**10. ORM**

**Jak v ORM implementovat vazbu 1:1 a 1:N, pokud je více možností, uveďte je**

**Jakým způsobem lze u ORM minimalizovat počet přístupů k databázi, případně omezit zatížení sítě?**

* Měli bychom využít znalosti konkrétní databáze a navrhnout ORM tak, abychom stahovali data s co nejmenším počtem dotazů (je např. lepší použít jeden SELECT dotaz, který vrátí milion výsledků, než 1000 SELECT dotazů, kde každý vrátí 1000 výsledků).
* Místo více dotazů tedy můžeme použít spojování tabulek. Měli bychom taky stahovat pouze ta data, které potřebujeme
* Pokud navrhujeme funkci, která generuje velký počet mezivýsledků, měla by být naimplementovaná na straně databáze (jako uložená procedura/funkce), aby se mezivýsledky nemusely zbytečně přenášet přes síť

**Aplikační objekty v ORM**

**Jaký význam mají parametrizované dotazy v JDBC nebo ADO.NET? Proč je používáme?**

* Parametrizované dotazy mají při různých hodnotách stejný string, díky tomu používají stejného plánu, tudíž po několikátém spuštění jsou provedeny rychleji
* Dále parametrizované dotazy řeší problém SQL Injection

## Popište, co je SQL injection a jak mu zabránit.

## SQL injection je spuštění SQL dotazu obsahujícího neošetřený kód od klienta, který může vést ke kompromitaci databáze

* U dynamického SQL
* Útočník může vkládat SQL dotaz do formuláře nebo zadat do URL
* Příklad:
  + Dotaz = "SELECT \* FROM [user] WHERE login = '" + login + "' AND pass = '" + pass + "'"

Execute(Dotaz)

* + pokud uživatel (např. ve formuláři) vyplní do loginu např. řetězec

asd' OR 1=1; --

* + tak se vrátí první uživatel z databáze (a klient se na něho přihlásí), i když nezná jeho přihlašovací údaje.

**Pokud by se ptal jak tomu zabránit (nejsem si jistý, někdy se může zeptat jen co je SQL injection)**

1. Sanizace vstupních řetězců
2. Parametrizované dotazy (implicitní, pokud se používá statické SQL) - správné řešení, které nejprve na server odešle dotaz s "placeholdery" pro vstupní data, a až poté samotná data. Zajistí se tak, že server bere vstup uživatele opravdu jako data, a ne jako direktivy dotazu - zároveň se tak dá optimalizovat spouštění více stejných dotazů s různými daty, protože server už má předpočtený plán dotazu a pouze ho spouští znovu nad různými daty
3. Omezit přístupová práva do ostatních částí DB

## Co je to doménový objekt a DAO v ORM? (Uveďte příklad)

* Doménový objekt reprezentuje objekt z domény aplikace (např. Auto v IS autopůjčovny)
* DAO (data access object) poskytuje rozhraní k práci s DTO (data transfer object), které slouží jako přepravky dat a jsou mapovány na konkrétní záznam v tabulce. DTO představuje jeden záznam (jeho data)
* DAO představuje obvykle tabulku samotnou a její funkce (vyhledání konkrétního záznamu, smazání konkrétního záznamu, načtení všech záznamů)

**--**

**ORM** je programovací technika zpřístupňující relační (či objektově relační) data pro objektové prostředí (např. C#, Java)

**Vlastnosti:**

* Práce s objektovým modelem
* Rychlejší vytváření aplikací vs menší výkon aplikace
* Přenositelnost mezi různými SŘBD vs využívání specifických vlastnosti SŘBD (datové typy, funkce..)

**DTO – Data Transfer Object**

* Např. třída Person pro tabulku Person
* Jeden objekt prezentuje jeden záznam – řádek tabulky

**DAO – Data Access Object**

* Jedna třída prezentuje jednu tabulku a její operace
* Např. třída PersonTable se statickými metodami Select, Insert… pro tabulku Person
* Používáme parametrizované operace z důvodu zabránění SQL injection

**Entita** v ORM je objekt, který je uložen v databázi (jeden záznam)

**Vazba 1:1**

* Vztahy jsou reprezentovány referencemi, ne FK
* Public class Uzivatel {… private Ucet ucet; …}
* Public class Ucet { … }

**Vazba 1:N**

* Public class Uzivatel {… …}
* Public class Ucet { … private User user; …}
* Nebo
* Public class Uzivatel {… private List <Ucet> ucet = new ArrayList <Ucet>(); …}
* Public class Ucet { … }