TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät für Informatik

Datenverwaltungssysteme

Bachelorarbeit

Neugestaltung der Datenbank des Chemnitzer Studentennetzes

Morris Jobke

Chemnitz, den 14. November 2012

Prüfer: Dr.-Ing. Frank Seifert

Betreuer: Dipl.-Inf. Johannes Fliege

Abstract

Das Chemnitzer StudentenNetz (CSN) ist eine studentische Initiative, die für das wohnheiminterne LAN und dessen Anbindung an das Campusnetz der TU Chemnitz verantwortlich ist. Damit einher gehen Verwaltungsaufgaben, die die Erfassung der Zugangsdaten der einzelnen Nutzer an über 2000 Netzwerkanschlüssen, die Inventarisierung, die Verwaltung der Mitarbeiter sowie die interne Buchführung einschließen. Um diese Aufgaben zu bewältigen wird eine seit 1999 bestehende Datenbank genutzt und gepflegt, wobei in zunehmendem Maße festzustellen ist, dass diese aufgrund der Weiterentwicklungen innerhalb des CSN nicht mehr den gegenwärtigen Anforderungen genügt. Dies äußert sich u.a. in der Zunahme des benötigten Wartungsaufwands.

Um den Betrieb des CSN weiterhin sicherzustellen, sollen die Anforderungen an die dort einzusetzende Datenbank neu erfasst werden. Darauf aufbauend soll ein Datenbankschema entwickelt werden, das diesen Anforderungen genügt und in der Lage ist, auch die bereits vorhandenen Datenbestände zu erfassen. Dabei ist es Ziel dieser Arbeit, dieses Datenbankschema in einem Testsystem prototypisch umzusetzen und, im Weiteren, Strategien zur automatisierten Migration der alten Datenbasis zu erarbeiten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

SQL-Anweisungen

Abkürzungsverzeichnis

CSN Chemnitzer StudentenNetz

DNS Domain Name Service

ID Identifikator

IP Internet Protocol

LAN Local Area Network

StuWe Studentenwerk Chemnitz-Zwickau

SQL Structured Query Language

URZ Universitätsrechenzentrum

VLAN Virtual Local Area Network

WLAN Wireless Local Area Network

1. Einleitung

Das Chemnitzer Studentennetz (CSN) ist eine studentische Initiative, deren Ziel es ist, im Rahmen von Forschung und Lehre den Bewohnern der Studentenwohnheime einen Internetanschluss über das Netz der Universität zur Verfügung zu stellen. Hierzu stehen dem CSN-Team ein monatliches Budget, welches über die Miete eingenommen wird, sowie eine Anbindung an das Universitätsnetz und somit das Deutsche Forschungsnetz zur Verfügung. Das CSN-Team ist für die Beschaffung, Einrichtung und Wartung sämtlicher Netzwerkkomponenten selbst zuständig.

Motivation

Die Datenbank des CSN ist bis dato ein funktionierendes Gebilde, welches von den jeweiligen Verantwortlichen immer mit Abstand betrachtet wurde. Zum Entwicklungszeitpunkt wurde versucht, so viele aktuelle Features des PostgreSQL-Datenbank-Management-Systems wie möglich zu nutzen. Damit wurde es für den Nachwuchs dieses Postens nicht einfacher. Veränderungen an der Datenbank sind meist nur oberflächlich hinzugefügt und Bestehendes kaum gewartet worden. Somit ist die Datenbank im aktuellen Zustand sehr mächtig, aber für die Mitarbeiter des CSN kaum wartbar. Aus diesem Grund wurde entschieden, dasselbige einer Generalüberholung zu unterziehen, Altlasten des Systems zu entfernen und die Anforderungen neu zu erfassen. Dieser

Schritt ist ein Teilabschnitt einer allgemeinenen Restrukturierung der Software-Basis des gesamten Netzes und somit nur folgerichtig.

Aufgabe

In dieser Arbeit soll die aktuelle Datenbank des CSN zum tieferen Verständnis der hinterlegten Daten vollständig auf ihre Funktionsweise untersucht werden. Darauf aufbauend sind die Anforderungen an ein neues Datenbank-Schema zu erfassen und selbiges zu entwerfen. Anschließend soll eine Evaluation durchgeführt werden und die Datenbank durch einen eigens entworfenen Migrationsvorgang befüllt werden.

Struktur der Arbeit

Das Kapitel 2 beschäftigt sich mit den Grundlagen und Vorkenntnissen, die für das tiefere Verständnis dieser Arbeit hilfreich sind. Hierzu zählt unter anderen die Beschreibung der Begrifflichkeiten, die im Datenbank-Umfeld gebräuchlich sind, sowie das Erklären der genutzten bildlichen Schemadarstellungen.

Im darauffolgenden Kapitel werden die Aufgabe und der Zustand des aktuell bestehenden System beschrieben und analysiert. Dabei wird auf sämtliche Tabellen der aktuellen Datenbank eingegangen und eine kurze Zusammenfassung der Kritikpunkte gegeben.

Im vierten Kapitel werden diese Kritikpunkte genauer betrachtet und es wird an Beispielen erläutert, zu welchen Problemen diese führen können.

Anschließend werden im Kapitel 5 konkrete Lösungsvorschläge für ebendiese Probleme aufgelistet.

Das neu entworfene Modell wird vollständig im sechsten Kapitel erklärt. Dies beinhaltet die verwendeten Datentypen, eine Beschreibung sämtlicher Tabellen samt ihrer Funktion sowie die Nennung aller Relationen zwischen den Tabellen. Zusätzlich verweist dieses Kapitel noch auf die konkreten Änderungen hinsichtlich der aktuellen Datenbank.

Das Kapitel 7 beschäftigt sich mit der Umsetzung dieses Modells. Hierbei wird auf die eingesetzte Hard- und Software, Probleme bei der praktischen Umsetzung, die Migrationsstrategie und Kenndaten zur Migration selbst eingegangen.

Im vorletzten Kapitel wird diese Umsetzung kurz beschrieben und bewertet, womit ein Urteil über die Anforderungserfüllung gefällt werden kann. Anschließend daran sind Verbesserungsvorschläge für die Zukunft des neue entworfenen Systems aufgeführt.

Das abschließende Kapitel enthält eine Zusammenfassung aller gewonnen Erkenntnisse aus dieser Arbeit.

2. Grundlagen

Die in dieser Arbeit genutzten Konventionen und Darstellungsformen werden in diesem Kapitel ebenso erklärt wie grundlegende Begriffe aus dem Datenbank-Umfeld.

2.1. Grundlegende Begriffe

Eine **Datenbank** dient zur persistenten Speicherung von großen Datenmengen. Dabei ist die Datenbank Teil eines **Datenbankmanagementsystems**, welches zusätzlich noch Metadaten, wie beispielsweise die Struktur der Daten, vorhält und verwaltet. Zur Struktur gehört unter anderen der Aufbau einzelner Tabellen, Datentypen der einzelnen Datensätze, Beziehungen zwischen Daten oder Speicherformate. Diese strukturelle Beschreibung der Datenbank nennt man **Datenbankschema**. [Elm02, Seiten 28 ff., 47 ff.] [Tü06, Seite 1 ff.]

Sämtliche Daten sind in **Tabellen** gespeichert, welche die Struktur der enthaltenen Datensätze vorgibt. Dabei sind die **Spalten** einer Tabelle die Eigenschaften jedes Datensatzes. Jede Zeile einer Tabelle enthält demnach einen kompletten **Datensatz**. [Elm02, Seite 226 ff.] [Tü06, Seite 120 ff.]

Die einzelnen Spalten werden durch Datentypangaben und **Beschränkungen** (engl. constraint) genauer beschrieben. Die Auszeichnung als **Primärschlüssel** (engl. prima-

ry key) gibt an, dass die Werte dieser Spalte als eindeutige Indentifikatoren für einen Datensatz gelten. Dies wird bei **Fremdschlüsseln** (engl. foreign key) genutzt, um einen Datensatz referenzieren zu können. Dabei wird die Spalte, welche als Fremdschlüssel definiert ist, strukturell als Verknüpfung zur entsprechenden Spalte beschrieben. Eine weitere Beschränkung ist die **Eindeutigkeit** (engl. unique), welche festlegt, dass sämtliche Werte dieser Spalte voneinder verschieden sein müssen. Ebenfalls kann die Kennzeichnung als **nicht-leer** (engl. not null) vorgenommen werden, die für jeden Datensatz die Angabe des entsprechenden Attributs voraussetzt. [Elm02, Seite 232 ff.] [Tü06, Seite 158]

2.2. Konventionen

In dieser Arbeit werden folgende Formatierungskonventionen genutzt. Tabellen werden grundsätzlich **fett** geschrieben. Die Spalten einer Tabelle werden *kursiv* markiert.

2.3. Schema-Darstellungen

Zur Darstellung des bisherigen und neu entworfenen Schema wurde die Software DbSchema von Wise Coders Solutions genutzt. [DbS] Die Darstellungsform wird nachfolgend erklärt.

Tabelle In Abbildung 2.1 ist ein Beispiel für eine Tabelle zu sehen. Im Kopf befindet sich der Name und im Bereich darunter sind die einzelnen Spalten aufgelistet.

Spalten/Attribute Jede Zeile in der Tabelle stellt eine Spalte dar. Ganz links ist an dem kleinen gelben Kreuz ersichtlich, ob die Spalte wertlos (*NULL*) sein darf. Wenn das

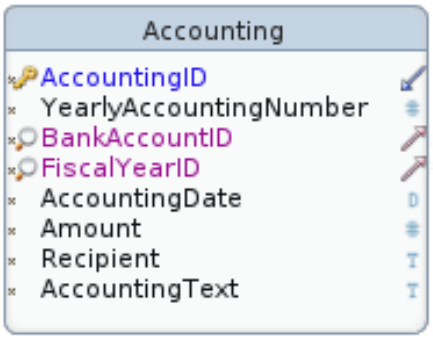

Abbildung 2.1.: Beispiel für eine Tabelle

Kreuz vorhanden ist, muss sie einen Wert ungleich *NULL* enthalten, andernfalls nicht. Auf der rechten Seite ist durch ein Symbol dargestellt, um welchen Datentyp es sich handelt.

Primärschlüssel Primärschlüssel sind durch einen gelben Schlüssel vor dem Namen und eine blaue Schriftfarbe kenntlich gemacht.

Fremdschlüssel Fremdschlüssel sind durch die lila Schriftfarbe dargestellt.

Index Wird für eine Spalte ein separater Index erstellt, so findet sich vor dem Namen ein Lupensymbol. Dies ist meist für Fremdschlüssel der Fall.

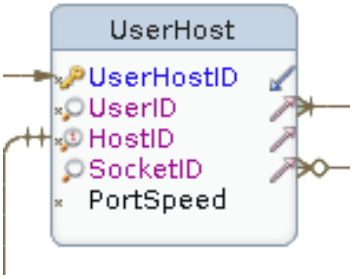

Abbildung 2.2.: Beispiel für Unique- und Index-Spalten

Eindeutige Werte Sollen innerhalb einer Spalte nur eindeutige Werte (UNIQUE-Constraint) zu finden sein, wird dies durch eine Lupe mit einer „1“ vor dem Namen kenntlich gemacht (siehe Abbildung 2.2).

Datentypen In den Schema-Darstellungen wird zwischen drei Grund-Datentypen und einer Schlüsselmarkierung unterschieden. Dies sind der numerische Datentyp (integer, numeric, float, money, . . .), repräsentiert durch eine Raute und der textuelle Datentyp (text, varchar, . . .), dargestellt durch ein großes „T“. Weiterhin ist eine Darstellung für den Datumstyp (datetime, timestamp, . . .), angezeigt durch ein großes „D“, sowie die Markierung für Primär- bzw. Fremdschlüssel dargestellt mittels Pfeil nach links unten bzw. rechts oben vorhanden. Falls keiner dieser Hinweise dargestellt ist, handelt es sich um einen eigenen Datentyp, der in der Schemabeschreibung aufgeführt ist.

Beziehungen zwischen Tabellen Zur besseren Lesbarkeit, welcher Fremdschlüssel zu welchem Primärschlüssel gehört, beginnt diese Beziehung immer auf Höhe des Fremdschlüssel-Eintrags und endet mit der Spitze auf dem Primärschlüssel-Eintrag. In Abbildung 2.3 sieht man den Fremdschlüssel *HouseID* in **Floor**, der auf den Primärschlüssel *HouseID* in **House** zeigt. Die Abbildung 2.2 zeigt auch, dass diese Beziehungen unterschiedliche Pfeile haben. Falls hinter dem Pfeil ein Kreis (in Abbildung 2.2 rechts unten) zu sehen ist, so ist diese Relation nicht zwingend für jeden Eintrag in dieser Tabelle vorhanden. Das bedeutet, dass der Fremdschlüsssel - hier *SocketID* - auch den Wert *NULL* annehmen kann und auf keinen anderen Eintrag verweist. Falls diese Verbindungslinie mit zwei senkrechten Strichen (siehe Abbildung 2.4 links) beginnt, so kommt der Fremdschlüssel in der Tabelle maximal einmal vor (UNIQUE-Constraint). Alle anderen (siehe Abbildung 2.4 rechts) Pfeile kennzeichnen normale Fremdschlüssel-Verweise ohne weitere Beschränkungen.

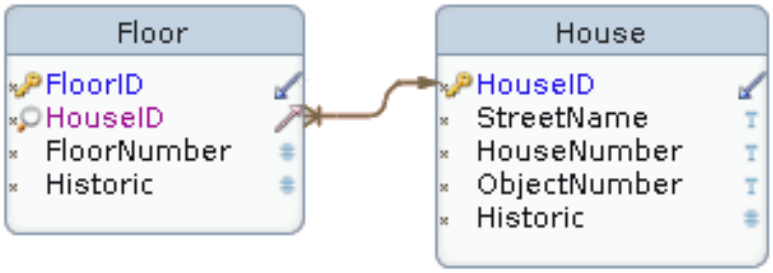

Abbildung 2.3.: Beispiel für eine Beziehung

Abbildung 2.4.: Beispiel für Beziehungsarten

3. Aufgabengebiet des CSN und derzeitiger Stand

Das Chemnitzer StudentenNetz (CSN) ist eine studentische Initiative, die sich zum Ziel gesetzt hat, die Studentenwohnheime auf dem Campus mit einem Netzwerk zu versorgen, dass über das Universitätsnetzwerk an das Internet angeschlossen ist. Hierzu gibt es ein Team, welches für den rechtlichen Rahmen der Vereinigung sorgt. Dieses verteilt zu erledigende Aufgaben, wie zum Beispiel die Verwaltung und Buchung der Finanzen oder die Datenbankadministration, an gewisse Verantwortliche innerhalb des Teams. Darüber hinaus gibt es noch zahlreiche sogenannte Etagenverantwortliche, welche die direkten Ansprechpartner der Studenten bei Netzwerk-technischen Problemen sind.

Insgesamt stellt das CSN ein Netzwerk für die circa 1.800 in den Wohnheimen wohnenden Studenten zur Verfügung. Dabei zählt das CSN-Team etwa zehn ständige Mitglieder, die die Kern-Aufgaben erledigen, und über 30 Etagenverantwortliche. Sämtliche Arbeit, die von den Mitgliedern für das CSN ausgeführt wird, ist ehrenamtlich.

3.1. Aufgabe der Datenbank

Die CSN-Datenbank ist Sammelpunkt für sämtliche Daten, die während des Betriebs des Netzwerkes und der damit verbundenen Aufgaben anfallen. Aufgrund des weiten Spektrums werden diese nachfolgend genauer aufgelistet und beschrieben.

Zu den gespeicherten Daten zählen die Nutzerinformation, sowie deren Wohnort, Rechnerdaten, verursachte Datenmengen und deren Status (z.B. gesperrt oder verwarnt). Ebenfalls ist sämtliche Netztechnik und deren Einstellungen hinterlegt, damit diese leicht änderbar und somit für die Mitarbeiter besser zu warten ist. Ein weiteres großes Gebiet ist der Finanzbereich, der sämtliche Buchungen des CSNs beinhaltet und somit die Finanzer des CSNs unterstützt. Anträge von Nutzer, die das CSN benutzen wollen, sind ebenso in der Datenbank enthalten wie sämtliche Um- sowie Auszüge. Schließlich werden auch Inventarinformationen zum gesamten CSN-Eigentum gespeichert. Zahlreiche einzelne und nicht kategorisierbare Daten, wie beispielsweise aktuelle Informationen, die auf der Webseite angezeigt werden, oder noch nicht zugewiesene IP-Adressen, runden die Datenbasis und somit die Aufgabe der Datenbank ab.

