**Többértékű attribútumok**

**kollekció atri JPAs lementéséhez: atrira @ElementCollection és egyszerű típusok, vagy ha osztálytípus, akkor @Embeddable az osztályra; gyermektábla külső kulccsal, gyermektábla konfigurálása: egyszerűnél @CollectionTable (és benne @JoinColumn), Column, osztály típusnál @CollectionTable és @AttributeOverride**

*eddig csak primitív vagy egy értéket tartalmazó atrikkal néztünk; mi van kollekció típusú atrikkal? hogyan lehet olyan entitást menteni, aminek pl. Lista atrija van?*

**Kollekcióba egyszerű típusok**(=primitívekhez tartozó csomagoló osztályok, String és dátum- idő objektumok), és az **@Embeddable** annotációval ellátott **osztály típusok** tárolhatóak akkor, ha azt akarjuk, hogy a **JPA automatikusan adatbázisba mentse**; magára a **kollekció atrira** pedig az **@ElementCollection** kell

**egy külön tábla kerül létrehozásra!**

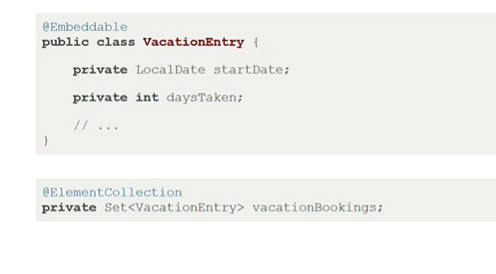
**Egyszerű típus**

egy külön tábla, egy **gyermektábla kerül létrehozásra külső kulccsal** és az adott alkalmazott beceneveivel:

****

**Példa embeddable**

létrejött gyermektábla: külső kulcs és startDate, daysTaken oszlopok

****

**Személyre szabás**

A gyermektábla konfigurálása

tábla nevének megadása; melyik oszlop legyen a külső kulcs hivatkozás, oszlop neve

@Embeddable-nél az @AttributeOverride-al adhatjuk meg a különböző oszlopok neveit



**Különbség @ElementCollection és @Entity között?**

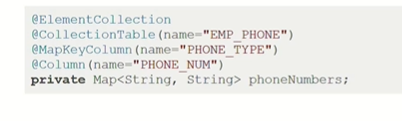
**With @ElementCollection, we can’t query, persist or merge target object independently of their parent object.**

**Map perzisztálása, lazy kapcsolat, lazy problémák és megoldás**

**(map atri automatikus JPAs lementéséhez: @MApKeyColum és @Column-al oszlopnevek testreszabása; lazy=csak szükség esetén tölti be az @ElementColleciton típusú atrit; betöltés selecttel vagy joinnal; tényleges szükség ( pl getter; default működés felülbírálása: @ElementCollection(FetchType=EAGER)-> nem jó gyakorlat; N+1 probléma: select a selectben defaultként a find()nál, helyette legyen külön metódus az atris lekérdezésre join fetchel; LazyInitializationException : csak Hibernatenél, lazy betöltésel nem hozza az atrikat, közben a metódus lezárja a pct-> ha ilyenkor hívok egy settert az atrira, hibát dob, mert nincs nyitott pc; ezt a problémát is megoldja a join fetch)**

**Map perzisztálása**

egyszerű és embeddable típusok tetszőleges kombinációja



**Lazy kapcsolat**

**ElementColleciton típusú atrik használatához kapcsolódik** a lazy betöltés fogalma

lazy: **csak szükség esetén tölti be az atri értéket**; a **gyermektáblát** vagy **külön selecttel** vagy **joinnal** lehet betölteni; ezt a JPA csak akkor csinálja meg, ha tényleg szükség van az atrik értékekre**, pl. ha rákérdezünk az értékre egy getterrel**

ez a **default működés**; ezt **felül lehet bírálni az @ElementCollection fetch attribútumával (=FetchType.EAGER);** ilyenkor az entitás betöltésekor az @Embeddable atrik is mindig betöltésre kerülnek; nem jó gyakorlat! statikus, minden esetre csinálni fogja, van helyette másik megoldás, hogy csak bizonyos esetekre

**Lazy betöltéssel kapcsolatos problémák**

**N+1 probléma**

**performancia probléma: a default lazy betöltés egy lekérdezés/select az összes az entitásra, majd egyenként az összesnél külön select az @Embeddable atrira** = 1 select + n darab select összesen; nem ok, mi a megoldás?

