INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Projektarbeit

Implementierung einer durchgängigen Sicherheitsarchitektur für eine WWW/SQL-Anwendung

Zhongshi Wang

Aufgabensteller:	Prof. Dr. Hegering
Betreuer:	Gerald Vogt
	Igor Radisic

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Der Lehrstuhl betreibt auf seinen Rechnern einen Apache Web-Server, über den sämtliche Informationen des Lehrstuhls abrufbar sind. Dies umfasst Informationen über die Vorlesungen, Seminare, Publikationen, Diplomarbeiten und FoPras/SEPs. Gleichzeitig gibt es weitere interne Schnittstellen für die Mitarbeiter des Lehrstuhls, die zur Wartung dieser Seiten und Aufruf weiterer Verwaltungsfunktionen dienen. Die Daten werden in einer PostgreSQL-Datenbank abgelegt und die Web-Seiten über Skripte in PHP erzeugt und gewartet.

1.1 Aufgabenstellung

Für die Datenbank gibt es verschiedene Benutzergruppen, die mit unterschiedlichen Rechten auf unterschiedliche Teile der Datenbank zugreifen dürfen. Im öffentlichen Bereich müssen sich Studenten für Übungsgruppen oder Klausuren anmelden. Dort sind Listen zum Beispiel über offene Diplomarbeiten abrufbar. Tutoren von Übungsgruppen bekommen darüber hinaus noch Zugriff auf einige nicht-öffentliche Teile, die während der Betreuung der Gruppen von Bedeutung sind. Die Mitarbeiter des Lehrstuhls haben prinzipiell volle Rechte auf der Datenbank, wobei aber auch hier im Normalbetrieb eine gewisse Zuordnung für Bereiche möglich sein sollte, um zum Beispiel den schreibenden Zugriff auf die Daten für eine Vorlesung nur den entsprechenden Betreuern zu ermöglichen.

Im Moment geschieht die Benutzerverwaltung und Vergabe der Rechte noch manuell in der Datenbank. Im Rahmen dieses Praktikums sollen hier Verwaltungsfunktionen implementiert werden, die folgende drei Aspekte auf der Ebene der Datenbank im Zusammenspiel mit dem Web-Server einfach und konsistent ermöglichen:

- Verwaltung der Benutzer
  Hier handelt es sich um die Einrichtung von neuen Benutzern, die Änderung und Entfernung der in der Datenbank existierenden Benutzer und die Zusammenfas-

sung der bestimmten Benutzer in Gruppen.

- Zuordung von Rechten an die Benutzer
  In Datenbank gibt es prinzipiell zwei Arten von Rechten, dies sind die Systemrechte und die Objektrechte. Die Systemrechte erlauben einem Datenbankbenutzer die Ausführung von `CREATE USER` oder `CREATE DATABASE`, also die Superuserrecht und das Recht, neue Datenbank einzurichten. Unter Objektrechte versteht man die Zugriffsberechtigungen(z.B. SELECT, INSERT, UPDATE/DELETE und RULE) auf die Datenbankobjekte. In PostgreSQL sind Tabellen(tables), Sichten(views) und Sequenzen(sequences) die drei Objektarten, auf die die Zugriffsrechte an die Benutzer vergeben werden können. Die Zugriffsberechtigungen werden von einem Superuser oder Besitzer der Objekte an die Benutzer oder Gruppen erteilt.

- Umsetzung dieser Zuordnung
  Durch die Befehl `GRANT` und `REVOKE` werden die Zugriffsrechte gewährt bzw. sie wieder entfernt.

Diese Funktionen sollen über den Web-Server aufrufbar sein und müssen sich in die bestehende Infrastruktur einfügen.

Durch den Vorgängerpraktikum[BM01] läuft bereits eine PostgreSQL Datenbank in Betrieb. Ziel des Praktikums ist eine betriebsfähige Implementierung dieses Verwaltungssystems am Lehrstuhl. Dies umfasst die notwendigen Skripte in PHP für die eigentliche Funktionalität sowie die eventuell notwendige Anpassung der bestehenden Datenschemata an die Erfordernisse eines sicheren Betriebs der Datenbank.

Kapitel 2

Sicherheitskonzept unter PostgreSQL

Ähnlich wie in Oracle werden Benutzer und Gruppen in PostgreSQL[pgs] für die Benutzerverwaltung verwendet (in Oracle wird das Konzept "Benutzer und Rollen" eingeführt). PostgreSQL verwaltet die Benutzer- und Gruppendaten in ihrem eigenen System. Sie sind nicht mit dem vom Betriebssystem verwendeten (z.B. Unix oder Windows) identisch.

Jede Verbindung zu PostgreSQL muss mit einem registrierten Benutzer erstellt werden, und jeder Benutzer kann zu einer oder mehrerer der definierten Gruppen gehören. Sowohl an die Benutzer als auch an die Gruppen können unterschiedliche Zugriffsberechtigungen auf die Datenbankobjekten vergeben werden.

2.1 Benutzermanagement

Um einen Benutzer zu identifizieren, wird die Benutzerkennung benötigt. Die Benutzerkennung muss eindeutig in PostgreSQL sein und ist wie schon erwähnt nicht notwendigerweise mit der vom Betriebssystem identisch. Alle Benutzerinformationen werden in einer Systemtabelle von PostgreSQL, die `pg_shadow` heißt, gespeichert. Auf diese Tabelle kann nur ein Superuser zugreifen. Parallel gibt es noch ein View `pg_user` von der Tabelle `pg_shadow`, welches für einen normalen Benutzer zugreifbar ist. Der wesentliche Unterschied zwischen `pg_user` und `pg_shadow` liegt darin, dass die Passwörter in `pg_user` nicht mehr in Klartext sondern mit Sternchen dargestellt werden, damit sie vor den normalen Benutzern geschützt werden.

Unter `usename` werden die Benutzerkennungen gespeichert, jede Benutzerkennung hat auch eine eindeutige SystemID zu PostgreSQL, die in der Spalte `usesysid` enthält ist. Die Attribute `usecreatedb` und `usesuper` repräsentieren die Systemberechtigungen

Attribut	**Typ**
usename	Text
usesysid	Integer
usecreatedb	Boolean
usetrace	Boolean
usesuper	Boolean
usecatupd	Boolean
passwd	Text
valuntil	Zeitspanne

Tabelle 2.1: Schema der Tabelle pg_shadow

von jedem Benutzer. Dies wird im folgenden Abschnitt beschrieben.

2.1.1 Benutzereinrichtung

Neue Datenbankbenutzer können nur von Superusers eingerichtet werden. Ein Defaultsuperuser in PostgreSQL ist `postgres`. Mit dem SQL-Befehl `CREATE USER` wird ein neuer Datenbankbenutzer erstellt. Der Befehl `CREATE USER` braucht nur einen Parameter: die Kennung des neuen Benutzers und hat folgende Syntax:

```
CREATE USER Kennung
[ WITH [ SYSID SystemID ][ PASSWORD 'Password' ] ]
[ CREATEDB | NOCREATEDB ]
[ CREATEUSER | NOCREATEUSER ]
[ IN GROUP Gruppename [, ...] ]
[ VALID UNTIL 'Zeitspanne' ]
```

In dieser Syntax ist die *Benutzerkennung* die einzige erforderliche Eingabe, alle anderen sind optional. Die Folgende ist die Detailbereibung aller optionalen Eingaben:

- `SYSID` *`SystemID`*:
  Mit `SYSID` *`SystemID`* wird eine benutzerdefinierte SystemID erzeugt. Diese SystemID ist eine IntegerZahl und muss wie die Benutzerkennung eindeutig in PostgreSQL existieren. Ohne Spezifizierung dieser Option wird eine passende SystemID zu diesem Benutzer von PostgreSQL automatisch gewählt.

- `PASSWORD` *`'Password'`*:
  Das Passwort ist nicht notwendigerweise in PostgreSQL erforderlich (Abhängig von der Konfiguration der Datei pg_hba.conf). Wenn ein Passwort bei der Einrichtung eines neuen Benutzers zugewiesen ist, muss es bei der Anmeldung zur Datenbank mit der Benuterkennung zusammen eingegeben werden.

- `CREATEDB | NOCREATEDB`:
  Wird die Option `CREATEDB` spezifiziert, wird das Systemrecht zur Erzeugung neuer Datenbanken sowie Entfernung der von diesem Benutzer eingerichteten Datenbanken an diesen Benutzer vergeben. Mit der explizierte Eingabe von `NOCREATEDB` wird das Recht wieder entfernt. Ohne Spezifizierung dieser Option impliziert die Eingabe von `NOCREATEDB` die Defaulteingabe.

- `CREATEUSER | NOCREATEUSER`:
  Bei Spezifizierung des Kennworts `CREATEUSER` wird das Systemrecht zur Einrichtung neuer Benutzer an diesen Benutzer zugeordnet. Dies impliziert die Entstehung eines neuen Superusers, d.h. dieser Benutzer hat alle Systemrechte und Zugriffsrechte auf jedes Datenbankobjekt in den sämlichen Datenbanken und zwar sogar das Recht, neue Datenbanken einzurichten, selbst wenn die Option `NOCREATEDB` spezifiziert ist. Da ein Superuser die Rechte aller anderen Superuser aufheben, sogar die anderen Superuser aus der Datenbank enfernen kann, muss es sehr vorsichtlich mit der Rechtzuordnung von `CREATEUSER` umgegangen werden. Wird `NOCREATEUSER` explizite eingegeben, entsteht nach der Ausführung des `CREATE USER` Befehls ein normaler Benutzer. Ohne Eingabe über diese Option ist `NOCREATEUSER` als Standardwert anzusehen.

- `IN GROUP` *Gruppename* `[, ...]`:
  Die Gruppenmitgliedschaft kann auch bei Einrichtung der neuen Benutzer bestimmt werden. Jeder Benutzer kann zu einer oder mehrerer Gruppen gehören. Mehrere Gruppennamen sollen durch Komma getrennt werden, dabei aufzupassen ist, dass die eingegebenen Gruppen bereits in der Datenbank existieren müssen.

- `VALID UNTIL` *'Zeitspanne'*:
  Mit Eingabe einer Zeitspanne wird die Gültigkeit des Passworts zeitlich begrenzt. d.h.nach diesem Zeitraum läuft das Passwort ab und muss mit einem Neuen geändert werden.

2.1.2 Benutzeränderung

Durch den SQL-Befehl `ALTER USER` kann der Zustand eines Datenbankbenutzers geändert werden. Diser Befehl hat folgende Syntax:

```
ALTER USER Kennung
[ WITH [ PASSWORD 'Password' ] ]
[ CREATEDB | NOCREATEDB ]
[ CREATEUSER | NOCREATEUSER ]
[ VALID UNTIL 'Zeitspanne' ]
```

Wie `CREATE USER` kann der Befehl `ALTER USER` nur von PostgreSQL Superusern ausgeführt werden.

2.1.3 Benutzerentfernung

Jeder Superuser in PostgreSQL kann durch den Befehl `DROP USER` die Datenbankbenutzer wieder entfernen. Wenn ein Benutzer eigne Datenbanken erstellt hat, müssen diese Datenbanken zuerst gelöscht werden, bevor dieser Benutzer aus dem System entfernt wird. Der Befehl hat folgende Syntax:

`DROP USER` *Kennung*

2.2 Gruppenmanagement

Gruppen dienen zur Vereinfachung der Zuordnung von Objektrechten. Normalerweise müssen die Zugriffsberechtigungen an jeden einzelnen Benutzer vergeben werden, und jedes Mal kann höchstens nur ein Benutzer bearbeitet werden. Wenn einige Datenbankbenutzer die gleichen Zugriffsberechtigungen haben sollen, muss die Rechtevergabe für jeden Benutzer wiederholt ausgeführt werden. Falls eine Änderung der Datenbank (z.B. Hinzufügen neuer Tabellen) ebenso eine Änderung der Benutzerberechtigungen notwendig macht, muss dieser Arbeitsvorgang wieder für jeden Benutzer wiederholt werden. Dies führt nicht nur zum hohen Arbeitsaufwand, sondern auch leicht zur Vernachlässigkeit der Zugriffsberechtigungen für die einzelnen Benutzer.

Um diese Probleme zu vermeiden, werden Gruppen in PostgreSQL eingeführt. Alle Datenbankbenutzer, die gleiche Berechtigungen haben sollen, können in einer Gruppe zusammengefasst werden. Die benötigten Zugriffsberechtigungen werden nicht mehr an jeden einzelnen Benutzer erteilt, sondern direkt an die Gruppe einmal vergeben. Alle Gruppenmitglieder haben dann die entsprechenden Berechtigungen. Wenn sich die Datenbank ändert, und damit eine Berechtigungsänderung verursacht ist, wird nur die Erneuerung der Berechtigungen von dieser Gruppe benötigt. Entsteht ein neuer Benutzer, der zu einer Gruppe gehören kann, wird dieser einfach in diese Gruppe hinzugefügt, anschließend hat diser Benutzer automatisch alle Berechtigungen, die bereits an diese Gruppe vergeben sind.