3.2. Aktueller Datenbestand der Datenbank

Dieses Teilkapitel soll einen groben Überblick über den aktuellen Umfang der Datenbank geben. Die Kategorisierung wurde selbst vorgenommen und dient lediglich der Übersichtlichkeit. Für die genaue Anzahl der Einträge und Spalten pro Tabelle wird auf die Übersichtstabelle A.1 im Anhang verwiesen.

Grundlage für diesen Abschnitt ist ein Datenbank-Schnappschuss, die teilweise Dokumentation aus dem Jahre 2004 von Ingo van Lil [Lil] sowie die Analyse des Quellcodes der aktuellen CSN-Webseite [CT].

Eine gesamte Schema-Darstellung befindet sich ebenfalls im Anhang. Die genannten Angaben an zu den Datensatz-Anzahl bezieht sich auf den 21. August 2012.

3.2.1. Finanzen

Diese Kategorie deckt die interne Buchhaltung und Finanzplanung des CSN ab. Sämtliche Einnahmen sowie Ausgaben buchen Mitarbeiter eigenständig und lassen diese dann quartalsweise von einem Steuerberater überprüfen. Hauptaugenmerk hierbei liegt in den Buchungen, Transfers und Abschlüssen.

Insgesamt enthalten die 15 Tabellen dieser Kategorie circa 11.000 Datensätze, welche seit dem Jahr 2004 erfasst wurden und für zahn Jahre gespeichert werden müssen.

3.2.2. StuWe

Das CSN erhält monatlich vom Studentenwerk Chemnitz-Zwickau (StuWe) 5 Euro pro Nutzer, welche über die Miete eingezogen werden. Dazu wird dem CSN ebenfalls monatlich eine Liste aller Zahlungen übermittelt, die in die Datenbank eingelesen wird. Nutzer, die keine Miete zahlen oder nicht den gesamten Monat im Wohnheim wohnten, zahlten folglich auch weniger, was aus diesen Listen ersichtlich ist.

Die zwei zu dieser Kategorie gehörenden Tabellen speichern etwas über 135.000 Datensätze, die beginnend im November 2006 monatlich eingelesen wurden und ebenfalls für die Dauer von zehn Jahren in der Datenbank vorgehalten werden.

3.2.3. VLAN

Innerhalb des CSNs gibt es verschiedene virtuelle LANs. Diese werden zur Abgrenzung einzelner Nutzergruppen benutzt. Zum Beispiel gibt es ein VLAN für alle Nutzer, die in Quarantäne sind, wodurch man den Netzzugang für diese Nutzergruppe explizit eingrenzen kann.

Die drei Tabellen enthalten etwas über 50 Datensätze.

3.2.4. Geräte

Sieben Tabellen sind unter dem Stichwort „Geräte“ zusammenfassbar. Hierbei handelt es sich um Netzwerktechnik und soll Metadaten, wie deren Status und Standort, zugehörige Rechnungen und mehr aufnehmen. Aufgrund der Vielzahl und Komplexität der erforderlichen einzutragenden Daten wurde dies kaum gepflegt und wird mittlerweile nicht mehr genutzt. Diese Tabellen haben lediglich einen Inventar-relevanten Einfluss und haben nichts mit der Funktionalität des Netzwerkes zu tun.

Zusammen umfasst diese Gruppe derzeit circa 80 Datensätze.

3.2.5. CSN- und Nutzerrechner

Diese Kategorie beschäftigt sich mit allen vom CSN und von Nutzern betriebenen Rechnern. Unter den vom CSN stammenden Rechnern sind zum Beispiel Router oder die Rechner des CSN-Labors zu verstehen. Hauptsächlich sind hier Daten über Anschlussort und Netzwerkdaten, wie IP- und MAC-Adresse gespeichert.

Diese sieben Tabellen enthalten insgesamt über 20.000 Datensätze.

3.2.6. Traffic

Diese Gruppe von Tabellen ist für das Speichern der Trafficwerte der Nutzer verantwortlich. Es wird viertelstündlich der Traffic eines jeden Rechners festgehalten. Genutzt werden die Daten, um Überschreitungen der in der CSN-Nutzerordnung festgehaltenen Grenze zu überwachen und eventuelle Verstöße zu ahnden. Insgesamt bleiben diese Daten für sechs Monate in der Datenbank.

Hier ist der größte Teil aller Datensätze zu verzeichnen - circa 8,7 Millionen Einträge listen die sieben Tabellen.

3.2.7. Haus- und Etagenverantwortliche

In jedem Wohnheim gibt es ehrenamtliche Mitarbeiter des CSN, die für die schnelle Hilfe beim Nutzer zuständig sind und bei kleineren Problemen mit dem Netz helfen. Vor einigen Jahren gab es noch sogenannte Hausverantwortliche, die unter anderen für das Einsammeln der Unkostenbeiträge mit zuständig waren, was jedoch mittlerweile -

wie beschrieben - über die Miete geschieht. Da diese Struktur mittlerweile entfallen ist, gibt es den Posten des Hausverantwortlichen nicht mehr.

Die vier Tabellen haben zusammen etwas mehr als 160 Datensätze.

3.2.8. Standort

Jeder CSN-Nutzer wohnt in einem Zimmer der zehn Wohnheime und in jedem dieser gibt es eine oder mehrere Anschlussdosen. Jedes Zimmer ist als ein Standort festgehalten und bietet die Grundlage dafür, dass ein Anschluss dem richtigem Nutzer zugeordnet wird.

Insgesamt enthalten diese fünf Tabellen über 15.000 Einträge.

3.2.9. Nutzer

Einige Tabellen der Datenbank beinhalten natürlich Daten der Nutzer. Dies fängt bei reinen Personendaten an, geht über gewisse Status der Nutzer bis hin zu Aufgabengebieten einzelner Nutzer innerhalb des CSNs oder eventuellen Sperren.

Die hierzu gehörenden zehn Tabellen beherbergen insgesamt fast 31.000 Datensätze.

3.2.10. Netztechnik

Die Netztechnik wird täglich automatisch aktualisiert und ist aus diesem Grund ebenfalls in der Datenbank abgebildet. Ein Stack besteht aus einer oder mehreren Units,

welche wiederum eine Vielzahl an Ports haben. Diese sind dann direkt mit einer Anschlussdose auf dem Zimmer verbunden.

Die vier Tabellen speichern über 3.600 Datensätze.

3.2.11. Login-Informationen

Die Nutzung von Shibboleth[1], dem zentralen Authentifizierungsdienst der Universität, wurde noch nicht im CSN realisiert. Daher benötigen wir die Zugangsdaten der Nutzer in einer anderen Form. Hierzu übermittelt das URZ Daten an das CSN, die unter anderem Nutzerkürzel und Passwort-Prüfsumme enthält. Diese Daten sind denen in Unix-Umgebungen gespeicherten Daten in **/etc/passwd** ähnlich. [pas]

Diese drei Tabellen enthalten insgesamt fast 30.000 Einträge.

3.2.12. DNS-Daten

Es gibt zwei Tabellen, die mit der DNS-Konfiguration in Verbindung gebracht werden können. Eine ist für den Update-Status der Konfiguration zuständig und eine weitere enthält Domain-Namen.

Die zwei Tabellen beinhalten insgesamt über 4.000 Einträge.

[1] http://www.tu-chemnitz.de/urz/www/auth/wtc.html

3.2.13. Anträge

Neue Nutzer füllen ein Formular zur Anmeldung bei CSN aus. Dabei müssen sie ihr Zimmer und Daten ihres PCs angeben. Dieser Antrag wird von einem Studentenwerk-Mitarbeiter bestätigt oder abgelehnt und resultiert - bei Annahme - in einem neu angelegten Nutzer samt Rechner.

In den drei Tabellen sind fast 28.000 Einträge zu Nutzeraufnahme- und Rechneranmeldeanträgen zu finden.

3.2.14. Änderungs-Logs

Um den Zustand der Nutzer- und Rechnerdaten für jeden Zeitpunkt rekonstruieren zu können, werden über diese Daten Änderungsverläufe gespeichert.

Die Daten in diesen beiden Tabellen beläuft sich auf ungefähr 7.500 Einträge.

3.2.15. Konfigurationstabellen

Zwei Tabellen enthalten die letzten gültigen Konfigurationseinstellungen für die Firewall und die Stacks.

Beide Tabellen enthalten annähernd 4.500 Einträge.

3.2.16. Sonstige Tabellen

Letztlich gibt es noch fünf Tabellen, die keiner Kategorie zuzuordnen sind. Sie enthalten eine Warteschlange für ausgehende E-Mails, Konstanten, wie das aktuelle Finanzjahr oder die Dauer einer Sperre, die aktuell noch verfügbaren IP-Adressen, die Ankündigungen, die auf der CSN-Webseite zu lesen sind, und Informationen über die aktuell eingespielte Version der CSN-Webseite.

Insgesamt umfassen diese über 7.700 Datensätze.

3.2.17. Obsolete Tabellen

DynShaper

Das CSN hat intern einen hervorragenden Netztechnik-Stand, jedoch ist die Außenanbindung über das Universitäts-Netzwerk beschränkt. Aus diesem Grund gab es eine Software, die das beschränkte Datenvolumen möglichst fair auf alle Nutzer verteilte. Diese Software nannte sich DynShaper und speicherte in insgesamt acht Tabellen ihre Daten. Da das CSN seine Anbindung an das URZ und selbiges seine Außenanbindung jedoch stetig ausbauen konnten, wurde von dieser Shaping-Maßnahme wieder Abstand genommen und ein festes tägliches Limit eingeführt. Somit wurden diese Tabellen überflüssig, sind dennoch in der Datenbank vorhanden.

Telefonnummern

Es bestehen zwei Tabellen, die zum Speichern der Telefonnummern der CSN-Mitarbeiter diente. Mittlerweile wurde jedoch dazu übergegangen diese Daten im internen Wiki zu speichern.

WLAN-Accounting

Bis das URZ die WLAN-Versorgung auf dem Campus übernahm, wurde dies vom CSN gepflegt. Der Server speicherte zur Authentifizierung der Nutzer Daten in einer Tabelle. Somit ist auch diese Tabelle bereits ungenutzt und enthält keinerlei Daten mehr.

3.3. Kritikpunkte

Viele Schwachstellen und Nachteile des aktuellen Datenbankschema wiederholen sich und sind in vier größere Kategorien zusammenfassbar.

Eine uneinheitliche Benennung von Tabellen und Spalten erschwert die Übersicht, Wartung und Pflege der Datenbank. Daneben finden sich eine Vielzahl an doppelten Daten oder Daten, die ebenso aus Beziehungen der Tabellen untereinander hervorgehen. Als dritter großer Kritikpunkt sind fehlende, falsche oder unnütze Verknüpfungen zwischen Tabellen bzw. Daten zu nennen. Letztlich sind noch obsolete Daten in einzelnen Spalten oder ganzen Tabellen vorhanden.

All diese Punkte sind eines sich ständig ändernden Umfeldes zu schulden, da oft kleinere Anpassungen vollzogen wurden und bestehende Strukturen nicht vollständig überarbeitet wurden.

Das nächste Kapitel beschäftigt sich genauer mit den einzelnen hier genannten Punkten und zeigt konkrete Beispiele dafür auf.

4. Problemanalyse

In diesem Kapitel wird auf konkrete Kritikpunkte eingegangen und Beispiele dafür werden aufgezeigt und erklärt.

4.1. Benennung

In der aktuellen Datenbank gibt es kein einheitliches Benennungsschema. Weder für Tabellen- noch für die Spaltennamen sind übergreifende Konventionen vorhanden oder ersichtlich.

4.1.1. Tabellen- und Spaltennamen

Englisch - Deutsch

Ein Großteil der Namen sind in Deutsch gehalten, aber es gibt eine nicht zu missachtende Anzahl an englischen Benennungen. Als Beispiele sind hier die Tabellen **traffic_current**, **traffic_overlimit** oder **hub_config_last** im Gegensatz zu **finanz_jahr**, **nutzerrechner** oder **standort** zu nennen.

Präfixe

Daneben haben zwei größere Gruppierungen von Tabellen teilweise ein einheitliches Präfix. Dies ist zum einen der Finanzbereich, bei dem lediglich zwei der 15 Tabellen nicht das Präfix „**finanz**_“ besitzen, und die Traffic-bezogenen Tabellen, die ein uneinheitliches Präfix besitzen. Zu letzterem sind beispielsweise die Tabellen **traffic_current** und **trafficstatistik** zu nennen.

Hierbei ist zu entscheiden, ob weitere Präfixe einzuführen sind oder ob diese nur uneinheitlich genutzten Präfixe entfernt werden sollen.

4.1.2. Fremdschlüssel

Aufgrund der besseren Verständlichkeit von Datenbankstrukturen ist es angebracht, Fremdschlüssel nach den referenzierten Primärschlüsseln zu benennen. Für die ID eines CSN-Nutzers (in der Tabelle **csn_nutzer** in der Spalte *person_id* hinterlegt) wurde fast in allen darauf Bezug nehmenden Tabellen die Bezeichnung *person_id* benutzt - außer in der Tabelle **nutzerrechner**, in welcher die Bezeichnung *sv* gewählt wurde. Dies sollte für „Systemverantwortlicher“ stehen und beschreibt die Funktion dieser Spalte treffend, jedoch dient sie keineswegs einer besseren Verständlichkeit, da man nicht auf einen Fremdschlüssel schließt, welcher auf einen Nutzer verweist.

4.2. Daten-Doppelungen

Im derzeitigen Zustand sind einige Daten doppelt in der Datenbank hinterlegt. Dies ist aus mehreren Gründen möglich, welche nachfolgend aufgelistet sind.

4.2.1. Mehrfach-Eintragung

Hier ist die Tabelle **kategorie** zu nennen, die die gleichen Daten wie **finanz_titel** enthält.

```
SELECT * FROM kategorie ORDER BY kategorie_id;

kategorie_id  |     katname     | katchar
--------------+-----------------+---------
            1 | Nutzungsgebuehr | N
            2 | Material        | M
            3 | Sonstiges       | S
[...]
(4 rows)

SELECT * FROM finanz_titel ORDER BY titel_id;

 titel_id |               titel                | titelchar
----------+------------------------------------+-----------
        1 | Nutzungsgebühr                     | N
        2 | Material                           | M
        3 | Sonstiges                          | S
[...]
(12 rows)
```

SQL-Anweisung 4.1: Redundante Daten bezüglich der Finanz-Abkürzungen

Ebenso sind Doppelungen in den Aufnahmeanträgen (siehe Trigger (Unterabschnitt 4.5.1)) oder den Tabellen rund um den Standort (**haus**, **standort**, **ev_bereich** - siehe Aus Verknüpfung ersichtlich (Unterabschnitt 4.2.2)) zu finden.

4.2.2. Aus Verknüpfung ersichtlich

Manche Tabellen enthalten Fremdschlüssel und zusätzliche Spalten, die Attribute speichern, welche bereits über ebendiese Fremdschlüssel verknüpfte Datensätze ersichtlich sind.

Ein Beispiel hierfür sind die beiden Tabellen **haus** und **standort**. In ersterer wird ein Haus mit Straße und Hausnummer definiert und in der zweiten Tabelle mittels dem

Fremdschlüssel **haus_id** darauf verwiesen. Damit müssten die Attribute „Straße“ und „Hausnummer“ nicht noch einmal in **standort** gespeichert werden, aber sie sind dennoch vorhanden.

4.3. Beziehungen

Oft werden Datensätze untereinander verknüpft. Durch die sich ändernden Anforderungen wurden im Nachhinein neue Datenstrukturen hinzugefügt und teilweise mit bestehenden verknüpft, weshalb gewisse Beziehungen beispielsweise doppelt vorkommen oder andere schlicht inhaltlich falsch sind.

4.3.1. Doppelte Beziehungen

In gewissen Tabellen sind Beziehungen doppelt abgebildet und sind somit überflüssig. Ein Beispiel hierfür ist die Tabelle **traffic_current**. Dort wird das viertelstündliche Datenaufkommen für eine IP-Adresse hinterlegt. Im Nachhinein wird dadurch der zugehörige Rechner bestimmt. Zusätzlich wird jedoch auch ein Verweis zur Person festgehalten. Dies ist jedoch nicht notwendig, da diese aus der Verknüpfung zum Rechner ersichtlich ist, welche wiederum auf den Rechnerinhaber verweist.

4.3.2. Inhaltlich falsche oder unnütze Beziehungen

Im geplanten Inventarsystem wurden zu jedem Gerät ein Lieferant vermerkt. Dies ist jedoch eher der falsche Ansatz, da der Lieferant meist noch Details wie Rechnungsnummer oder Kaufdatum benötigt, was aus dieser Beziehung nicht ersichtlich ist und

händisch gesucht werden muss. Somit ist diese Verknüpfung unnütz und der falsche Ansatzpunkt.

