KALENDER	RUBRIK_ID	3
FODB_CONTACT_ANFRAGE	PK_CO_MA_ID	1
FODB_CONTACT_ANFRAGE	FK_USER_ID	1
FODB_CACHE_LSF_PERSON_ROLES	FK_ROLE_ID	0
FODB_CACHE_LSF_PERSON_ROLES	FK_PERS_ID	0
FODB_CACHE_LSF_PERSON_ROLES	FK_INST_ID	0
FODB_CACHE_LSF_INSTITUTE	FK_FODB_LSF_ID	0
FODB_CACHE_LSF_PERSON_STATUS	FK_PERS_ID	0
FODB_CACHE_LSF_PERSON	FK_INST_ID	0
FODB_FO_BERICHT_ADMIN	EINRID	0
FODB_CACHE_ZENTRUM	ZENTRUM_ID	0
FODB_CACHE_SWORT	SCHLAGWORT_ID	0
FODB_CACHE_KOOP	KOOP_ID	0

Tabelle A.8: ID-Spalten ohne Index - Teil 2

TABLE_NAME	COLUMN_NAME	NUM_ROWS
FODB_ADMIN_FLAGS	FK_USER	22
FODB_CACHE_LSF_PERSON_STATUS	FK_PERS_ID	0
FODB_CONTACT_ANFRAGE	FK_USER_ID	1
FODB_CONTACT_BEWERTUNG	FK_USER_ID	6
FODB_CONTACT_BEWERTUNG	FK_KIND_QU_ID	6
FODB_CONTACT_BEWERTUNG	FK_CO_VO_OP_ID	6
FODB_CONTACT_BEWERTUNG2	FK_CO_VO_OP_ID	4
FODB_CONTACT_BEWERTUNG2	FK_KIND_QU_ID	4
FODB_CONTACT_MAIN	FK_KIND_QU_ID	293
FODB_CONTACT_MAIN	FK_CO_VO_OP_ID	293
FODB_CONTACT_MAIN	FK_USER_ID	293
FODB_CONTACT_NO_REG_USER	FK_CO_MA_ID	205
FODB_EINRICHTUNG	FK_ORT	81
FODB_FORSCHUNGSPARTNER	FK_PROJEKT	0
FODB_FORSCHUNGSPARTNER	FK_PARTNER	0
FODB_KAT_ANSPRECHPARTNER	FK_EINR	4
FODB_KAT_ATLAS_EBENE2	FK_EBENE1	21
FODB_KAT_ATLAS_EBENE3	FK_EBENE2	94
FODB_KAT_ATLAS_PROJEKT	FK_ATLAS	475
FODB_KAT_ATLAS_PROJEKT	FK_PROJEKT	475
FODB_KOOPERATION_REF	FK_USER	1359
FODB_KOOPERATION_REF	FK_STRUCT	1359
FODB_KOOPERATION_REF	FK_KOOP	1359
FODB_KOOP_WORT	FK_KOOP	0
FODB_KOOP_WORT	FK_WORT	0
FODB_PATENTS_IMAGES	FK_PATENTS_ID	0
FODB_PATENTS_IMAGES	FK_IMAGES_ID	0
FODB_PATENTS_OFFERS	FK_PATENTS_ID	0
FODB_PATENTS_SECTORS	FK_SECTORS_ID	0
FODB_PATENTS_SECTORS	FK_PATENTS_ID	0

Tabelle A.9: FK-Spalten ohne FK-Constraint - Teil 1

TABLE_NAME	COLUMN_NAME	NUM_ROWS
FODB_PROJEKT_AUSNAHME	FK_USER	106
FODB_PROJEKT_AUSNAHME	FK_PROJEKT	106
FODB_PROJEKT_SCHLAGWORT	FK_PROJEKT	34016
FODB_PROJEKT_SCHLAGWORT _TEMP	FK_PROJEKT	86960
FODB_PROJEKT_WORT2	FK_WORT	647998
FODB_PROJEKT_WORT2	FK_PROJEKT	647998
FODB_PUBLIKATION_EXTERN	FK_CATID	15496
FODB_PUBLIKATION_FAV_OLD	FK_CATID	0
FODB_PUBLIKATION_FAV_OLD	FK_USER	0
FODB_PUBLIKATION_STRUKT	FK_PUBL_CATID	63499
FODB_PUBLIKATION_STRUKT	FK_STRUKT	63499
FODB_PUBLIKATION_USER	FK_USER	51185
FODB_PUBLIKATION_USER	FK_CATID	51185
FODB_PUBLIKATION_USER_SAVE _OLD	FK_CATID	13200
FODB_PUBLIKATION_USER_SAVE _OLD	FK_USER	13200
FODB_SPONSOREN_BILD	FK_SPONSOR	1
FODB_SPONSOREN_PLANER	FK_SPONSOR	98
FODB_SPONSOREN_RESET	FK_SPONSOR	0
FODB_SPONSOR_RECHNUNG	FK_SPONSOR	91
FODB_THEMA_FKZUORD	FK_ZUORDNUNG	185
FODB_THEMA_FKZUORD	FK_THEMA	185
FODB_THEMEN_ERINNERUNG	FK_USER	17
FODB_USER	FK_ORT	2608
FODB_USER_NEWS_IMAGE	FK_USER_NEWS_ID	50
FODB_USER_PORTRAIT	FK_USER	336
FODB_USER_PORTRAIT_LOG	FK_PORTRAIT	1609
FODB_USER_PROFIL_SENDEN	FK_USER	587
FODB_USER_PUBL_TOP5	FK_USER	478
FODB_USER_PUBL_TOP5	FK_PUBL	478