Wie bei Benutzerverwaltung werden alle Gruppeninformationen in einer Systemtabelle, die `pg_group` genannt wird, gespeichert. In den Spalten `groname` und `grosysid` sind alle Gruppennamen und die zugehörigen GruppenID enthalten und unter `grolist` werden die BenutzerID aller Gruppenmitglieder gespeichert.

2.2.1 Gruppeneinrichtung

Jede neue Gruppe in PostgreSQL kann durch Superuser mit dem Befehl `CREATE GROUP` eingerichtet werden. Hier ist die Syntax:

Attribut	Typ
groname	Text
grosysid	Integer
grolist	Text

Tabelle 2.2: Schema der Tabelle pg_group

`CREATE GROUP` *Gruppenname*
`[ WITH [ SYSID` *SystemID* `][ USER` *Benutzerkennung* `[, ...] ]`

Der Befehl `CREATE GROUP` *Gruppenname* erstellt eine leere Gruppe ohne Mitglieder. Mit der optionalen Eingabe `SYSID` *SystemID* wird eine SystemID für diese Gruppe erzeugt. Da PostgreSQL die Systemtabellen von Benutzern und Gruppen separat verwahltet, ist es erlaubt, dass eine SystemID für Benutzer mit einer SystemID für Gruppe identisch ist. Bei der Erzeugung einer neuen Gruppe kann die Gruppenmitgliedschaft gleichzeitig bestimmt werden, indem man unter Verwendung des Kennworts `USER` die Kennungen, dessen Benutzer zu dieser Gruppe gehören sollen, durch Komma getrennt eingibt.

2.2.2 Gruppenänderung

Durch den SQL Befehl `ALTER GROUP` können Benutzer in eine Gruppe hinzugefügt oder aus einer Gruppe entfernt werden. Die Syntax hat folgende Form:

`ALTER GROUP` *Gruppename* { `ADD | DROP` } `USER` *Benutzerkennung* `[, ...]`

In der Abhängigkeit davon, ob das Kennwort `ADD` oder `DROP` spezifiziert ist, werden die eingegebenen Benutzer in die gewünschte Gruppe hinzugefügt oder die entsprechenden Gruppenmitglieder entfernt.

2.2.3 Gruppenentfernung

Jeder Superuser kann mit dem Befehl `DROP GROUP` eine Gruppe entfernen. Eine Gruppe kann ohne Warnung gelöscht werden, selbst wenn sie noch Gruppenmitglieder enthält, die noch nicht aus der Gruppen entfernt sind. Hier ist die Syntax:

`DROP GROUP` *Gruppename*

Bei der Entfernung einer Gruppe werden ihre zugehörigen Berechtigungen nicht mitgelöscht. Alle Zugriffsberechtigungen für die bereits entfernte Gruppe werden an die SystemID, die zu der Gruppe vor Entfernung gehört, zugeordnet. Eine unabsichtlich gelöschte Gruppe kann deshalb wieder hergestellt werden, indem man eine neue Gruppe

Schlüsselwort	**Symbol**	**Beschreibung**
SELECT	r	Erlaubt den Berechtigungen, Tabellen, Sichten, Sequenzen abzufragen.
INSERT	a	Erlaubt den Berechtigungen, neue Zeilen in Tabellen einzufügen.
UPDATE, DELETE	w	Erlaubt den Berechtigungen, Zeilen in Tabellen zu aktualisieren oder aus Tabellen zu löschen. Wenn eine der beiden Berechtigungen vergeben ist, ist die andere implizite auch vergeben.
RULE	R	Erlaubt den Berechtigungen, eine wiederschreibbare Rule auf Tabellen oder Sichten zu erstellen.
ALL	arwR	Repräsentiert alle vier Berechtigungen.

Tabelle 2.3: Privilegien in PostgreSQL[WD01]

mit der identischen SystemID von der zuvor Gelöschten einrichtet.

2.3 Zugriffsberechtigungen

PostgreSQL verwaltet intern eine Menge von Zugriffskontrollisten, die sogenannte Access Control Lists (ACL)[WD01]. Diese Informationen beschreiben den Zustand der Zugriffsberechtigungen in einer Datenbank, nämlich welchen Benutzern oder Gruppen ist es erlaubt, mit welchen Privilegien auf die Datenbankobjekte zugreifen zu können. Die Datenbankobjekte in PostgreSQL, auf die eine Zugriffsberechtigung vergeben werden kann, sind z.B. Tabellen (tables), Sichten (views) und Sequenzen (sequences). Es gibt vier Typen von Privilegien: SELECT (read), INSERT (append), UPDATE/DELETE (write) und RULE (eine Bereichtigung zur Erstellung einer wiederschreibbaren Rule auf eine Tabelle). Die detailierte Beschreibung wird in Tabelle 2.3 angezeigt.

Ein Benutzer ist dann ein Besitzer in PostgreSQL, wenn er ein Datenbankobjekt (z.B. eine Tabelle) erstellt hat. Ein Besitzer hat volle Rechte auf alle von ihm erstellten Objekte. Damit die anderen Benutzer den Zugriff auf diese Objekte erhalten, muss der Besitzer oder ein Superuser die entsprechenden Privilegien durch den Befehl `GRANT` an sie zuordnen. Mit dem Befehl `REVOKE` werden die vergebenen Privilegien wieder entfernt.

2.3.1 GRANT Privilegien

Der Befehl `GRANT` hat folgende Syntax:

```
GRANT privilege [, ...] ON object [, ...]
TO { PUBLIC | username | GROUP groupname }
```

Mit dem Schlüsselwort `PUBLIC` werden die Privilegien an alle Benutzer in der Datenbank erteilt. Bei Verwendung des Schlüsselworts `ALL` werden alle vier Privilegien für die spezifizierten Objekte gleichzeitig vergeben. Die vergebenden Privilegien auf jedes Datenbankobjekt werden in einer PostgreSQL Systemtabelle (pg_class) gespeichert. Durch einen SQL Befehl `SELECT` kann der Status der Zugriffsberechtigungen für das gewählte Objekt angefragt werden. Von dem *psql* Client kann man auch als Alternative den Backslash Befehl \z benutzen, um die Berechtigungsinformation für die Datenbankobjekte zu erfahren. Im folgenden wird eine Verwendung von Backslah Befehl gezeigt.

```
  mnm=# \z aufgabe
  Access permissions for database "mnm"
  Relation   |       Access permissions
------------+-------------------------------------
  aufgabe    | {"=","wangz=rw","group mnmteam=arwR"}
  (1 row)
```

In diesem Beispiel werden alle Zugriffsrechte auf die Tabelle `aufgabe` als Ergebnis angezeigt. Die linke Seite von dem Zeichen „=" in jedem Term repräsentiert einen Benutzer oder eine Gruppe, auf der rechten Seite stehen die entsprechenden Rechte, mit denen der Benutzer oder die Gruppe auf diese Tabelle zugreifen kann. Der erste Term wird immer für das Schlüsselwort `PUBLIC` (d.h. für alle Benutzer in der Datenbank) reserviert. Werden keine Zugriffsrechte an alle Benutzer gemeinsam erteilt, bleibt die rechte Seite auch leer.

2.3.2 REVOKE Privilegien

Der Befehl `REVOKE` hat folgende Syntax:

```
REVOKE privilege [, ...] ON object [, ...]
FROM { PUBLIC | username | GROUP groupname }
```

Die Struktur von dem Befehl `REVOKE` ist identisch wie `GRANT`, mit der Ausnahme, dass hier anstelle `TO` das Schlüsselwort `FROM` eingesetzt wird. Obwohl das Schlüsselwort

`PUBLIC` an dieser Stelle die gleiche Bedeutung wie beim `GRANT` hat, hat die Verwendung von dem Befehl `REVOKE` auf `PUBLIC` keine Auswirkungen auf der jenigen Benutzern, dessen Zugriffsberechtigungen explizite an sie durch `GRANT` vergeben wurden.

2.4 Realisierung der Datenbanksicherheit

Die Datenbanksicherheit in PostgreSQL umfasst das Management von Benutzern, Gruppen und Zugriffsberechtigungen. SQL bietet verschiedene Befehle, um Sichterheismechanismen in PostgreSQL zu realisieren.

Mit dem Befehl `CREATE USER` legt man einen Benutzer neu an und richtet damit einen Benutzernamen und ein Passwort für den Benutzer ein, gleichzeitig hat auch die Möglichkeit, den Benutzer in eine oder mehrerer bereits bestehenden Gruppen hinzuzufügen. Um den Zustand eines Datenbankbenutzers zu verändern (z.B. das Passwort zu ändern), wird der Befehl `ALTER USER` verwendet. Der Befehl `DROP USER` entfernt einen in der Datenbank schon exitierenden Benutzer wieder.

Eine Gruppe in PostgreSQL kann mit dem Befehl `CREATE GROUP` eingerichtet werden. In einer Gruppe werden die jenigen Datenbankbenutzer aufgenommen, die nach gewissen Kriterien zusammengehören können. Mit dem Befehl `ALTER GROUP` kann man neue Benutzer in eine Gruppe hinzufügen oder die Gruppenmitglieder entfernen. Der Befehl `DROP GROUP` löscht eine Gruppe.

Damit ein Benutzer überhaupt Arbeiten in der Datenbank ausführen kann, werden ihm die entsprechenden Zugriffsberechtigungen auf den Datenbankobjekten durch den Befehl `GRANT` gegeben. Die Objekte in PostgreSQL, auf die eine Zugriffsberechtigung vergeben kann, sind z.B. Tabellen, Sichten und Squenzen. Der Befehl `REVOKE` entzieht einem Benutzer oder einer Gruppe diese Berechtigungen wieder. Es gibt zwei Typen von Privilegien: Systemprivilegien und Objeteprivilegien. Die Systemprivilegien sind diejenigen Rechte, die einem Benutzer erlaubt, neue Datenbanken einzurichten oder neue Benutzer in die Datenbank hinzuzufügen, die Objekteprivilegien in PostgreSQL umfassen die vier Zugriffsberechtigungen(SELECT, INSERT, UPDATE/DELETE und RULE) auf unterschiedlichen Datenbankobjekten. Die Systemprivilegien können nur von einem Superuser erteilt werden, während die Objekteprivilegien von einem Superuser oder dem Besitzer (Owner) dieser Objekte in PostgreSQL vergeben können.

Nach dem Sichterheitskonzept unter PostgreSQL sollen alle Benutzer wie in Abbildung 2.2 gezeigt verwaltet werden. An jeden Datenbankbenutzer werden entsprechende Zugriffsberechtigungen auf bestimmte Objekte zugeordnet werden. In einer Gruppe haben alle Mitglieder gemeinsam die Zugriffsrechte, die direkt an diese Gruppe erteilt werden. Die Gruppenmitglieder können nicht nur die gemeinsamen als auch individuelle Zugriffsrechte erhalten. Ein Beispiel in der Praxis dafür: Während der Vorlesungszeit gibt es im Lehrstuhl mehrere Übungsbetriebe für entsprechende Vorlesungen, für eine Vorlesung (z.B. Info3) bilden dann die Tutoren als Benutzer eine eigne Gruppe in der Datenbank. Alle Tutoren in diese Gruppe bekommen den Zugriff auf die Objekte (wie z.B. Übungsaufgaben, Lösungen), die während der Übungsbetreuung von Bedeutung sind. Gleichzeitig erhält jeder Übungsleiter noch zusätzlich individuellen Zugriff auf die Information über die Übungsgruppe, die direkt von ihm betreut wird.