4.3.3. Aufwendige Beziehungsabbildungen

Wenn einem Eintrag in einer Tabelle mehrere Einträge einer anderen Tabelle zuzuordnen sind, dann wird für diese Abbildung eine separate Tabelle und nicht nur eine zusätzliche Spalte in der Ausgangstabelle zur Speicherung benötigt. Dies gilt lediglich für n:m-Beziehungen, nicht jedoch für 1:1- oder 1:n-Beziehungen. Für letztere gibt es jedoch eine solche aufwendige Abbildung der Beziehung über eine separate Tabelle. Ein Nutzerrechner kann an genau einer Anschlussdose innerhalb des Wohnheimes angemeldet und somit angeschlossen werden. Somit würde eine zusätzliche Spalte in der Tabelle **nutzerrechner** genügen um dies kenntlich zu machen. Umgesetzt wurde das jedoch über die Tabelle **angeschlossen_an**, in der je eine Dosen- und Rechner-ID zusammen abgespeichert werden.

4.3.4. Fehlende Beziehungskennzeichnung

Eine fehlende Kennzeichnung der Beziehungen innerhalb einer Datenbank schränkt die Funktionsweise dieser nicht ein, da die Verknüpfung der Datensätze oft in der programmierten Logik hergestellt wird. Für die Analyse, das Verständnis und die Wartbarkeit einer Datenbank ist dies jedoch eine Arbeitserleichterung und Hilfe. Deshalb sollten Beziehungen bzw. Fremdschlüssel kenntlich gemacht werden. Beispiele für eine fehlende Markierung sind die Tabellen **mailingliste** sowie **bezieht_liste**.

4.4. Obsolete Daten

Aufgrund zahlreicher Änderungen im CSN über die Jahre hinweg, hielten neue Softwaresysteme Einzug und alte nicht mehr benötigte wurden abgeschafft. Meist wurde dabei jedoch die recht komplexe Datenbank außer Acht gelassen und ungenutzte Strukturen blieben zurück.

4.4.1. Tabellen und Spalten

Nur teilweise eingeführte Systeme, wie das Inventarsystem, welches nie richtig genutzt wurde, führten unter anderem dazu, dass Tabellen um gewisse Spalten ergänzt oder sogar ganze Tabellen hinzugefügt wurden. Diese ungenutzen Strukturen erschweren die Arbeit mit der Datenbank. Die Tabellen der Kategorie Geräte (Unterabschnitt 3.2.4) zählen hierzu.

4.4.2. Ungenutzte Vererbung

Von Tabellen-Vererbung wurde bei der CSN-Datenbank ebenso Gebrauch gemacht und hat an den meisten Stellen auch Sinn. Es gibt jedoch auch Stellen an denen eine Vererbung im aktuellen Zustand zwecklos ist und somit nur einen gewissen ungenutzten Mehraufwand darstellt. Ein Beispiel hierfür ist die Tabelle **person** von der lediglich **csn_nutzer** erbt und von der innerhalb der aktuellen CSN-Software kein weiterer Gebrauch gemacht wird. An dieser Stelle würde demnach die Tabelle **csn_nutzer** vollkommen ausreichen.

4.5. Funktionen und Trigger

Mit Funktionen und Triggern sind in der Datenbank abgespeicherte Prozeduren gemeint. Funktionen erweitern die Bearbeitungsmöglichkeiten innerhalb der Datenbank und Trigger sind Funktionen, die auf gewisse Ereignisse (Erstellen, Bearbeiten oder Löschen von Datensätzen) reagieren und daraufhin ausgeführt werden.

4.5.1. Trigger

Daten werden von einem Programm aus in die Datenbank übertragen. Nach dem Eintragen werden sogenannte Trigger ausgelöst, die diese Daten dann weiterverarbeiten. Für Kleinigkeiten, wie das Aktualisieren des Wertes für zum Beispiel die letzte Bearbeitung ist dies sinnvoll. Jedoch gibt es auch ebensolche Trigger, die komplexe Datenverarbeitungen durchführen, die das eintragende Programm an sich schon ausführen könnte und somit nur für eine Verteilung der Programmlogik auf mehrere Stellen (Datenbank und externe Programme) zur Folge hat.

Ein Beispiel hierfür sind die Nutzeraufnahmeanträge. Diese werden vorübergehend in der Tabelle **neuantrag** gespeichert. Von dort werden die Daten auf zwei Tabellen verteilt - einmal in eine Tabelle für die Aufnahme eines neuen Nutzers und einmal in eine Tabelle für die Registrierung eines neuen Rechners.

4.5.2. Funktionen

Funktionen innerhalb der Datenbank sind für die Arbeit direkt in der Datenbank hervorragend geeignet. Jedoch soll dies nicht bzw. nur noch in Ausnahmefällen stattfinden.

Sämtliche Bearbeitungen der Daten soll von einem externen Programm oder Skript ausgeführt werden, was eine eigene Anbindung an die Datenbank hat und keinerlei komplexere Logik in der Datenbank nutzt - mit Ausnahme der Funktionen, die die Datenbank bereits mitbringt.

Die Funktionen in der Datenbank stehen unter keiner Versionsverwaltung, wie die restliche Software im CSN, und ist somit nicht schnell anpassbar und einfach wartbar. Aus diesem Grund sollen möglichst alle datenbankinternen Funktionen ausgelagert werden, um einen zentralen Anlaufpunkt für die Programmlogik zu haben.

5. Anforderungen an das neue System

Aus der Betrachtung der aktuellen Datenbasis (siehe Aktueller Datenbestand der Datenbank (Abschnitt 3.2)), der daraus folgenden Problemanalyse (Kapitel 4) und mehreren Rücksprachen mit dem CSN-Team sind folgende Anforderungen aufgestellt worden.

01 - Abbildung sämtlicher benötigter Daten

Das neue Schema soll natürlich sämtliche Daten speichern können, die das CSN benötigt. Darunter fallen verschiedene Kategorien, welche nachfolgend aufgeschlüsselt sind.

Nutzer Nutzerdaten, Nutzerstatus und Sperren im Speziellen, Nutzeraufgaben, Etagenverantwortliche, Trafficausnahmen

Wohnort Wohnort-Abbildung

Infrastruktur Abbildung der technischen und logischen Infrastruktur

Mietbestätigung Validierung der Ein-, Aus- und Umzüge

Rechner Allgemeine Rechnerdaten (MAC-Adressen, IP-Adressen), Nutzerrechner, Nutzerrechnerstatus

Traffic Trafficaufkommen pro Nutzerrechner, Trafficstatistiken

Finanzen Abbildung des internen Finanzsystems (Konten, Buchungen, Transfers, Abschlüsse, Kategorisierung), Studentenwerks-Abrechnungen

02 - Einfügen hilfreicher Daten

Bei der Benutzung des aktuellen Systems traten Probleme auf, die mit ungeeigneten Mitteln gelöst wurden. Ein Beispiel hierfür ist die Reservierung gewisser Rechnernamen, die durch das Eintragen von Dummy-Rechnern gelöst wurde. Hierzu sollen Datenstrukturen geschaffen werden, die die aktuellen Administrationsaufgaben des Netzwerkes hilfreich unterstützen.

03 - Betrachtung zukünftiger Probleme

Das zu entwerfende Schema soll auch in Hinblick auf die Zukunft sinnvoll gestaltet werden, damit eventuell auftretende Probleme unterbunden werden können. Ein Beispiel für diese Anforderung ist die Betrachtung der Vergabe von IPv6-Adressen und deren Speicherung.

04 - Benennungskonventionen schaffen und beachten

Es sollen einheitliche Benennungsregeln geschaffen werden, die für das gesamte Schema gelten. Somit soll gewährleistet werden, dass man sich schneller zwischen den einzelnen Strukturen des Schemas zurechtfindet.

05 - Mehrfache Datenspeicherung vermeiden

Daten, die doppelt eingetragen sind, bergen das Risiko, dass sie nicht korrekt gepflegt werden und somit falsch sind. Im neuen Schema soll demnach keinerlei Redundanz entstehen können.

06 - Sämtliche Beschränkungen im Schema aufführen

Sämtliche Beschränkungen (Constraints - Primär-, Fremdschlüssel, Eindeutigkeit von Werten, ...) sollen in der Datenbank definiert und nicht nur aus der dazugehörigen Programmlogik ersichtlich sein. Mit dieser Anforderung soll die Struktur bereits sehr gut ersichtlich sein, ohne eine komplexe Dokumentation des Schemas hinzuzuziehen.

07 - Keine überflüssigen Datenstrukturen schaffen

Es sollen ledig tatsächlich gebrauchte Datenstrukturen (Tabellen, Spalten, Datentypen, ...) geschaffen werden.

08 - Keine Datenbank-Prozeduren einführen

Da das CSN sich als Ziel gesetzt hat, sämtliche Skripte und Quelltexte aus der Datenbank zu entfernen und an einem zentralen Ort zu speichern, sollen keine Datenbank-Prozeduren geschaffen werden. Somit soll eine bessere Übersichtlichkeit über die Quelltexte geschaffen und die Versionierung jener erleichtert werden.

6. Änderungen

Die vorgesehenen Konsequenzen aus den Problemen und Kritikpunkten im vorherigen Kapitel Problemanalyse (Kapitel 4) werden in diesem Teil behandelt.

6.1. Benennungskonventionen

Die Benennung für Tabellen und Spalten ist ein wichtiger Schritt der Datenbank-Schemamodellierung. Er trägt zu einem besseren Verständnis der gespeicherten Daten bei und verdeutlicht die Struktur. Deshalb sollten gewisse Regeln für die Benennung aufgestellt werden, die bei jeder neuen Tabelle und deren Spalten eingehalten werden. Diese Regeln oder Konventionen sind nachfolgend aufgelistet und sind so für das neue Schema anzuwenden.

- Die Tabellen haben den Singular des repräsentierten Objektes als Namen zu tragen. Ein Beispiel ist hier die Bezeichnung **User** anstatt **Users**.
- Die Spalten sind nach dem Singular des repräsentierten Attributes zu wählen. **FirstName** ist anstatt von **FirstNames** zu wählen.
- Eine Mischung von deutschen und englischen Begriffen ist zu vermeiden. Um die Mitarbeit am CSN auch ausländischen Studenten zu vereinfachen wurde sich innerhalb des CSNs auf die englische Sprachwahl festgelegt. Zusätzlich machen

englische Begriffe das Lesen anderer CSN-Quelltexte einfacher, da Programmiersprachen meist an das Englische angelehnt sind und somit einheitlicher sind. Es werden ausschließlich englische Begriffe wie **User** anstatt von **Nutzer** gewählt.

- Die Tabellen- und Spaltennamen sind im PascalCase [BS09, S. 28] zu schreiben. Das heißt, dass Wörter z.B. nicht durch Unterstriche sondern durch deren Großschreibung getrennt werden. Ebenso ist der erste Buchstabe groß zu schreiben. Diese Schreibweise wird auch Binnenmajuskel genannt und ist z.B. in **UserHasStatus** ersichtlich. Ein Gegenbeispiel hierfür wäre **user_has_status**.
- Der Primärschlüssel einer Tabelle endet prinzipiell auf *ID*.
- Die Fremdschlüssel haben den gleichen Namen wie ihre zugehörigen Primärschlüssel.

6.2. Vermeidung von Doppelungen

In Daten-Doppelungen (Abschnitt 4.2) ist ersichtlich, dass es einige Strukturen in der aktuellen Datenbank gibt, die gleiche Daten enthalten. Dies birgt eine Inkonsistenzgefahr und ist zusätzlich schwerer zu pflegen. Deshalb wird beim Entwurf des neuen Schemas darauf geachtet, möglichst viele solcher Strukturen zu entfernen bzw. nicht zu übernehmen. Weiterhin sollte überprüft werden, welche Doppelungen in separate Tabellen auslagerbar sind.

6.3. Überarbeitung der Beziehungen

Wie aus Beziehungen (Abschnitt 4.3) erkennbar, gibt es aktuell noch Schwachstellen bezüglich der Beziehungen. Das Problem, dass Verknüpfungen doppelt abgebildet wer-

den, ist fast immer vermeidbar. Jedoch gibt es auch hierfür Ausnahmen, in denen man es auf Grund aufwendiger Rekonstruktion von Beziehungen über eine Vielzahl von Tabellen in Kauf nimmt, dass eine Verknüpfung doppelt abgebildet wird.

Sämtliche inhaltlich falschen und unnützen Beziehungen sind aus dem neuen Schema zu entfernen. Beziehungen, die zurzeit zu aufwendig abgebildet werden, müssen einfacher umgesetzt werden.

Das Problem der fehlenden Kennzeichnung von Beziehungen wird automatisch beim Neuentwurf entfallen, da bei der Planung diese explizit angegeben werden, damit die Struktur verständlich ist. Die fehlenden Markierungen resultieren aus dem nachträglichen und spontanen Hinzufügen in das bestehende System ohne die Durchführung von Planungsschritten. Die Beziehungen sind dadurch zwar aus dem Quelltext ersichtlich, jedoch nicht explizit in der Datenbank hinterlegt.

6.4. Entfernung obsoleter Daten

Tabellen, bei denen sich herausgestellt hat, dass sie nicht mehr benötigt werden, brauchen bei dem Neuentwurf des Schemas nicht weiter beachtet werden und entfallen somit von selbst. Bei obsoleten Spalten passiert das auf die gleiche Art und Weise. Der Sinn und Zweck einer jeden Spalte wird hinterfragt und bei der Feststellung der Obsoleszenz wird diese nicht weiter beachtet.

6.5. Funktionen und Trigger

Sämtliche in der Datenbank hinterlegten Funktionen werden nicht übernommen, da das CSN sämtliche Bearbeitungen von Daten in ein separates System auslagert, damit diese an einem zentralen Ort pflegbar bleiben.

Nahezu ähnlich wird mit Triggern zu Tabellen verfahren. Lediglich Trigger, die das Datum der letzten Änderung eintragen werden hinzugefügt.

7. Modell

In diesem Kapitel wird auf die einzelnen Tabellen des neuen Modells detailiert eingegangen und Gründe für eben diese Struktur aufgezeigt. Die Regeln aus Benennungskonventionen (Abschnitt 6.1) werden nicht noch einmal explizit aufgeführt, da sie als gegeben betrachtet werden.

Eine gesamte Schema-Darstellung befindet sich im Anhang.

7.1. Datentypen

Jede Spalte einer Tabelle hat einen gewissen Datentyp. Für das neue Schema der CSN-Datenbank, welches in diesem Kapitel vorgestellt wird, werden folgende Datentypen verwendet. Zur Feststellung des passenden Datentyps wurde das Kapitel 8 („Data Types“) der PostgreSQL-Dokumentation genutzt.

Primär- und Fremdschlüssel

Sämtliche Primärschlüssel sind vom Typ „serial“, sofern dies im Text nicht anders angegeben ist. Dieser stellt sicher, dass es über 2,1 Milliarden eindeutige Werte gibt, was

für jeden Primärschlüssel in unserer Datenbank ausreichend ist. Eine zusätzliche Eigenschaft von „serial“ ist das automatische Vergeben eines eindeutigen Wertes.

Die auf Primärschlüssel verweisenden Fremdschlüssel sind demzufolge vom Typ „integer“, um den Wertebereich abdecken zu können.

Datums- und Zeitangaben

Sämtliche Zeitangaben sind vom Datentyp „timestamptz“, der neben dem Datum und der Uhrzeit auch die Zeitzone speichert. Falls lediglich ein Datum - also eine maximale Auflösung von einem Tag - benötigt wird, ist auf den Datentyp „date“ zurückzugreifen.

Text und Wahrheitswert

Sämtliche textuellen Werte, die gespeichert werden sollen, verwenden den Datentyp „varchar“. Dies ist eine Empfehlung der PostgreSQL-Dokumentation für textuelle Daten, da es keine wesentlichen Geschwindigkeitsvorteile zwischen den drei infrage kommenden Datentypen gibt. [Pos12, Seite 110]

Für Wahrheitswerte wird der darauf konzipierte Datentyp „boolean“ eingesetzt.

Spezielle Datenformate

Für das neue Schema der CSN-Datenbank wurde von drei besonderen Datentypen, die PostgreSQL zur Verfügung stellt, Gebrauch gemacht. Zum einen sind dies der Datentyp für IP-Adressen und Netzwerke „inet“, welcher IPv4 und IPv6 handhaben kann.

Der zweite Typ ist „macaddr“, welcher für die Speicherung von MAC-Adressen konzipiert ist, und der letzte ist „money“, welcher Geldbeträge mit zwei Nachkommastellen speichert.