Tabelle A.10: FK-Spalten ohne FK-Constraint - Teil 2

TABLE_NAME	COLUMN_NAME	NUM_ROWS
FODB_WOERTERBUCH_REF	FK_TYPE	0
FODB_WOERTERBUCH_REF	FK_REF_ID	0
FODB_WOERTERBUCH_REF	FK_WORT	0
FODB_ZENTRUM_EINR	FK_EINR	46
FODB_ZENTRUM_EINR	FK_ZENTRUM	46
FODB_ZENTRUM_WORT	FK_WORT	0
FODB_ZENTRUM_WORT	FK_ZENTRUM	0
FODB_ZENTRUM_WORT2	FK_WORT	0
FODB_ZENTRUM_WORT2	FK_ZENTRUM	0

Tabelle A.11: FK-Spalten ohne FK-Constraint - Teil 3

TABLE_NAME	NUM_ROWS
FODB_PROJEKT_GERAET_OLD	654
FODB_PROJEKT_KOOPERATION_OLD	6104
FODB_PUBLIKATIONEN_OLD	48837
FODB_PUBLIKATION_FAV_OLD	0
FODB_PUB_SAVE_DELETE_OLD	15207
FODB_PUB_SAVE_DOPPELT_OLD	29685
FODB_PROJEKT_PUB_SAVE_OLD	820
FODB_PROJEKTE_PUB_MAP_SAVE_OLD	809
FODB_PUBLIKATION_USER_SAVE_OLD	13200
FODB_PUB_LINK_SAVE_OLD	155
FODB_USER_PUBL_TOP5_SAVE_OLD	82

Tabelle A.12: Alte Tabellen

TABLE_NAME	INDEX_NAME	NUM_COLS	NUM_ROWS
ADMIN	ADM_PK	6	15
FODB_PUBLIKATION	SEARCH_FODB_PUB _LM_TEST	6	61682
FODB_FO_BERICHT _PUB_2003	SEARCH_PUB_2003	6	5838
FODB_FO_BERICHT _PUB_2004	SEARCH_PUB_2004	6	6030
FODB_FO_BERICHT _PUB_2005	SEARCH_PUB_2005	6	7303
FODB_FO_BERICHT _PUB_2006	SEARCH_PUB_2006	6	6224
FODB_FO_BERICHT _PUB_2007	SEARCH_PUB_2007	6	6201
FODB_FO_BERICHT _PUB_2008	SEARCH_PUB_2008	6	6256
FODB_EINRICHTUNG _UNTERSTRUKTUR	FODB_IDX_UNTER STRUKTU_ID_TOP	4	566
FODB_PROJEKTE _PUB_MAP	FODB_PROJEKTE _PUB_MAP_U01	4	1707
FODB_KOOPERATION _REF	FODB_KOOPERATION _REF_U01	3	1319
FODB_WOERTER BUCH_REF	FODB_WOERTER BUCH_REF_PK	3	0
FODB_CACHE_LSF _PERSON	IDX_LSF_PER_FNA ME_LNAME_TITLE	3	0
FODB_USER	IDX_USER_VNAME _NAME_TITLE	3	2568
PROJEKTE	PERF01_PROJEKTE	3	3856
PROJEKTE	PERF02_PROJEKTE	3	3856
PROJEKTE	PERF03_PROJEKTE	3	3856
SERVER	SERV_PK	3	2
SERVER	SER_PK	3	2
FODB_PUBLIKATIO NEN_OLD	SYS_C005727	3	48837
FODB_PUBLIKATION _STRUKT	SYS_C005739	3	62507
SERVER	SYS_C006010	3	2
FODB_KAT_ATLAS _PROJEKT	FODB_ATLAS_PRO JEKT_UNIQUE	2	475
FODB_EINRICHTUNG _UNTERSTRUKTUR	FODB_PK_UNTER STRUKTUR	2	566
FODB_PROJEKT _WORT2	FODB_PROJEKT _WORT2_U01	2	647998
FODB_PUBLIKATION _USER	FODB_PUBLIKATION _USER_U01	2	51185
FODB_SCHEDULED _JOBS	FODB_SCHEDULED _JOBS_U01	2	0