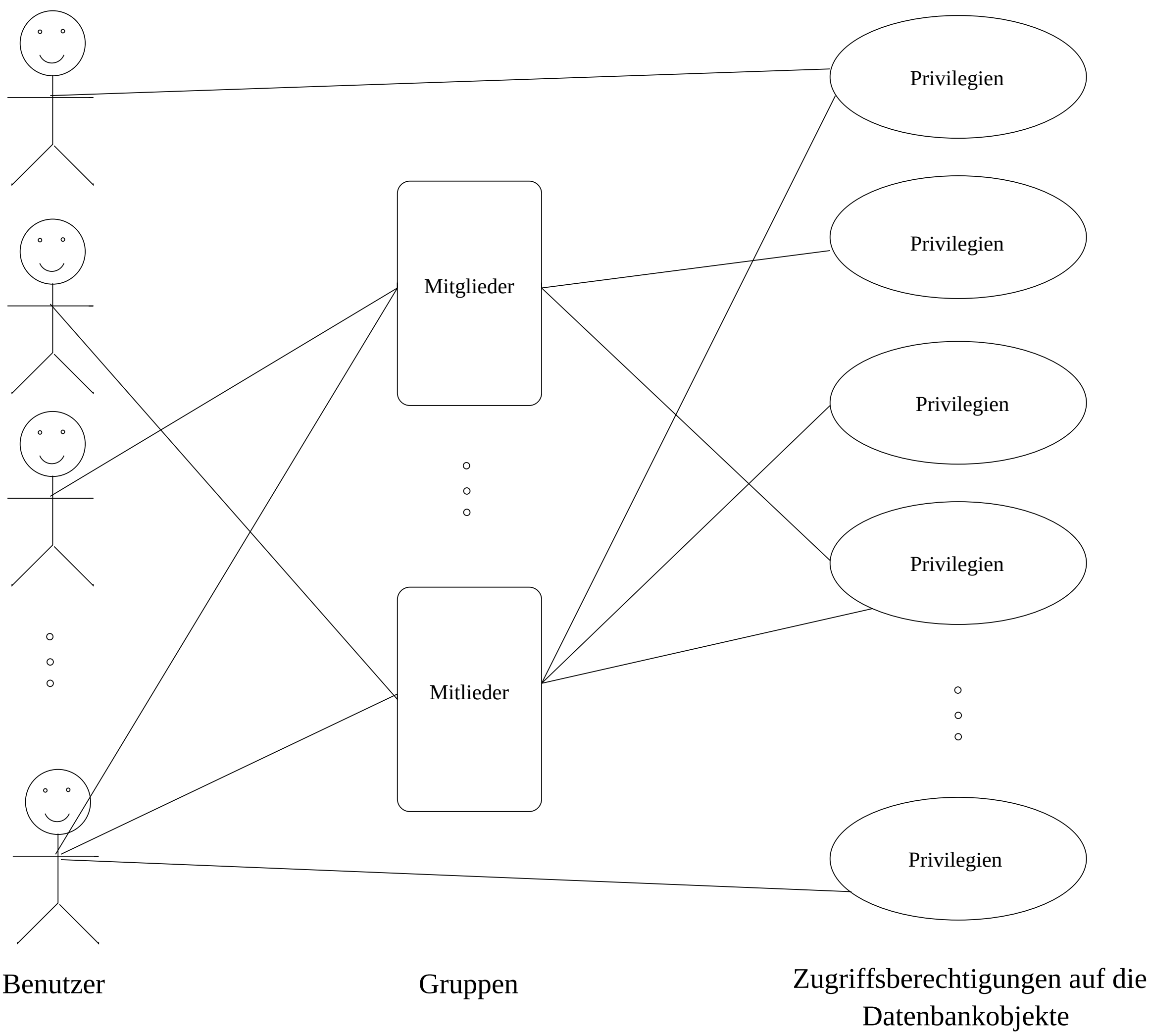

Abbildung 2.1: Sichterheitskonzept

Kapitel 3

Entwurf eines Benutzerverwaltungssystems

Im Rahmen dieses Praktikums soll ein Benutzerverwaltungssystem entwickelt werden, das die notwendigen Verwaltungsaufgaben nicht nur auf der Ebene der Datenbank, sondern auch im Zusammenspiel mit dem Web-Server erledigen kann.

3.1 Verwaltungsaufgaben

Die datenbankspezifischen Verwaltungsaufgaben wurden in Kapitel 2 bereits beschrieben. Es handelt sich dabei um die folgende drei Aspekte:

- Verwaltung der Benutzer

- Zuordnung von Zugriffsrechten an die Benutzer

- Umsetzung dieser Zuordnung

Im Zusammenspiel mit dem Web-Server sollen folgende Aufgaben erfüllt werden:

- Zertifikatausstellung
  Alle Informationen, die nicht öffentlich zugänglich sein sollen, werden über das HTTPS-Protokoll verschlüsselt übertragen. Um auf solche Daten zuzugreifen, erhält jeder Mitarbeiter des Lehrstuhls sowie Befugte ein Zertifikat.

- Speicherung der Benutzerinfodaten
  Zur Anmeldung an der Datenbank werden gewisse Benutzerdaten benötigt. Diese Daten werden in der Datei `.dbmap` gespeichert. Für jeden Benutzer ist eine Zeile in dieser Datei reserviert und jede Zeile besteht aus drei Teilen, nämlich dem Name des Benutzers (Vor- und Nachname), der Benutzerkennung und dem Passwort.

- Verbindungsaufbau für die Datenbankanmeldung
  Der Verbindungsaufbau von dem Web-Server zum Datenbankserver wird von den entsprechenden Funktionen in PHP-Skript übernommen.

3.2 Anmeldung an der Datenbank

Jeder Benutzer in der Datenbank kann prinzipiell zwei Kennungen haben.

- Datenbankkennung(DB-Kennung)
  Diese Kennung ist normalerweise für Mitarbeiter des Lehrstuhls der Login am Unix-Rechner, für Studenten die Cip-Kennung.
- WWW-Kennung
  Die WWW-Kennung besteht aus der Datenbankkennung des Benutzers und einer gefolgten Zeichenkette "_www" .

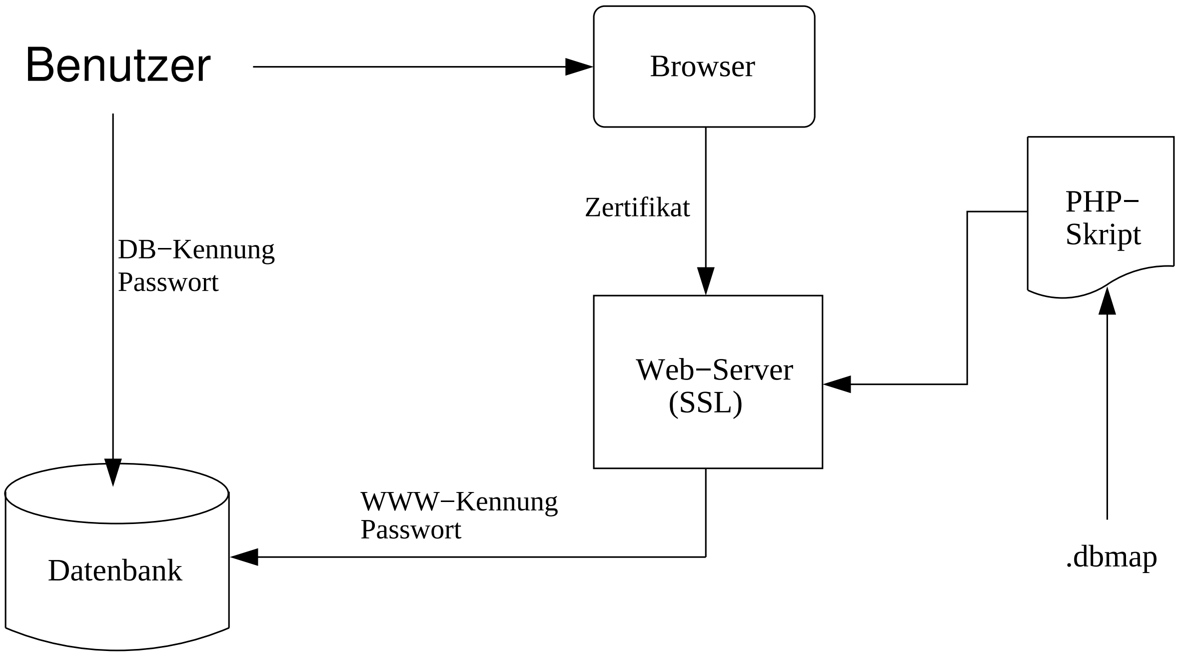

Abbildung 3.1: Datenbankanmeldung

Das Anmeldungsverfahren wird in der Abbildung 3.1 angezeigt. Mit der Datenbankkennung und einem dazugehörigen Passwort meldet man sich direkt an der Datenbank an, um dort zu arbeiten.

Für den Zugriff auf WWW wird die WWW-Kennung benötigt. Wie schon erwähnt, um auf den sicheren Web-Server (SSL) zugreifen zu können, bekommt jeder befugten Benutzer in der Datenbank ein Zertifikat. Zum Zweck der Authentifizierung wird das Zertifikat zuerst von dem Web-Server überprüft. Bei gültigem Zertifikat wird das zuständige PHP-Skript aufgerufen. Dies liest die Infodatei `.dbmap` zeilenweise ein, und findet diejenige Zeile, in der der Benutzername (Vor- und Nachname) mit dem auf Zertifikat stehenden Namen übereinstimmt. Die in dieser Zeile liegende Kennung und das Passwort werden dann aus der Datei geholt und als Parameter der Verbindungsfunktion in PHP übergeben. Diese Funktion erstellt anschließend die Verbindung zum Datenbankserver mit dieser Kennung und dem Passwort. Anders als die direkte Anmeldung braucht man in diesem Fall die WWW-Kennung und das Passwort überhaupt nicht zu bemerken. Alle Anmeldungsschritte werden von dem Web-Server und den PHP-Funktionen übernommen.

3.3 Herkömmliche Arbeitsvorgänge

Bisher muss man die Benutzerverwaltungsaufgaben in mehreren Arbeitsvorgängen erledigen. Ein Beispiel für die Einrichtung von neuen Benutzer:

- direkte Anmeldung an der Datenbank
  Man meldet sich zuerst an der Datenbank an, um die datenbankspezifischen Verwaltungsaufgaben zu erfüllen.

- Benutzerkennungeinrichtung und Gruppenzuordnung
  Dies geschieht, indem man die SQL-Befehle `CREATE USER` sowie `CREATE GROUP` in der Datenbank ausführt.

- Erteilung von Zugriffsberechtigungen
  Durch den Befehl `GRANT` vergibt man die entsprechenden Zugriffsrechte auf bestimmte Datenbankobjekte an den Benutzer.

- Zertifikatausstellung
  Dazu muss man aus der Datenbank auskommen, und die entsprechenden Programme für die Zertifikatausstellung aufrufen.

- Aktualisierung der Infodatei `.dbmap`
  Zum Schluss aktualisiert man die Datei `.dbmap` mit einer neuen Zeile von dem Eintrag "Benutzerkennung:Vor- und Nachname des Benutzers:Passwort". Der Benutzername hier muss mit dem auf Zertifikat stehenden Namen identisch sein.

Die Nachteile von solcher Arbeitsweise sind leicht zu erkennen:

- manuelle Eingabe erforderlich
  Alle Benutzerverwaltungsaufgaben werden in diesem Fall noch manuell in der Datenbank ausgeführt. Dazu muss man zuerst die entsprechenden SQL Befehle mit sämtlichen Parametern sowie den notwendigen Optionseingaben formulieren, und dann auf dem PostgreSQL Client *psql* ausführen. Dies geschieht entweder durch direkte Texteingabe oder durch den Einsatz von einem SQL-Skript für jeden Arbeitsvorgang. Ein Beispiel dafür ist ein im Betrieb eingesetztes SQL-Skript **grant.mnmteam**, das bewirkt, dass sämtliche Mitglieder der Gruppe **mnmteam** sowohl alle vier Zugriffsrechte(SELECT, INSERT, UPDATE/DELETE, RULE) auf alle Tabellen als auch Lesezugriffe(SELECT) auf alle Sichten(views) in der Datenbank erhalten.

```
GRANT ALL ON arbeitet_fuer_praktikum TO GROUP mnmteam;
GRANT ALL ON arbeitet_fuer_vorlesung TO GROUP mnmteam;
GRANT ALL ON aufgabe TO GROUP mnmteam;
GRANT ALL ON calls TO GROUP mnmteam;
GRANT ALL ON diplomarbeit TO GROUP mnmteam;
GRANT ALL ON doktorandenkolloquium TO GROUP mnmteam;
GRANT ALL ON einreichung TO GROUP mnmteam;
GRANT ALL ON fopra TO GROUP mnmteam;
GRANT ALL ON gruppe_hat_ausarbeitung_abgeg TO GROUP mnmteam;
GRANT SELECT ON termine_wwwpublic TO GROUP mnmteam;
GRANT SELECT ON fopra_wwwpublic TO GROUP mnmteam;
GRANT SELECT ON diplomarbeit_wwwpublic TO GROUP mnmteam;
GRANT SELECT ON oberseminarterm_wwwpublic TO GROUP mnmteam;
GRANT SELECT ON mitarbeiter_wwwpublic TO GROUP mnmteam;
GRANT SELECT ON student_wwwpublic TO GROUP mnmteam;
GRANT ALL ON klausur TO GROUP mnmteam;
GRANT ALL ON klausur_in_raum TO GROUP mnmteam;
GRANT ALL ON konferrenz TO GROUP mnmteam;
GRANT ALL ON mailadressenmitarbeiter TO GROUP mnmteam;
GRANT ALL ON mailadressenstudent TO GROUP mnmteam;
.
.
.
.
.
.
GRANT ALL ON vorlesung_vorlesungnummer TO GROUP mnmteam;
```

  Wie oben angezeigt wird, muss für jede einzelne Tabelle und Sicht in dieser Datenbank ein entsprechender `GRANT` Befehl manuell in diesem Skript eingetippt werden. In diesem Fall sind die Formulierungen, die Aktualisierungen und die Wartungen der Benutzerverwaltung sehr aufwendig durchzuführen.