Aufzählungstypen

Falls es eine Wertemenge gibt, die eine Spalte annehmen kann, dann empfiehlt sich der Aufzählungstyp. Dieser wird vorher definiert und kann dann wie jeder andere Datentyp in der Tabellendefinition benutzt werden.

```
CREATE TYPE NetworkClass AS ENUM (
    'Default',
    'Quarantine'
);

CREATE TABLE "VLAN" (
    "VLANID"            serial PRIMARY KEY,
    "HouseID"           integer NOT NULL REFERENCES "House"( "HouseID" )
       ,
    "SubnetworkID"      integer NOT NULL REFERENCES
                           "Subnetwork"( "SubnetworkID" ),
    "NetworkClass"      NetworkClass NOT NULL
);
```

SQL-Anweisung 7.1: Aufzählungstyp

Dies ist ein Beispiel, wie solch ein Aufzählungstyp definiert und eingesetzt wird. In die Spalte *NetworkClass* der Tabelle **VLAN** kann nun entweder „Default“ oder „Quarantine“ als Wert eingetragen werden. Für alle anderen Werte schlägt das Eintragen fehl.

Falls die Wertemenge eines solchen Typs sich ändert, kann das „ALTER TYPE“-Statement genutzt werden. Dies ermöglicht ein umfangreiches Bearbeiten des Datentyps, wie das Hinzufügen, Ändern, Entfernen oder Umsortieren der Elemente.

7.2. Nutzer

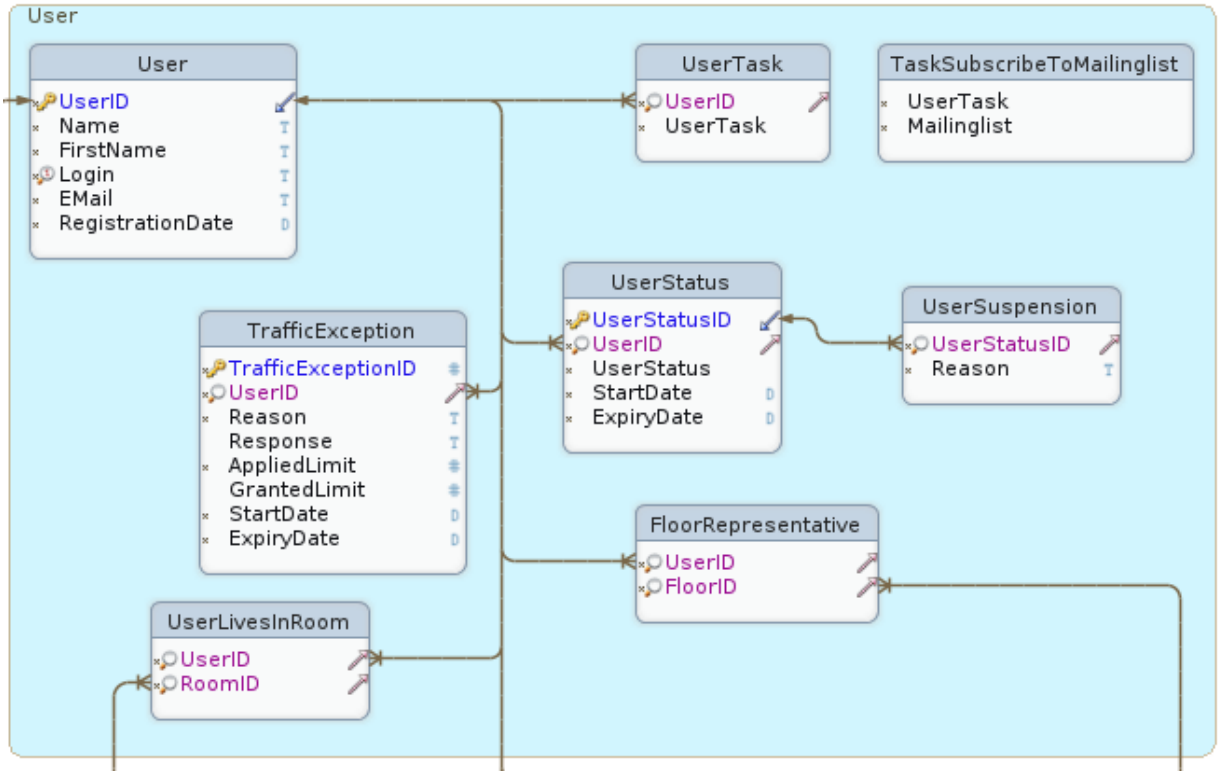

Abbildung 7.1.: Nutzer-Tabellen

Die Kerntabelle dieser Kategorie ist die Tabelle **User**. Sie enthält die Informationen über den Namen, die E-Mail-Adresse, das URZ-Login und das Anmeldedatum. All dies ist mit einer eindeutigen Identifikationsnummer verknüpft, damit auf einen speziellen Nutzer verwiesen werden kann. Wie man sieht, wurden alle Attribute außer der Fremdschlüssel für das Zimmer übernommen. Das ist darin begründet, dass die Verknüpfung zwischen Nutzer und seinem Zimmer in eine separate Tabelle Namens **UserLivesInRoom** ausgelagert wurde. Damit ist es möglich, dass ein Nutzer in mehreren Zimmern wohnen kann.

7.2.1. Status

Ein Nutzer kann mehrere Status haben. Darunter zählt zum Beispiel eine verhängte Sperre oder ob der Nutzer aufgrund eines Viren- oder Wurmbefalls in Quarantäne ist. Hierfür sind zwei Tabellen zuständig. All diese Status sind in einem Aufzählungstyp mit dem Namen *UserStatus* festgehalten. Dieser wiederum wird in der Tabelle **UserStatus** neben einer Status-ID, der Nutzer-ID und der Gültigkeitsdauer festgehalten.

Für gewisse Sperren ist es nötig, dass ein Grund hinterlegt wird, was der Fall bei Viren- oder Wurmbefall eines Nutzer-Rechners ist. Die Gründe sind einem individuellen Status (ein Eintrag in **UserStatus**) zugeordnet und in der Tabelle **UserSuspension** gespeichert.

7.2.2. Aufgaben

Ein Nutzer kann innerhalb des CSN-Teams ehrenamtliche Aufgaben übernehmen. Diese Aufgaben werden meist nicht nur von einer einzelnen Person wahrgenommen, was die Notwendigkeit eines geeigneten Kommunikationsweges erfordert. Der größte Teil der Kommunikation findet über interne Mailinglisten statt, welche den einzelnen Aufgabenbereichen zugeordnet sind.

Die einzelnen Aufgabengebiete sind im Aufzählungs-Datentyp *UserTask* und die Mailinglisten in *Mailinglist* festgehalten. Die Verknüpfung einer Aufgabe zu einem Abonnement einer Mailingliste wird in **TaskSubscripeToMailinglist** gespeichert. Dies ist notwendig, da gewisse Aufgaben mehreren Mailinglisten zuordenbar sein müssen, aber auch eine Mailingliste in unterschiedlichen Aufgabengebieten genutzt werden kann. Ein Nutzer kann mehrere Aufgaben innerhalb des CSNs haben, wie zum Beispiel Finanz- oder Datenbankverantwortlicher. Diese Beziehung wird mittels der Tabelle **UserTask** definiert.

7.2.3. Etagenverantwortliche

Die Etagenverantwortlichen sind die ersten Ansprechpartner im CSN für die Nutzer. Meist betreut eine Person eine oder mehrere Etagen (vergleiche Standort (Abschnitt 7.3)) in der Nähe seines eigenen Zimmers. Welche Etagen eine Person betreut, ist in der Tabelle **FloorRepresentative** ersichtlich, die Verknüpfungen zwischen einer Person und einer Etage enthält.

7.2.4. Vergleich zum alten Schema

Zu den kleinen Änderungen zählen die Auslagerung des Nutzer-Wohnortes und der Entfernung der zusätzlichen ID im Nutzerstatus.

Zusätzlich sind die Tabelle für die verschiedenen Status des Nutzers, die vorhandenen Aufgabengebiete im CSN und die Auflistung der Mailinglisten entfallen, welche nun über einen Aufzählungstyp realisiert sind. Somit ist die Lokalisierung der Begriffe einfacher und zentraler möglich. Sämtliche Übersetzungen befinden sich nun ausschließlich im Quellcode und nicht wie im alten Schema teilweise auch in der Datenbank.

7.3. Standort

Diese Kategorie wurde einer größeren Änderung unterzogen, da hier einige Daten doppelt in der alten Datenbank vorhanden waren. Sie enthält die drei Tabellen **House**, **Floor** und **Room**, welche die Wohnheime und ihre Zimmer sehr gut abbilden.

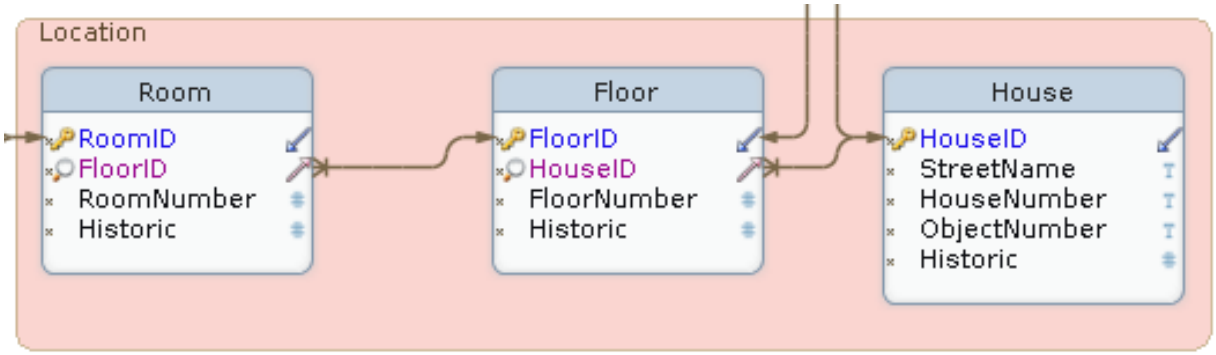

Abbildung 7.2.: Standort-Tabellen

Jeder Eintrag in **House** besitzt eine ID, einen Straßennamen, eine Hausnummer sowie die vom StuWe vergebene Obekt-Nummer. Die nächstkleinere Einheit ist die Etage, welche in **Floor** gespeichert ist. Jede Etage hat eine ID und eine Etagennummer, sowie einen Verweis auf das Haus, in dem sie sich befindet. Die kleinste Einheit in dieser Kategorie ist das Zimmer, dass in **Room** abgebildet ist und ebenso eine ID und eine Zimmernummer hat und sich in einer Etage befindet.

In jeder dieser drei Tabellen gibt es zusätzlich noch eine Spalte *Historic*. Diese ist per Standard auf „false“ gesetzt. Für den Fall, dass ein Zimmer, eine Etage oder gar ein Haus nicht mehr existiert - zum Beispiel auf Grund einer Renovierung -, dann wird dieser Wert auf „true“ gesetzt und zeigt an, dass das Zimmer vorhanden war, aber nun nicht mehr existiert. Das Vorgehen ist notwendig, damit für den Fall, dass z.B. ein Raum nicht mehr existiert und normalerweise aus der Datenbank gelöscht wird, die bisherigen Mieter-Listen vom Studentenwerk (vergleiche Studentenwerk-Abrechnungen (Abschnitt 7.7)) nicht an Gültigkeit verlieren.

7.3.1. Vergleich zum alten Schema

Wie bereits erwähnt waren hier sehr viele doppelte Daten zu finden. Die drei Tabellen **haus**, **standort** und **ev_bereich** enthielten allesamt Daten, die in wenigstens einer der anderen Tabellen zu finden war. In allen drei war zum Beispiel Hausnummer und Straßenname gespeichert. Zusätzlich speicherten **standort** und **ev_bereich** die Etagennummer. Weiterhin war in **standort** ein Verweis auf **ev_bereich** hinterlegt. Durch die Neugestaltung dieser Tabellen gibt es nun keinerlei doppelten Daten mehr.

Das Kürzel des Wohnheims ist nun gleichzeitig die ID in der Tabelle **House**, da dieses bereits eindeutig ist.

7.4. Infrastruktur und VLAN

Die beiden Kategorien Infrastruktur und VLAN stellen die hardwareseitige (Infrastruktur) und softwareseitige (VLAN) Abbildung der Netzwerkinfrastruktur dar. Dies sind einmal die Netzwerkgeräte und die logische Unterteilung in Netzwerke.

7.4.1. VLAN

Ein VLAN ist ein virtuelles Teilnetz eines real existierenden Netzwerkes. Darüber ist es möglich die einzelnen Nutzer in gewisse Nutzergruppen zu unterteilen. Hauptsächlich wird dies zur Isolation der in Quarantäne befindlichen Rechner benutzt.

Innerhalb des CSNs besitzt jedes Wohnheim seinen eigenen IP-Adressraum - Subnetz genannt. Ebenso hat jedes dieser Subnetze ein eigenes Gateway, über welches der Zugang

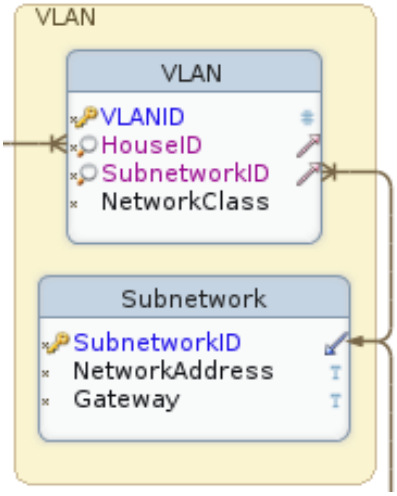

Abbildung 7.3.: VLAN-Tabellen

zum restlichen Netz des CSN, dem Universitätsnetzes und dem Internet abgewickelt wird. All diese Definitionen des Adressbereiches, sowie des Gateways sind in **Subnetwork** gespeichert.

Die Unterteilung in normale Nutzung und Quarantäne wird über den Aufzählungstyp *NetworkClass* definiert.

Die genaue Definition eines VLANs findet in **VLAN** statt. Die VLANs werden durch eine eindeutige Verknüpfung von Netzklasse - repräsentiert durch obigen Aufzählungstyp - , Subnetz und Haus und der Zuordnung einer ID definiert. Diese ID wird von einer Software benutzt, um die VLANs aufzubauen.

7.4.2. Infrastuktur

Die hardwareseitige Abbildung besteht aus den vier Tabellen **Stack**, **Unit**, **Port** und **Socket**. Die hardwaretechnisch größte zusammengehörige Einheit stellt ein Stack dar. Dieser ist mit einem Eintrag in **Host** verknüpft, woher er seine Rechnereigenschaften wie

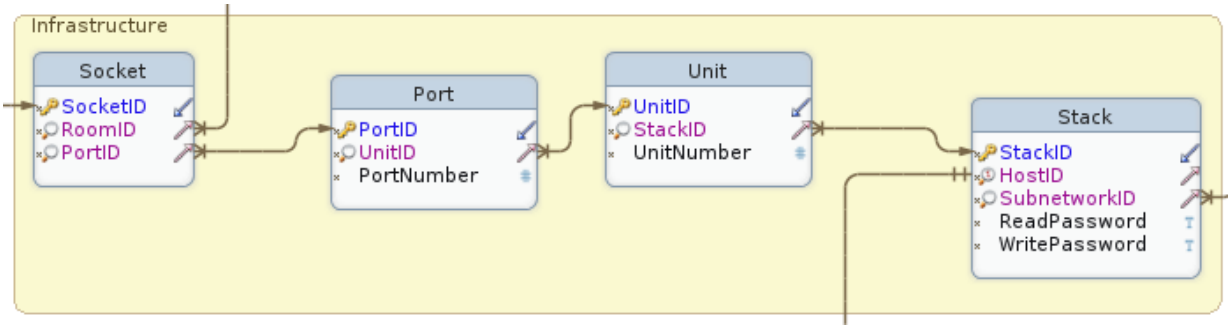

Abbildung 7.4.: Infrastruktur-Tabellen

Name, Anmeldedatum und MAC-Adresse bezieht. Zusätzlich wird noch ein Verweis auf das zugehörige Subnetz und die Zugangsdaten für die Konfiguration des Stacks definiert.

Ein Stack besteht aus einer oder mehreren sogenannten Units. Jede dieser hat eine ID, einen Verweis auf den Stack, in dem sie eingebaut ist und eine Unit-Nummer. Dies ist in **Unit** hinterlegt.

Eine Unit besitzt eine Vielzahl an Ports, an welchen dann ein Kabel zu einer Dose jedes Wohnheimzimmers steckt. Jeder dieser Ports ist in **Port** gespeichert und besitzt eine ID, einen Verweis auf die zugehörige Unit und eine Nummer.

Sämtliche Anschlußdosen aller Wohnheimzimmer werden in der Tabelle **Socket** hinterlegt. Diese speichert eine ID, einen Verweis auf den Raum, in dem sich die Dose befindet, sowie den Port, an dem sie angeschlossen ist.

