**Was haben wir gelernt?**

1. **Was ist eine relationale DB?**

Eine relationale DB ist eine DB mit einer strukturierten Datenhaltung.

Daten hängen zusammen/stehen in einer Beziehung (mit FKs). Die Daten werden miteinander verknüpft, diese Verknüpfung nennt man Relation. Für gewöhnlich nutzt man die ID als eine Art Key.

Beispiel: SQL 🡪 MySQL: Relationales Datenbankmanagementsystem

1. **Was ist NoSql?**

Keine Relationen (Tabellen) mit Reihen und Attributen, keine Datenstruktur

Sind in einem JSON-ähnlichem Format, alles in einer Zeile.

NoSQL ist schneller, als SQL nur bestehen überhaupt keine relationen.

Somit hat NoSQL keine Foreign Keys, nosql ‘SolR’ bietet fulltext suchen an.

1. **Was ist eine Relation?**

Die Relation ist die Beziehung zwischen Datensätzen von verschiedenen Tabellen. The Connection between Records in different tables.

1. **Was ist ein Schema?**

Ist die Struktur einer Datenbank, innerhalb einer Datenbank kann es mehrere Schemen geben.

1. **Was ist DNS?**

DNS ist einfach nur das System im Internet, dass IP4 Adressen in Namen auflöst.

Löst IP-Adressen in Namen auf.

1. **Was ist JDBC?**

Definition: Java Database Connectivity

JDBC ist eine Datenbankschnittstelle für Java, wird über den Drivermanager aufgerufen.

1. **Was ist eine API?**

Ist eine Programmierschnittstelle // Application Programming Interface

1. **Was ist ein Primary Key?**

Ein PK ist eine eindeutig identifizierbare Spalte in einer Tabelle, meist eine ID. Dieser Wert darf nicht null sein und darf nicht öfter als 1 mal vorkommen.

1. **Was ist ein Foreign Key?**

Ein FK wird bei einer Verknüpfung zwischen Tabellen benutzt. Dabei wird der PK einer Tabelle in einer anderen Tabelle gespeichert um auf die Daten zuzugreifen. Durch diesen Vorgang wird die Relation zwischen den zwei Tabellen geschaffen.

1. **Was ist eine Query?**

Ist eine Anfrage an die DB bzw. eine Aktion auf der DB (select, insert, update, ....)

1. **Basic SQL**
   1. Insert

Insert into games(id, title) values(1, “Madden 20”);

Insert Into games values (1,12321,alle werte eintragen) // geht auch

* 1. Update

Update games

Set title = „TemTem“

Where title = „Madden 20“;

NUR ZUM Ändern von VALUE!

* 1. Delete

Delete from games

Where title = „TemTem“;

Zum Löschen von Datensätzen! Kann gerollbackd werden

(Drop kann nicht gerollbackd werden und hat kein where, löscht Table)

* 1. Select, from where, order, limit

Where: where year > 2000

Where name like “%berg%”

Select \*

From games g

Where year\_published > 2000

Order by year\_published

Limit 5;

Es warden alle Games, die nach dem Jahr 2000 veröffentlicht wurden ausgegeben. Dabei wurde aufsteigend nach dem Jahr sortiert und es werden nur die 1. 5 Spiele ausgegeben.

Order by:

Order by year\_published desc: absteigend nach Jahren sortieren

Limit:

Limit 5, 5: Die 1. 5 Spiele warden übersprungen, nach diesen werden die nächsten 5 Spiele ausgegeben

* 1. Inner join (implizit/explizit)

Inner join passiert implizit mit where

Explizit:

Select \*

From games g

Inner join games\_platform gp on g.id = gp.game\_id

Inner join platforms p on p.id = gp.platform\_id;

* 1. Left join

LEFT JOIN gibt alle Datensätze aus der linken Tabelle (Tabelle 1) und die übereinstimmenden Datensätze aus der rechten Tabelle (Tabelle 2) zurück. Das Ergebnis ist von der rechten Seite NULL, wenn keine Übereinstimmung vorliegt

* 1. Table alias

Select \*

From games **g**

Select \*

From games **as g**

* 1. Create database

Create database db;

Create database if exists db;

* 1. Create table

Create table games(

Id int not null auto\_increment primary key,

Title varchar(100)

);

* 1. Truncate <table>

Truncate games;

Inhalte werden aus einer Tabelle gelöscht, die Tabelle selbst wird nicht gelöscht

* 1. Drop table

Drop table games;

Drop table if exists games;

* 1. Select count(\*)

Select count(\*)

From games g

Where g.title like „%a”;

Zählt alle Spiele, die mit dem Buchstaben A anfange.

1. **Was ist ACID?**

Atomacity: Queries in einer DB werden alle als ein Ganzes gespeichert, schlägt eine Query innerhalb einer Transaction fehl, wird nichts gespeichert

Consistency: Regeln warden eingehalten, z. B.: der FK games\_id gehört zur ID von games

Isolation: Änderungen einer Transaction werden erst nach commit von außen sichtbar

Durability: Änderungen einer Transaction sind in der DB gespeichert & verfügbar, auch wenn ein Systemfehler auftritt

1. **Was ist eine Transaction?**

Einer Transaction sind Queries, die zusammengehören. D.h., dass entweder alle Queries erfolgreich abgeschlossen werden oder nicht

* 1. setAutocommit(false|true)

false: Transactions werden nicht automatisch commited

true: Transactions werden automatisch commited

* 1. start transaction;

kennzeichnet den Beginn einer transaction

* 1. commit();

Daten werden commited (dauerhaft gespeichert)

* 1. rollback();

Änderungen von Daten werden zurück genommen (geht nur solange nicht commited wurde)

1. **ResultSet**
   1. getString(1)

String mit Index 1 aus ResultSet holen, der Parameter kann auch ein String sein (Name der Spalte oder Table Alias)

* 1. getInt()

Parametertyp kann String oder int sein, bei int wird der Index übergeben, bei String der Name der Spalte oder der Table Alias

* 1. next()

springt beim ResultSet in die nächste Zeil, liefert true wenn die nächste Zeile gültig ist, liefert false wenn die nächste Zeile ungültig ist

* 1. getBoolean()

Parametertyp kann String oder int sein, bei int wird der Index übergeben, bei String der Name der Spalte oder der Table Alias

1. **Was ist die 3. Normalform?**

Die 3. Normalform, dient dazu Redundanzen zu verhindern.