Tabelle A.13: Zusammengesetzte Indexe - Teil 1

TABLE_NAME	INDEX_NAME	NUM_COLS	NUM_ROWS
FODB_WOERTER BUCH	FODB_UNQ_WOER TERBUCH	2	172709
FODB_ZENTRUM _WORT2	FODB_ZENTRUM _WORT2_U01	2	0
GERAETE	GER_PK	2	356
FODB_SCHLAGWORT	IDX_ID_SCHLAG WORT	2	19536
FODB_PUB_SAVE _DOPPELT_OLD	IDX_ID_TITEL	2	29685
PROJEKTLEITER	PERF01_PROJEKT LEITER	2	1229
PROJEKTLEITER	PL_NAME	2	1229
STATISTIK	PRKEY_STATISTIK	2	17593
FODB_EMP_AUTHOR	SYS_C005502	2	1610
FODB_EMP_EMAIL	SYS_C005510	2	617
FODB_EMP_INFO	SYS_C005518	2	1595
FODB_USER_PAT TERN	SYS_C005836	2	1738
FODB_USER_PUBL _TOP5	SYS_C005856	2	442
GERAETE	SYS_C005917	2	356
GERAETE	SYS_C005918	2	356
FODB_CACHE_UNIVIS	UNIQUE_IP_EMAIL	2	3340
PROJEKT_KOOP_E	UNQ_PROJEKT _PARTNER_E	2	2672
PROJEKT_KOOP_I	UNQ_PROJEKT _PARTNER_I	2	787
PROJEKT_GERAET	UNQ_PROJ_GERAET	2	458
PROJEKT_PUBLIKA TION	UNQ_PROJ_PUB	2	2164
PROJEKT_SCHLAG WORT	UNQ_PROJ_SW	2	14368

Tabelle A.14: Zusammengesetzte Indexe - Teil 2

TRIGGER_NAME	TRIGGER_TYPE	TABLE_NAME
DELETE_TOPIC	AFTER STATEMENT	FODB_THEMEN
GERAET_LOESCHEN	AFTER STATEMENT	FODB_GERAET
FODB_GERAET _STRUK	AFTER EACH ROW	FODB_GERAET _STRUK
MAIL_LOESCHEN	AFTER STATEMENT	FODB_MAIL_MAIL
DEL_PROJEKT	BEFORE EACH ROW	PROJEKTE
FODB_PROJEKTE _PUB_MAP_AD	AFTER STATEMENT	FODB_PROJEKTE
FODB_KOOPERATION	AFTER EACH ROW	FODB_KOOPERATION
TRANSFER_LOESCHEN	AFTER STATEMENT	FODB_USER
FODB_GERAET	AFTER EACH ROW	FODB_GERAET
FODB_USER_LEVEL_AD	AFTER STATEMENT	FODB_PROJEKTE
USER_LOESCHEN	AFTER STATEMENT	FODB_USER
FODB_KOOPERA TION_STRUK	AFTER EACH ROW	FODB_KOOPERA TION_STRUK

Tabelle A.15: Lösch-Trigger

TABLE_NAME	COLUMN_NAME	NUM_ROWS
PROJEKTE	BETRAG_UNI_ECU	3856
PROJEKTE	KOOPPART_EXT	3856
PROJEKTE	KOOPPART_INT	3856
PROJEKTE	KOORDINATION	3856
FODB_FO_BERICHT	DISS	545
FODB_KAT_ATLAS_EBENE3	FK_PARTNER4	94
FODB_RENTNER	NACHNAME	22
FODB_RENTNER	VORNAME	22
GAESTEBUCH	ANTWORT	16
FODB_HILFE	BESCHREIBUNG_E	8
FODB_EMP_MAIL_MODULE	SORT_ID	7
FODB_GAESTEBUCH2	FROM_ID	4
FODB_CACHE_PROFIL_USER	SUCHE	3
FODB_CACHE_PROFIL_USER	TELEFON	3

