WORKSHOP

zum

OBJEKT RELATIONALES DATENBANK

Autoren: Rituraj Singh | AmirHossein Roshanzadeh | Saeide Dana

Gruppe 1

Prof. Dr. Kay-Michael Otto

Datenbanken und Informationssysteme (INFM1200)
Sommersemester 2021

1 INHALTSVERZEICHNIS

in der Dokumentation wird die folgende Terminologie verwendet, die in der folgenden Tabelle beschrieben ist

VERWENDETE TERMINOLOGIEN

Abkürzung	Englische Bedeutung	Deutsche Bedeutung
RDBMS	Relational Database Management System	Relationales Datenbank Management System
ORDBMS	Object Relational Database Management System	Objekt Relationales Datenbankverwaltungssystem
ADT	User Defined Abstract Data Type	Benutzerdefinierte abstrakte Datentypen
UDR	User Defined Routines	Benutzerdefinierte Routinen

§ 2 LERNZIELE DER DOKUMENTATION

Dieses Dokument wird versuchen, alle wichtigen Details für objekt relationale Datenbankmanagementsysteme zu erklären, die auch für einen Einsteiger leicht verständlich sind. Das Dokument verdeutlicht die Unterschiede zwischen rein relationalen und objektrelationalen Datenbankmanagementsystemen und wie man letztere in der Praxis einsetzen kann. Um die Dinge richtig zu verstehen, muss man sich die Hände schmutzig machen, indem man tatsächlich einen Code implementiert. Wir haben einige Aufgaben mithilfe verschiedener Tools erstellt, die dem besseren Verständnis des Konzepts der objektrelationalen Datenbank im besseren Sinne helfen können. Letztendlich haben wird versucht, einige Fragen angemessen zu beantworten die beim Lernen über objektrelationale Datenbank Management systeme auftreten können.Für unseren Workshop wird **Postgresql** als objektrelationale Datenbank ausgewählt, da es als "**The World's Most Advanced Open Source object Relational Database**" gilt.

Ablauf des praktischen Teils des Workshops

Unser Workshop ist grundsätzlich in theoretische und praktische Teile unterteilt. Was den praktischen Teil betrifft, so besteht er ebenfalls aus zwei Teilen ~

- **Implementierung mit Postgres**
  Um die objektrelationalen Erweiterungen von SQL besser zu erklären, haben wir online-Tools wie **SQL Fiddle** verwendet,damit die Kursteilnehmer dafür keine zusätzliche Software installieren müssen. Die zu erledigenden Aufgaben werden als **.sql-Dateien** erstellt und während des Workshops zur Verfügung gestellt und werden auch hier erwähnt.

- **Implementierung mit Java**
  Um die Umsetzung mit Java zu zeigen, haben wir wieder kleine Aufgaben mit **IntelliJ IDE** und **EDU Tool** Plugin erstellt. Wir empfehlen für diesen Workshop eine **vollständige Offline-Vorbereitung** Ihres Systems wie in Installationsanweisungen beschrieben.

Zusammenfassung der im Workshop verwendeten Tools

Implementierung mit Postgres	Implementierung mit Java
SQL Fiddle	**IntelliJ IDE**
--------	**EDU Tool Plugin**
-------	**Code Together/Code with me**

§ 3 THEORETISCHER ÜBERBLICK

Dieser Abschnitt wird die theoretischen Aspekten einer objektrelationalen Datenbank klären und was sie von rein relationaler Datenbank unterscheidet.

3.1 Übersicht über ORDBMS und RDBMS

Quellenangaben im Literaturverzeichnis - 1 | 2 | 3

RDBMS ~ RDBMS ist ein Datenbank verwaltungssystem, das auf einem relationalen Datenmodell basiert. Dieses Modell verwendet mathematische Beziehungen und seine Komponenten sind Tabellen und Werte. Beispielsweise enthält eine Beziehung mit der Bezeichnung **User** mit Datensätzen, die Informationen zu jedem **User** enthalten. klassifizieren wir diese Art von Daten als "**einfach**" , weil diese Datensätzen Standarddatentypen (wie varchar,float, int) verwenden. Bekannte RDBMS sind **MS SQL Server**, **MySQL**, **SQLite und MariaDB.**

ORDBMS ~ Das objektrelationale Datenbankverwaltungssystem ist ein Datenbank verwaltungssystem, das auf dem relationalen Modell und dem objektorientierten Datenbank Modell basiert. Da ORDBMS eine Kombination zwischen relationalen und dem objektorientierten Modell ist. Es versucht, die Lücke zwischen diesen modellen zu schließen, damit in Programmiersprachen wie Java, C ++, Visual Basic .NET oder C# verwendet werden. Ein objektorientiertes Datenbankmodell enthält eine Klasse, z. B. **User**, **mit einem Objekt**, das Informationen zu jedem **User** enthält.(**in der folgenden Abbildung beschrieben**)

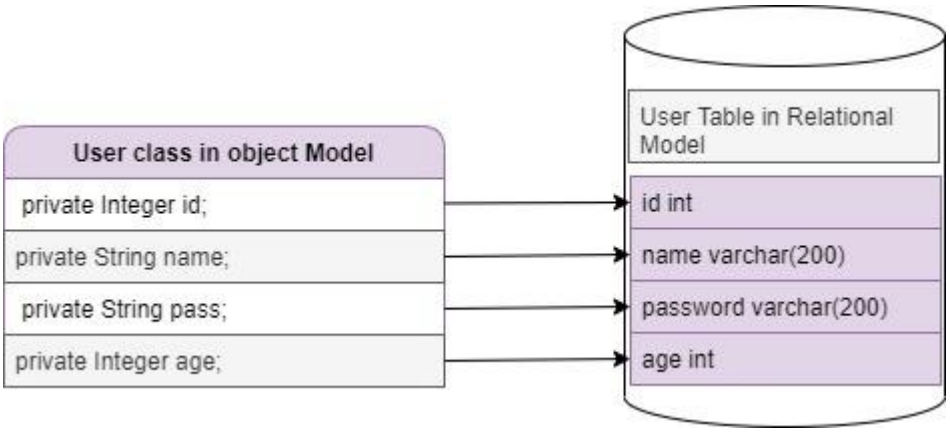

Abbildung 1 ~ Wie komplexe Objekte in relationalen Tabellen gespeichert werden.

Mit anderen Worten, können mit einem objektrelationale Datenbank zusätzlich zu den relationalen Daten weitere komplexe oder multimediale Datentypen (wie Bilder, Videos usw.) hinzufügen. **Oracle Database**, **PostgreSQL und Microsoft SQL Server** sind einige Beispiele für ORDBMS

3.2 Erweiterungen des objekt relationalen Modells

Quellenangaben im Literaturverzeichnis - 4

Das ORDBMS erweitert die Funktionen des relationalen Modells, damit Objekte in den Spalten einer relationalen Datenbank gespeichert werden können. Ein ORDBMS wird manchmal als hybrides DBMS bezeichnet. Der hybride DBMS-Ansatz wurde 1990 von Michael Stonebraker vorgeschlagen , um komplexere Entitäten und Regeln zu speichern.
Das objektrelationale Datenmodell ist eine Erweiterung des relationalen Modells mit folgenden Funktionen:

- Ein Feld kann ein Objekt mit Attributen und Operationen enthalten.
- Komplexe Objekte(oder Benutzerdefinierte Datentypen) können in relationalen Tabellen gespeichert werden

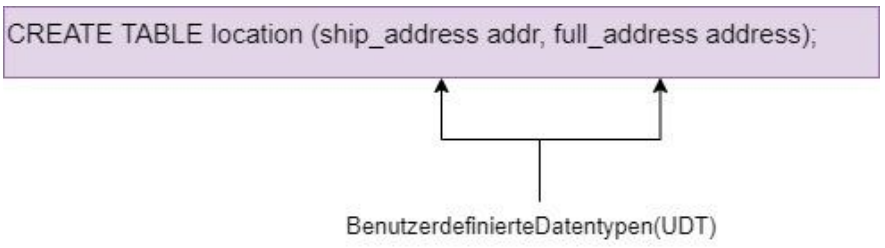

Abbildung 2 ~ **addr** und **adresse** sind Komplexe Objekte oder Benutzerdefinierte Datentypen. Dies wird in der Datei **UDT.sql** erklärt.