Die Attribute *SocketID* und *UnitID* sind nicht vom Typ „serial“ sondern „varchar“. Dies begründet sich darin, dass die Dosen in jedem Wohnheimzimmer beschriftet sind und diese Beschriftung zur besseren Orientierung für die Nutzer und die CSN-Mitarbeiter übernommen wurde. Die *UnitID* setzt sich aus dem Stackstandort und der Unitnum-

mer zusammen, womit schneller aus der *UnitID* auf die konkrete Hardware vor Ort geschlossen werden kann.

„Ve64#0#15.2“ ist die Dosenbezeichnung für die Dose im Wohnheim Vettersstraße 64, Erdgeschoss, Zimmer 15, Dose 2. „mpr–52-kr#4“ ist die Unit-Bezeichnung für die Unit im Wohnheim Vettersstraße 52, Keller, rechter Serverschrank, Unit-Nummer 4.

7.4.3. Vergleich zum alten Schema

Die Grundstruktur wurde größtenteils übernommen. Jedoch wurde die alte Tabelle **hub** mit der treffenderen Bezeichnung **Stack** versehen. Die Vererbung von einer allgemeinen Rechner-Tabelle wurde aufgehoben und stattdessen ein Verweis auf einen Eintrag in selbiger hinterlegt. Weiterhin ist die Tabelle **kabeltyp** nicht mehr in Gebrauch und wurde ebenso wie nicht mehr benötigte Spalten der anderen Tabellen entfernt.

7.5. Rechner

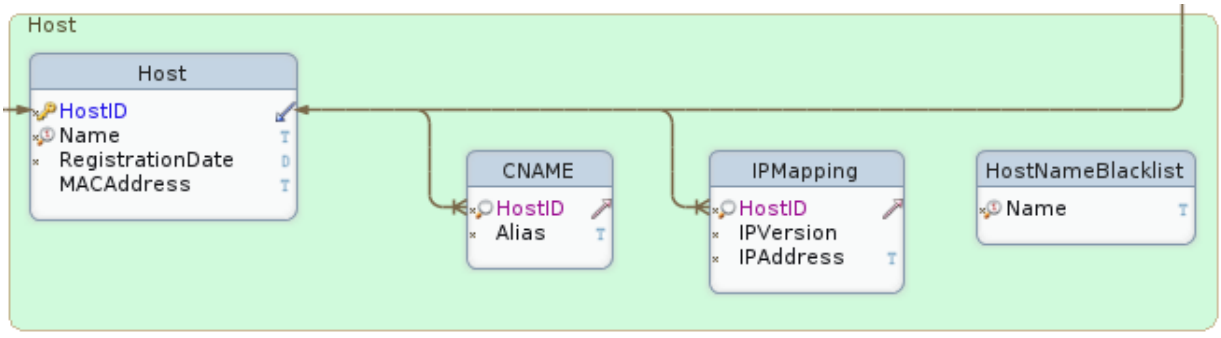

Abbildung 7.5.: Rechner-Tabellen

Innerhalb des Netzwerks des CSNs gibt es natürlich eine Vielzahl an Rechner, welche in zwei Kategorien unterteilt werden. Zum einen die Rechner der Nutzer und zum anderem vom CSN betriebene Rechner, die für die Funktionsfähigkeit des Netzes und weitere Dienste zuständig sind. Beide weisen gleiche Attribute auf, wie einen Hostnamen, ein Anmeldedatum, die IP- und MAC-Adresse, sowie eine eindeutige Nummer (Host-ID). Diese Informationen - mit Ausnahme der IP-Adresse - sind in **Host** hinterlegt. Für die zukünftige Bereitstellung von IPv6 wird die Zuordnung einer IP-Adresse in der Tabelle **IPMapping** geregelt. Jedem Host können hier IP-Adressen zugeordnet werden. Dazu wird neben der Host-ID die Version der IP-Adresse (IPv4 oder IPv6) und die IP-Adresse selbst hinterlegt. Während der Umstellung kann dann beides parallel ausgelesen werden. Sobald nur noch IPv6 verwendet wird, kann diese separate Tabelle in eine Spalte in **Host** übernommen werden. Die Spalte **IPVersion** in **IPMapping** stellt einen Aufzählungstyp mit zwei verschiedenen Einträgen - IPv4 und IPv6 - dar.

Für zusätzliche DNS-Aliase ist **CNAME** verantwortlich. Dort kann ein Alias angelegt und einem Rechner zugewiesen werden.

Damit innerhalb des CSNs Nutzer nicht Hostnamen wählen dürfen, die sich das CSN für eigene Rechner vorbehalten will oder die strittig sind, gibt es eine Hostnamen-Blacklist. In der Tabelle **HostNameBlacklist** können diese Namen eingetragen werden.

7.5.1. Nutzerrechner

Die Tabelle **UserHost** enthält ebenso wie **Stack** einen Verweis auf einen Eintrag in **Host** und verfügt dadurch über die Grundeigenschaften eines Rechners. Zusätzlich sind noch die Verweise für den Nutzer, dem der Rechner gehört, die Dose, an dem der Rechner angeschlossen ist, und die maximale Geschwindigkeit des Anschlussports hinterlegt.

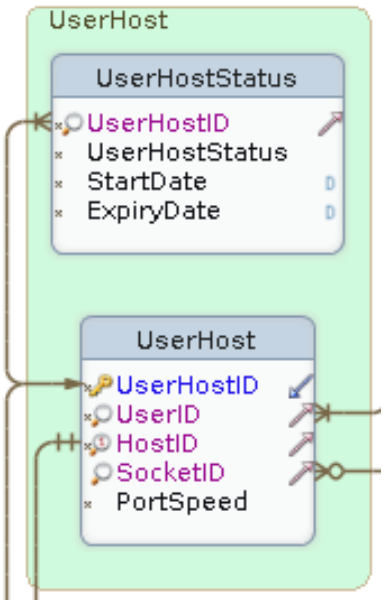

Abbildung 7.6.: Nutzerrechner-Tabellen

Jeder Nutzerrechner kann einen gewissen Status haben, was über die Tabelle **UserHostStatus** abgebildet ist. In dieser werden alle Status mit der Rechner-ID verknüpft und ein Gültigkeitszeitraum zugeordnet. Die Status sind über einen Aufzählungstypen in der Spalte *UserHostStatus* hinterlegt.

7.5.2. Vergleich zum alten Schema

Die größten Änderungen stellen die Eingliederungen der alten Tabelle **angeschlossen_an** in die neue **UserHost** und **nutzerrechner_status** in den Aufzählungstypen *UserHostStatus* in **UserHostStatus** dar. Es wurden neue Tabellen für die zukünftige Bereitstellung von IPv6 im CSN und das Blacklisting von Hostnamen erstellt. Ansonsten sind lediglich ungenutzte IDs entfernt worden.

7.6. Traffic

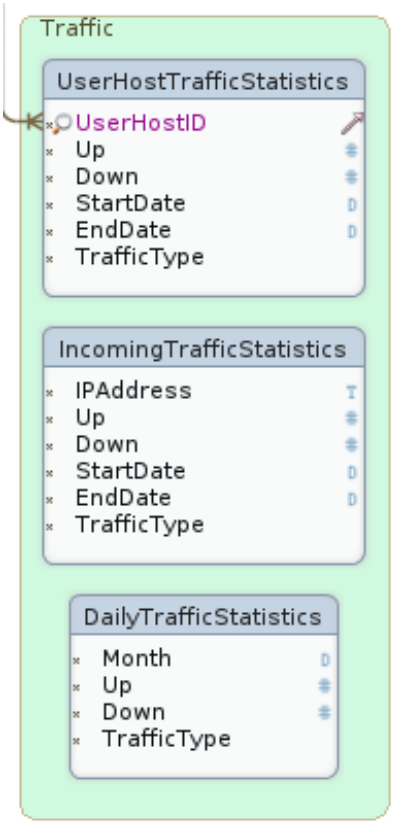

Abbildung 7.7.: Traffic-Tabellen

Innerhalb des CSNs wird das Datenaufkommen viertelstündlich erfasst. Die Information für hoch- und heruntergeladene Daten wird zusammen mit der IP-Adresse, die sie verursachte, und der Zeit in die Tabelle **IncomingTrafficStatistics** gespeichert.

Daraus werden die Datensätze einzeln entnommen und einem gewissen Nutzerrechner zugeordnet. Diese Zuordnung wird dann in die Tabelle **UserHostTrafficStatistics** übernommen, wo die IP-Adresse durch eine Rechner-ID ersetzt wird. Insgesamt bleiben diese Daten dann für 10 Tage in unserer Datenbank.

Daneben wird aus **IncomingTrafficStatistics** noch eine Statistik des gesamten CSNs für jeden Tag erstellt und in **DailyTrafficStatistics** gespeichert.

Die in allen drei Tabellen vorkommende Spalte *TrafficType* ist von einem Aufzählungstypen, der zwei Werte für die Kennzeichnung des Traffics besitzt - einen für externen und einen für internen.

7.6.1. Vergleich zum alten Schema

Die Tabellen **traffic_current** und **traffic_log** sind nun zusammengefasst, da sie jeweils gleiche Daten für unterschiedliche Zeiträume enthielten.

Weiterhin wurden in **IncomingTrafficStatistics** keine Spalten übernommen, die für die Zuordnung des genauen Hostnamens und der Host-ID zuständig sind, da dies über die IP-Adresse erfasst wird und dann in der Tabelle **UserHostTrafficStatistics** ersichtlich ist. Die Markierung, ob ein Datensatz in **IncomingTrafficStatistics** bearbeitet wurde oder noch nicht, entfällt ebenso, da die bearbeiteten Datensätze schlicht gelöscht werden.

7.7. Studentenwerk-Abrechnungen

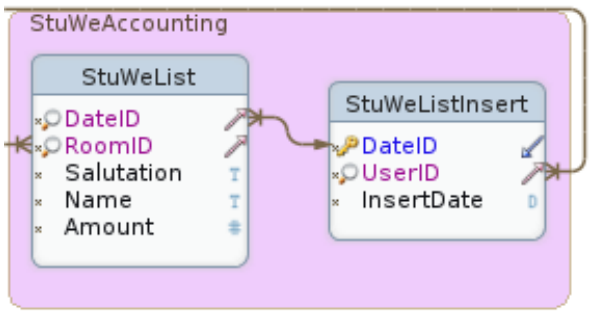

Abbildung 7.8.: StuWe-Tabellen

Vom Studentenwerk werden für uns Gelder für die Unterhaltung des Netzwerks über die Miete eingenommen. Die Daten für den Betrag, den ein Nutzer in einem gewissen

Monat über die Miete an uns zahlt, sind aus der Tabelle **StuWeList** ersichtlich. Diese Liste wird monatlich von Studentenwerk an das CSN übermittelt und muss - mittels Skript - in die Datenbank eingelesen werden. Sie enthält die Daten für den Monat, den Wohnort, die Anrede, den Name und den Betrag. Um nachzuvollziehen, wer diese Liste eingelesen hat, wird das Import-Datum, sowie ein Verweis auf den Nutzer und die Liste in **StuWeListInsert** hinterlegt.

Bei diesen Tabellen gab es keinerlei inhaltliche Änderungen, weshalb kein Vergleich zum alten Schema aufgeführt ist.

7.8. Einzugs-, Auszugs- und Umzugs-Validierung

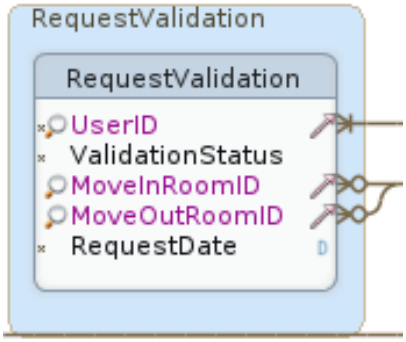

Abbildung 7.9.: Validierungstabellen

Die Tabelle **RequestValidation** enthält sämtliche Ein-, Aus- und Umzüge aller CSN-Nutzer, die vom Studentenwerk noch nicht bestätigt worden sind. Es wird der Nutzer, der Validierungsstatus (z.B. Einzug, Umzug - vom Typ Aufzählung), das Einzugszimmer, das Auszugszimmer sowie das Datum vermerkt. Somit können Nutzer auch in mehrere Zimmer einziehen und z.B. aus nur einem ausziehen oder umziehen.

7.8.1. Vergleich zum alten Schema

Die Status werden nun nicht mehr in einem Textfeld gespeichert, sondern haben einen eigenen Aufzählungstyp. Somit können hier Falscheingaben erkannt werden.

7.9. Finanzen

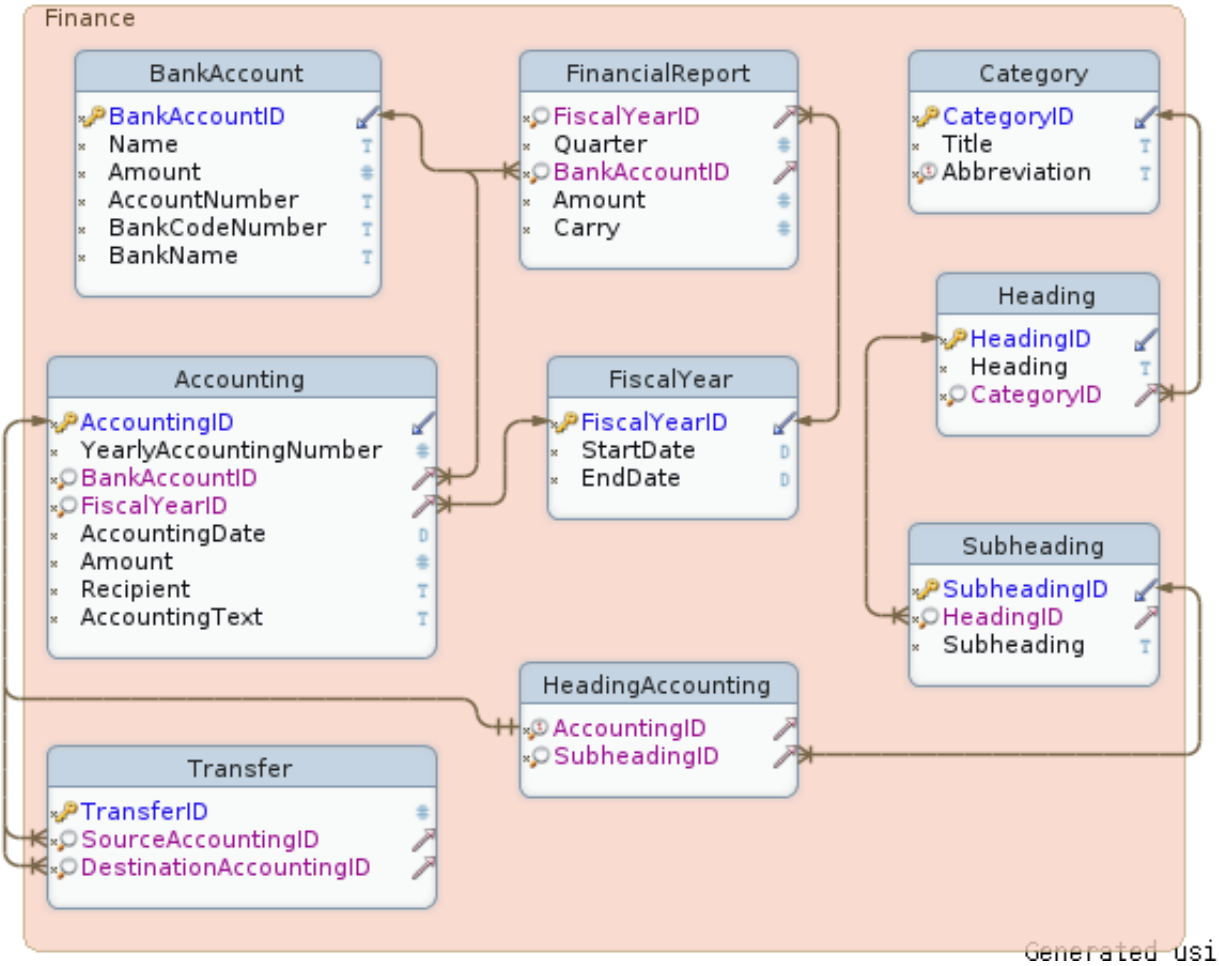

Abbildung 7.10.: Finanz-Tabellen

In **BankAccount** sind sämtliche CSN-Konten mit einer ID, Name, aktuellen Betrag, Kontonummer, Bankleitzahl und Bankname eingetragen.

Unsere Buchungen werden quartalsweise in einem Abschluss zusammengefasst. Dieser Abschluss beinhaltet für jedes unserer Konten den Betrag und den Übertrag aus dem vorherigen Quartal, sowie die Angabe des Quartals und Jahres. Gespeichert werden diese Informationen in **FinancialReport**. Das Finanzjahr ist jeweils in **FiscalYear** mit Beginn und Ende, sowie einer ID gelistet.