- Wechsel zwischen Systemen
  Bei der Arbeit muss man immer zwischen Systemen wechseln. Zuerst arbeitet man in der Datenbank und dann kommt aus der Datenbank aus, um die entsprechenden Programme für Zertifikatausstellung auszuführen und anschließend aktualisiert die Datei `.dbmap` mit einem Texteditor.

- separate Ausführung
  Alle einzelnen Schritte sind ohne Berücksichtigung der Ergebnisse von anderen Arbeitsschritten separat ausgeführt. Dies kann dazu führen, dass Inkonsistenzen entstehen.

3.4 Aufbau des Benutzerverwaltungssystems

Um den gesamten Arbeitsaufwand zu verringern, ist ein Benutzerverwaltungssystem zu implementieren, das die sämtlichen Verwaltungsaufgaben auf der Ebene der Datenbank im Zusammenspiel mit dem Web-Server einfach und konsistent erledigen kann. Damit man die Anbindung der Datenbank an den Web-Server erstellen sowie die unterschiedlichen Formulardaten auf dem Web-Server auswerten kann, wird die Skriptsprache **PHP** zum Einsatz gebracht. Das Auswahlverfahren ist dem vorgängigen Fopra[BM01] zu entziehen.

3.4.1 Einsatz von PHP

PHP ist eine auf der Serverseite ablaufende, plattformunabhängige Skriptsprache. Sie wird direkt im HTML-Code untergebracht. Durch die Dateierweiterung wie .PHP, .PHP3, oder PHP4 entsteht eine PHP-Skriptdatei.

Das vom Client geforderte PHP-Skript wird von dem PHP-Engine, der im Apache-Server als Modul betrieben wird, interpretiert. Enthält das Skript SQL-Befehle, erstellt PHP eine Verbindung zum Datenbankserver und schickt dadurch die SQL-Befehle an die Datenbank. Dort werden diese Befehle ausgeführt, und die zugehörigen Ergebnisse werden anschließend durch diese Verbindung wieder an dem PHP-Engine zurückgeliefert. Dort werden sie als HTML-Code in die Seite eingebaut und die fertige Seite wird dem Webserver übergeben, der sie als reines HTML dem Browser zurücksendet.

3.4.2 Auswertung der Formulare mit PHP

Ein Formular besteht aus unterschiedlichen Elementen wie zum Beispiel Textfelder zur direkten Texteingabe, Listenfelder zum Einträgeauswählen und Buttons zum Anklicken.

Ein HTML-Formular stellt die Möglichkeit dar, die vom Nutzer durch Eingeben, Auswählen, und Anklicken gesammelten Daten zum Web-Server zu übertragen und damit zu dem zuständigen PHP-Skript zu gelangen. In HTML werden die Formulare mittels `<FORM>`-Tag definiert:

```
<FORM ACTION="URl" METHOD="GET"|"POST">

... Formularinhalte ...

</FORM>
```

Das URL vom Attribut `ACTION` ist in unserem Fall das Zielskript auf dem Web-Server, das die übertragenen Formulardaten auswertet bzw. weiterleitet. Die Übertragung der Daten zum Web-Server übernimmt HTTP mit den Methoden GET oder POST. HTTP-Nachrichten enthalten drei Bestandteile, das URI, das Kommando und die Nachricht. Die Nachricht besteht aus Header und Body(der Nachrichtinhalt). Die Methode GET nutzt das URI zur Übertragung, hängt die Daten an die URL mit an, die Methode POST hingegen verfasst die Daten im Bodyteil einer Nachricht. Jedes Formularelement ist durch die Attribute **name** und **value** als Wert gekennzeichnet. Im PHP-Skript wird der Name eines Formularelements als eine Variable und der Value als den entsprechenden Wert dieser Variable ausgewertet.

Ab der aktuellen Version von PHP(4.1.0) hat der PHP-Team[php] aus Sicherheitsgründen empfohlen, die Einstellung von *register_globals* in der Konfigurationsdatei `php.ini` auf *"off"* als Standardwert zu setzen, damit alle Formularelemente, die sowohl durch GET-Methode als auch durch POST-Methode übertragen wurden, nicht mehr direkt mit dem Elementnamen als Variable in PHP-Skript ansprechbar sind. Um sie in Anspruch nehmen zu können, bietet der PHP-Team einige neue spezielle Variablen:

- $_GET[]
  In dieser Arrayvariable sind alle Formularvariablen enthalten, die durch die Methode **GET** übertragen sind.

- $_POST[]
  Diese Arrayvariable enthält alle Formularvariablen, die durch die Methode **POST** gesendet sind.

- $_REQUEST[]
  Das ist eine Zusammenfassung von den GET-Variablen, den POST-Variablen sowie den Cookie-Variablen.

Diese speziellen Variablen sind assoziative Arrayvariablen. Die Formularelemente sind nur noch erreichbar, indem man den Elementnamen als Schlüssel gemäß der Übertragungsmethode, die Variable `$_GET[]` oder `$_POST[]` einsetzt. Um das im folgenden Beispiel angezeigte Inputelement im PHP-Skript auswerten zu können, verwendet man nicht wie gewohnt die Variable `$username`, sondern die Variable `$_POST["username"]` oder `$_REQUEST["username"]`.

```
<FORM ACTION="ziel.php" METHOD="POST">
<INPUT TYPE="TEXT" NAME="username" VALUE="loginname">
... andere Formularelemente ...
</FORM>
```

Diese Umstellung führt dazu, dass man die Formularvariablen für alle bisherigen Anwendungen entsprechend ändern soll. Diejenigen Anwendungen, die nur bei der Einstellung von `register_globals` auf **"on"** ablaufen können, sogar nicht mehr funktionieren. Dieser Fopra hat diese Umstellung inzwischen erlebt, und dies hat einen großen zusätzlichen Arbeitsaufwand verursacht. Laut dem PHP-Team sind die alten globalen Variablen wie beispielsweise `$HTTP_GET_VARS` oder `$HTTP_POST_VARS` zwar noch einsetzbar, aber es lohnt sich, die neuen Variablen zu benutzen.

3.4.3 Architektur

Die Verwendung von Formularen ermöglicht, die für die Erledigung der Benutzerverwaltungsaufgaben notwendigen Daten vom Client zum Web-Server zu übertragen. Durch den Einsatz von PHP-Skripten werden nicht nur die Kommunikation zwischen dem Datenbankserver und Web-Server hergestellt, sondern auch die vom Client zum Web-Server geschickten Daten auf dem Server ausgewertet und in den entsprechenden SQL-Befehlen für die Benutzerverwaltung formuliert.

Die Realisierung des Benutzerverwaltungssystems wird in Abbildung 3.2 dargestellt. Alle Benutzerverwaltungsaufgaben wie zum Beispiel Einrichtung von neuen Benutzern, Zuordnung der Gruppenmitgliedschaft, Erteilung der Zugriffsberechtigungen usw. werden zuerst in den unterschiedlichen Formularen formuliert, die wiederum in den entsprechenden PHP-Skripten eingeblendet werden, damit man mit Hilfe von Browser die für die Ausführung dieser Aufgaben notwendigen Parameter sowie die entsprechenden Optionseingaben durch Eintippen, Auswählen, und Anklicken zusammenfassen kann. Anschließend werden die gesammelten Daten an den Web-Server geschickt und dort von den PHP-Skripten bearbeitet und zum nächsten Schritt weitergeleitet. Werden inzwischen die Informationen aus der Datenbank gebraucht, besorgt die Verbindungsfunktion in PHP eine Verbindung zum Datenbankserver, fordert die benötigten Daten mit entsprechenden SQL-Befehlen und übergibt diese an die kommenden zwischenformulare. Wenn alle Formulare für einen Arbeitsvorgang durchgearbeitet werden, formuliert PHP die endgültigen SQL-Befehle, die die unterschiedlichen Benutzerverwaltungsfunktionen darstellen und sie an die Datenbank schicken. Werden diese Befehle dort erfolgreich ausgeführt, dann endet dieser Arbeitsvorgang.

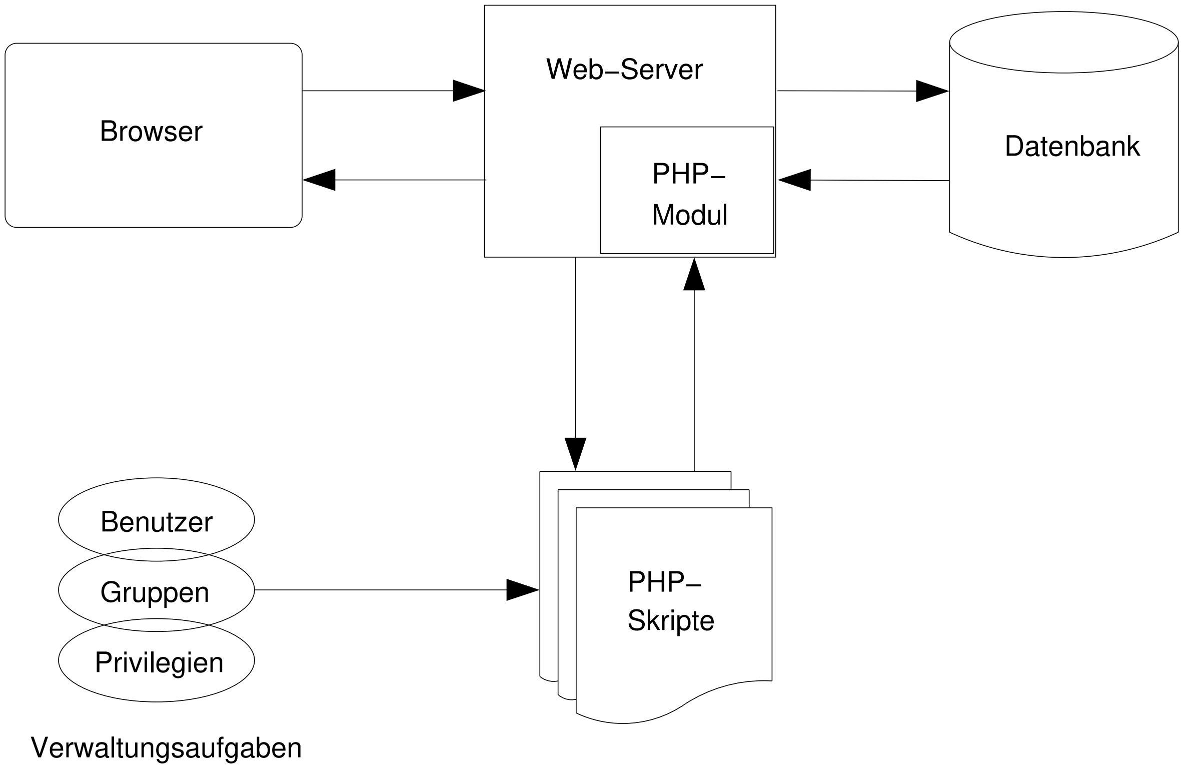

Abbildung 3.2: Architektur

Kapitel 4

Implementierung

4.1 Übersicht des Verwaltungssystems

Durch den Einsatz von PHP-Skripten wurde das Benutzerverwaltungssystem, das in Abbildung 4.1 angezeigt wird, implementiert.