Tabelle A.16: Tabellen-Spalten, die nur einen Wert enthalten

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
TEST_WORDS	NAME	1	1	C
FODB_MENU	MENU_KEY	47	47	C
FODB_PROJEKT _PUB_SAVE_OLD	ID	820	820	C
FODB_PUB_SAVE_DOP PELT_OLD	CATID	29685	29685	U
FODB_PUB_SAVE_DE LETE_OLD	ID	15207	15207	C
VERTRAGSPARTNER	FP_NR	9	9	R
FODB_USER_PRO FIL_SENDEN	FK_USER	587	587	R
FODB_USER_PORT RAIT	FK_USER	321	321	R
FODB_USER_PATTERN	ID	1738	1738	R
FODB_THEMEN_ERIN NERUNG	FK_USER	17	17	R
FODB_SPONSOR _MAIL	FK_SPONSOR	20	20	R
FODB_PUBLIKATION _EXTERN	FK_CATID	14372	14372	R
FODB_EMP_MELDUNG	EMP_ID	318	318	R
FODB_EMP _MAIL_BUFFER	FK_EMP_MAIL	1899	1899	R
FODB_EINRICHTUNG _UNTERSTR_MUA	FROM_ID	13	13	R
FODB_EINRICHTUNG _UNTERSTRUKTUR	ID	566	566	C
ZENTREN	ZENTRUM_NR	161	161	C
ZENTREN	ZENTRUM	161	161	C
VERTRAGSPARTNER	LFD_NR	9	9	C
VERTRAGSPARTNER	FP_NR	9	9	C
VERBUNDTHEMEN	NAME	7	7	C
VERBUNDTHEMEN	KUERZEL	7	7	C
THEMENTYP	TYP	7	7	C
THEMENTYP	TYPNAME	7	7	C
STYLE	EINR_NR	15	15	U
STYLE	EINR_NR	15	15	C
SERVER	SERV_ID	2	2	U
SERVER	HTTP	2	2	C
SERVER	BESCHREIBUNG	2	2	C

Tabelle A.17: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 1

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
SCHLAGWORTE	SCHLAGWORT	8347	8347	U
SCHLAGWORTE	SCHLAGWORT _NR	8347	8347	C
SCHLAGWORTE	SCHLAGWORT	8347	8347	C
PUBLIKATIONEN	PUBLIKATION	2101	2101	U
PUBLIKATIONEN	PUBLIKATION _NR	2101	2101	C
PUBLIKATIONEN	PUBLIKATION	2101	2101	C
PROJEKTLEITER	LFD_NR	1229	1229	C
PROJEKTE	FP_NR	3856	3856	C
PROJEKTE	LAST_ENTRY	3856	3856	C
MURX	FP_NR	1	1	C
MURX	LFD_NR	1	1	C
MURX	PROJEKTTITEL _D	1	1	C
MURX	P_BESCHREI BUNG_D1	1	1	C
MURX	MITTELGEBER	1	1	C
MURX	PROJEKTBEGINN	1	1	C
MURX	LAST_ENTRY	1	1	C
KOOPERATION_I	PARTNER_INT	528	528	U
KOOPERATION_I	KOOP_I_NR	528	528	C
KOOPERATION_I	PARTNER_INT	528	528	C
KOOPERATION_E	PARTNER_EXT	2161	2161	U
KOOPERATION_E	KOOP_E_NR	2161	2161	C
KOOPERATION_E	PARTNER_EXT	2161	2161	C
GERAETE	GERAET	356	356	U
GERAETE	GERAET_NR	356	356	C
GERAETE	GERAET	356	356	C
GAESTEBUCH	EINTRAG_NR	16	16	C
GAESTEBUCH	KOMMENTAR	16	16	C
GAESTEBUCH	DATUM	16	16	C
FORSCH_BERICHT	FACHGRUPPE_NR	3	3	C
FODB_ZENTRUM_SFB	ID	9	9	C
FODB_ZENTRUM	ID	153	153	C
FODB_WOERTER BUCH_REFTYPE	NAME	1	1	U
FODB_WOERTER BUCH_REFTYPE	ID	1	1	C
FODB_WOERTER BUCH_REFTYPE	NAME	1	1	C

Tabelle A.18: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 2

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_WOERTER BUCH2	WORT	122504	122504	U
FODB_WOERTER BUCH2	ID	122504	122504	C
FODB_WOERTER BUCH2	WORT	122504	122504	C
FODB_VERANSTAL TUNG_TYP	ID	1	1	C
FODB_VERANSTAL TUNG_TYP	NAME	1	1	C
FODB_VERANSTAL TUNG_TYP	NAME_E	1	1	C
FODB_VERANSTAL TUNG_TYP	SORT	1	1	C
FODB_VERANSTAL TUNG_FLYER	ID	134	134	C
FODB_VERANSTAL TUNG_FLYER	URL	134	134	C
FODB_VERANSTAL TUNGEN	ID	1593	1593	C
FODB_USER_TRANS FER	FROM_ID	1466	1466	C
FODB_USER_TOP _PR_SENDEN	ID	14	14	C
FODB_USER_TOP _PL_SENDEN	ID	17	17	C
FODB_USER_TITEL	ID	41	41	C
FODB_USER_START HILFE	USER_ID	2190	2190	C
FODB_USER_REMINDER	FROM_ID	1813	1813	C
FODB_USER_REMINDER	DATUM	1813	1813	C
FODB_USER_PROJEKT BERICHT	ID	2479	2479	C
FODB_USER_PROFIL _SENDEN	FK_USER	587	587	C
FODB_USER_PORTRAIT _LOG	ID	1540	1540	C
FODB_USER_PORTRAIT _KEYWORD	ID	1365	1365	C
FODB_USER_PORTRAIT _CATEGORY	ID	9	9	C
FODB_USER_PORTRAIT	FK_USER	321	321	C