Abgesehen von benutzerdefinierte Typen. Eine objektrelationale Datenbank ermöglicht gegenüber einer rein relationalen Datenbank einige andere Erweiterungen, z.B. Nicht-atomare Attribute , Geschachtelte Relationen , Vererbung und Benutzerdefinierte Routinen, die anhand von Beispielen im Implementierungs Teil unseres Workshops geklärt wird.

3.3 Funktionen von ORDBMS

Quellenangaben im Literaturverzeichnis - 5

ORDBMS verfügen über vier Funktionen, die sie von RDBMS unterscheiden ~
• **Benutzerdefinierte abstrakte Datentypen (ADTs)** ~ ADTs ermöglichen neue Datentypen mit Strukturen, die für bestimmte zu definierende Anwendungen geeignet sind.
• **Benutzerdefinierte Routinen (UDRs)** ~ UDRs bieten die Möglichkeit, benutzerdefinierte Serverfunktionen zu schreiben.
• **Smart BLOBs** ~ Hierbei handelt es sich um festplattenbasierte Objekte mit der Funktionalität von Dateien mit wahlfreiem Zugriff.
• **Erweiterbare und flexible Indizierung** ~ Die R-Baum- oder GiST-Indizierung (Generalized Search Tree) für mehrdimensionale Daten ermöglicht die schnelle Suche nach bestimmten ADTs in einer Tabelle.

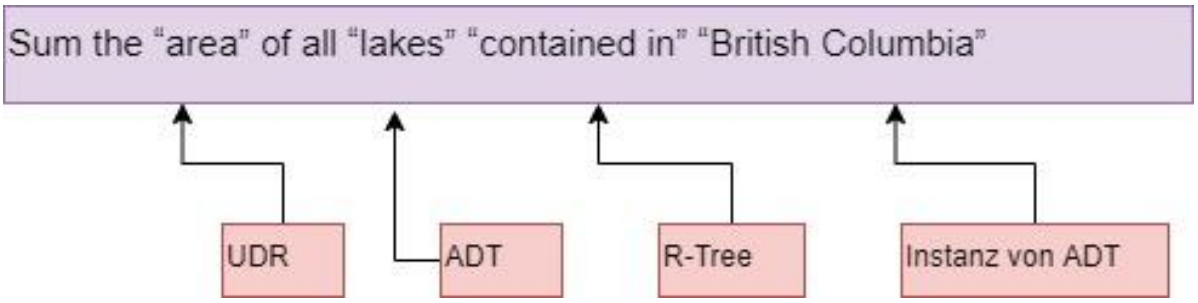

Abbildung 3~ Beispiel

das obige Beispiel versucht zu erklären, wie alle oben genannten Funktionen in einer SQL-Abfrage verwendet werden können. **z.B** area ist eine **benutzerdefinierte Routine oder Funktion**, die die **Fläche** eines bestimmten **abstrakten Datentyps** oder **Benutzerdefinierte DatenTyp**, in diesem Fall eines "**Lake**", berechnen wird. British Columbia ist eine "**Instanz**" eines anderen benutzerdefinierten Datensatzes namens "**Provinz**"."**contained in**" ist ein Suchmechanismus, der einen R-Baum verwendet, um alle "Lakes" zu suchen, die in der Provinz **British Columbia** vorhanden sind. und die SQL-Abfrage berechnet die Fläche aller in **British Columbia** vorhandenen **Lakes**.

3.4 Relevanz des objektrelationalen Modells in der Praxis

*Quellenangaben im Literaturverzeichnis - **

Das folgende Diagramm aus objektrelationale Datenbank Verwaltungssystem - Michael Stonebraker klassifiziert Datenbanken nach Verwendung.

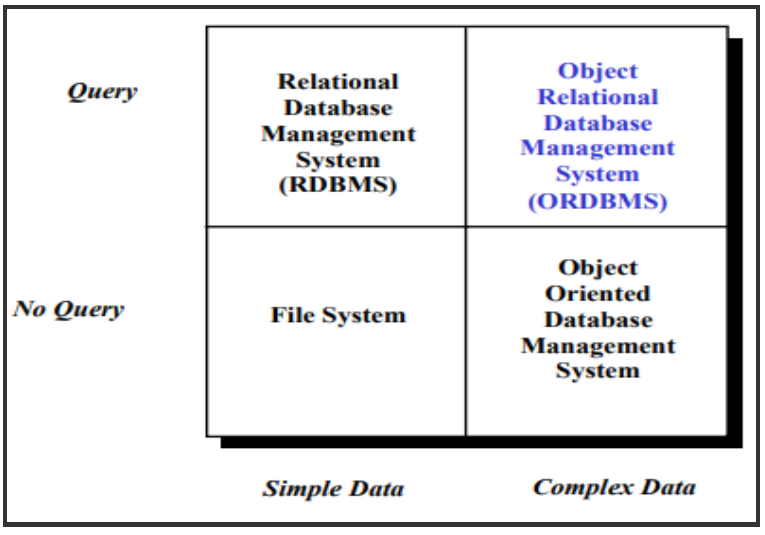

Abbildung 4 : einfache Datenbank Klassifizierungsmatrix.

Diagramm zeigt an, dass die Verwendung eines ORDBMS bestimmte Vorteile bietet, wenn Abfragen in einer Datenbank mit komplexen Daten durchgeführt werden sollen.

3.5 Unterscheidung zwischen ORDBMS und RDBMS

Quellenangaben im Literaturverzeichnis - 6

Mit Hilfe der folgenden Tabelle versucht, die relevanten Unterscheidungen zwischen rein relationalen und objektrelationalen Datenbank Verwaltungssystemen zu erläutern.