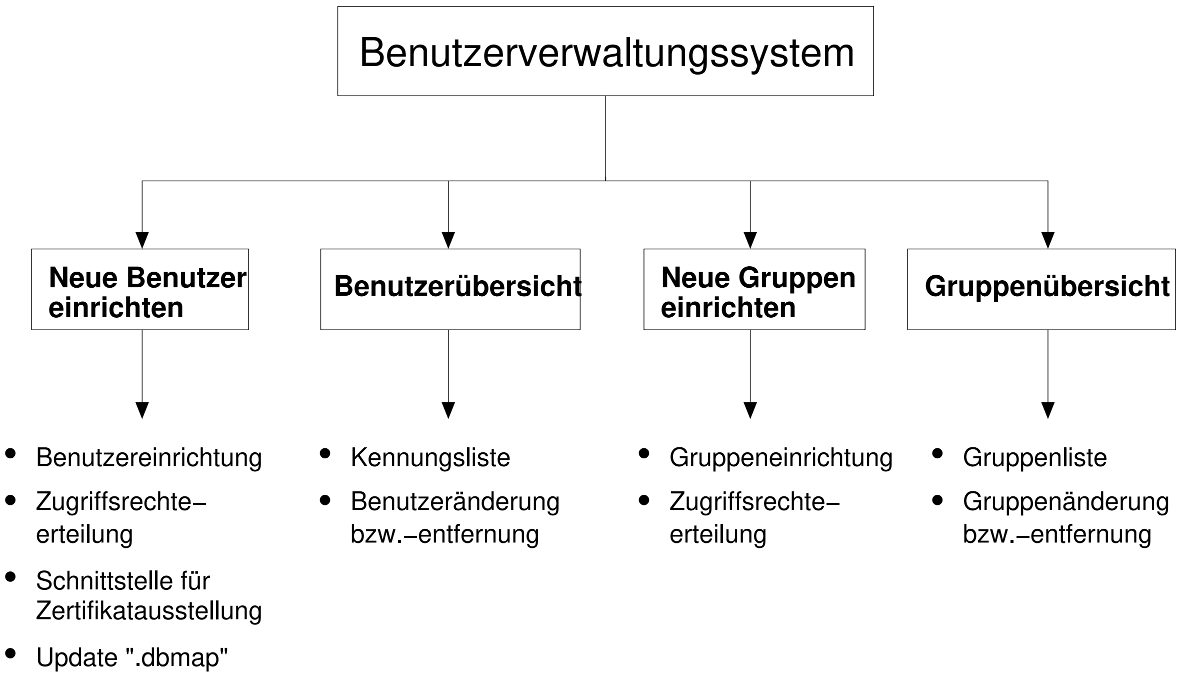

Abbildung 4.1: Verwaltungssystem

Alle Benutzerverwaltungsaufgaben werden sowohl auf der Ebene der Datenbank als auch im Zusammenspiel mit dem Web-Server in einem System zusammengefasst. Die einzelnen Verwaltungsaufgaben werden in verschiedenen PHP-Skripten formuliert. Durch den Aufruf einer Folge von bestimmten PHP-Skripten werden die entsprechenden Verwaltungsaufgaben durchgeführt.

In diesem Verwaltungssystem stellen die vier Links (Neue Benutzer einrichten, Benutzerübersicht, Neue Gruppen einrichten, Gruppenübersicht) als Zugänge dar, die zur Ausführung der unterschiedlichen Verwaltungsaufgaben führen.

Um neue Benutzer einzurichten, klickt man auf den ersten Link an. Alle darunter liegenden Skripte werden anschließend nacheinander aufgerufen. Für die Benutzereinrichtung und Zugriffsrechteerteilung werden zuerst unterschiedliche HTML-Formulare angezeigt. In diesen Formularen kann man alle notwendigen Optionseingaben für die Ausführung von SQL-Befehlen in der Datenbank bestimmen und diese Daten zum kommenden Skript abschicken. Die abgeschickte Eingabedaten werden dann in diesem Skript ausgewertet und in den SQL-Befehlen umwandelt. Solche SQL-Befehle werden durch die Verbindung nach Datenbankserver gesendet und dort ausgeführt. Anschließend werden die Funktionen für die Zertifikatausstellung und die Aktualisierung der Datei `.dbmap` nacheinander aufgerufen.

Die `Benutzerübersicht` zeigt die Liste aller Datenbankbenutzer an. Hier werden die Benutzer mit den allgemeinen Attributen wie zum Beispiel dem Loginnamen, der Benutzeridnummer usw. nach Loginnamen sortiert. Wählt man einen Benutzer aus, gelangt man zum Detailübersicht für diesen Benutzer. Hier werden neben den allgemeinen Attributen auch alle Gruppen, zu denen dieser Benutzer gehört, angezeigt. Jeder Gruppenname stellt einen Link dar, der zum Detailübersicht für diese Gruppe führt. Im Benutzerdetailübersicht gibt es ein Formular `UserManager`. In diesem Formular kann man unterschiedliche Verwaltungsfunktionen auswählen, um den Benutzerzustand zu verändern.

Durch `Neue Gruppen einrichten` unter `GroupAktion` werden neue Gruppen mit den zugehörigen Benutzern erstellt und gleichzeitig die entsprechenden Zugriffsrechte auf unterschiedliche Datenbankobjekte an die Gruppe vergeben.

In `Gruppenübersicht` werden die sämtlichen Gruppen in der Datenbank und die zugehörigen Benutzernamen angezeigt. Jeder Gruppenname stellt einen Link dar, der genauso wie bei `Benutzübersicht` zur detailierten Darstellung dieser Gruppe führt. Im Detailübersicht werden alle Mitglieder der Gruppen tabellarisch angezeigt. Jeder Benutzername stellt wieder einen Link dar, der zum Deteilübersicht für diesen Benutzer führen kann. Ein Formular `GroupManager` steht hier zur Verfügung, damit man die gewünschten Verwaltungsfunktionen für diese Gruppe auswählen und anschließend sie ausführen kann.

In den folgenden Abschitten werden die vier Auswahlmöglichkeit in der Startseite detailiert beschrieben.

4.2 Einrichtung von neuen Benutzern

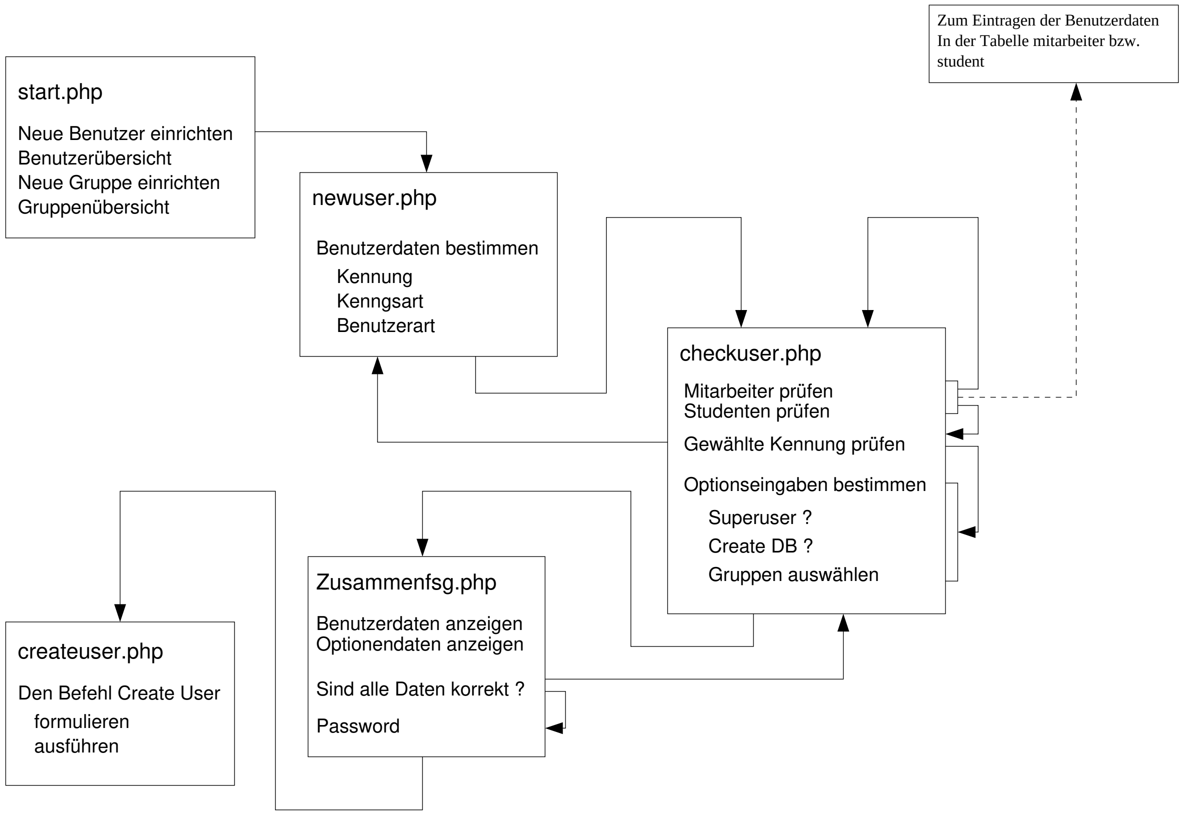

Abbildung 4.2: Einrichtung von neuen Benutzern

Wählt man in der Startseite die Option `Neue Benutzer einrichten`, wird das Skript `newuser.php` aufgerufen. In dieser Datei handelt es sich im wesentlichen ein Formular. In diesem Formular soll man die Kennung für den Benutzer eingeben. Diese Kennung ist nämlich die Datenbankkennung, mit der der Benutzer sich bei der Datenbank anmelden kann. Die WWW-Kennung entsteht durch die Eingabe der Datenbankkennung und gefolgt mit der Zeichenkette "_www". Nach der Festlegung der Kennung bestimmt man die Kennungsart durch eine Checkbox, d.h. welche Kennungen der Benutzer besitzen soll, die Datenbankkennung oder die WWW-Kennung oder die Beiden. Als Standardeinstellung wird es auf "beide "gesetzt. Am Schluss wählt man eine Benutzerart von Mitarbeiter und Student aus. Wenn es sich um einen Studenten handelt, soll man die Matrikelnummer eingeben. Das Button `Start` führt die Datenübertragung zum Skript `checkuser.php`.

Es gibt in der Datenbank zwei Tabellen `mitarbeiter` und `student`, die für die Verwaltung von Benutzerinfodaten zuständig sind. Die Informationen wie beispielsweise der Vorname, Nachname, die Arbeitskennung von Mitarbeitern, Cipkennung, Matrikelnummer von Studenten usw. für alle Mitarbeiter am Lehrstuhl und diejenigen Studenten, die einen Fopra oder Diplomarbeit an diesem Lehrstuhl gemacht haben, werden in der entsprechenden Tabelle gespeichert.

Das Skript `checkuser.php` prüft zuerst mit der Verbindung zur Datenbank, ob die Benutzerinfodaten schon in diesen Tabellen eingegangen sind. Wenn es nicht der Fall ist, unterbricht es das Programm, gibt eine Warnung und eine Eingabe des Links, der zur Eintragung dieser Daten führt, aus. Da es sich um eine andere Anwendung in der Datenbank handelt, wird ein neues Fenster eröffnet. Dort kann man durch entsprechende Tools die Benutzerinformation eintragen. Wenn man damit fertig ist, schließt man das Fenster und kehrt wieder zurück zu der Stelle, an der das Programm unterbrochen wurde. Durch das Anklicken des Buttons `Weiter` wird das Programm von sich selbst gestartet. Damit die von vorherigen Formular übertragenen Daten nicht verlierengehen, werden sie durch den Einsatz von versteckten(Hidden) Inputfeldern an sich neu mitgeschickt. Das Programm prüft nochmal die Infodaten. Falls sie jetzt in der Datenbank vorhanden sind, geht das Programm zum nächsten Schritt weiter.

In diesem Schritt wird die eingegebene Kennung überprüft, ob diese Kennung in der Datenbank bereits von anderen Benutzern besitzt oder reserviert wird. Die Datenbankkennung und WWW-Kennung unterscheiden sich nur von dem konstanten Suffix "_www", d.h. wenn eine von der Beiden an einen Benutzter vergeben wird, wird die andere ausschließlich für diesen Benutzer reserviert, obwohl sie noch nicht durch den Befehl `CREATE USER` in der Datenbank eingetragen ist. Wenn es festgelegt wird, dass diese Kennung bereits von einem Benutzer besitzt oder reserviert wird, bricht das Programm an dieser Stelle ab und man muss zum Anfang zurückkehren, um eine neue Kennung einzugeben. Falls diese Kennung noch frei ist, wird sie an diesen Benutzer vergeben.