7.9.1. Buchungen

Die Haupttabellen der Finanz-Kategorie sind die Buchungstabellen **Accounting** und **HeadingAccounting**. Erstere enthält eine ID, die Jahresbuchungs-ID, die Konto-ID, das Buchungsdatum, das Finanzjahr, den Betrag, den Empfänger und einen Buchungstext. Für Buchungen, die einem Haushaltstitel zugeordnet sind, wird der Eintrag in **Accounting** mit einem Untertitel mittels der Tabelle **HeadingAccounting** verknüpft.

Für den Fall, dass ein Transfer von einem CSN-eigenen Konto zu einem anderen CSN-eigenen Konto stattfindet, werden diese beiden Buchungen (Aus- und Einzahlung) über deren Buchungs-IDs in **Transfer** gespeichert.

7.9.2. Titel und Untertitel

In **Subheading** werden zu jedem Untertitelname eine ID und ein Verweis auf seinen zugehörigen übergeordneten Titel gespeichert. Diese werden wiederum mit Name, ID und einem Verweis auf die Kategorie in **Heading** hinterlegt. Die Kategorieinformationen wie Name und Abkürzung werden in **Category** abgelegt.

7.9.3. Vergleich zum alten Schema

In **BankAccount** wurden die Spalten für IBAN, Swift-BIC und, ob ein Konto aktiv ist, nicht übernommen, da sie nicht mehr genutzt werden. Ebenso wurde derselben Tabelle die Spalte Übertrag entfernt, da dieser aus **FinancialReports** ersichtlich ist. Bei den Buchungen sind die Attribute kommentar und zu_pruefen entfallen, da sie nicht genutzt werden. Weiterhin ist in **HeadingAccounting** die Spalte host_id nicht mehr enthalten, welche früher für das händische Einsammeln der CSN-Gebühren genutzt wurde.

Es gibt keine separate Tabelle für Buchungen, die mit Haushaltstiteln verknüpft sind, da diese im neuen Schema über eine vorhandene Verknüpfung von Buchung und Haushaltstitel in **HeadingAccounting** erkenntlich sind.

Die alten Tabellen **kategorie** und **finanz_titel** enthielten gleiche Werte, welche nun in die Tabelle **Category** zusammengefasst hinterlegt sind.

7.10. Weggefallene Tabellen

Es gab eine Vielzahl an Tabellen, die aktuell nicht mehr genutzt werden. Nachfolgend sind diese mit Grund des Wegfalls aufgelistet.

Die Tabellen der Kategorie „Geräte“ entfallen vollständig. Dazu gehören **ansprechpartner**, **geraete**, **geraetestatus**, **geraeteart**, **hersteller**, **kontakt** und **rechnung**. Wie bereits erwähnt wurden sie aufgrund der zu großen Menge an Informationen, die eingepflegt werden sollten (z.B. *firmware* oder *abschreibungszeitraum*), kaum genutzt. Diese Informationen werden vom Finanzberater des CSNs rechtlich verbindlich eigenständig gepflegt.

Das Amt des Hausverantwortlichen gibt es seit mehreren Jahren nicht mehr. Deshalb wurden die zugehörigen Tabellen **hv_bereich** und **hv_betreut** und damit auch deren enthaltenen Informationen nicht übernommen.

Die DynShaper-Software ist ebenso seit mehreren Jahren außer Betrieb, weshalb deren Datenbestand ebenso nicht übernommen wird. Hierzu zählen die Tabellen **ds_vorhersagen_current**, **ds_vorhersagen**, **ds_vorhersagen_tmp**, **ds_gruppen_v2**, **ds_nutzergruppen_v2**, **ds_statistik_v2**, **ds_ausnahmen_v2** und **ds_allgemein_v2**.

Die Telefondaten der CSN-Team-Mitglieder werden seit geraumer Zeit innerhalb des internen Wikis gepflegt. Die Tabellen **team_telefon** und **telefon_typ** werden somit nicht in das neue Schema eingearbeitet.

Sämtliche Nachrichten des CSN sollen in Zukunft über eine einheitliche Schnittstelle kommuniziert werden, womit diese Aufgabe in den CSN-Blog übergeht. Dadurch entfällt die Tabelle **aktuelle_nachrichten**.

Die Tabelle **person** ist ebenso ungenutzt, da lediglich **csn_nutzer** von ihr erbt und keinerlei Daten ausgelesen noch hineingeschrieben werden. Somit reicht es, wenn **csn_nutzer** übernommen wird.

In Verbindung mit dem Nutzer wurden uns vom URZ Login-Informationen übertragen, mittels derer wir die Nutzer authentifizieren konnten. Da nun das vom URZ angebotene Shibboleth auch im CSN umgesetzt wird, entfallen diese Daten und somit auch die Tabellen **ypmap**, **ypmap_tmp** und **ypmap_todel**.

Für das Auffinden von inaktiven Nutzern war die Tabelle **inaktiventabelle** zum zwischenzeitlichen Speichern von eventuellen Nutzern vorhanden. Diese Funktionalität soll jedoch in ein Datenbank-externes Programm ausgelagert werden, womit die Tabelle obsolet wird.

Durch die Übernahme der Verwaltung und Pflege des WLANs auf dem Campusgelände durch das URZ, benötigt man keine Tabelle für die Unterhaltung desselbigen innerhalb des CSNs mehr. Damit entfällt **radacct** aus dem alten Schema.

Jegliche Spalten, die Bezug auf die Inventarisierung nehmen, werden nicht in das neue Modell übernommen, da wir in Zukunft auf das Inventarsystem unseres Steuerberaters aufbauen wollen. Dieses beinhaltet alle verbindlichen Informationen, die wir bereitstellen müssen. Dadurch entfallen die Tabellen **inventar** und **csn_rechner** gänzlich. Letztere, da sie lediglich die Tabelle **rechner** um inventar-bezogene Informationen ergänzte. Sämtliche Informationen zu CSN-eigenen Rechnern können somit in dieser gespeichert werden.

7.11. Logging

Aus Gründen der internen Nachvollziehbarkeit und der Nachweispflicht des CSN gegenüber dem URZ, ist es unverzichtbar, gewisse Datenbestände zu loggen. Das CSN hat einen Vereinbarung mit dem URZ, in dem sich das CSN verpflichtet, dass die IP-Adresse über einen gewissen Zeitraum einer Person zuordenbar ist.

Für jede Tabelle, die geloggt werden soll, wird eine Tabelle mit dem gleichen Namen und dem Postfix **Log** erstellt. Diese enthält die gleichen Spalten wie die eigentliche Tabelle und zusätzlich noch die Spalten *Action*, welche die getätigte Aktion in Form eines Aufzählungstyps beherbergt (INSERT, UPDATE, DELETE). Weiterhin werden noch die Spalten *ActionDate* vom Datumstyp, in der das Datum und die Uhrzeit der Änderung gespeichert wird, und *User*, in der der ausführende Datenbanknutzer hinterlegt wird, angelegt.

Das Logging wird durch alle nachfolgend genannten Tabellen zur Verfügung gestellt. In Klammern ist aufgeführt, welche Tabellen aus dem altem Schema dadurch abgedeckt werden.

- **UserLog** (csn_nutzer_aend_log und csn_nutzer_log)
- **UserTaskLog** (hat_aufgabe_log)
- **UserStatusLog** (csn_nutzer_hat_status_log)
- **TrafficExceptionLog** (traffic_ausnahmen_log)
- **FloorRepresentativeLog** (ev_betreut_log)
- **RequestValidationLog** (validation_log)
- **HostLog** (rechner_log und csn_rechner_log)
- **UserHostLog** (nutzerrechner_log und angeschlossen_an_log)
- **UserHostStatusLog** (nutzerrechner_hat_status_log)
- **BankAccountLog** (finanz_konto_log)

Für das Einfügen der Daten sind dann Trigger verantwortlich. Sie werden bei allen veränderlichen Aktionen (INSERT, UPDATE, DELETE) in den entsprechenden Tabellen ausgeführt. Dabei wird dann der neue Datensatz mit der Aktion, dem ausführenden Datenbanknutzer und der aktuellen Zeit in der Log-Tabelle gespeichert.

7.11.1. Vergleich zum alten Schema

In vorangegangener Auflistung ist bereits zu sehen, dass einige Log-Tabellen des alten Schemas in einer des Neuen zusammengefasst wurden.

Weiterhin sind die Log-Tabellen entfallen, bei denen die zugehörige Tabelle nicht im neuen Schema wiederzufinden ist. Hierzu zählen **aktuelle_information_log**,

ds_vorhersagen_log, **hv_betreut_log**, **konto_log**, **neuantrag_log**, **person_log**, **rechner-neuantrag_log** und **webupdate_log**.

8. Implementierung

In diesem Kapitel wird auf die konkrete Umsetzung der Datenbank eingegangen und die aufgetretenen Probleme mit ihrem Lösungsweg genauer erklärt.

8.1. Hardware

Im CSN wird seit einiger Zeit fast ausschließlich auf virtualisierte Server gesetzt. Dadurch soll eine strikte Trennung aller erforderlichen Dienste ermöglicht werden, sodass meist einer dieser Server nur einen einzigen Dienst beherbergt. Das ist auch der Fall für den Datenbank-Dienst. Der zugrunde liegende Virtualisierungs-Host besteht aus einem Supermicro H8DGU-F Mainboard, zwei Prozessoren vom Typ AMD Opteron 6134 (8 x 2,3 GHz), 16 x 4 GB DDR3-RAM und zwei 160 GB SAS-Festplatten als Host-Systemfestplatten. Die Festplatten der Gast-Systeme werden über das Netzwerk vom Storage-Server des CSNs bezogen.

Dem Gast-System wurden vorerst ein Prozessorkern mit 2,3 GHz, 2 GB Arbeitsspeicher und 20 GB Festplattenspeicher zugeteilt.

8.2. Eingesetzte Software

Das CSN setzt intern ausschließlich auf die Serverversion von Ubuntu. Hierbei haben sich die LTS-Versionen eingesetzte System durchgesetzt. Dabei handelt es sich um eine aller zwei Jahre erscheinende Ausgabe, die einen Unterstützungszeitraum von fünf Jahren hat. Somit kann immer auf ein sich als stabil erwiesenes System aufgebaut werden. Derzeit ist dies Ubuntu 12.04, welches auf dem neuen Datenbank-Server eingesetzt werden soll.

Als eingesetztes Datenbanksystem wird auf PostgreSQL gesetzt. Ein Grund für den Einsatz ist, dass PostreSQL sich innerhalb des CSNs bewährt hat und die im CSN eingesetzte Software gut mit diesem Datenbank-Management-System zusammenarbeitet. Weiterhin sind die Mitarbeiter mit dem PostgreSQL-System vertraut und benötigen keine weitere Einarbeitungszeit. Ebenso steht PostgreSQL kostenlos zur Verfügung und wird in Ubuntu mit Aktualisierungen versorgt. In Ubuntu 12.04 ist die Version 9.1 enthalten [Ubu].

8.3. Probleme während der praktischen Umsetzung

Nach der theoretischen Auseinandersetzung mit der Aufgabe folgte die prototypische Umsetzung. Diese verlief nicht immer komplikationsfrei und resultierte in neuen Erkenntnissen und teilweise auch der Umgestaltung der theoretischen Ergebnisse. In diesem Abschnitt sind die aufgetretenen Probleme dokumentiert.

8.3.1. Vererbung in PostgreSQL

In PostgreSQL ist es möglich von einer anderen Tabelle zu erben. Dabei werden alle Spalten dieser Tabelle übernommen. Hierbei muss an das CREATE TABLE-Statement der Befehl INHERIT und der Tabellename angehangen werden. [Pos12, Seite 66 ff. (Kapitel 5.8 Inheritance)]

```
CREATE TABLE "Host" {
    "HostID"    serial PRIMARY KEY,
    "Name"      varchar
};

CREATE TABLE "UserHost" {
    "UserName"  varchar
} INHERIT ("Host");
```

SQL-Anweisung 8.1: Vererbung

Danach besitzt die Tabelle **UserHost** die drei Spalten *HostID*, *Name* und *UserName*. Falls nun ein Eintrag in der Kindtabelle getätigt wird, erscheint dieser ebenso bei einer Abfrage der Eltern-Tabelle. Diese Verhaltensweise kann man mit dem Schlüsselwort ONLY nach FROM unterbinden. [Pos12, Seite 1425 ff.]

```
INSERT INTO "Host" ("Name") VALUES ("GeneralHost1");
INSERT INTO "UserHost" ("Name", "UserName") VALUES ("UserHost1", "User5")
   ;

SELECT "Name" FROM "Host";
Name
--------------
"GeneralHost1"
"UserHost1"
(2 rows)

SELECT "Name", "UserName" FROM "UserHost";
Name        | UserName
------------+----------
"UserHost1" | "User5"
(1 row)

SELECT "Name" FROM ONLY "Host";
Name
--------------
```

```
"GeneralHost1"
(1 row)
```

SQL-Anweisung 8.2: SELECT-Statements bei Vererbung

Genau diese Funktionalität ist für die Datenbank des CSNs von großem Vorteil. Betrachtet man die Allgemeinheit aller Rechner innerhalb des CSN, so hat jeder von diesen beispielsweise eine IP-, eine MAC-Adresse sowie einen Hostnamen. Nun können noch zwei speziellere Arten von Rechnern unterschieden werden. Zum einen die Nutzerrechner, die zusätzlich noch den Nutzer, dem der Rechner gehört, und die Dose, an dem der Rechner angeschlossen ist, speichert. Die zweite Art von Rechner sind die Stacks. Dies ist ein logischer Verbund von mehreren physikalischen Netzwerkgeräten und werden von CSN-Mitarbeitern administriert. Dafür werden in der Datenbank zusätzlich zu den allgemeinen Rechnereigenschaften noch Passwörter für den lesenden und schreibenden Zugriff sowie ein Subnetz, was dieser Stack verwaltet, gespeichert.

Der Vorteil dieser Herangehensweise ist es, dass man nun für die Generierung der DHCP- und DNS-Konfiguration lediglich die Elterntabelle abrufen muss, da dort alle benötigten Informationen vorhanden sind.

Probleme bei der Vererbung

In PostgreSQL ist die Vererbung nur teilweise funktionstüchtig umgesetzt. Dazu zählen unter anderen das Übernehmen der Spalten in die Kindtabelle und das Auflisten der Kindtabellen-Einträge in der Elterntabelle. [Pos12, Seite 66 ff. (Kapitel 5.8 Inheritance)]

Das Vererben von Constraints wird ebenso wie das Anlegen von Constraints über mehrere Tabellen nicht unterstützt. Dies ist jedoch essentiell, da ansonsten Fremdschlüssel-, Primärschlüssel- und Eindeutigkeits-Definitionen entfallen. Beispielsweise ist es in einer Kindtabelle dadurch möglich gleiche Werte in eine Spalte einzufügen, die in der

Elterntabelle als „UNIQUE“ gekennzeichnet war. Umgangen werden kann das, indem man diese Definitionen für die Kindtabelle erneut angibt, jedoch ist dies dann nur in Bezug auf diese Tabelle. Das bedeutet, dass in Kind- und Elterntabelle in der gleichen Spalte ein gleicher Wert auftaucht, obwohl diese Spalte als „UNIQUE“ gekennzeichnet ist. [Pos12, Seite 68 ff. (Abschnitt 5.8.1 Caveats)]

Aus diesen Gründen wurde trotz den Vorteilen auf die native Vererbung innerhalb der Datenbank verzichtet und eine pragmatische Umsetzung gewählt. [Kow12, Seite 65 ff.] Dabei gibt es die Elterntabelle in genau der gleichen Art und Weise. Sie beinhaltet alle gemeinsamen Daten z.B. aller Rechner. Die „Kindtabellen“ enthalten nun jedoch nicht die Daten der „Elterntabelle“, sondern lediglich einen Verweis auf einen Eintrag in selbiger. Somit sind die gemeinsamen Daten in einer Tabelle zu finden. Falls man nun für einen Eintrag in einer „Kindtabelle“ auch die Daten der Elterntabelle benötigt, so kann man dies durch ein simples JOIN-Statement bewerkstelligen.