In der ersten Normalform werden alle Daten in einer Tabelle angesetzt (alles Atomar ohne wiederholungsgruppen), in der zweiten und dritten werden die Daten, jeweils in mehrere kategorische Tabellen zugeordnet. Die zweite NF darf keine Spalte haben die du nicht mittels des Primärschlüssels identifizieren kannst. Die dritte NF hat keine transitiven Abhängigkeiten, alles muss durch den Primärschlüssel erkennbar sein 🡪 mehr Kategorien/ Tabellen.

1. **Was ist normalisiert?**

Daten wurden innerhalb einer DB so organisiert, dass keine bzw. möglichst wenig Redundanzen auftreten. Tabellen werden in kleinere Tabellen kategorisiert, sind aber durch einen Primärschlüssel verbunden.

1. **Was ist denormalisiert?**

Redundanzen werden hinzugefügt, um die Durchführung von Queries schneller zu machen

1. **Was ist ein model?**

Im package model werden POJO-Klassen (Plain Old Java Object) gespeichert. Sind Abbildungen der Realität (z.B. Car, Human, ...)

1. **Was ist eine JDBC Connection?**

Wird vom DriverManager returned, damit kann man Statements erzeugen, mit denen man eine Query ausführen kann. Um eine Verbindung zur Datenbank aufzustellen.

1. **Was ist ein Prepared Statement?**

Am Server kompiliert, sicherer vor SQL-Injections, optimal Ausführgeschwindigkeit

Prep.Statements schützen vor SQL-Injections (Hackerangriff). Das Prepare Statement bereitet eine Query vor. Ab dem zweiten mal in dem ein SQL-Statement genutzt wird lohnt sich der Prepared Statement und man spart dadurch Zeit.

* Connection.prepareStatement(select viewed from games where id = ?)
* Connection.setAutoCommit(false) damit der Statement blockweise commited wird
* preparedStatement.addBatch(); wenn du einen weiteren Record eintragen willst.
* Anschließend wird durch preparedStatement.executeBatch(); oder .execute().

1. **SQLException**

Die SQL Exception ist eine Checked Exception und wird bei Fehlern in der Datenbank bzw. Bei Datenbank abfragen geworfen. Auch bei Connection-Fehlern (alles was mit SQL zutun hat).

1. **Connection immer im finally Block schließen**

Die Connection muss immer geschlossen werden. Der Finally block wird immer ausgeführt und dort muss die Connection immer geschloßen werden, damit die Connection seitens der Datenbank nicht offengehalten wird. Sonst kommt die Datenbank an ein Limit wegen der Menge an Connections.

1. **Was ist Datenmapping?**

Es ist eine Programmiertechnik, um Daten aus relationalen Datenbanksystemen umzuwandeln. Die Tabellen werden dabei in Objekte umgewandelt.

1. **Separation of Concerns, Class/Method responsibility**

Methoden und Klassen machen nur eine Sache. Wenn eine Klasse mehrere Aufgaben hat sollte sie auf neue Klassen verteilt werden. Refaktoring ist dabei das Vorgehen wie du zu dem Ergebnis kommst.

1. **Was ist JPA?**

JPA ist ein Java Framework um Objekte in eine DB zu speichern. Java Persistance Application Programming Interfaces und Hybernate ist davon eine implementierung. Hybernate ist ein Mapping Tool, welches als Implementation in der JPApi genutzt wird, um eine Kommunikation zu Datenbanken und deren Managementsystemen zu gewährleisten.

1. **Was ist Hibernate?**

Hibernate ist eine Implementierung von JPA, es ist ein Object-relational Mapping Tool.

1. **Was ist Mapping?**

FRAG CHRISTIAN, WEIL ES SCHON IN DATENMAPPING BEANTWORTET WURDE!

Objekt relationship entity?

1. **Was macht @Table**

Man kann dann diese Annotation in einer Klasse verwenden, sodass diese Klasse zu einer Tabelle wird. Diese Klasse ist auch eine Tabelle in der Datenbank.

1. **Was macht @Entity**

Annotation um eine Notation darzustellen, welche OneToMany oder ManyToMany etc. Ist.

1. **Was ist OneToMany, ManyToOne, OneToOne**

OneToMany ist eine 1-N Notation, dass z.B. ein Spiel nur einen Publisher haben kann.

ManyToOne ist eine N-1 Notation, dass z.B. der Publisher mehrere Spiele produzieren kann.

OneToOne ist eine 1-1 Notation, ein Ehemann kann nur eine Ehepartnerin haben.

1. **Lazy Loading**

Methode um die Daten zu laden, erst wenn du sie brauchst. Eager Loading ist das gegenstück!

1. **Hibernate kann Tabellen erzeugen**

CHRISTIAN FRAGEN!!! AHHHHHHH

1. **Was macht der EntityManager?**

EntityManager API is used to access a database in a particular unit of work. It is used to create and remove persistent entity instances, to find entities by their primary key identity, and to query over all entities. This interface is similar to the Session in Hibernate.

1. **Was ist @Id**

@ID ist eine Annotation und verweist auf den Primary Key.