Zur Vorbereitung für die Ausführung des `CREATE USER` Befehls werden alle zugehörigen Optionseingaben in einem Formular, das im Skript als nächsten Schritt enthalten ist, zum Auswählen tabellarisch dargestellt. Es gibt folgende Formularelemente: ein Optionsfeld (Inputfeld, wobei der Attributwert von Type gleich "radio" ist.) für die Entscheidung, ob der Benutzer ein Superuser ist; ein Optionsfeld für die Entscheidung, ob der Benutzer neue Datenbank einlegen kann; ein Optionsfeld für die Entscheidung, ob die Gültigkeit des Passworts zeitlich begrenzt ist. Wenn die Option `Ja` gesetzt wird, stehen zwei Auswahlboxen, die von den geschachteten <OPTION>-Tags in <SELECT>, zur Verfügung. Die erste besteht aus Zahlen von (1 bis 9), die Zweite besteht aus Zeiteinheiten (Woche, Monat, Jahr), damit eine Zeitspanne dargestellt werden kann. Die alle drei Optionsfelder werden auf `NEIN` als Standardeinstellung gesetzt; es gibt noch eine Checkbox, die zur Entscheidung für die Gruppenmitgliedschaft dient. Die sämtlichen Gruppennamen in der Datenbank werden angezeigt, durch Anklicken der entsprechenden Kästchen werden eine oder mehrere Gruppen ausgewählt. Aus Sicherheitsgründen erfolgt die Eingabe eines Passworts im letzten Schritt vor der Ausführung des `CREATE USER` Befehls. Sind alle Entscheidungen getroffen, kann man per Button `Weiter` zu dem Skript `zusammenfsg.php` gelangen.

Hier werden alle Daten, nämlich die Benutzerinfodaten, die als Hidden-Variablen mitgeschickt werden und die Optiondaten für den `CREATE USER`-Befehl, zusammengefasst. Man hat deswegen die letzte Gelegenheit vor der Befehlausführung, um die Eingabedaten zu überprüfen. Wenn alle Daten mit der eignen Vorstellung übereinstimmen, gibt man ein Passwort für die Datenbankkennung ein. Durch Anklicken des Buttons `Bestätigen` gelangt man zu `createuser.php`.

In diesem Skript werden alle Hidden-Variablen sowie das übertragene Passwort ausgewertet und zu einem komplexen `CREATE USER`-Befehl für jede ausgewählte Kennung formuliert. Für WWW-Kennung wird ein zufallsgeneriertes, 16 zeichenlanges Passwort eingesetzt. Nach der Formulierung werden die Befehle zu der Datenbank gesendet, dort ausgeführt. Nach der erfolgreichen Ausführung entsteht der neue Benutzer in der Datenbank. Wird eine WWW-Kennung frisch eingerichtet, wird die Datei `/proj/www/pcheger3/Port443/etc/.dbmap` mit einer neuen Zeile von `WWW-Kennung:Vorname Nachname:Passwort` aktualisiert, damit man ein Zertifikat für diese Kennung ausstellen kann.

4.3 Aktualisierung von Benutzern

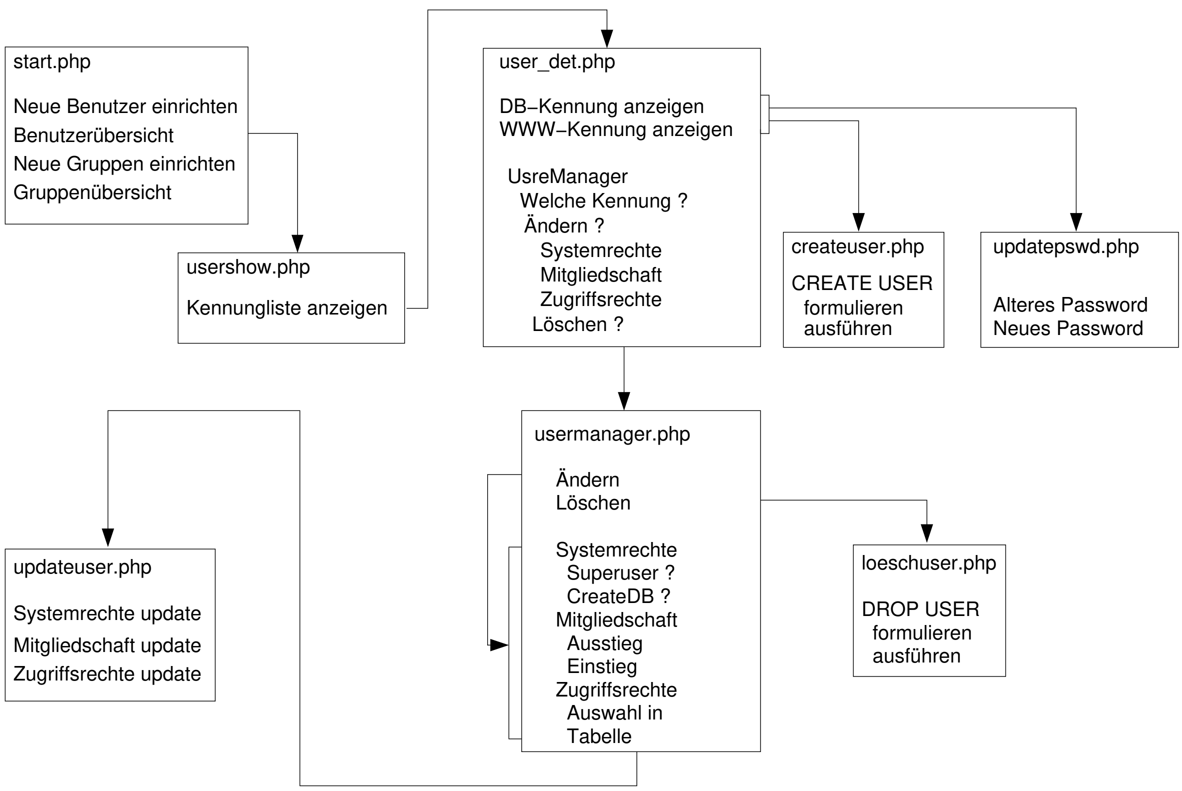

Abbildung 4.3: Aktualisierung von Benutzern

Um den Zustand eines Datenbankbenutzers zu verändern, wählt man zuerst in der Startseite den Link `Benutzerübersicht` aus. In Benutzerübersicht werden die gesamten Benutzerkennungen mit den entsprechenden Eigenschaften tabellarisch angezeigt. In dieser Tabelle kann man erfahren, ob ein Benutzer ein Superuser ist, ob er das Recht hat, neue Datenbanken einzurichten. Jede Benutzerkennung stellt wieder einen Link dar, der zu dem Skript `user_det.php` führt. Dort bekommt man eine detaillierten Übersicht von der gewählten Kennung. Die beiden Kennungen (DB-Kennung und WWW-Kennung) für einen Benutzer werden hier gemeinsam angezeigt. Wenn eine von den beiden fehlt, steht eine Option zur Verfügung, die zur Einrichtung dieser Kennung führt. Neben den allgemeinen Eigenschaften werden alle Gruppen, zu denen diese Kennung gehört, angezeigt. Wählt man eine Gruppe aus, gelangt man zur Deteilübersicht dieser Gruppe. Es gibt auch die Möglichkeit, das Passwort zu verändern. Wenn es sich dabei um die Passwortänderung von einer Datenbankkennung handelt, muss man zuerst das aktuelle Passwort eingeben und dann das neue Passwort zweimal wiederholen. Durch die Bestätigung wird ein neues Passwort für das DB-Kennung erzeugt. Bei der Passwortänderung von einer WWW-Kennung braucht man nur diese Aktion zu bestätigen. Ein zufallgeneriertes Passwort wird automatisch erzeugt, und anschließend wird die Datei `.dbmap` mit dem neuen Passwort aktualisiert.

In dem HTML-Formular `UserManager` gibt es unterschiedliche Verwaltungsfunktionen, die den Benutzerzustand verändern können. Wenn die beiden Kennungen (DB-Kennung und WWW-Kennung) für diesen Benutzer bereits in der Datenbank eingerichtet sind, kann man sich in einer Auswahlliste entscheiden, ob man die beiden Kennungen gleichmäßig betrachten oder nur die einzelne Kennung verändern möchte. Die einzelne Änderung kann zu unterschiedlichen Zuständen beider Kennungen führen. Da die beiden Kennungen für den selben Benutzer in Prinzip zusammengehören sollen, bleibt die gleiche Behandlung beider Kennungen als Standardeinstellung. Aber es gibt jedoch die Möglichkeit, die einzelne Kennung zu verändern. In diesem Fall wird eine Warnung zuerst ausgegeben.

Anschließend kann man wählen, ob die Kennungen geändert oder aus der Datenbank entfernt werden sollen. Bei der Benutzerkennungveränderung kann man feststellen, welche Änderungsoptionen durchgeführt werden. Dabei handelt es sich um die Veränderung der Systemrechte, Gruppenmitgliedschaft und Zugriffsberechtigungen. Nach der Festlegung wird das Skript `usermanager.php` aufgerufen.

In diesem Skript werden alle Optionsdaten ausgewertet. Wenn es um die Benutzeränderung geht, wird es weiter untersucht, welche Änderungsoptionen gewählt werden. Für jede gewählten Änderungsoption werden die entsprechenden Formular-Elementen eingeblendet. Bei der Änderung von Systemrechten handelt es sich um zwei Optionsfelder (type="radio"), die zur Entscheidung für das Superuserrecht und das Recht, neue Datenbanken einzulegen, dienen. Bei der Änderung von Gruppenmitgliedschaft werden alle Gruppen, zu denen der Benutzer bisher gehört, in einer Tabelle angezeigt. Jede Gruppe stellt ein Input-Element (type="checkbox") dar. Durch Anklicken kann man sich entscheiden, ob die gewählten Gruppen aus der bisherigen Gruppenliste von dem Benutzer entfernt werden sollen. Die restlichen Gruppen werden auf der gleichen Weise tabellarisch angezeigt. Durch Auswählen der entsprechenden Checkboxen werden die neuen Gruppen dem Benutzer hinzugefügt. Bei der Bearbeitung von Zugriffsrechten werden die sämtlichen Datenbankobjekte (Tabellen, Sichten und Sequenzen), auf die bestimmte Zugriffsrechte vergeben werden können, tabellarisch angezeigt. Neben jedem Objekt stehen fünf Input-Elemente (type="checkbox"), die die unterschiedlichen Privilegien (Select, Insert, Update/Delete und Rule) repräsentieren. Wenn der Benutzer bestimmte Zugriffsberechtigungen auf dieses Objekt besitzt, werden die entsprechenden Checkboxen gedruckt dargestellt. Durch Anklicken auf die Checkboxen werden die neuen Zugriffsrechte an den Benutzer vergeben bzw. die bestehenden Zugriffsrechte wieder entfernt. Nach der Feststellung aller zu verändernden Optionen in dem Formular kann man per Button `Ausführen` alle Optionsdaten zu dem Skript `update_user.php` schicken und das Skript starten.

Die übertragenen Daten werden in entsprechenden SQL-Befehlen formuliert. Dabei handelt es sich um die Befehle von `ALTER USER`, `ALTER GROUP` und `GRANT oder REVOKE`. Nach der erfolgreichen Ausführung dieser Befehle wird der Benutzerzustand aktualisiert.

Das Skript `loeschuser.php` wird aufgerufen, wenn man sich für die Entfernung der Kennungen entscheidet. Vor der Ausführung des Befehls `DROP USER` werden alle Zugriffsrechte und Gruppenmitgliedschaft zuerst entfernt. Wenn die WWW-Kennung entfernt werden soll, muss die entsprechende Zeile in der Datei `.dbmap` auch gelöscht werden.

4.4 Einrichtung der neuen Gruppen

Der Link `Neue Gruppen einrichten` in der Startseite führt direkt zu dem Skript `newgroup.php`. In diesem Skript braucht man nur einen neuen Gruppenname einzugeben. Durch Anklicken auf die Button `Weiter` wird das Skript `checkgroup.php` aufgerufen. Zuerst wird es überprüft, ob es in der Datenbank bereits eine Gruppe mit dem gleichen Name gibt. Wenn es der Fall ist, werden eine Warnung und zwei Links ausgegeben. Ein Link davon führt zur neuen Eingabe von Gruppennamen und mit dem anderen gelangt man zur Dateiübersicht dieser Gruppe. Falls kein Fehler bei der Überprüfung auftaucht, wird ein Formular zur Vorbereitung von dem Befehl `CREATE GROUP` eingeblendet.

In diesem Formular soll man die Gruppenmitgliederzuordnung und die Zugriffsberechtigungen dieser Gruppe festlegen. Bei der Gruppenmitgliederzuordnung werden alle Benutzerkennungen in der Datenbank tabellarisch angezeigt. Jede Kennung stellt ein Input-Element (type="checkbox") dar. Durch Anklicken auf die gewünschten Kennungen werden alle Gruppenmitglieder festgelegt. Bei der Zuordnung von Zugriffsrechten werden ebenfalls alle Datenbankobjekte (Tabellen, Sichten und Sequenzen) tabellarisch angezeigt. Neben jedem Objekt stehen fünf Input-Elemente (type="checkbox"), die die unterschiedlichen Privilegien (Select, Insert, Update/Delete und Rule) repräsentieren. Durch Anklicken auf die Checkboxen werden die neuen Zugriffsrechte auf das Objekt an den Benutzer vergeben. Nach der Bearbeitung sämtlicher Objektelisten wird die Zuordnung von Zugriffsberechtigungen für diese Gruppe umgesetzt. Per die Button `Ausführen` werden alle ausgewählten Daten nach dem Skript `creategroup.php` übertragen.