Vergleichsmetriken	RDBMS	ORDBMS
Standard Definition	SQL2	SQL3/4
Unterstützung für OOP	schlecht	Beschränkt
Idee	Basiert auf der Idee der Normalisierung, um die Speichereffizienz, Datenintegrität und Skalierbarkeit zu verbessern.	Basiert auf der Idee, RDBMS objektorientierte Unterstützung Wie Vom Benutzer erweiterbare Typen , Vererbung und ,Polymorphismus anzubieten.
Komposition	Zweidimensionale Tabellen mit Zeilen und Spalten, die verwandte Tupel oder Datensätze enthalten. Dies ist eine schlechte Darstellung von Entitäten der „realen Welt"	Versucht, diesen Nachteil zu überwinden, indem ein Feld oder eine Spalte Objekte als Werte enthält.
Umfang der verarbeiteten Daten	"Einfache" Daten werden gemäß den SQL-92-Standards behandelt	Unterstützt für komplexe Datenverarbeitungen (wie Bilder, Videos usw.)
Eindeutige Kennung	Primärschlüssel	Objektkennung (OID)
Visualisierung	ER-Diagramm wird bevorzugt	Klassendiagramm wird bevorzugt
Produktreife	Sehr mündig	Unreif; Erweiterungen sind neu und relativ unbewiesen
Kosten	Nicht so kostenintensiv	Kostenintensiv
Indizierung - Techniken	B-Tree für den schnellen Zugriff auf skalare Daten	Generics B-Tree oder R-Tree für den schnellen Zugriff auf 3-dimensionale Daten

3.6 Objektrelationalen Erweiterungen der SQL

Quellenangaben im Literaturverzeichnis - 7 | 8| 17 | 18

Eine objektrelationale Datenbank ermöglicht gegenüber einer rein relationalen Datenbank einige andere Erweiterungen, z.B. benutzerdefinierte Typen ,Nicht-atomare Attribute , Geschachtelte Relationen , Vererbung und Benutzerdefinierte Routinen, um all diese objektrelationalen Funktionalitäten zu unterstützen, wurden neue Erweiterungen in SQL eingeführt. Viele dieser Funktionen sind selbsterklärend, da sie mit der Funktionalität von oops verglichen werden können. Abgesehen von Nicht-atomare Attribute. Eine der Grundregeln des relationalen Modells ist, dass die Attribute einer Relation atomar sind. Eine objektrelationale Datenbank wie Postgres

hat diese Einschränkung nicht; Attribute können selbst Sub-Werte enthalten, auf die von der Abfragesprache aus zugegriffen werden kann. Sie können zum Beispiel Attribute erstellen, die Arrays von Basistypen sind.
Da für diesen Workshop **PostgreSQL** verwendet wird, wird in der folgenden Tabelle die relevanten SQL-Erweiterungen erläutert.

Erweiterungen	Info / Schlüsselwort	Aussehen von Abfragen
benutzerdefinierte Typen	TYPE oder DOMAIN - Wird verwendet, um einen benutzerdefinierten Datentyp zu erstellen	CREATE **TYPE** address AS (city VARCHAR(90), street VARCHAR(90)); CREATE **DOMAIN** addr VARCHAR(90) NOT NULL DEFAULT 'N/A';
Geschachtelte Relationen	ARRAY - Postgres hat nicht die Möglichkeit, verschachtelte Tabellen zu verwenden, da alternativ Arrays verwendet werden	CREATE TABLE Event ("Event_id" serial PRIMARY KEY, "description" EventDescription **ARRAY**);
Vererbung	INHERITS -wird verwendet, um ein Tabellen Attribut von einem anderen zu erben	CREATE TABLE guitarist (maximum_picking_speed int, guitar_of_choice VARCHAR) **INHERITS**(musician);
Nicht-atomare Attribute	ARRAY - Arrays werden als nicht atomare Attribute verwendet	CREATE TABLE SAL_EMP (name text, pay_by_quarter **int4[]**, schedule **char16[][]**);

CREATE TYPE vs CREATE DOMAIN

CREATE TYPE registriert einen neuen Datentyp zur Verwendung in der aktuellen Datenbank. Der Benutzer, der einen Typ definiert, wird sein Besitzer wohingegen **CREATE DOMAIN** erstellt eine neue Domäne. Eine Domäne ist im Wesentlichen ein Datentyp mit optionalen Einschränkungen (Einschränkungen der zulässigen Wertemenge). Der Benutzer, der eine Domain definiert, wird ihr Besitzer.

§ 4 PRAKTISCHER TEIL DES WORKSHOPS

Quellenangaben im Literaturverzeichnis - 11

Wie bereits erwähnt, ist der Workshop in zwei Phasen unterteilt **Implementierung mit Postgres** und **Implementierung mit Java.** (Schauen Sie sich die verwendeten Tools an)
Hinweis : Alle Fragen zu den Aufgaben werden während des Workshops gestellt

Implementierung mit Postgres
Zuerst werden einige der Grundlagen rein relationaler Datenbankmodellabfragen erläutert.Damit Sie die Funktionsweise relationaler Datenbank Abfragen verstehen. Dann haben wir versucht, die Erweiterungen des objektrelationalen Modells gegenüber dem relationalen Modell wie

benutzerdefinierte Typen, nicht-atomare Attribute, geschachtelte Relationen, Vererbung usw. zu erklären.

Implementierung mit Java

Wir haben kleine Aufgaben mithilfe **IntelliJ IDE und EDU Tools** plugin implementiert, die versuchen, einem Anfänger zu erklären, wie er mit Bildern in PostgreSQL umgehen kann. Außerdem haben wir Aufgaben erstellt, um zu klären, wie ein einfaches Java-Objekt auf die Datenbank abgebildet werden kann und wie einfach CRUD-Funktionen auf diesem vordefinierten Java-Objekt implementiert werden können.

Bevor wir fortfahren, möchten wir einen kurzen Überblick über das Erscheinungsbild von Abfragen in Postgres geben. Grundsätzlich gibt es drei Hauptkategorien von SQL-Befehlen **DDL – Data Definition Language** **(z.B CREATE , DROP , ALTER)** kann verwendet werden, um das Datenbankschema zu definieren. Es behandelt lediglich Beschreibungen des Datenbankschemas und wird zum Erstellen und Ändern der Struktur von Datenbankobjekten in der Datenbank verwendet. **DQL – Data Query Language** **(z.B SELECT)** kann zum Ausführen von Abfragen für die Daten in Schema Objekten verwendet werden. Der Zweck des DQL-Befehls besteht darin, eine Schema-Beziehung basierend auf der an ihn übergebenen Abfrage abzurufen. **DML – Data Manipulation Language** **(z.B INSERT , UPDATE , DELETE)** befasst sich mit der Manipulation von Daten, die in der Datenbank vorhanden sind und zu DML oder Data Manipulation Language gehören, und dies schließt die meisten SQL-Anweisungen ein.

Allgemeine Übersicht der verwendeten Befehle

Befehle	Erläuterung
`DROP TABLE IF EXISTS books, authors;`	löscht die books, authors-Tabellen, falls sie bereits vorhanden sind
`DROP TYPE IF EXISTS address;`	löscht den Typ , address, falls diese bereits vorhanden sind
`DROP TABLE IF EXISTS musician , country , guitarist CASCADE;`	CASCADE drop wird verwendet, wenn Tabellen irgendwie miteinander verbunden sind
`CREATE TABLE IF NOT EXISTS authors ( id serial PRIMARY KEY, name VARCHAR(25) );`	Erstellt eine Tabelle authors mit der ID als Primärschlüssel und Namensattribut, falls diese Tabelle nicht bereits vorhanden sind
`CREATE TYPE address AS (city VARCHAR(90), street VARCHAR(90));`	Definiert einen benutzerdefinierten Typ(UDT) mit CREATE TYPE

4.1 Implementierung mit Postgres

Im ersten Abschnitt wird die Grundlagen rein relationaler Datenbank Abfragen erläutert, die über eine Reihe von SQL-Befehlen ausgeführt werden können. Dafür wird eine SQL-Datei **Rdb.sql(↓)** erstellt, die zwei Tabellen books and authors erstellt und dann fügt einige Daten in sie ein. Diese SQL-Datei dient nur dazu, sich mit den Konzepten relationaler Datenbanken und einfacher SQL-Abfragen vertraut zu machen. *Jetzt wird in Abschnitt zwei* **vier kleine Aufgaben** als SQL-Dateien erstellt, um die Erweiterungen des objekt relationalen Modells gegenüber dem relationalen Modell zu erklären.