Tabelle A.19: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 3

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_USER_PATTERN	ID	1738	1738	U
FODB_USER_PATTERN	ID	1738	1738	C
FODB_USER_NEWS _IMAGE	ID	42	42	C
FODB_USER_NEWS _IMAGE	LOCATION	42	42	C
FODB_USER_NEWS	ID	187	187	C
FODB_USER_NEWS	DATUM	187	187	C
FODB_USER_MAIL	ID	2906	2906	C
FODB_USER_MAH NUNG_SENDEN	ID	16	16	C
FODB_USER_LEBENS LAUF	FROM_ID	799	799	C
FODB_USER_HPSENDEN _BADMIN	ID	256	256	C
FODB_USER_HPSENDEN	ID	809	809	C
FODB_USER_CLICKED	USER_ID	2424	2424	U
FODB_USER_CLICKED	USER_ID	2424	2424	C
FODB_USER_ANGEBOT	FROM_ID	499	499	C
FODB_USER	ID	2568	2568	C
FODB_UMFRAGEN	ID	21	21	C
FODB_UMFRAGELOGIN	IP	6232	6232	C
FODB_UEBERSETZUNG	ENGLISCH	1	1	C
FODB_THEMEN_ZUORD NUNG	ID	14	14	C
FODB_THEMEN_ZUORD NUNG	BEZEICHNUNG	14	14	C
FODB_THEMEN_ERIN NERUNG	FK_USER	17	17	C
FODB_THEMEN	ID	132	132	C
FODB_SUPPORT_TYPE	ID	4	4	C
FODB_SUPPORT_TYPE	ENGLISH	4	4	C
FODB_SUPPORT_TYPE	GERMAN	4	4	C
FODB_SUPPORT_SEC TION	ID	8	8	C
FODB_SUPPORT_SEC TION	ENGLISH	8	8	C
FODB_SUPPORT_SEC TION	GERMAN	8	8	C

Tabelle A.20: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 4

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_SUPPORT	ID	22	22	C
FODB_SUPPORT	INSCRIBED	22	22	C
FODB_SPONSOR_RECH NUNG	ID	90	90	C
FODB_SPONSOREN_RE SET	FK_SPONSOR	1	1	C
FODB_SPONSOREN_RE SET	DATUM	1	1	C
FODB_SPONSOREN	ANMELDEDATUM	211	211	C
FODB_SPONSOREN	ID	211	211	C
FODB_SESSION	SESS_ID	4	4	C
FODB_SESSION	SESS_EXPIRES	4	4	C
FODB_SCHEDULED _JOB_TYPE	NAME	1	1	U
FODB_SCHEDULED _JOB_TYPE	ID	1	1	C
FODB_SCHEDULED _JOB_TYPE	NAME	1	1	C
FODB_SCHEDULED _JOBS	VALUE	4	4	U
FODB_SCHEDULED _JOBS	ID	4	4	C
FODB_SCHEDULED _JOBS	VALUE	4	4	C
FODB_PUBLIKATION _EXTERN	FK_CATID	14372	14372	U
FODB_PUBLIKATION _EXTERN	FK_CATID	14372	14372	C
FODB_PUBLIKATIONEN _LINK	ID	320	320	C
FODB_PUBLIKATIONEN _OLD	ID	48837	48837	C
FODB_PUBLIKATION	CATID	61682	61682	U
FODB_PUBLIKATION	CATID	61682	61682	C
FODB_PROJEKT _TYP_WEITERE	ID	1	1	C
FODB_PROJEKT _TYP_DRITTMITTEL	ID	2	2	C
FODB_PROJEKT_TYP	ID	9	9	C

Tabelle A.21: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 5

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_PROJEKT_PUB LIKATION	ID	2599	2599	C
FODB_PROJEKT_KO OPERATION_OLD	ID	6104	6104	C
FODB_PROJEKT _CLICKED	PROJEKT_ID	13019	13019	C
FODB_PROJEKT_AN HANG	PROJEKT_ID	50	50	C
FODB_PROJEKTE_TEMP	ID	8741	8741	C
FODB_PROJEKTE_BACK UP	ID	8705	8705	C
FODB_PROJEKTE	ID	10747	10747	C
FODB_ONLINE	SID	1	1	C
FODB_ONLINE	ZEIT	1	1	C
FODB_ONLINE	GUELTIG	1	1	C
FODB_NTR_OPERATOR	ID	14	14	C
FODB_NTR_OPERATOR	VORNAME	14	14	C
FODB_NEW_PROJEKTE _PUB_MAP	CAT_ID	137	137	C
FODB_MITTELGEBER	ID	22	22	C
FODB_MAIL_MAIL_LISTE	ID	12759	12759	C
FODB_MAIL_MAIL_EINR	ID	1979	1979	C
FODB_MAIL_MAIL	ID	538	538	C
FODB_MAIL_LISTE	ID	22	22	C
FODB_MAIL_GRUPPE	ID	4	4	C
FODB_MAIL_GRUPPE	SORT	4	4	C
FODB_KOOPERATION	ID	7673	7673	C
FODB_KOOP	ID	1513	1513	C
FODB_KAT_ATLAS _EBENE3	ID	94	94	C
FODB_KAT_ATLAS _EBENE3	NAME_DE	94	94	C
FODB_KAT_ATLAS _EBENE2	ID	21	21	C
FODB_KAT_ATLAS _EBENE2	NAME_DE	21	21	C
FODB_KAT_ATLAS _EBENE1	ID	5	5	C
FODB_KAT_ATLAS _EBENE1	NAME_DE	5	5	C