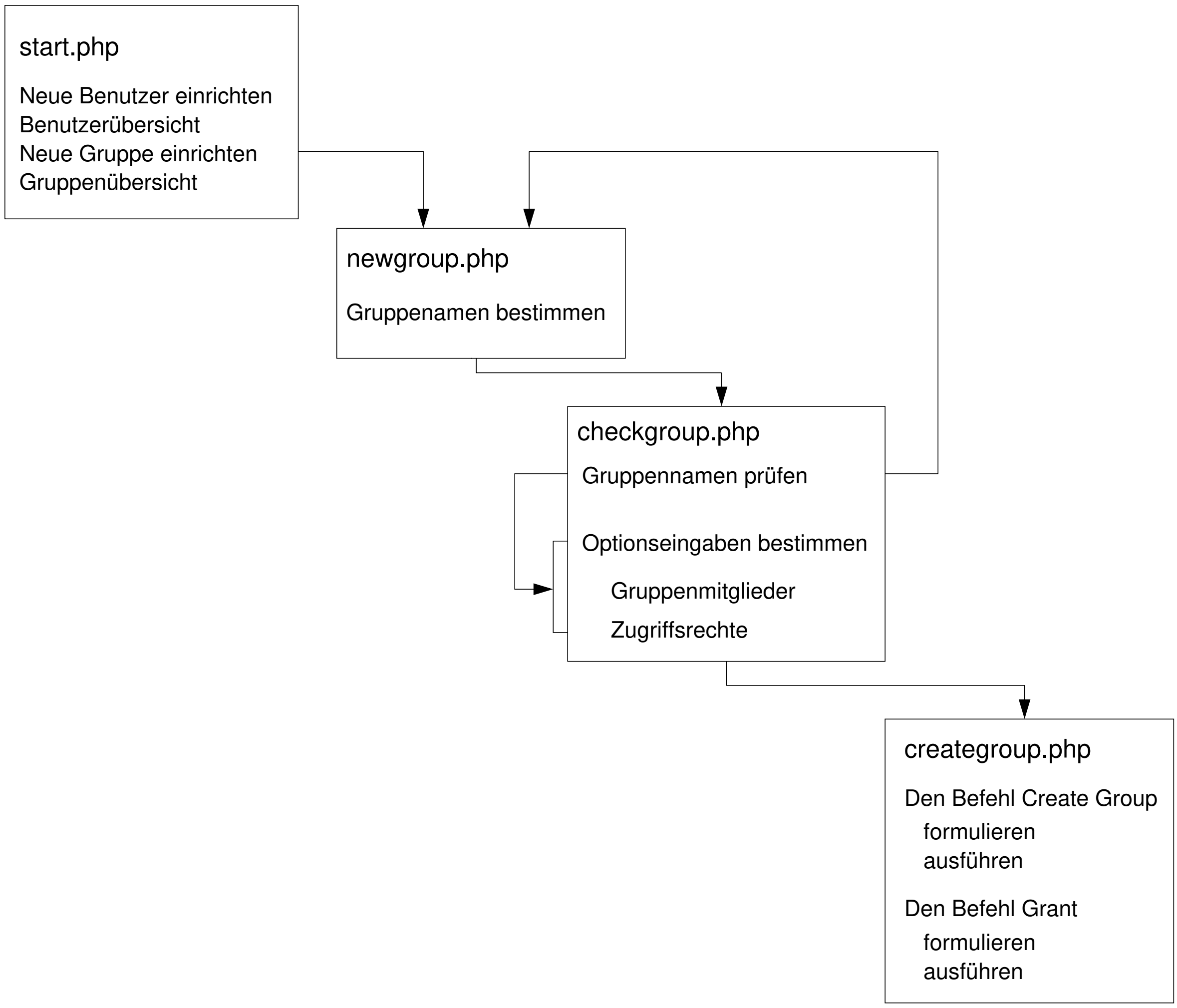

Abbildung 4.4: Einrichtung von neuen Gruppen

Die mitgeschickten Daten werden im Skript `creategroup.php` ausgewertet und in den SQL-Befehlen `CREATE GROUP` und `GRANT` formuliert. Nach der erfolgreichen Ausführung dieser Befehle wird die neue Gruppe mit den gewählten Zugriffsberechtigungen in der Datenbank eingerichtet.

4.5 Aktualisierung von Gruppen

Wenn man die bestimmte Gruppe ändern möchte, wählt man in der Startseite den Link `Gruppenübersicht`. Dieser Link führt zur Übersicht der gesamten Gruppen in der Datenbank und der jeweiligen Gruppenmitglieder. Jeder Gruppenname stellt einen Link dar. Wählt man die gewünschte Gruppe in der Gruppenliste, wird das Skript `group_det.php` aufgerufen.

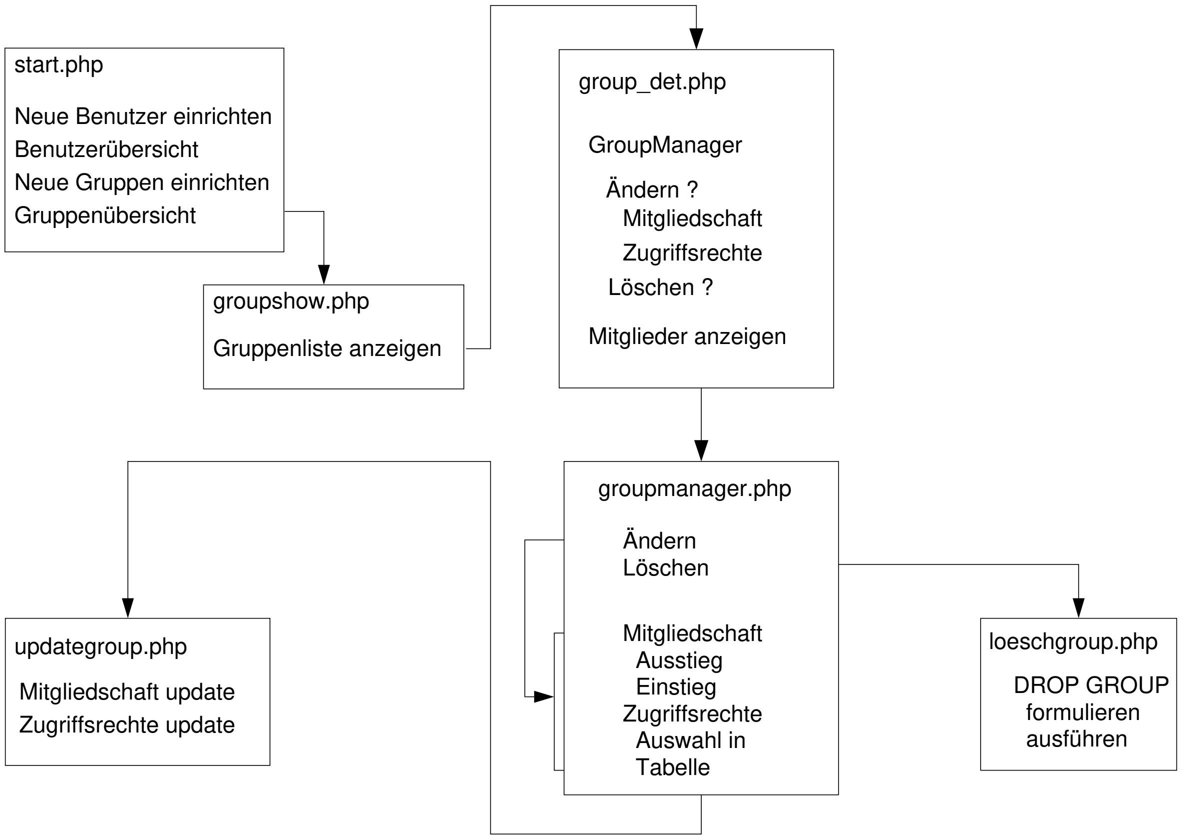

Abbildung 4.5: Aktualisierung von Gruppen

In diesem Skript werden alle Gruppenmitglieder in Detail angezeigt. Jeder Mitgliedername stellt wieder einen Link dar, der zur Änderung dieses Benutzers führt. In dem Formular `GroupManager` kann man die gewünschten Änderungsoptionen festlegen. Dabei handelt es sich um die Gruppenänderung oder Gruppenentfernung. In dem Fall der Gruppenänderung soll man sich entscheiden, welche Änderungen durchgeführt werden, nämlich die Gruppenmitgliedschaft, die Zugriffsrechte oder die beide. Nach der Feststellung gelangt man per Anklicken auf die Button `Start` zu dem Skript `groupmanager.php`.

Die mitgeschickten Optionsdaten werden ausgewertet. Bei der Gruppenänderung werden die entsprechenden Formular-Elemente eingeblendet. Für die Änderung von Gruppenmitglieder werden zuerst alle Mitglieder, die zur Zeit zu dieser Gruppe gehören, tabellarisch angezeigt. Jeder Mitglieder stellt ein Input-Element (type="checkbox") dar. Weil es sich dabei um die Mitglieder in dieser Gruppe handelt, werden alle Checkboxen gedruckt angezeigt. Klickt man auf bestimmte Checkboxen an, werden die entsprechenden Mitglieder aus der Gruppe entfernt. Die üblichen Datenbankbenutzer werden in einer anderen Tabelle angezeigt. Durch Auswählen von den Checkboxen werden die entsprechenden Benutzer in diese Gruppe hinzugefügt. Bei der Bearbeitung von Zugriffsberechtigungen geht es genau so wie bei Aktualisierung von Benutzern (siehe Abschnitt 4.3).

In diesem Formular werden alle Änderungen festgelegt. Durch Anklicken auf die Button `Ausführen` werden alle Daten nach dem Skript `update_group.php` übertragen. Dort werden sie in SQL-Befehlen `ALTER GROUP`, `GRANT` bzw. `REVOKE` formuliert. Nach der erfolgreichen Ausführung wird der Gruppenzustand aktualisiert. Bei der Gruppenentfernung werden alle Gruppenmitglieder und Zugriffsberechtigungen vor der Ausführung des Befehls `DROP GROUP` entfernt.

Kapitel 5

Funktionsreferenz

Die sämtlichen Hilfsfunktionen, die bei der Implementierung zum Einsatz gebracht wurden, sind in einer Include-Datei ***function.inc*** nach Funktionsnamen sortiert zusammengefasst. In diesem Kapitel werden die davon wichtigen Funktionen nach ihrer Funktionalität beschrieben.