Aufgabe-1

UDT.sql(↓) befasst sich damit, wie wir unseren eigenen benutzerdefinierten Datentyp(z.B addr , address) erstellen und in einer anderen Tabelle(location) verwenden können.

wichtige SQL-Befehle in der Datei UDT.sql

```
CREATE DOMAIN addr VARCHAR(90) NOT NULL DEFAULT 'N/A';
                                    -- Definieren Sie einen benutzerdefinierten Typ(UDT) mit CREATE DOMAIN
CREATE TYPE address AS (city VARCHAR(90), street VARCHAR(90));
                                    -- oder Definieren Sie einen benutzerdefinierten Typ(UDT) mit CREATE TYPE
CREATE TABLE location (ship_address addr, full_address address);
                                    -- Verwenden Sie die bereits erstellten UDTs in der Location-tabelle
INSERT INTO location VALUES ('ship_address1', ('Northampton1', 'Tower St1')::address);
INSERT INTO location VALUES ('ship_address2', ('Northampton2', 'Tower St2')::address);...
-- Daten in die Tabelle einfügen. Beachten Sie, dass das zweite Tupel als Adresse eingegeben wurde. weil es vom Adresstyp
                                    sein sollte.
```

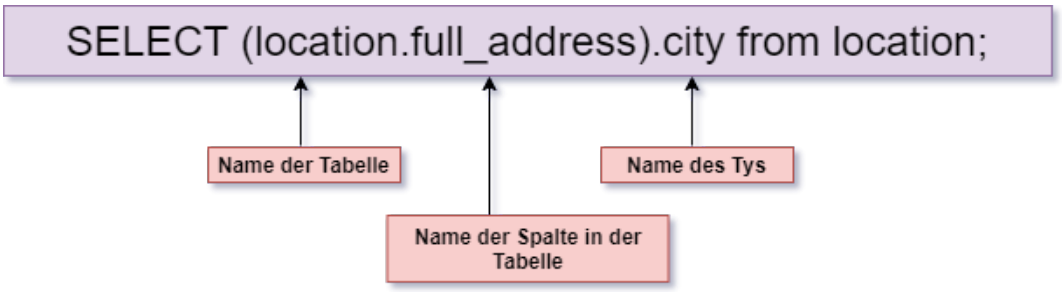

Abbildung 5 : *Auswahlkriterien*: Sie können mit der Punktnotation auf die Typen zugreifen

Fragen :

-- Überprüfen Sie, wie alle Spalten und Zeilen in der Tabellenposition aussehen.

-- Zeigen Sie den Namen der Straße an, deren Stadtname Northampton 4 ist

-- Verwenden Sie die Punktnotation, um die Informationen über die Stadt und die Straße aus der Standorttabelle anzuzeigen

Lösung von Fragen / verbleibenden Befehlen in UDT.sql

```
SELECT * FROM location;
                                                    --Die Ausgabe der Spalte full_address ist ein zusammengesetztes Tupel
SELECT (location.full_address).street from location where (location.full_address).city = 'Northampton4';
SELECT (location.full_address).city, (location.full_address).street from location;
                                                    --Sie können mit der Punktnotation auf die Typen zugreifen
```

Aufgabe-2

INHERITANCE.sql(⇩) befasst sich mit Vererbung. Hier legen wir drei Tabellen Musician , Country und Guitarist an. So, dass Musician einen Verweis auf die Tabelle Country hat und die Tabelle Guitarist erbt die Tabelle Musician. Es ist wichtig zu beachten, dass Guitarist nur Zugriff auf seinen eigenen Inhalt hat, erhält jedoch Attribute aus der übergeordneten Tabelle(Musician),die ihre Vorgänger Tabelle ist. Und normalerweise hat der Musician Zugriff auf seine eigenen Daten, aber auch auf die Daten der Gitarristen, weil sie seine Eltern sind.

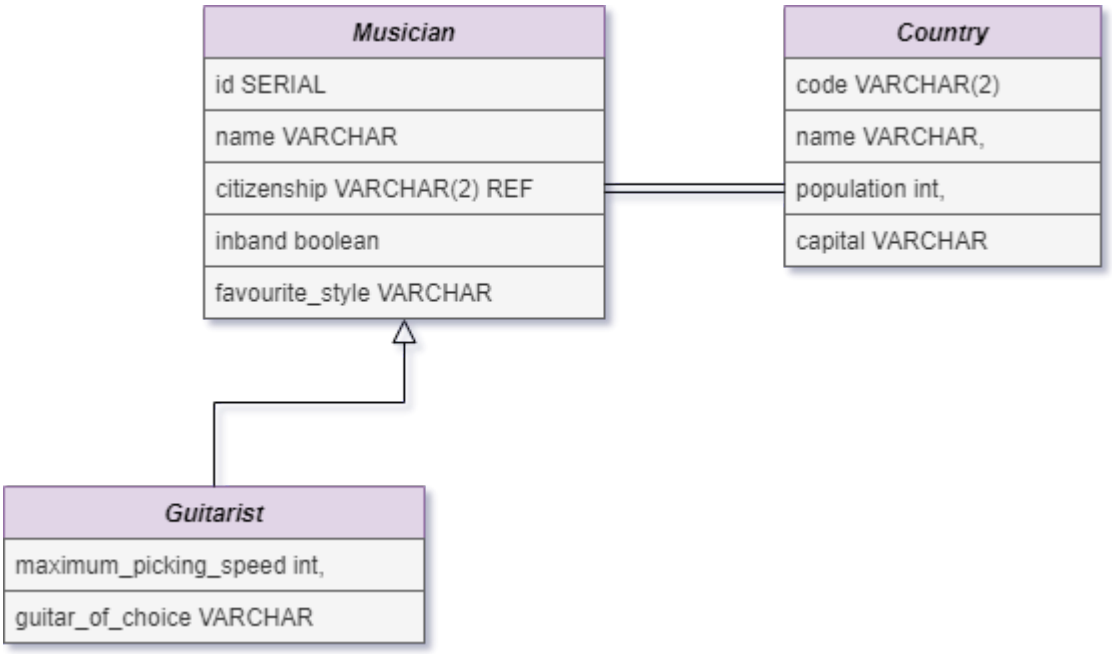

Abbildung 6 : Vererbung Beispiel

wichtige SQL-Befehle in der Datei INHERITANCE.sql

```
CREATE TABLE country (
code VARCHAR(2) PRIMARY KEY,
name VARCHAR,
population int,
capital VARCHAR
);
                                                    -- Erstellen Sie die country-tabelle
CREATE TABLE musician (
```

```
id SERIAL PRIMARY KEY,
name VARCHAR,
citizenship VARCHAR(2) REFERENCES country(code),
inband boolean,
favourite_style VARCHAR
);
                                        -- Erstellen Sie eine musician-Tabelle mit Bezug auf Tabelle country.
CREATE TABLE guitarist (
maximum_picking_speed int,
guitar_of_choice VARCHAR
) INHERITS(musician);
                                -- Der guitarist erbt die Attribute von der übergeordneten Tabelle mit dem Namen musician
                                        -- Hinweis : Der guitarist erbt die Attribute, nicht den Inhalt
INSERT INTO country VALUES ('US', 'United States of America',25000,'Washington');.....
                                                -- Daten in country-tabelle einfügen
INSERT INTO musician(name, citizenship, inband, favourite_style) VALUES ('Bongoman',
'US',false,'speedybongo');.....
                                                -- Daten in musician-tabelle einfügen
INSERT INTO guitarist(name, citizenship, inband,
favourite_style,maximum_picking_speed,guitar_of_choice) VALUES
('Runnerboy1', 'IN',true,'bongo1',30,'choice1');.....
                                                -- Daten in guitarist-tabelle einfügen
```

Fragen :

-- Zeigen Sie alle Spalten der Gitarristentabelle an, um zu überprüfen, welche Spalten von der übergeordneten Tabelle übernommen wurden.