Das Beispiel von oben ist nachfolgend einmal mit der für die soeben beschriebene Herangehensweise aufgezeigt.

```
CREATE TABLE "Host" {
    "HostID"    serial PRIMARY KEY,
    "Name"      varchar
};

CREATE TABLE "UserHost" {
    "HostID"    integer REFERENCES "Host" ("HostID"),
    "UserName"  varchar
};

INSERT INTO "Host" ("Name") VALUES ("GeneralHost1");
INSERT INTO "Host" ("Name") VALUES ("UserHost1");
INSERT INTO "UserHost" ("HostID", "UserName") VALUES ((SELECT "HostID"
    FROM "Host" WHERE "Name" = 'UserHost1'), "User5");

SELECT "Name" FROM "Host";
Name
--------------
"GeneralHost1"
"UserHost1"
(2 rows)
```

```
SELECT "Name", "UserName" FROM "UserHost" JOIN "Host" USING ("HostID");
Name        | UserName
------------+----------
"UserHost1" | "User5"
(1 row)

SELECT "Name" FROM "Host" WHERE "HostID" NOT IN (SELECT "HostID" FROM
    "UserHost");
Name
--------------
"GeneralHost1"
(1 row)
```

SQL-Anweisung 8.3: Adaptierte Vererbung

Wie man sieht, ist ein zusätzliches INSERT-Statement für den Nutzerrechner vonnöten. Ebenso beinhaltet die vollständige Abfrage eines Nutzerrechners das erwähnte JOIN-Statement. Die Abfrage der Rechner, die nicht in der Kindtabelle enthalten sind, ist ebenso komplexer geworden.

Im Gegensatz zur Erhöhung der Komplexität der Abfragen steht die Sicherstellung der Eindeutigkeit gewisser Spalten (UNIQUE-Constraint), was eine zwingende Anforderung an diese Tabellen war.

Diese Lösung fand in den Tabellen **Host** und **Accounting** als „Elterntabellen“ und **Stack**, **UserHost** und **HeadingAccounting** als „Kindtabellen“ Anwendung.

8.4. Migrationsstrategie

Neben der Erstellung eines an die Bedürfnisse des CSN angepassten Datenbank-Schema ist natürlich die Migration der bestehenden Daten ein wichtiger Punkt. Hierzu werden aus der bestehenden Datenbank die Daten abgerufen, die für das neue Schema relevant sind.

Für jede Tabelle des neuen Schemata, soll ein INSERT-Statement erstellt werden, was alle benötigten Daten aus der alten Datenbank enthält. Dabei ist die Vorgehensweise bei allen Tabellen nahezu identisch.

8.4.1. Vorgehensweise

Zuerst muss eine SELECT-Anweisung erstellt werden, die sämtliche für die neue Tabelle benötigten Daten enthält. Diese wird ausgeführt und die erhaltenen Daten in eine INSERT-Anweisung für die neue Datenbank gespeichert. Dazwischen finden noch kleinere Anpassungen, wie das Umwandeln zwischen verschiedenen Kodierungen, statt, da die neue Datenbank auf „UTF–8“ setzt und die bestehende Datenbank ihre Daten in „ISO–8859–1“ speicherte.

Folgende Anweisung stellt die Ausgangsabfrage an die bisherige Datenbank dar.

```
SELECT strasse, hausnr, hkz, objekt_nr FROM haus ORDER BY haus_id;
        strasse         | hausnr | hkz   | objekt_nr
------------------------+--------+-------+-----------
 Thueringer Weg         | 3      | Tw3   |      8235
 Reichenhainer Strasse  | 35     | Rh35  |      8110
 Reichenhainer Strasse  | 37     | Rh37  |      8120
 Reichenhainer Strasse  | 51     | Rh51  |      8190
 Vettersstrasse         | 52     | Ve52  |      8130
 Vettersstrasse         | 54     | Ve54  |      8140
 Vettersstrasse         | 64     | Ve64  |      8151
 Vettersstrasse         | 66     | Ve66  |      8152
 Vettersstrasse         | 70     | Ve70  |      8170
 Vettersstrasse         | 72     | Ve72  |      8180
(10 rows)
```

SQL-Anweisung 8.4: Datenexport aus bestehender Datenbank

Daraus wird durch das Skript nachfolgende Anweisung für die neue Datenbank.

```
INSERT INTO "House" ("StreetName", "HouseNumber", "Abbreviation",
    "ObjectNumber") VALUES
('Thüringer Weg', '3', 'Tw3', 8235),
('Reichenhainer Straße', '35', 'Rh35', 8110),
```

```
('Reichenhainer Straße', '37', 'Rh37', 8120),
('Reichenhainer Straße', '51', 'Rh51', 8190),
('Vettersstraße', '52', 'Ve52', 8130),
('Vettersstraße', '54', 'Ve54', 8140),
('Vettersstraße', '64', 'Ve64', 8151),
('Vettersstraße', '66', 'Ve66', 8152),
('Vettersstraße', '70', 'Ve70', 8170),
('Vettersstraße', '72', 'Ve72', 8180);
```

SQL-Anweisung 8.5: Generiertes INSERT-Statement für neue Datenbank

Wie man sieht gab es noch kleinere Anpassungen der Schreibweisen, welche im Skript erledigt wurden.

8.4.2. Gruppierung der Tabellen

Die Gruppierung der Tabellen in logisch zusammengehörige Mengen war bei der Erstellung der Migrationsanweisungen ein grober Anhaltspunkt. Für jede dieser Gruppierungen wurde ein Skript erstellt, welches sich mit der bestehenden Datenbank verbindet, die Daten ausliest und fertige SQL-Statements zurückgibt.

8.4.3. Dynamisches Auslesen von IDs

Da die Primärschlüssel größtenteils automatisch und fortlaufend generiert werden, können diese nicht direkt übernommen werden. Deshalb muss auf andere eindeutige Eigenschaften zurückgegriffen werden, um den Primärschlüssel des neu erstellten Eintrages zu erhalten.

Hier sieht man die Ausgangs-Anweisung für die bestehende Datenbank.

```
SELECT hkz, etage, zinr FROM standort JOIN haus USING (haus_id) ORDER BY
    hkz, etage, zinr;
```

```
 hkz  | etage | zinr
------+-------+------
 Rh35 |     0 |   11
 Rh35 |     0 |   12
 Rh35 |     0 |   13
 ...
 (2370 rows)
```

SQL-Anweisung 8.6: Eindeutige Eigenschaften als Hilfsmittel für ID-Ermittlung

Daraus resultiert folgende Anweisung für die neue Datenbank.

```
INSERT INTO "Room" ("FloorID", "RoomNumber") VALUES
((SELECT "FloorID" FROM "Floor" WHERE "FloorNumber" = 0 AND
    "HouseID" = 'Rh35'), 11),
((SELECT "FloorID" FROM "Floor" WHERE "FloorNumber" = 0 AND
    "HouseID" = 'Rh35'), 12),
((SELECT "FloorID" FROM "Floor" WHERE "FloorNumber" = 0 AND
    "HouseID" = 'Rh35'), 13),
...;
```

SQL-Anweisung 8.7: INSERT-Statement mit Abfrage für ID-Ermittlung

In diesem Beispiel sieht man, dass auf den eindeutigen Wert der Abkürzung eines Wohnheimes in Zusammenhang mit der Etagennummer gesetzt und damit der zugehörige Primärschlüssel ermittelt wurde.

8.5. Migrationsdurchführung

Für die Migration sämtlicher Daten ist es wichtig, dass während des Exports des aktuellen Datenbestandes keinerlei Änderungen an selbigen durchgeführt werden. Aus diesem Grund werden Testläufe durchgeführt, um den Wartungszeitraum abschätzen zu können.

Nachfolgend sind die Zeiten und Datenmengen aufgeführt, die während des Testens der Migration gemessen wurden. Für das Messen der benötigten Zeiten wurde auf

den Linux-Befehl *time* gesetzt, welcher die Ausführungszeit eines Aufrufes anzeigt. Die Dateigrößenangaben basieren, wie die Abkürzung zeigt, auf den Binärpräfixen. Somit sind 1024 Byte mit 1 KiB übereinstimmend.

Gruppierungs-Name	Exportzeit in Sekunden	Datei-Größe	Importzeit in Sekunden	Anzahl an Datensätzen
Finance	0,267	221,4 KiB	0,64	999
Host	0,234	723,9 KiB	2,67	9.883
Infrastructure	0,242	846,4 KiB	12,77	7.106
Location	0,134	200,2 KiB	1,13	2.461
RequestValidation	0,151	1,5 KiB	0,12	6
StuWe	8,965	26,6 MiB	515,44	137.090
Traffic	476,931	1,2 GiB	25487,21	6.723.836
UserHost	0,331	3,9 MiB	18,67	10.034
User	0,777	1,4 MiB	114,63	28.938
VLAN	0,103	4,0 KiB	0,07	41
Datenbank-Schema	—	12,3 KiB	1,87	—

Tabelle 8.1.: Migrations-Daten (Exportdatum: 16. Oktober 2012 15:24 Uhr)

Aus der Tabelle 8.1 Migrations-Daten ist eine Exportdauer von etwas über 8 Minuten ersichtlich. Für den Import werden über 7 Stunden benötigt. Bei die derzeitigen Nutzung als Testdatenbank für die neue CSN-Software ist dieses Vorgehen vollkommen ausreichend. Falls man bei der tatsächlichen Umstellung des Datenbanksystems jedoch nicht einen derart großen Wartungszeitraum veranschlagen möchte, kann man versuchen parallel ausführbare INSERT-Statements nebenläufig zu starten. Beispielsweise sind sämtliche Einträge des extrem zeitintensiven Imports der Kategorie „Traffic“ voneinander unabhängig und somit für eine parallele Ausführung geeignet.

9. Evaluation

In diesem Kapitel wird auf das praktische Resultat der Arbeit eingegangen und in wie weit die Anforderungen erfüllt wurden.

01 - Abbildung sämtlicher benötigter Daten

Die erste Anforderung enthielt eine Liste an Daten, die definitiv vom neuen Schema abbildbar sein müssen. Diese Anforderung wurde vollständig erfüllt, da alle genannten Datenstrukturen mit dem erstellten Schema korrekt abbildbar sind.

02 - Einfügen hilfreicher Daten

Mit dieser Anforderung sollten Schwachstellen des alten Systems umgangen werden. Hierzu wurden mehrere Strukturen eingefügt, die so nicht im alten Schema vorhanden waren.

Die Tabelle **HostNameBlacklist**, die für das Sperren gewisser Rechnernamen zuständig ist, zählt ebenso zu diesen Verbesserungen, wie die Spalten *Historic* in den Tabellen der Räume, Etagen und Häuser.

03 - Betrachtung zukünftiger Probleme

Unter diesem Punkt fallen zum Beispiel die Möglichkeit, dass mit dem entworfenen Schema beispielsweise die IPv6-Adressvergabe parallel zur IPv4-Adressvergabe möglich ist. Dies wurde durch die Tabelle **IPMapping** gelöst. Daneben ist noch zu nennen, dass der Einzug eines CSN-Nutzers in mehrere Zimmer nun abbildbar ist.

Seitens des CSNs gab es keine weiteren Vorschläge für zukunftsorientierte Datenstrukturen.

04 - Benennungskonventionen schaffen und beachten

Im Kapitel Benennungskonventionen (Abschnitt 6.1) wurden diese aufgeführt. Sämtliche Tabellen, Spalten und Typenbezeichnungen im entworfenen Schema richten sich danach, womit der Anforderung gerecht wird.

05 - Mehrfache Datenspeicherung vermeiden

Dieser Anspruch wurde so gut wie möglich und praktikabel umgesetzt. Im entwickelten Schema ist eine Verknüpfung redundant. Die Tabelle **UserHost** benötigt eine Verknüpfung zum Benutzer, welcher der Eigentümer des Rechners ist und somit für den entstandenen Traffic und eventuelle Verstöße zur Rechenschaft gezogen werden kann. Die Auflösung dieser Beziehung ist möglich, indem von der Nutzerrechner-Tabelle aus über die Tabellen **Socket**, **Room** und **UserLivesInRoom** auf den Nutzer geschlossen werden kann. Hierbei wurde sich jedoch bewusst dagegen entschieden, damit auch Rechner auf einen Nutzer angemeldet werden können, die nicht im Zimmer des Nutzers stehen.

Dies ist der Fall, wenn beispielsweise Studentenclubs Rechner im Clubraum betreiben wollen.

06 - Sämtliche Beschränkungen im Schema aufführen

Sämtliche Primär- und Fremdschlüssel werden im Schema aufgeführt. Weiterhin wurde auch - wo möglich und sinnvoll - von Eindeutigkeitsmarkierungen (UNIQUE-Constraint) und Markierungen für als zwingend erforderliche Werte (NOT NULL-Constraint) Gebrauch gemacht.

07 - Keine überflüssigen Datenstrukturen schaffen

Nach der Präsentation des Schemas beim CSN wurde in Rücksprache mit diesem das Schemamodell als zweckmäßig und geeignet befunden. Es wurden keine fehlenden Strukturen festgestellt.

08 - Keine Datenbank-Prozeduren einführen

Während des Entwurf des Schemas wurde in Hinblick auf diese Anforderung gänzlich auf das Entwickeln von Datenbank-Prozeduren verzichtet.

Zusammenfassung

Alle Anforderungen wurden mit Ausnahme von Nummer 5 erfüllt. Bei dieser Ausnahme ist jedoch zu beachten, dass diese nur an einer einzigen Stelle im Schema aufgrund einer Verbesserung der Abbildbarkeit von Sachverhalten nicht erfüllt wurde.

Im Ganzen ist also zu sagen, dass die Anforderungen sehr gut umgesetzt sind.

	Anforderung	**Anmerkung**
✓	01 - Abbildung sämtlicher benötigter Daten	
✓	02 - Einfügen hilfreicher Daten	Aktuelle Schwachstellen vollständig abgebildet
✓	03 - Betrachtung zukünftiger Probleme	Keine weiteren Vorschläge vorhanden
✓	04 - Benennungskonventionen schaffen und beachten	
	05 - Mehrfache Datenspeicherung vermeiden	Praktisch sinnvolle Umsetzung
✓	06 - Sämtliche Beschränkungen im Schema aufführen	
✓	07 - Keine überflüssigen Datenstrukturen schaffen	
✓	08 - Keine Datenbank-Prozeduren einführen	

Tabelle 9.1.: Anforderungserfüllung

9.1. Verbesserungsansätze

Anschließend an die Bewertung der Umsetzung sollen in diesem Abschnitt noch auf Verbesserungsvorschläge eingegangen werden.

Vererbung

Im Abschnitt Vererbung in PostgreSQL (Unterabschnitt 8.3.1) wurde die unzureichende Umsetzung von Vererbung in PostgreSQL aufgezeigt. Hier kann man den weiteren Entwicklungsverlauf in PostgreSQL betrachten und eventuell von der hier erklärten Umsetzung auf native Vererbung in PostgreSQL umsteigen.

Tabellenübergreifende Constraints

Für gewisse Datenstrukturen ist es teilweise nötig eine Beschränkung (z.B. UNIQUE-Constraint) über gleiche Spalten unterschiedlicher Tabellen anzulegen. Jedoch wird das von der aktuellsten PostgreSQL-Version ebenso noch nicht unterstützt.

Zusätzliche Constraints

Für die noch bessere und konsistente Datenspeicherung, kann eventuell die Reglementierung weiterer Spalten in Betracht gezogen werden.

10. Zusammenfassung

In dieser Arbeit galt es die Aufgabe zu bearbeiten, ein komplett neues Schema für das CSN zu entwerfen. Dabei sollte sich grundlegend am alten Schema orientiert werden. Hierzu war es notwendig dieses vorerst zu verstehen, da keinerlei Dokumentation während der Entstehungsphase angelegt wurde. Einige Jahre später wurde versucht dies nachzuholen, was nur teilweise glückte. Seitdem versuchten sich bereits einige Freiwillige mit den Eigenarten des bisherigen Schemas zu arrangieren und es zu verstehen. Mit dieser Arbeit wurde all dies ausgearbeitet und das bestehende System evaluiert. Anschließend sind die Erfahrungen und Verbesserungsvorschläge in ein komplett neues Schema eingeflossen, welches das alte vollständig ersetzt.

Zu der Neugestaltung des kompletten Datenbankschemas gehört ebenso die Migration sämtlicher bestehender Datensätze. Exemplarisch wurde die Machbarkeit anhand einer Migrationsstrategie unter Beweis gestellt. Damit ist aufgezeigt, dass das entworfene Modell den Anforderungen gerecht wird und alle benötigen Daten in ihrer Struktur abbilden kann.

Daran anschließend wurde untersucht, ob die zu Beginn aufgestellten Anforderungen erfüllt sind. Schließlich wurden Verbesserungsvorschläge für die Zukunft genannt.

Mit dieser Arbeit wird dem CSN ein vollständiges Schema samt Migrationsstrategie übergeben, auf dem aufbauend die weitere Neugestaltung der Software-technischen

Grundlage fortgeführt werden kann. Damit kann das CSN-Team seinem Anspruch gerecht werden, eines der modernsten und fortschrittlichsten Studentennetze Deutschlands zu sein, welches ausschließlich auf das ehrenamtliche Engagement seiner Mitarbeitern setzt.