`aktion_grouppriv($conn, $listen, $groupname, $art, $hr)`

Diese Funktion führt die SQL-Befehle `GRANT` oder `REVOKE` für eine Gruppe aus. `$conn` stellt die Verbindung zur Datenbank dar. Alle Privilegien, die eine Gruppe erhalten soll, werden in einem assoziativen Array (`$listen`) gespeichert. Die Objektnamen (Tabellen, Sichten und Sequenzen), auf die die Zugriffsrechte vergeben werden, werden als Schlüssel des assoziativen Array, und die entsprechenden Zugriffsberechtigungen (Select, Insert, Delete/Update und Rule) als den Wert von dem Schlüssel gespeichert. Der Parameter `$art` entscheidet, ob es sich dabei um den Befehl `GRANT` oder `REVOKE` handelt. Wenn es sich um `GRANT` handelt, wird der Parameter `$hr` auf *TO*, sonst auf *FROM* gesetzt.
Einsatz in: **creategroup.php, loeschgroup.php und update_group.php**

`aktion_userpriv($conn, $listen, $username, $art, $hr)`

Diese Funktion führt die SQL-Befehle `GRANT` oder `REVOKE` für einen Benutzer aus. `$conn` stellt die Verbindung zur Datenbank dar. Alle Privilegien, die ein Benutzer erhalten soll, werden in einem assoziativen Array (`$listen`) gespeichert. Die Objektnamen (Tabellen, Sichten und Sequenzen), auf die die Zugriffsrechte vergeben werden, werden als Schlüssel des assoziativen Array, und die entsprechenden Zugriffsberechtigungen (Select, Insert, Delete/Update und Rule) als den Wert von dem Schlüssel gespeichert. Der Parameter `$art` entscheidet, ob es sich dabei um den Befehl `GRANT` oder `REVOKE` handelt. Wenn es sich um `GRANT` handelt, wird der Parameter `$hr` auf *TO*, sonst auf *FROM* gesetzt.
Einsatz in: **loeschuser.php, update_user.php**

`altergroup($login, $listen, $art, $conn)`

Diese Funktion führt den Befehl `ALTER GROUP` für einen gegebenen Benutzer. Jeder Benutzer in der PostgreSQL Datenbank kann zu beliebigen Anzahl von Gruppen gehören. Bei der Änderung der Gruppenmitgliedschaft von einem Benutzer wird diese Funktion verwendet. `$conn` bezeichnet die Verbindungskennung zur Datenbank, `$login` ist die Benutzerkennung, `$listen` ist ein Array, in dem die Grupppennamen gespeichert werden, die neu hinzugefügt oder entfernt werden sollen. Der Parameter `$art` enthält zwei Werte, nämlich *ADD* oder *DROP*.
Einsatz in: **loeschuser.php, update_user.php**

`alteruser($conn, $login, $name1, $name2)`

Diese Funktion erzeugt ein HTML-Formular für die Ausführung des SQL-Befehls `ALTER USER`. Dabei handelt es sich um die Änderung von Systemrechten für den gegebenen Benutzer, nämlich das Superuserrecht und das Recht, neue Datenbanken einzurichten. Dafür sind zwei Input-Elemente (type="radio") zuständig. Mit Auswählen der entsprechenden Optionen werden die notwendigen Daten gesammelt und in beiden Variablen `$name1` und `$name2` gespeichert.
Einsatz in: **usermanager.php**

`checklogin($conn, $kennung)`

Bei der Einrichtung von neuen Benutzern wird es zuerst überprüft, ob die von dem Nutzer eingegebene Kennung bereits in der Datenbank existiert oder von den anderen reserviert. Wenn es der Fall ist, werden eine Warnung und ein Link zum Anschauen dieser Kennung ausgegeben. Jeder Benutzer in der Datenbank kann prinzipiell zwei Kennungen (Datenbank-Kennung und WWW-Kennung) haben. Während die Datenbank-Kennung zur direkter Anmeldung an der Datenbank dient, wird die WWW-Kennung für den Zugriff auf WWW benutzt. Die WWW-Kennung besteht aus der Datenbank-Kennung und einer Zeichenkette "_www". Wenn eine der beiden Kennungen für einen Benutzer eingerichtet wird, ist die andere automatisch für diesen Benutzer in der Datenbank reserviert.
Einsatz in: **checkuser.php**

`create_objectrightlist($conn, $login, $art)`

Bei der Bearbeitung der Zugriffsberechtigungen von einem Benutzer oder einer Gruppe handelt es sich um die sämtlichen Objekte (Tabellen, Sichten und Sequenzen) in der Datenbank und die entsprechenden Zugriffsrechte (Select, Insert, Delete/Update und Rule). Diese Funktion erzeugt ein HTML-Formular mit einer tabellarische Darstellung von den Objekten und Privilegien. Jedes Objekt wird in einer Zeile angezeigt. In der Tabelle gibt es sieben Spalten, nämlich Objektname, Objekttyp (Tabelle, Sicht oder Sequenz) und fünf Privilegienarten (Select, Insert, Delete, Update und Rule). Für jede Privilegiensart wird ein Input-Element (type="checkbox") von HTML-Formular eingesetzt. Wenn der Benutzer oder die Gruppe bestimmte Zugriffsrechte auf das Objekt hat, werden die entsprechenden Checkboxen von den Privilegien gedruckt angezeigt. Durch

Anklicken auf die Checkbox von Privilegien werden die entsprechenden Zugriffsberechtigungen auf das Objekt gesetzt oder wieder entfernt. Nach der Bearbeitung aller Zeilen werden die gesamten Zugriffsrechte auf alle Objekte in der Datenbank für den eingegebenen Benutzer oder die Gruppe bestimmt. Der Parameter `$conn` stellt die Verbindung zur Datenbank dar, `$login` bezeichnet den Benutzernamen oder Gruppennamen. Wenn `$art` gleich "group"ist, handelt es sich um die Bearbeitung von einer Gruppe, sonst geht es um einen Benutzer.
Einsatz in: **checkgroup.php, groupmanager.php, usermanager.php**

`createtable($listen, $anzahl, $name, $anzeige)`

Diese Funktion erzeugt eine tabellarische Darstellung für jeden Arrayeintrag in `$listen` und jeder Eintrag stellt ein Input (type="checkbox") dar. Mit `$anzahl` kann man bestimmen, wieviel Spalten diese Tabelle haben soll. Die ausgewählten Daten werden in der Variable `$name` gespeichert. Wenn `$anzeige` "checked" ist, werden alle Checkboxen gedruckt dargestellt, sonst werden sie normal angezeigt.
Einsatz in: **checkgroup.php, checkuser.php, groupmanager.php, usermanager.php, user_det.php**

`form_grouplist($grouplist)`

Bei der Bearbeitung von Gruppenmitgliedern sollen die Datenbank-Kennung und WWW-Kennung von einem Benutzer gleich behandelt werden. Diese Funktion untersucht alle Mitglieder in einer Gruppe, fasst die Datenbank-Kennung und die WWW-Kennung von einem Benutzer zusammen, um die mögliche Inkonsistenz zu vermeiden. z.B die beiden Kennungen "aaa" und "aaa_www" werden in der Form "aaa(DB & WWW)" zusammengefasst.
Einsatz in: **groupmanager.php**

`get_allgrouplist($conn)`

Diese Funktion fragt die Systemtabelle *pg_group* in PostgreSQL ab, und gibt im Erfolgfall die gesamten Gruppennamen in einem Array zurück.
Einsatz in: **checkuser.php, usermanager.php**

`get_alluserlist($conn)`

Diese Funktion fragt die View *pg_user* von der Systemtabelle *pg_shadow* in der Datenbank ab, und gibt im Erfolgfall die gesamten Benutzerkennunglisten in einem Array zurück.
Einsatz in: **groupmanager.php**

`get_mygrouplist($conn, $login)`

Diese Funktion fragt die Systemtabelle *pg_group* in PostgreSQL ab, und gibt im Erfolgfall die gesamten Gruppennamen für einen eingegebenen Benutzer (`$login`) in einem Array zurück.

Einsatz in: **loeschuser.php, update_user.php, usermanager.php**

`get_mymemberlist($conn, $groupname)`

Diese Funktion fragt die View *pg_user* von der Systemtabelle *pg_shadow* in der Datenbank ab, und gibt im Erfolgfall die gesamten Benutzerkennunglisten für die Gruppe (`$groupname`) in einem Array zurück.
Einsatz in: **groupmanager.php, loeschgroup.php, update_group.php**

`get_myrights($conn, $login, $art)`

Die Systemtabelle *pg_class* in PostgreSQL enthält alle vergebenen Zugriffsberechtigungen auf jedes Datenbankobjekt. Diese Funktion fragt diese Tabelle ab, und gibt im Erfolgfall die gesamten Zugriffsberechtigungen auf alle Datenbankobjekte von einem eingegebenen Benutzer oder einer Gruppe in einem Array zurück. Der Parameter `$conn` stellt die Verbindungskennung dar, `$login` bezeichnet einen Benutzernamen oder Gruppennamen. Wenn die Eingabe von `$art` "group" ist, handelt es sich dabei um eine Gruppe, sonst geht es um einen Benutzer.
Einsatz in: **loeschgroup.php, loeschuser.php, update_group.php, update_user.php**

`get_userform($conn, $kennungsart, $kennung, $vorname, $nachname)`

Für die Ausführung des SQL-Befehls `CREATE USER` werden bestimmte Optionseingaben wie beispielsweise die Eingabe über die Systemrechte, Eingebe über die Gruppenzuordnung usw. benötigt. Diese Funktion erzeugt deshalb ein HTML-Formular, um solche Eingaben zu erfassen. Es gibt folgende Formularelemente: ein Optionsfeld (Input-Element, wobei der Attributwert von Type gleich "radio" ist) für die Entscheidung, ob der Benutzer ein Superuser ist; ein Optionsfeld für die Entscheidung, ob der Benutzer neue Datenbanken einlegen kann; ein Optionsfeld für die Entscheidung, ob die Gültigkeit des Passworts zeitlich begrenzt ist. Wenn die Option `Ja` gesetzt wird, stehen zwei Auswahlboxen, die von den geschachteten <OPTION>-Tags in <SELECT>, zur Verfügung. Die Erste besteht aus Zahlen von (1 bis 9), die Zweite besteht aus Zeiteinheiten (Woche, Monat, Jahr), damit eine Zeispanne dargestellt werden kann; Es gibt noch eine Checkbox, die zur Entscheidung für die Gruppenmitgliedschaft dient. Die sämtlichen Gruppennamen in der Datenbank werden angezeigt, durch Anklicken der entsprechenden Kästchen werden eine oder mehrere Gruppennamen ausgewählt. Um eine bessere Übersicht bei der neuen Benutzereinrichtung zu bekommen, benutzt man auch einige Benutzerinfodaten, die als Parameter eingegeben werden. Dabei handelt es sich um `$kennungsart` (Datenbank-Kennung, WWW-Kennung), `$kennung`, `$vorname` und `$nachname`.
Einsatz in: **checkuser.php**

`getpassword($stellen)`

Diese Funktion erzeugt ein zufallgeneriertes Passwort, das aus Zeichen und Zahlen mit Variablenlänge besteht. Mit dem Parameter `$stellen` kann man die Stellenanzahl des

Passworts bestimmen.
Einsatz in: **createuser.php, updatepswd.php**

`group_id2name($conn, $groupids)`

In der Systemtabelle `pg_group` werden alle Gruppenmitglieder nur als BenutzerID angezeigt. Diese Funktion setzt die Zuordnung von BenutzerID und Benutzernamen für jede Gruppe um. Der Parameter `$conn` bezeichnet die Verbindung zur Datenbank, `$groupids` ist ein Array, in dem alle zubearbeitenden GruppenID-Nummer gespeichert werden.
Einsatz in: **groupshow.php**

`updatefile_addln($pfad, $login, $name, $pswd)`

Bei jeder neuen Einrichtung von der WWW-Kennung wird ein Zertifikat für den Zugriff auf den sicheren Web-Server ausgestellt. Für die Datenbankanmeldung über den sicheren Web-Server werden gewisse Benutzerdaten benötigt. Diese Daten werden in einer Textdatei (.dbmap) gespeichert. Dabei handelt es sich um den Benutzernamen (Vor- und Nachname), die WWW-Kennung und das entsprechende Passwort. Diese Funktion aktualisiert die ".dbmap"-Datei mit einer neuen Zeile von Benutzernamen, Kennung und Passwort.
Einsatz in: **createuser.php, updatepswd.php**

`updatefile_detln($pfad, $login)`

Diese Funktion löscht den Eintrag von der eingegebenen Kennung (`$login`) in der Datei ".dbmap".
Einsatz in: **loeschuser.php, updatepswd.php**

Kapitel 6

Zusammenfassung und Ausblick

In diesem FoPra wurde ein Benutzerverwaltungssystem entworfen und implementiert, das die sämtlichen Verwaltungsaufgaben nicht nur auf der Ebene der PostgreSQL-Datenbank, sondern auch im Zusammenspiel mit dem Web-Server zusammenfasst.

Dieses System automatisiert alle herkömmlichen manuellen Arbeitsvorgänge bei der Benutzerverwaltung, integriert die zusammengehörigen Verwaltungsfunktionen und führt sie nacheinander sequenziell aus, damit die Vereinfachung bei der Einrichtung von neuen Benutzern bzw. Gruppen und Vergabe von Zugriffsberechtigungen, die Vermeidung von Inkonsistenzen und die Erleichterung des Verwaltungsaufwands ermöglicht werden.

Zur Vervollständigung des Benutzerverwaltungssystems ist die Implementierung der automatischen Ausstellung für Zertifikate. Diese Aufgabe wird in einem anderen Fopra erfüllt. Zur Einbindung solcher Funktionen wurden entsprechende Schnittstellen in das System zur Verfügung gestellt.

Literaturverzeichnis

[BM01] C. Barth and S. Marick.
Evaluation der Anbindung einer SQL-Datenbank an einen Apache Webserver.
Technische Universität München, Fopra, 2001.

[pgs] Postgresql 7.1 administrator's guide, http://techdocs.postgresql.org/.
The PostgreSQL Global Development Group.

[php] http://www.php.net/. The PHP Group.

[WD01] John C. Worsley and Joshua D. Drake.
Practical PostgreSQL.
O'Reilly, 2001.