-- Schreiben Sie die Abfrage, um alle Daten der Musikertabelle anzuzeigen

-- Schreiben Sie die Abfrage, um nur die Daten der Musiktabelle anzuzeigen

Lösung von Fragen / verbleibenden Befehlen in INHERITANCE.sql

```
SELECT * FROM guitarist;
        --guitarist hat nur Zugriff auf seinen eigenen Inhalt, erhält jedoch Attribute aus der übergeordneten Tabelle(musician)
SELECT * FROM musician;
                --musician hat Zugriff auf seine eigenen Daten, aber auch auf Gitarristen Daten, da es seine Eltern sind.
                                        --Mit dieser Abfrage können wir also alle Daten abrufen.
SELECT * FROM only musician;
                                --Mit dieser Abfrage können wir nur die Daten der musician-tabelle abrufen
```

Aufgabe-3

ARRAYS.sql(↓) befasst sich mit Arrays. Diese Datei klärt, wie wir Arrays in Postgres erstellen und darauf zugreifen können. Es ist wichtig zu beachten, dass Arrays in Postgres **nicht 0-indiziert** sind und Arrays können mit einfachen (**wie Integer**) oder komplexen (**wie Text**) Datentypen definiert werden.Es ist auch **Slicing** bei der Auswahl von Daten aus einem Array möglich, wie in jeder anderen Programmiersprache heutzutage.

wichtige SQL-Befehle in der Datei ARRAYS.sql

```
---------------------------------------------------------------------------------------------------- Beispiel-1
CREATE TABLE tictactoe (
   squares   integer[3][3]
);
                          --Erstellen Sie eine Tabelle mit einem zweidimensionalen Ganzzahl Array mit dem Namen "Quadrat"
INSERT INTO tictactoe VALUES('{{1,2,3},{4,5,6},{7,8,9}}');
                                                                    --Werte in das Array einfügen
SELECT squares[3][2] FROM tictactoe;
                                                                --8 wird die Ausgabe dieser Auswahlabfrage sein


---------------------------------------------------------------------------------------------------- Beispiel-2
CREATE TABLE messages (
    msg  text[]
) ;
                                                    --Erstellen Sie eine Tabelle mit einem Text Array mit dem Namen "msg"
INSERT INTO messages VALUES ('{"hello", "world"}') ;
INSERT INTO messages VALUES ('{"I", "feel", "so", "free"}') ;
                                                                    --Werte in das Array einfügen
SELECT msg[2] FROM messages ;
                  --Position 2 wird ausgewählt. Daher besteht die Ausgabe aus zwei Zeilen mit den Werten "world" und "feel".
SELECT msg[2:4] FROM messages;
                                                                --Slicing ist auch möglich
      --Position 2 bis 4 wird ausgewählt. Daher besteht die Ausgabe aus zwei Zeilen mit den Werten "world" und "feel", "so",
                                                                                                        "free".
```

Aufgabe-4

NESTED_TABLES.sql(↓) befasst sich mit Geschachtelte Relationen. Es ist wichtig zu beachten, dass Postgres den Begriff der verschachtelten Tabelle nicht unterstützt. Es werden jedoch Arrays beliebiger Typen unterstützt, das wohl ein gleichwertiges Konzept ist. In dieser Aufgabe haben wir einen Enum-Typ (EventDescriptionType) erstellt, der als vordefinierte Tabelle interpretiert werden kann und dieser EventDescriptionType wird als Datentyp in einem anderen benutzerdefinierten Datentyp namens EventDescription verwendet. Später wird diese EventDescription als Array verwendet, wodurch eine verschachtelte Tabellenbeziehung entsteht. Beim Einfügen der Daten in die Tabelle sollte das Array vor dem Einfügen auf EventDescription[] typisiert werden.

wichtige SQL-Befehle in der Datei NESTED_TABLES.sql

```
CREATE TYPE EventDescriptionType AS ENUM (
 'felt report',
 'Flinn-Engdahl region',
 'local time',
 'tectonic summary',
 'nearest cities',
```

```
 'earthquake name',
 'region name'
);
                                                    --Erstellen Sie einen enum typ "EventDescriptionType"
CREATE TYPE EventDescription AS (
 "text" text,
 "type" EventDescriptionType
);
                --Erstellen Sie einen anderen Typ EventDescription, der den EventDescriptionType als Argument verwendet
CREATE TABLE Event (
 "Event_id"              serial PRIMARY KEY,
 "description"           EventDescription ARRAY
);
                                                --Erstellen Sie die Tabelle Event mit der EventDescription als Array
INSERT INTO Event values (1,ARRAY[('L','felt report')]::EventDescription[]);
INSERT INTO Event values (2,ARRAY[('R','region name'),('P','nearest cities')]::EventDescription[]);
                                                                --Werte in Tabelle einfügen
                                                --Das Array sollte vor dem Einfügen in EventDescription[] typisiert werden
```

Fragen :

-- Schreiben Sie eine Abfrage, um den gesamten Inhalt der Tabelle Ereignis anzuzeigen

-- Schreiben Sie eine Abfrage, um nur das erste Element der Tabelle Arrays from Event anzuzeigen

-- Schreiben Sie eine Abfrage, um nur das "type" attribut des ersten Elements der Arrays from Event-Tabelle anzuzeigen

Lösung von Fragen / verbleibenden Befehlen in NESTED_TABLES.sql

```
SELECT * FROM Event;
SELECT Event.description[1] FROM Event;
                                        --Verwenden Sie die Punktnotation, um die erste description spalte auszuwählen
SELECT (Event.description[1]).type FROM Event;
                        --Verwenden Sie die Punktnotation, um den Typ aus der Spalte mit der Erstbeschreibung auszuwählen
```

4.2 Implementierung mit Java

Quellenangaben im Literaturverzeichnis - 9 | 12

in diesem Teil wird erklärt wie man mit Bildern in PostgreSQL umgehen kann. Außerdem wird Aufgaben erstellt, um zu klären, wie ein einfaches Java-Objekt auf die Datenbank abgebildet werden kann und wie einfach **CRUD(CREATE,READ,UPDATE,DELETE)-Funktionen** auf diesem vordefinierten Java-Objekt implementiert werden können. Um eine Verbindung mit der Postgres-Datenbank auf unserem lokalen Rechner von Java aus herzustellen, haben wir **JDBC+Maven** als Anwendungsprogrammierschnittstelle(API) verwendet. Es gibt sechs kleine Aufgaben(Hier verfügbar), die wir mit dem **IntelliJ IDE- und EDU Tool-Plugin** erstellt haben, die

im Folgenden beschrieben werden. Wir empfehlen für diesen Workshop eine **vollständige Offline-Vorbereitung** Ihres Systems wie in Installationsanweisungen beschrieben.