Tabelle A.22: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 6

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_INFOPORTAL _LINK	ID	44	44	C
FODB_INFOPORTAL _LINK	SORT	44	44	C
FODB_INFOPORTAL_ART	ID	6	6	C
FODB_INFOPORTAL_ART	ART	6	6	C
FODB_GESCHICHTE	ID	14	14	C
FODB_GERAET	ID	955	955	C
FODB_GEO_ORTE	ID	1868	1868	C
FODB_GAESTEBUCH _BLOCK	ID	129	129	C
FODB_GAESTEBUCH _BLOCK	EMAIL	129	129	C
FODB_GAESTEBUCH2	ID	4	4	C
FODB_GAESTEBUCH2	DATUM	4	4	C
FODB_GAESTEBUCH2	TEXT	4	4	C
FODB_FO_BERICHT _ZEITRAUM	ID	1	1	C
FODB_FO_BERICHT _ZEITRAUM	JAHR	1	1	C
FODB_FO_BERICHT _PUB_TYP_ALLG	ID	7	7	C
FODB_FO_BERICHT _PUB_TYP_ALLG	NAME	7	7	C
FODB_FO_BERICHT _PUB_TYP_ALLG	NAME_PL	7	7	C
FODB_FO_BERICHT _PUB_AUSNAHMEN	PUB_ID	20	20	C
FODB_FO_BERICHT _PUBLIKATIONEN	ID	10	10	C
FODB_FO_BERICHT _PROJEKTE	ID	26061	26061	C
FODB_FO_BERICHT _BILD	BESCHREIBUNG	21	21	C
FODB_FO_BERICHT _BEREICH	ID	835	835	C
FODB_FO_BERICHT _BEREICH	DATUM	835	835	C
FODB_EMP_STATUS	STATUS_TAG	7	7	U
FODB_EMP_STATUS	STATUS_ID	7	7	C
FODB_EMP_STATUS	STATUS_TAG	7	7	C

Tabelle A.23: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 7

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_EMP_MELDUNG	EMP_ID	318	318	C
FODB_EMP_MELDUNG	MELDUNG_DATE	318	318	C
FODB_EMP_MAIL _MO-DULE	ID	7	7	C
FODB_EMP_MAIL _MO-DULE	TEXT	7	7	C
FODB_EMP_MAIL _MO-DULE	TITLE	7	7	C
FODB_EMP_MAIL _BUF-FER	FK_EMP_MAIL	1899	1899	C
FODB_EMP_MAIL _BUF-FER	SCHEDULED	1899	1899	C
FODB_EMP_MAIL	ID	2103	2103	C
FODB_EMP_MAIL	INSERT_DATE	2103	2103	C
FODB_EMP_EMP	EMP_ID	351	351	C
FODB_EMP_EMAIL	EMAIL_ID	560	560	C
FODB_EINRICHTUNG _UNTERSTR_MUA	FROM_ID	13	13	C
FODB_EINRICHTUNG _UNTERSTR_ANG	FROM_ID	26	26	C
FODB_EINRICHTUNG _UNTERSTRUKTUR	ID	566	566	C
FODB_EINRICHTUNG _UNTERSTRUKT	ID	566	566	C
FODB_EINRICHTUNG _STRUKTUR	ID	24	24	C
FODB_EINRICHTUNG _EADMIN	EINRICHTUNG	65	65	C
FODB_EINRICHTUNGS _TYP	ID	5	5	C
FODB_EINRICHTUNGEN _TYP_BED	ID	4	4	C
FODB_EINRICHTUNGEN _BED	ID	3	3	C
FODB_EINRICHTUNG	ID	81	81	C
FODB_CONTACT_VOTE _OPTIONS	PK_CO_VO_OP_ID	5	5	C
FODB_CONFIG	PROPERTY	2	2	C
FODB_BLOCK_WORTE	WORT	54	54	C