**nem a sima find()al,** egyszerű selectel kérünk le, hanem **hozzá joinoljuk az embeddable atrikat** is, és a JPA csak egy selectet ad ki n+1 helyett; ha **join fetcht** használunk

**LazyInitializationException**

**csak Hibernate esetén**

**entitás betöltésekor, ha az embeddable atrija lazy betöltésű**, ha **lezárjuk a persistence contextet (metódus végén),** utána az entitás még **hozzáférhető pl getterrel, de már csak detachedként; ha ezen az entitáson lekérjük az embeddable atri érétkét, akkor nem fogja betölteni és ezt az exceptiont kapok**; a háttérben nem töltődnek be az atri értékei, viszont getter setter esetén a hibernate megpróbálja; de mivel **nincs nyitott pc**, ez nem lehetséges, és **exception** lesz

megoldások: még nyitott pcnél kérdezünk rá az atrira (nem elegáns, plusz kód); helyette jobb gyakorlat **a selectben megmondani a kapcsoló atrik betöltését** **join fetch**-el

**cél: töltse be az @ElementCollectiont is, de ne legyen n+1 probléma (select a selectben, EAGER)= join fetch**

**Többértékű attribútumok - gyakorlat**

*Egy entitáshoz hogyan veszünk fel Collection típusú attribútumokat és hogyan kerülnek ezek letárolásra?*

1. *új Employee atri Set<String> nickNames; @ElementCollection annotációval+* getter;
2. *void testNickNames();*

new Employee, setNickNames(Set.of(\*\*\*)); save, futtatás -> mi történt, mik lettek kiírva a konzolra, és mik történtek az adatbázisban?

olvassuk ki: anotherEmployee = .findEmployeeById(e.getId()); sout nicknames > Hibernate exceptiont dob; mi történik a háttérben? metódus hívásakor kinyitom az entitymanagert, elindul egy pc, betöltöm az employeet, de ez kizárólag az employees táblából történik, nicknames táblához nem szagol hozzá; close után az employee példány detached státuszba került; a **getNicknamesnél próbál adatbázishoz kapcsolódni, de nem tud, mert nincsen nyitott persistence context -> megoldás, amig nyitott pc van adidg töltsük be az atri értékét!** close() elé sout getNicknames a findEmployeeById()ban -> a teszt is lefut mert ezzel fel is töltötte, a getter hatására(nem elegáns)

helyette: (mindkét problémát megoljda: n+-1et és az exceptiont is) magába a selectbe írjuk bele, hogy kérdezze le a beceneveket is, vagy az atrinál beállítjuk, **hogy eager módon kerüljön betöltésre: @ElementCollection(fetch = fetchType.EAGER)**

* futtatásra a Hibernate leftouter joint adott ki, egy selectben lekérdezte az embeddablet is; performanciában ez para lehet, nem javasolt, ne használjuk ezt a fetchType.EAGER-t; **helyette saját új metódus a Daoban!**
* **Employee findEmployeeByIdWithNickNames()** -> creatquery(select e form Employee e join fetch e.nicknames where e.id = :id”, Employee.class).setParamameter(„id” id).getSingleResult; teszt átírása erre a metódusra és futtatás, majd assertEquals nicnevekre

1. *generált gyermek tábla személyre szabása*

@CollectionTable(name = „nicknames”, joinColumns = @JoinColumn(name = „emp\_id”)); @Column(name = „nickname”)

1. *kollekció osztálytípusú elemekkel*

új atri: VacationEntry -> LocalDate és int days; getter setter, konstruktorok, osztályra @Embeddable

teszteset: void testVacations(); new Employee, setVacationBookings(két VacationEntry); majd save employee és futtatás; mi történt? mi jött létre?

visszaolvasás: anotherEmployee = .findEmployeeByIdWithVacations(employee.getId()); -> findEmployeeByIdWithVacations metódus implementálás: entitymanager, createqurey(selct e ftom Employee e **join fetch** e.vacationbookins where id = : id”, Employee.class).getsingleResult(); close(), return employee; majd assert vacationbookings set méretére

1. *tábla szofisztikálása*

@CollectionTable (bookings, emp\_id); @AttributeOverride(startDate -> start\_date, daystaken->days

1. *map alkalmazása*

új atri: Map<String, String> phonenumbers; @ElementCollection, @CollectionTable(name = „phone\_numbers”, joincolumns = @JoinColumn(name = „emp\_id”)); @MapKeyColumn(name =); &Column (name =), getterek setterek

void testPhonenumbers(), new Employee, sethonenumbers(home-1234 stb.), saveemployee, anotherEmployee= .fdnebyidwithphnumbers() -> metódus implementáció: em.createQuery(select e form Employee e join fetch e.phoneNumbers where d = :id”, .class).setparam().getSingle().close(); return emp; testbe assertEquals anothere.getphonenum.get(home)

**Kérdések**

*Mit jelentenek a többértékű attribútumok?*

*Milyen típusok lehetnek többértékű atrik?*

*Milyen annotációkkal látjuk el a többértékű atrikat ahhoz, hogy a JPA kezelje?*

*Mit csinál a többértékű atrikkal a JPA?*

*Mit nem lehet csinálni a többértékű atrikkal JPA-val, a kapcsolati atrikkal szemben?*

*Milyen táblák jönnek létre egyszerű és embeddable típusok esetén?*

*Milyen annotációkkal tudjuk személyre szabni a gyermektáblát?*

*Milyen annotációkat használunk map többértékű atrinál?*

*Mit jelent a lazy betöltés?*

*Hogy tudjuk felülbírálni a lazi beötltést?*

*Milyen problémát okoz a default lazy betöltés @Embeddable atriknál?*

*Hogy védjük ki ezt a problémát? (mit használunk sima find(id) helyett)?*

*Hibernate használata esetén mikor találkozhatunk LazyInitialExceptionnel, és mi az?*

*Hogy tudjuk ezt kivédeni?*