Hinweis: Es war nicht möglich, die Erläuterungen zu jeder Java-Codezeile zu geben, die wir in dieser Dokumentation oder in diesem Workshop implementiert haben. Daher haben wir versucht, das Konzept zu erklären. Aber wir haben die Erklärung für jede Zeile als Kommentar in den Aufgaben Dateien angegeben. Ein weiterer Punkt, auf den Sie achten sollten, ist, dass die kleinen Aufgaben oder die von den Teilnehmern zu füllenden Zeilen als Lücken in den Aufgaben erwähnt werden, die wir mit dem Edu Tools-Plugin implementiert haben.

Aufgabe-1

task1 (↓) beschäftigt sich damit, wie wir mit JDBC als API eine Verbindung zur Postgres-Datenbank herstellen können. In dieser Aufgabe können Sie lernen, wie eine JDBC-URL aussieht und wie man sie mit der java.sql-Bibliothek verwenden kann, um die Verbindung mit der Datenbank herzustellen. **Name unserer Datenbank : test**

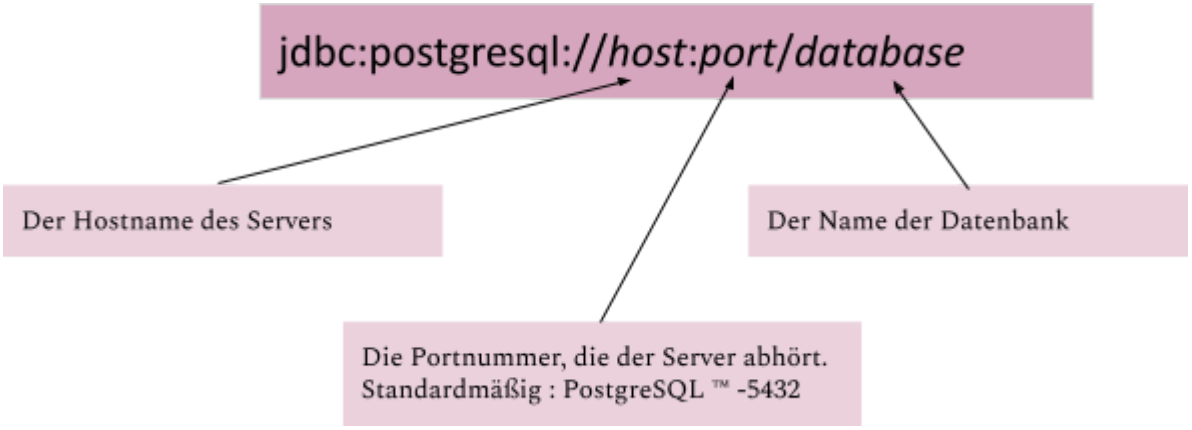

Abbildung 7 : JDBC URL

Um eine Verbindung herzustellen, müssen Sie eine Connection-Instanz von JDBC erhalten. Dazu verwenden Sie die Methode DriverManager.getConnection():

```
Connection db = DriverManager.getConnection(url, username, password);
```

Aufgabe-2

task2 (↓) Da Postgres komplexe Datentypen wie "**bytea**" erlaubt, die zum Speichern von Bildinhalten verwendet werden können. Daher beschäftigt sich diese Aufgabe damit, wie wir ein Bild in unserer Datenbank speichern können. Um den Inhalt einer bereits vorhandenen Bild zu lesen, haben wir die **FileInputStream-Bibliothek** in Java verwendet.

Aufgabe-3

task3 (↓) Da wir nun den Inhalt des Bildes in unsere Datenbank geschrieben haben, macht es Sinn, den Inhalt aus der Datenbank abzurufen und dem Endbenutzer anzuzeigen. Daher befassen

wir uns in dieser Aufgabe mit dem Abrufen des Bildes aus der Datenbank und zeigen es dann dem Benutzer mithilfe der **JFrame-GUI** von Java an.

Aufgaben-4.5.6

Diese Aufgaben befassen sich mit dem Mapping(siehe oben) einer einfachen Java-Klasse namens **User** mit den Attributen **id**, **name** , **password & age** auf eine Tabelle namens users in der Datenbank und wie CRUD-Funktionalitäten mit diesem Objekt implementiert werden können.Um CRUD-Funktionalitäten zu implementieren, haben wir das DAO-Muster verwendet.
Das Data Access Object Pattern oder DAO-Muster wird verwendet, um Low-Level-APIs oder Operationen für den Datenzugriff von High-Level-Geschäfts Diensten zu trennen. Im Folgenden sind die Teile des Data Access Object Pattern aufgeführt.

- **Data Access Object Interface** - Diese Schnittstelle definiert die Standardoperationen, die auf einem Modellobjekt (oder mehreren) ausgeführt werden.*(UserDao)*
- **Konkrete Klasse Data Access Object** - Diese Klasse implementiert die oben genannte Schnittstelle. Diese Klasse ist dafür verantwortlich, Daten aus einer Datenquelle zu holen, die eine Datenbank / xml oder ein anderer Speichermechanismus sein kann.*(UserDaoImpl)*
- **Model Object oder Value Object** - Dieses Objekt ist ein einfaches POJO, das Get-/Set-Methoden enthält, um die mit der DAO-Klasse abgerufenen Daten zu speichern.*(User)*

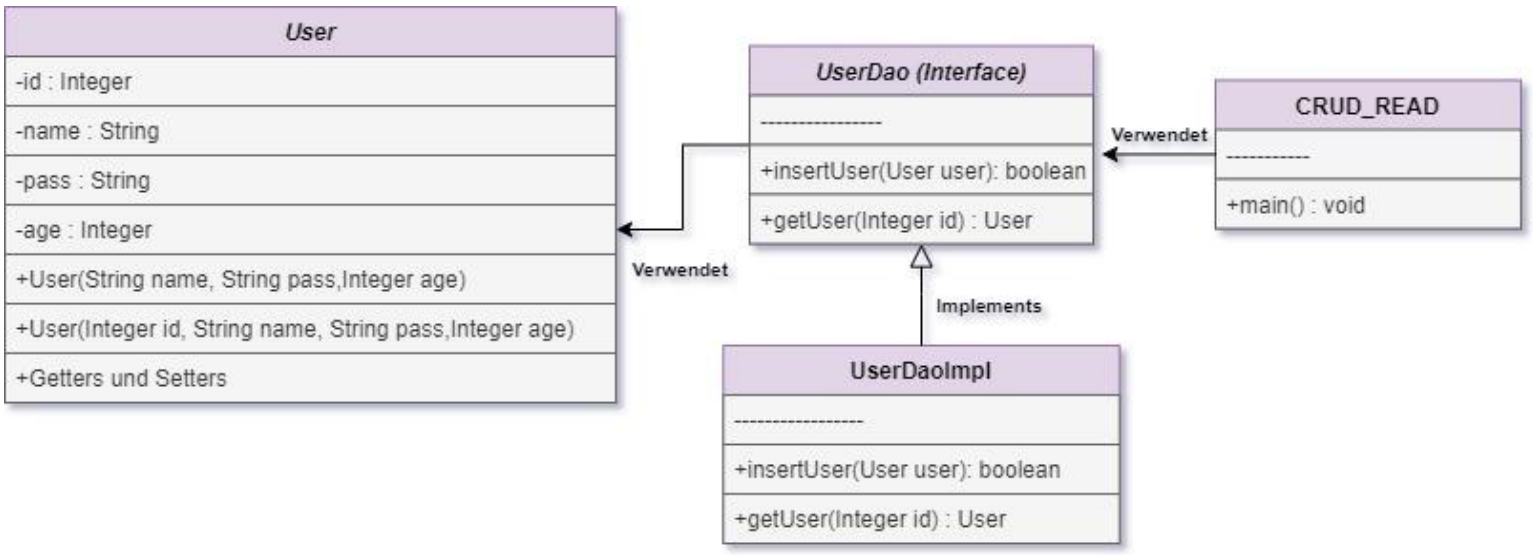

Abbildung 8 : Data Access Object Pattern

Aufgabe-4

task4 (↓) erstellt einfach eine Tabelle namens **users** mit den oben genannten Attributen

Aufgabe-5

task5 (↓) In dieser Aufgabe implementieren wir zunächst die Methode **insertUser()** in unserer Klasse **UserDaoImpl**, um damit Benutzer in unsere Datenbank einzufügen. Dann erstellen wir in

unserer Klasse **CRUD_INSERT**, die die **main()** hat, vier Benutzer und fügen sie nacheinander in unsere Tabelle "**users**" ein.