Tabelle A.24: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 8

TABLE _NAME	COLUMN _NAME	NUM _ROWS	NUM _DIS TINCT	CONS TRAINT _TYPE
FODB_BEWERTUNG	ID	5	5	C
FODB_BEWERTUNG	TEXT_DE	5	5	C
FODB_BEWERTUNG	TEXT_EN	5	5	C
FINANZEN	FP_NR	5	5	C
FAKULTAET	FAKULTAET_NR	89	89	C
FACHGRUPPE	FACHGRUPPE_NR	444	444	C
EINRICHTUNGEN	EINR_NR	14	14	C
EINRICHTUNGEN	HOMEPAGE	14	14	C
EINRICHTUNGEN	ADRESSE	14	14	C
EINRICHTUNGEN	NAME	14	14	C
DATEI	DATEI_ID	7	7	C
DATEI	DATEINAME	7	7	C
DATEI	BESCHREIBUNG	7	7	C
BILD	FP_NR	815	815	C
BEARBEITER	FP_NR	2044	2044	C
ANWENDUNG	ANWENDUNG_NR	1	1	C
ANWENDUNG	ANWENDUNG	1	1	C

Tabelle A.25: Spalten mit Schlüsseleigenschaft, aber ohne Primärschlüssel - Teil 9

A.2 Quelltexte

```
1  SELECT
2  fodb_projekte.id AS pid,
3  fodb_projekte.projekttitel_d AS ptitel_de,
4  fodb_projekte.projekttitel_e AS ptitel_en,
5  fodb_projekte.projektbesch_d AS beschreibung_de,
6  fodb_projekte.projektbesch_e AS beschreibung_en,
7  fodb_projekte.projektbeginn AS beginn,
8  fodb_projekte.projektende AS ende,
9  fodb_projekte.from_id,
10 fodb_projektbearbeiter.name AS pbname,
11 fodb_user.id AS plid
12 FROM fodb_projekte, fodb_user, fodb_projektbearbeiter
13 WHERE ( (fodb_user.id = 4559 AND (1=1))
14 OR fodb_projekte.id in (
15 SELECT fodb_projekte.id
16 FROM fodb_projekte,
17 fodb_projektbearbeiter,
18 fodb_user,
19 (SELECT id, struktur1_nr, struktur2_nr, struktur3_nr,
20 '%'||REGEXP_REPLACE(trim(nachname),
21 '([^a-z,-]|ö|ü|ä| ß |oe|ue|ae)','%',1,0,'i')||'%' AS comp,
22 REGEXP_REPLACE(trim(nachname),
23 '([^a-z,-]|ö|ü|ä| ß |oe|ue|ae)',
24 '.{1,2}',1,0,'i') AS comp_detail
25 FROM fodb_user WHERE id='4559') comp
26 WHERE
27 fodb_projekte.fp_nr = fodb_projektbearbeiter.fp_nr(+)
28 AND fodb_projekte.from_id=fodb_user.id
29 AND fodb_projekte.id not in
30  (select fk_projekt as ID from fodb_projekt_ausnahme
31 Where fk_user='4559')
32 AND (fodb_user.struktur1_nr=comp.struktur1_nr
33 AND (fodb_user.struktur2_nr=comp.struktur2_nr
34  OR fodb_user.struktur3_nr=comp.struktur3_nr))
35 AND fodb_projektbearbeiter.name LIKE comp.comp
36 AND REGEXP_LIKE(fodb_projektbearbeiter.name,
37 comp.comp_detail)
38 )
39 ) AND
40 fodb_projekte.status < 3 AND
41 fodb_projekte.from_id = fodb_user.id AND
42 substr(fodb_projekte.projektende, 1, 4) >= '1990' AND
43 fodb_projekte.fp_nr = fodb_projektbearbeiter.fp_nr(+)
44
45 ORDER BY ptitel_de
```

Listing A.1: SQL-Quellcode zur Ermittlung von Projekten eines bestimmten Nutzers

```
1  select fodb_user_status.status, fodb_projekte.id AS prid,
2  fodb_projekte.PROJEKTBEGINN AS p_start,
3  fodb_projekte.PROJEKTENDE AS p_ende,
4  fodb_projekte.projekttitel_d AS prtitle
5  FROM fodb_projekte
6  LEFT JOIN fodb_projekte_suche
7  ON fodb_projekte.id = fodb_projekte_suche.id
8  LEFT JOIN fodb_user
9  ON fodb_user.id = fodb_projekte.from_id
10 LEFT JOIN fodb_projektbearbeiter
11 ON fodb_projektbearbeiter.fp_nr = fodb_projekte.fp_nr
12 LEFT JOIN fodb_user_status
13 ON fodb_user_status.id = fodb_projekte.from_id
14 WHERE (fodb_user_status.status <> 2
15    OR fodb_user_status.status IS NULL)
16 AND ( ( lower(suche_d) LIKE lower('%mobilisierung%')
17    OR lower(fodb_user.nachname)
18             like '%mobilisierung%'
19    OR lower(fodb_projekte.projekttitel_d)
20             like '%mobilisierung%'
21    OR lower(fodb_projekte.projekttitel_e)
22             like '%mobilisierung%'
23    OR lower(fodb_user.vorname)
24             like '%mobilisierung%'
25    OR lower(fodb_projektbearbeiter.name)
26             like '%mobilisierung%' ) )
27 AND fodb_projekte.STATUS < 3
28 AND fodb_projekte.einr_nr IN (
29  SELECT id AS einr_nr
30  FROM fodb_einrichtung
31  WHERE deactivated='0')
32 ORDER BY prtitle;
```