Literaturverzeichnis

[BS09] Berner, Elisabeth ; Siehr, Karl H.: *Sprachwandel und Entwicklungstendenzen als Themen im Deutschunterricht: fachliche Grundlagen - Unterrichtsanregungen - Unterrichtsmaterialien*. Universitätsverlag Potsdam, 2009. – ISBN 3869560037

[CT] CSN-Team: *CSN SVN-Repository*. `svn://csn-server.csn.tu-chemnitz.de/branches/rel-3-0-branch`, Abruf: 13.09.2012

[DbS] *DbSchema*. `http://www.dbschema.com/`, Abruf: 01.11.2012

[Elm02] Elmasri, Shamkant B. Ramez; Navathe N. Ramez; Navathe: *Grundlagen von Datenbanksystemen*. Pearson Studium, 2002. – ISBN 3827370213

[Kow12] Kowarschick, Wolfgang: *Vorlesungsskript Multimedia-Datenbanksysteme*. `http://mmdb.hs-augsburg.de/medium/text/lehre/2012sose/mmdb/MMDB.pdf`. Version: 2012, Abruf: 29.09.2012

[Lil] Lil, Ingo van: *DbAllgemeines < Csnpublic < TWiki*. `https://www.csn.tu-chemnitz.de/twiki/bin/view/Csnpublic/DbAllgemeines`, Abruf: 26.08.2012

[pas] *passwd(5): password file - Linux man page*. `http://linux.die.net/man/5/passwd`, Abruf: 12.08.2012

[Pos12] PostgreSQL Global Development Group: *PostgreSQL 9.1.6 Documentation*. 2012 `http://www.postgresql.org/files/documentation/pdf/9.1/postgresql-9.1-A4.pdf`

[Tü06] Türker, Gunter Can; S. Can; Saake: *Objektrelationale Datenbanken - Ein Lehrbuch*. dpunkt.verlag, 2006. – ISBN 3898641902

[Ubu] *Ubuntu – Informationen über Paket postgresql in precise*. `http://packages.ubuntu.com/precise/postgresql`, Abruf: 28.10.2012

Anhang A.

Tabellenübersicht

Tabellenname	Anzahl an Einträgen	Anzahl an Spalten
aktuelle_information	224	9
angeschlossen_an	2.228	4
ansprechpartner	0	4
aufgabengebiet	17	3
bezieht_liste	27	3
cnames	9	5
csn_nutzer	3.366	9
csn_nutzer_aend	3.354	13
csn_nutzer_hat_status	22.227	7
csn_nutzerstatus	14	4
csn_rechner	1.112	15
dnsupdate	4.406	5
dose	3.331	3
ds_allgemein_v2	15	2
ds_ausnahmen_v2	7	3
ds_gruppen_v2	60	3

ds_nutzergruppen_v2	3.366	5
ds_statistik_v2	26.261	7
ds_vorhersagen	273	5
ds_vorhersagen_current	1.109	5
ds_vorhersagen_tmp	255	6
ev_bereich	78	5
ev_betreut	74	5
finanz_abschluesse	136	6
finanz_buchungen	5.535	12
finanz_jahr	9	4
finanz_konto	4	12
finanz_plan	9	4
finanz_plan_eintrag	143	6
finanz_plan_eintrag_gehoert_zu	207	3
finanz_titel	12	3
finanz_titelbuchungen	5.515	14
finanz_titeluebertraege	0	8
finanz_transfers	101	3
finanz_uebertragsbuchungen	0	3
finanz_untertitel	32	3
firewall_anschluesse_last	2.216	4
geraete	29	18
geraeteart	26	6
geraetestatus	6	5
hat_aufgabe	121	5
haus	10	6
hersteller	9	4

hub	54	21
hub_config_last	2.228	11
hv_bereich	9	3
hv_betreut	3	5
inaktiventabelle	0	5
inventar	1.121	6
kabeltyp	4	2
kategorie	4	3
konstanten	35	5
kontakt	8	10
konto	4	9
mailingliste	7	3
mailqueue	0	8
netzklasse	3	8
neuantrag	8.869	10
nutzeraufnahmeantrag	8.869	8
nutzerrechner	4.474	8
nutzerrechner_aend	4.400	10
nutzerrechner_hat_status	5.539	7
nutzersperre	564	4
person	3.366	7
port	3.363	4
radacct	0	24
rechner	5.588	7
rechnerneuantrag	10.103	8
rechnerstatus	4	4
rechnung	0	3

standort	2.237	8
stuwe_liste	135.338	5
stuwe_liste_bearbeiter	69	3
subnetz	21	6
subnetz_vlan_pro_haus	30	4
team_telefon	29	3
telefon_typ	5	2
traffic	0	12
traffic_ausnahmen	80	10
traffic_current	37.372	8
traffic_log	8.864.370	8
traffic_overlimit	3.336	6
traffic_rechner_statistik	6.691	7
trafficart	2	5
trafficstatistik	8.815	5
unit	86	8
validierungs_log	9.985	7
verfuegbare_ips	4.497	1
webupdate	2.246	5
ypmap	17.173	7
ypmap_tmp	11.119	7
ypmap_todel	1	7

Tabelle A.1.: betrachtete Tabellen aus dem alten Datenbank-Schema (Stand: 21. August 2012)

Anhang B.

Schema-Darstellungen

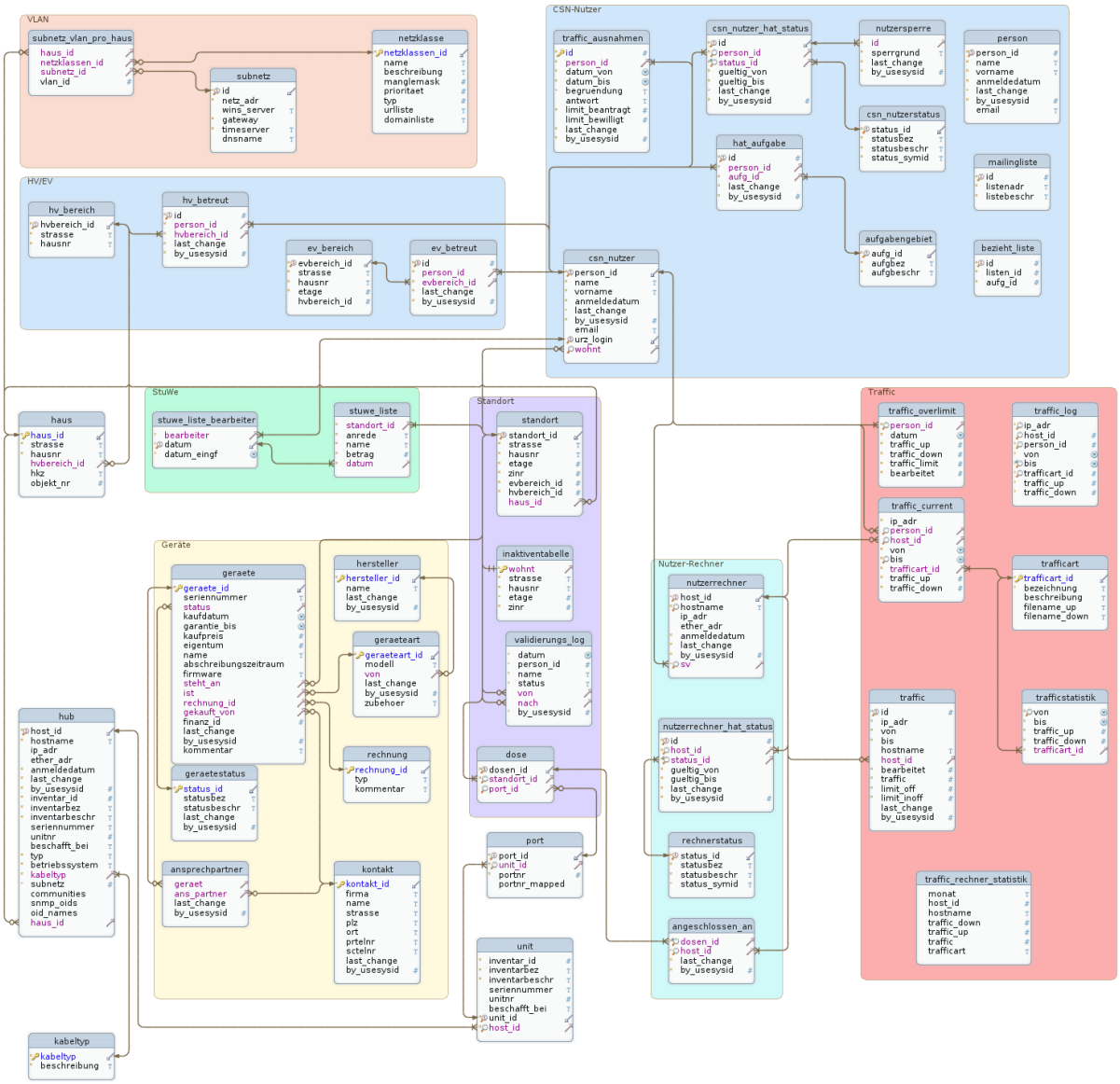

Abbildung B.1.: Bisheriges Datenbank-Schema (Teil 1)

Abbildung B.2.: Bisheriges Datenbank-Schema (Teil 2)

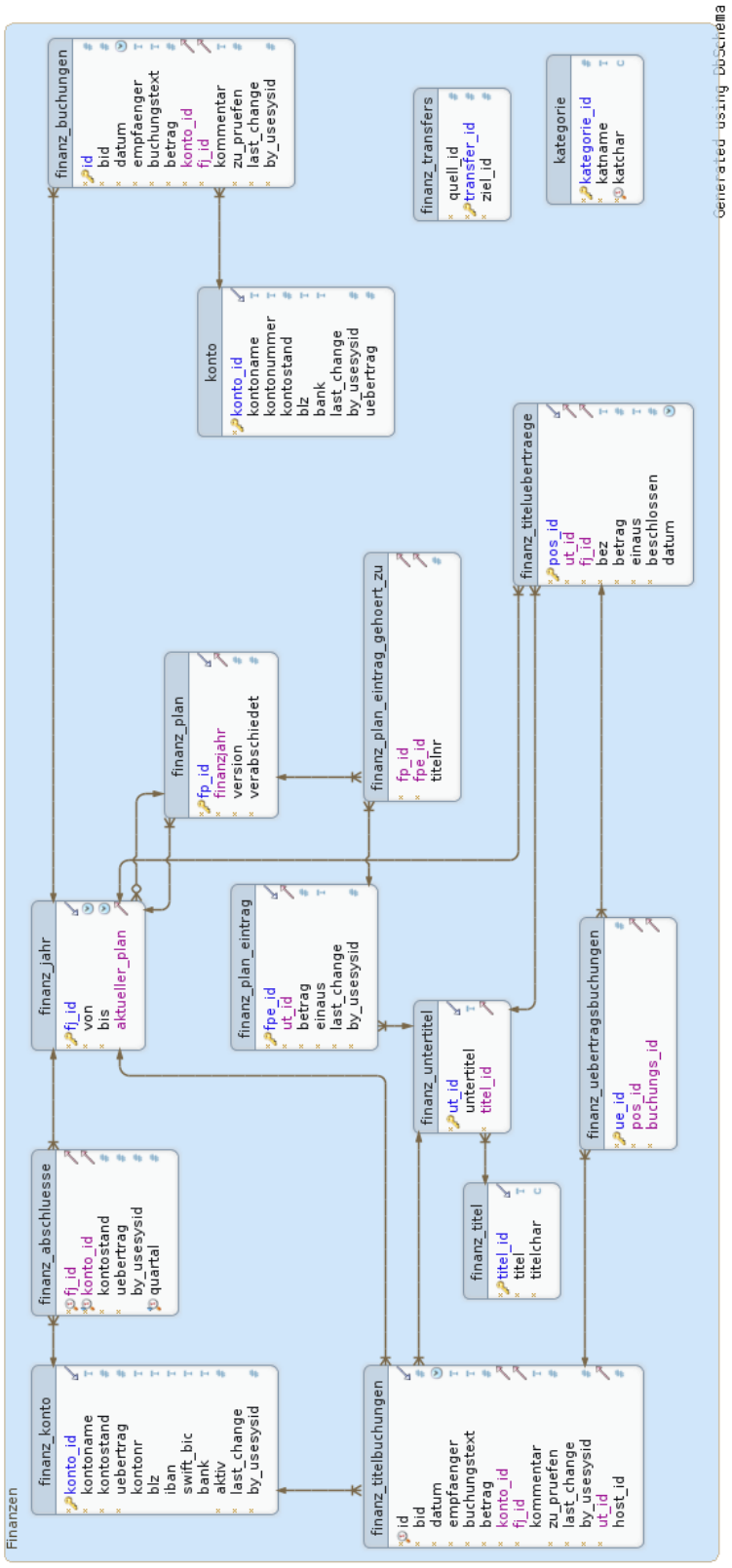

Abbildung B.3.: Bisheriges Datenbank-Schema (Teil 3) - Finanzen

Abbildung B.4.: Bisheriges Datenbank-Schema (Teil 4) - Logs

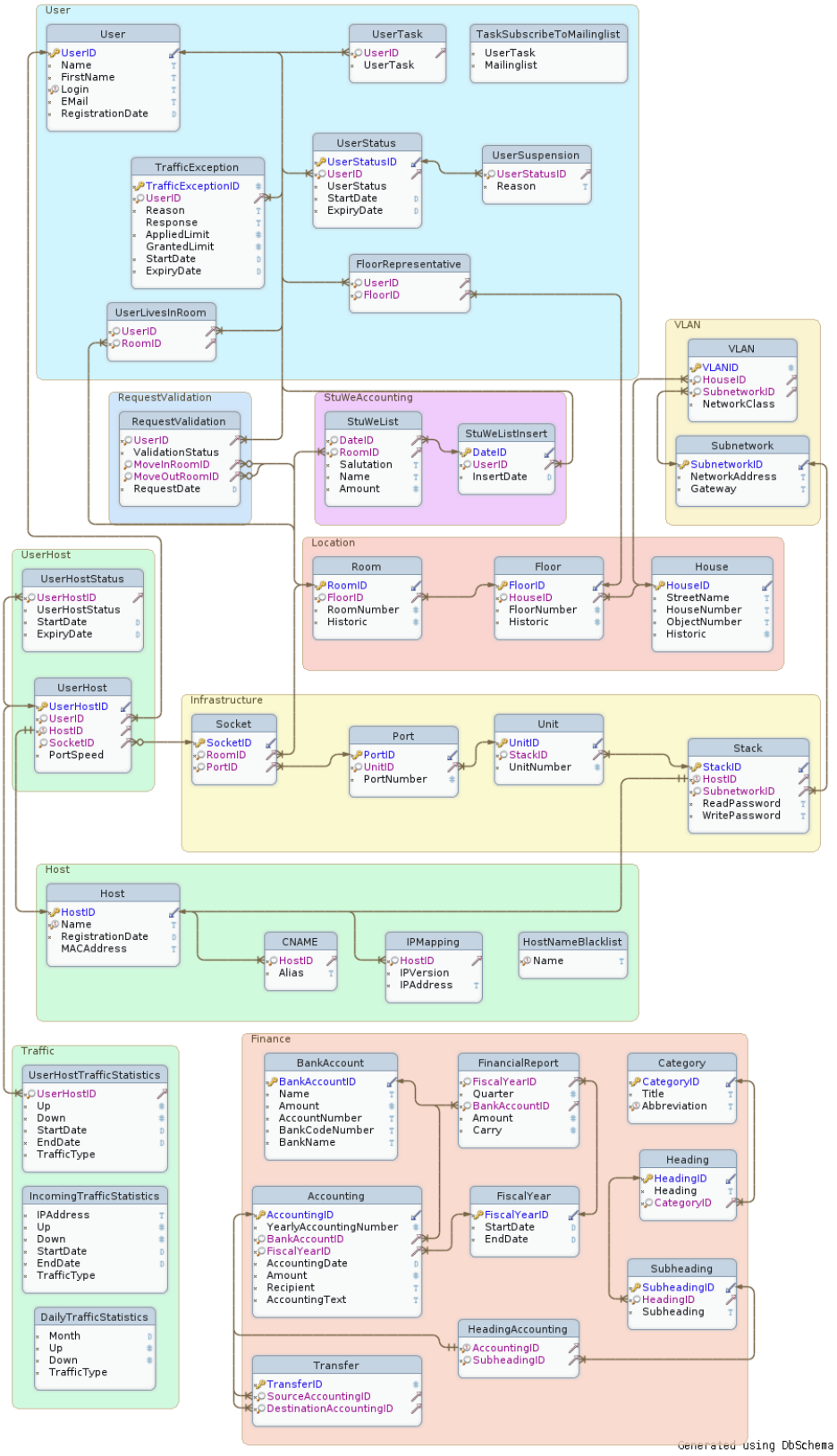

Abbildung B.5.: Entworfenes Datenbank Schema