Aufgabe-6

task6 (↓) In letzter Aufgabe implementieren wir zunächst die Methode **getUser(id)** in unserer Klasse **UserDaoImpl**, um einen Benutzer mit einer bestimmten ID abzurufen. Dann erstellen wir in unserer Klasse **CRUD_READ**, die die main() hat, um einen Benutzer abzurufen und die gewünschten Details anzuzeigen.

Wenn Sie nun die Java-Implementierung des Codes richtig verstanden haben, können Sie die **UPDATE- und DELETE-Funktionalitäten als Herausforderung **selbst implementieren****

§ 5 INTERESSANTE FRAGEN ZU ORDBMS

Dieser Abschnitt wird einige der Fragen beantworten, die sich aus unserem Thema der objektrelationalen Datenbank auftreten könnten.

5.1 Was sind die Vorteile der relationalen Objektdatenbank ?

Quellenangaben im Literaturverzeichnis - 15

zu den vielen Vorteilen der Verwendung von objektrelationalen Datenbanken gegenüber reinen relationalen Datenbanken gehört die Unterstützung von objektorientierten Konzepten wie Vererbung, Polymorphismus, langfristige Transaktionen. Und auch die Unterstützung für komplexe Objekte und Datenkapselung. Eines der anderen Ziele der objektrelationalen Datenbank ist die Integration von Datenbanken und Programmiersprachen. Auf die objektrelationale Datenbank kann gleichzeitig zugegriffen werden und In der Teamarbeit ist es möglich, die Ergebnisse anderer bei der Arbeit zu verwenden.

5.2 In welcher Bereiche werden ORDBMS verwendet ?

Quellenangaben im Literaturverzeichnis - 5 | 16

ORDBMS wurde in den letzten Jahren von der Industrie weitgehend übernommen. Einige der weltweit führenden Unternehmen wie BMW, Vereinte Nationen und Bloomberg nutzen ORDBMS. ORDBMS wird nicht nur in einer Branche eingesetzt, sondern fast überall. Vom Automobil zum Wassermanagement. Von der Archäologie bis zum Wetterdienst. ORDBMS ist heutzutage überall und die Anzahl der Bereitstellungen nimmt weltweit zu. Die Finanzindustrie, staatliche GIS-Daten, Fertigung, Web-Technologie und NoSQL-Workloads sowie wissenschaftliche Daten zeigen einige Beispiele dafür, wie ORDBMS in der realen Welt verwendet wird oder werden kann.

Hinweis ~ **PostGIS** ist eine Erweiterung zur Unterstützung von geografischen Daten mit vielen nützlichen Datentypen von Funktionen.

5.3 Was sind die Stärken und Schwächen von ORDBMS ?

Quellenangaben im Literaturverzeichnis - 6 | 19 | 20

Stärken	Schwächen
Datenmodellierung Attribute, die nicht nur aus einem Wert bestehen, sondern aus einer Wertemenge können je nach Anforderung als variables Array, Set, List oder Multi Set modelliert werden.	**hohe Komplexität und Kosten** Aufgrund der Komplexität, der Unausgereiftheit und der hohen Kosten, die mit der Wartung von ORDBMS verbunden sind, versuchen Unternehmen normalerweise, bei den üblichen relationalen Datenbanken zu bleiben.
Datenbankentwurf Erstellung von Datenbankmodell anhand eines ER-Diagramms oder UML ist erleichtert.	**Geringe Einfachheit** Befürworter von RDBMS sind der Meinung, dass die wesentliche Einfachheit und Reinheit relationaler Modelle bei diesen Arten von Erweiterungen verloren geht.
Bewahrt Relationale Merkmale bewahrt den bedeutenden Wissens- und Erfahrungsschatz, der in die Entwicklung relationaler Anwendungen eingeflossen ist	**Angebot von Objektmodellen** ORDBMS-Anbieter versuchen immer, Objektmodelle als Erweiterungen des relationalen Modells mit einigen zusätzlichen Komplexitäten darzustellen. Diese Möglichkeit verfehlt den Punkt der Objektorientierung und unterstreicht die große semantische Lücke zwischen diesen beiden Technologien.

5.4 Gibt es Programmierschnittstellen ?

Quellenangaben im Literaturverzeichnis -15

Eine Programmierschnittstelle z.B API dient dazu, Informationen zwischen einer Anwendung und einzelnen Programmteilen standardisiert auszutauschen. Die API ermöglicht es, die Programmierung zu modularisieren und dadurch zu vereinfachen. Die einzelnen über eine API angebundenen Programmteile erfüllen spezifische Funktionen und sind vom Rest der Applikation klar getrennt. Für verschiedene Programmiersprachen stehen eine Reihe von Anwendungs Programmierschnittstellen (siehe unten) zur Verfügung, mit denen wir die ORDBMS-Funktionen nutzen können.

Python	SQLAlchemy , psycopg2 2.8.6 (um eine Verbindung zu postgresql herzustellen)
C | C++	pqxx/pqxx(um eine Verbindung zu postgresql herzustellen)
Ruby	Pg ist die Ruby-Schnittstelle zum PostgreSQL-RDBMS.
PHP	Doctrine , pg_connect(um eine Verbindung zu postgresql herzustellen)
Java	Java Database Connectivity(um eine Verbindung zu postgresql herzustellen)

5.5 Welche Relevanz hat CAP-Theorem ?

Quellenangaben im Literaturverzeichnis - 13

In der theoretischen Informatik besagt das CAP-Theorem, **dass es für einen verteilten Datenspeicher unmöglich ist, mehr als zwei der folgenden drei Garantien gleichzeitig bereitzustellen**:

- *Consistency (Konsistenz) :* bedeutet, dass alle Knoten in einem Cluster gleichzeitig dieselben Daten sehen.
- *Availability (Verfügbarkeit) :* Lesen und Schreiben müssen immer erfolgreich sein. Mit anderen Worten, ein Knoten muss zu jedem Zeitpunkt für Benutzer verfügbar sein.
- *Partition tolerance (Partitionstoleranz) :* Dies bedeutet, dass das System auch dann weiter funktioniert, wenn unterwegs beliebige Nachrichten verloren gehen. Ein Netzwerkpartition Ereignis tritt auf, wenn auf ein System nicht mehr zugegriffen werden kann.

Hinweis ~ PostgreSQL, Oracle, DB2 usw. stellen Ihnen CAp ("konsistent" und "verfügbar") zur Verfügung, während NoSQL-Systeme wie MongoDB und Cassandra Ihnen cAP ("verfügbar" und "partitionstolerant") zur Verfügung stellen. Aus diesem Grund wird NoSQL oft als letztendlich konsistent bezeichnet.