Listing A.2: SQL-Quellcode zur Suche nach Projekten

Literaturverzeichnis

[BM72] Rudolf Bayer and Edward McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Information*, 1(3):173–189, 1972. (zitiert auf Seite 7)

[Bru09] Jake Brutlag. Speed matters for google web search, 2009. http://services.google.com/fh/files/blogs/google_delayexp.pdf [Online, abgerufen am 08.11.2012]. (zitiert auf Seite 2)

[Bur10] D.K Burleson. *Oracle Tuning: The Definitive Reference*. Rampant TechPress, 2010. S. 749. (zitiert auf Seite 10)

[Cor05] Oracle Corporation. SQL Reference, 2005. http://docs.oracle.com/cd/B19306_01/server.102/b14200.pdf [Online, abgerufen am 30.01.2013]. (zitiert auf Seite 39)

[Cor08] Oracle Corporation. Oracle database performance tuning guide, 2008. http://docs.oracle.com/cd/B19306_01/server.102/b14211.pdf [Online, abgerufen am 01.10.2012]. (zitiert auf Seite 12)

[Cor09] Oracle Corporation. Reference, 2009. http://docs.oracle.com/cd/B19306_01/server.102/b14237.pdf [Online, abgerufen am 16.01.2013]. (zitiert auf Seite 20)

[Cor10] Oracle Corporation. 2 day + performance tuning guide, 2010. http://docs.oracle.com/cd/B19306_01/server.102/b28051.pdf [Online, abgerufen am 09.01.2013]. (zitiert auf Seite 12)

[Fou] International DOI Foundation. DOI Handbook Introduction. http://www.doi.org/doi_handbook/1_Introduction.html [Online, abgerufen am 27.01.2013]. (zitiert auf Seite 28)

[GWC12] Official google webmaster central blog: Using site speed in web search ranking, 21.09.2012. http://googlewebmastercentral.blogspot.de/2010/04/using-site-speed-in-web-search-ranking.html [Online, abgerufen am 21.09.2012]. (zitiert auf Seite 2)

[Mit03] Sitansu S. Mittra. *Database performance tuning and optimization: Using Oracle*. Springer professional computing. Springer, New York and NY, 2003. (zitiert auf Seite 11)

[MS05] Dinesh P. Mehta and Sartaj Sahni. *Handbook of data structures and applications*. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, Boca Raton (Fla.), 2005. S. 9-15. (zitiert auf Seite 7)

[Nie00] Jakob Nielsen. *Designing web usability*. New Riders Publ, Indianapolis and Ind, [nachdr.] edition, 2000. S. 42. (zitiert auf Seite iii und 1)

[Pre05] Manfred Precht. *Angewandte Statistik*. Oldenbourg, München, 7 edition, 2005. S. 35. (zitiert auf Seite 26)

[SB03] Dennis Shasha and Philippe Bonnet. *Database tuning: Principles, experiments, and troubleshooting techniques*. Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, Amsterdam, 2003. S. 1ff., S 77ff., S. 84ff., S. 143ff. (zitiert auf Seite 7, 10, 11, 12, 19, 21, 26 und 35)

[SM85] Israel Spiegler and Rafi Maayan. Storage and retrieval considerations of binary data bases. *Information Processing & Management*, 21(3):233 – 254, 1985. (zitiert auf Seite 7)

[Spr09] Sylvia Springer. 10 Jahre Forschungsportal Sachsen Anhalt - Virtueller Marktplatz für Wissens- und Technologietransfer weltweit nachgefragt. *Wissenschaftsmanagement*, 15(2):44, 2009. (zitiert auf Seite 15)

[SSH10] Gunter Saake, Kai-Uwe Sattler, and Andreas Heuer. *Datenbanken: Konzepte und Sprachen*. Biber-Buch. mitp Verl.-Gruppe Hüthig Jehle Rehm, Heidelberg and Hamburg, 4 edition, 2010. S. 394f., S. 451f. (zitiert auf Seite 8 und 9)

[SSH11] Gunter Saake, Kai-Uwe Sattler, and Andreas Heuer. *Datenbanken: Implementierungstechniken*. mitp/bhv, 2011. S. 2ff., S. 133ff., S. 158ff., S. 263ff., S. 337, S. 411. (zitiert auf Seite ix, 5, 6, 7, 9 und 10)

[TW07] S. Thomas and L. Williams. Using automated fix generation to secure SQL statements. In *Proceedings - ICSE 2007 Workshops: Third International Workshop on Software Engineering for Secure Systems, SESS'07*, 2007. http://collaboration.csc.ncsu.edu/laurie/Papers/ICSE_SESS_2007.pdf [Online, abgerufen am 28.01.2013]. (zitiert auf Seite 9)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 13.02.2013