6 Fazit

Diese Studie wird durchgeführt, um mehr über objektrelationale Datenbanken zu erfahren. Nachdem Sie diesen Dokumentation gelesen haben, sind Sie nun mit den Konzepten objektorientierter Datenbanken und relationaler Datenbanken vertraut, wie sie funktionieren und wie sie sich unterscheiden. Und auch haben Sie festgestellt, dass das objektrelationale Datenmodell eine Erweiterung des relationalen Modells ist, jedoch eigene Sonderfunktionen hat und Sie haben sich mit seiner Herangehensweise vertraut gemacht. In Folgenden kannten Sie Funktionen von ORDBMS, welche Möglichkeiten stellen sie uns zur Verfügung und Datenbank Suchmethode mit komplexen Daten. In Implementierung mit Postgres verstanden Sie die Funktionsweise relationaler Datenbank Abfragen durch die Erkenntnis von einige der Grundlagen rein relationaler Datenbankmodellabfragen, dass wir es im Artikel als kleine Aufgabe in SQL-Dateien erwähnt. Und in Implementierung mit Java lernten Sie, dass wie Sie mit **CRUD** Funktionalität in PostgreSQL mit hilfe **IntelliJ IDE und EDU Tools** umgehen kann. Am Ende der Dokumentation haben wir auch versucht, einige Fragen zu beantworten, die wir für relevant halten und bei der Beschäftigung mit diesem Thema auftreten könnten.Alles in allem haben wir versucht, alle theoretischen und praktischen Aspekte objektrelationaler Datenbanken abzudecken, die sich für Anfänger mit ein wenig Programmierkenntnissen oder sogar dem Willen, sie zu lernen, als würdig erweisen können.

7 Anhang

Dateiname	Beschreibung
Gitlab Repository , Hier verfügbar	GitLab Repository des durchgeführten Workshops
Tools	
SQLFiddle, Hier verfügbar	Online Datenbank Tool
PostgreSQL, Hier verfügbar	Objektrelational Datenbank
IntelliJ, Hier verfügbar	Integrated Development Environment
Sql-Dateien	
Rdb.sql ,Hier verfügbar	Behandelt Grundlagen relationaler Datenbankabfragen
UDT.sql ,Hier verfügbar	Befasst sich mit benutzerdefinierten Datentypen
INHERITANCE.sql ,Hier verfügbar	Befasst sich mit Vererbung
ARRAYS.sql ,Hier verfügbar	Befasst sich mit Arrays
NESTED_TABLES.sql,Hier verfügbar	Befasst sich mit dem Konzept der verschachtelten Tabellen
VORBEREITUNG DES COMPUTERS, Hier verfügbar	ANLEITUNG ZUR VORBEREITUNG DES COMPUTERS FÜR TEILNEHMER

8 Add-ons

Hier finden Sie einige Java-Dateien, die unter bestimmten Szenarien nützlich sein können. Diese Dateien und deren Details finden Sie im **Add-on-Ordner im Git-Repo** :

- **JavaPostgreSqlWriteImage** und **JavaPostgreSqlReadImage** sind zwei Dateien, die sich mit dem Schreiben und Lesen eines Bildes aus der Datenbank befassen
- **JavaPostgreSqlPopulate** und **JavaPostgreSqlRetrieve** sind zwei Java-Dateien, die hier sich mit dem Einfügen und Abrufen von Daten befassen
- **JavaPostgreSqlPopulate_UDT** und **JavaPostgreSqlColumnHeaders_UDT** sind zwei Java-Dateien, die hier sich mit dem Einfügen und Abrufen von Daten befassen

9 Literaturverzeichnis

Quelle

1.“Relational Database Management System.” Wikipedia, Wikimedia Foundation, 26 Mar. 2019. (**)

2.“Object-Relational Database.” Wikipedia, Wikimedia Foundation, 8 July 2018. (**)

3.“What is an Object-Relational Database Management System (ORDBMS)? , techopedia.com (**)

4.”The Definitive Guide to db4o pp 31-46” | Comparing the Object and Relational Data Models (**)

5.Applications of Object Relational Database Management Systems at BCS (**)

6.Comparison of RDBMS,OODBMS & ORDBMS | Gheorghe SABAU Bucharest (**)

7.”Non-Atomic Values” | Advanced Postgresql Features, postgresql.org, 20th May 2021(**)

8.PostgreSQL 13.3 Documentation, postgresql.org, 20th May 2021 (**)

9.”Create, read, update and delete” Wikipedia, Wikimedia Foundation, 18 May 2021 (**)

10.”Connecting to the PostgreSQL™ Database” | Chapter 3. Initializing the Driver(**)

11.”SQL | DDL, DQL, DML, DCL and TCL Commands” , geeksforgeeks.org, 07 Apr, 2021 (**)

12.Data Access Object Pattern | Implementation ,tutorialspoint.com (**)

13.”CAP theorem” Wikipedia, Wikimedia Foundation, 12 May 2021 (**)

14.”Storing Binary Data”, PostgreSQL 7.4.30 Documentation, 20th May 2021 (**)

15.PostgreSQL.” Wikipedia, Wikimedia Foundation, 20 May 2021. (**)

16.”SOLUTIONS – WHO USES POSTGRESQL” | cybertec-postgresql.com. (**)

17.”CREATE TYPE” | PostgreSQL 9.5.25 Documentation , postgresql.org, 20th May 2021 (**)

18.”CREATE DOMAIN” | PostgreSQL 9.5.25 Documentation , postgresql.org, 20th May 2021 (**)

19.”Weaknesses of the object-oriented model in the database” | webinux.ir (**)

20.”Objektrelationale Datenbanken” | MICHAEL SPRINGMANN (AUTOR) (**)

Ressourcen zur Code Implementierung

1.”PostgreSQL Java” | zetcode.com, July 6, 2020. (**)

2.”Java – Read a file from resources folder” | mkyong.com, September 4, 2020. (**)

3.”Working with PostgreSQL in Java” | stackabuse.com. (**)

4.”Building Simple Data Access Layer Using JDBC” | dzone.com. January 08, 16. (**)

5.Data Access Object Pattern | Implementation ,tutorialspoint.com (**)

Bilder mit freundlicher Genehmigung

1. Object-Relational DBMSs – Tracking the Next Great Wave, M. Stonebraker, Morgan Kaufmann, 1999

andere in der Dokumentation verwendete Bilder wurden von uns mit draw.io erstellt.

10 Aufgabenverteilung

Rituraj Singh (Mat.Nr.: 19539):

- entscheiden, welche Tools und Datenbank für den Workshop verwendet werden sollen
- Durchführung des gesamten praktischen Teils des Workshops
- den Workshop verwalten ~ inklusive Dokumentation , PPTX für den Workshop und Code

AmirHossein Roshanzadeh(Mat.Nr.: 19660):

- Add kommentar für weitere Erklärungen von Java Code
- Sammlung von wichtigen Inhalt in theoretischen Teil
- Hilfe bei der Erstellung von Bildern für die Dokumentation bieten
- Zusammenfassung des DokumentationsInhalte

Saeide Dana (Mat.Nr.:XXXXX):

- Add kommentar für weitere Erklärungen von Java Code
- recherchieren und Antworten auf auftretende Fragen geben
- Hilfe bei der Gestaltung der Dokumentation bieten
- Fehler in der Dokumentation finden