**Haladó adatbáziskezelő-rendszerek**

**In-memory db, Archiválás(trigger, tárolt eljárás)**

**Buzás Ádám-SMLZV2**

# Bevezetés:

A projekt során az volt a célom, hogy egy korábban készült projektemet továbbfejleszek az óránt tanult technológiákkal. Ez a projekt egy mysql, spring és angular stacket használ. Az alkalmazásom egy informatikai könyveket árusító webshop. Felhasználóknak lehetősége van a könyvek közötti keresésre, kategória szerinti szűrésre. Minden felhasználónak joga van az egyes könyvek részletes adatait megtekinteni. Lehetőségük van könyveket kosárba rakni, majd a kosár tartalmát megvizsgálni.

Az alkalmazás tartalmaz egy JWT tokenes autentikációt, mely segítségével a felhasználók tudnak regisztrálni, bejelentkezni, illetve a bejelentkezett felhasználók kijelentkezni. A felhasználóknak két szerepkört definiáltam: USER és ADMIN.

A USER szerepkörbe tartozó felhasználóknak joga van rendelés leadására. Az ADMIN jogosultsággal rendelkező felhasználó láthatja a felhasználók által leadott rendeléseket. A rendszerhez adhat új kategóriát, illetve új terméket. Ezeket a menüpontokat csak ADMIN jogosultsággal lehet elérni.

Új funkciók közé kitaláltam, hogy létrehozok egy lokális fejlesztői környezetet, ahol az eddig használt mysql helyett egy h2 in-memory adatbázist alkalmazok. És létrehozok egy ”dev” környezetet, ami a meglévő mysql adatbázisra fog csatlakozni. A következő fejlesztés, hogy a megrendeléseket lehetősége lesz az ADMIN felhasználónak archiválni. Ehhez angularban létrehozok egy új képernyőt, ahol a felhasználó nyomon tudja követni az archivált éveket és hogy az adott évben mennyi rendelés lett archiválva, illetve lehetősége lesz év kiválasztásával egy adott évet archiválni. Végezetül egy replikációt is megfogok valósítani.

# In-memory db:

Az in-memory adatbázisnak egy h2 adatbázist használtam. Ehhez a spring alkalmazásban kialakítottam egy új local és dev nevezetű profilt. Ez egy jó gyakorlat, mert a h2 lehetővé teszi, hogy az alkalmazás izoláltan és biztonságosan működjön, valós adatbázis terhelése nélkül. A h2 másik nagy előnye, hogy minden alkalmazásindításkor tiszta állapotról indul, így elkerülhetők a korábbi tesztelésekből visszamaradt, esetleg hibás vagy irreleváns adatok. Ez segít a fejlesztőknek abban, hogy mindig egy friss és konzisztens adatbázisállapottal dolgozzanak, ami stabilabb és kiszámíthatóbb fejlesztési élményt biztosít.

A local profil konfigurációja tartalmazza a h2 drivert, a h2 db url, felhasználónevet és a jelszót. A spring a data.sql fájlt minden indításkor lefuttatja köszönhetően a spring.jpa.defer-datasource-initialization=true beállításnak. Ez az sql fájl tartalmazza azokat az insert-ket, melyek az adatbázisba illesztik a konzistens adatokat. Az adatbázis sémája automatikusan jön létre egy JPA – Hibernate funkciónak köszönhetően. Ez a funkció egyszerűen a @Entity, @Table, @Column stb. annotációk alapján hozza létre az adatbázis sémát, minden indításkor.

A konfigurációk beállítását követően az alkalmazást indítva már a h2 adatbázis fut a backend alatt. Ez a console-ban is látható:

A screenshot of a computer

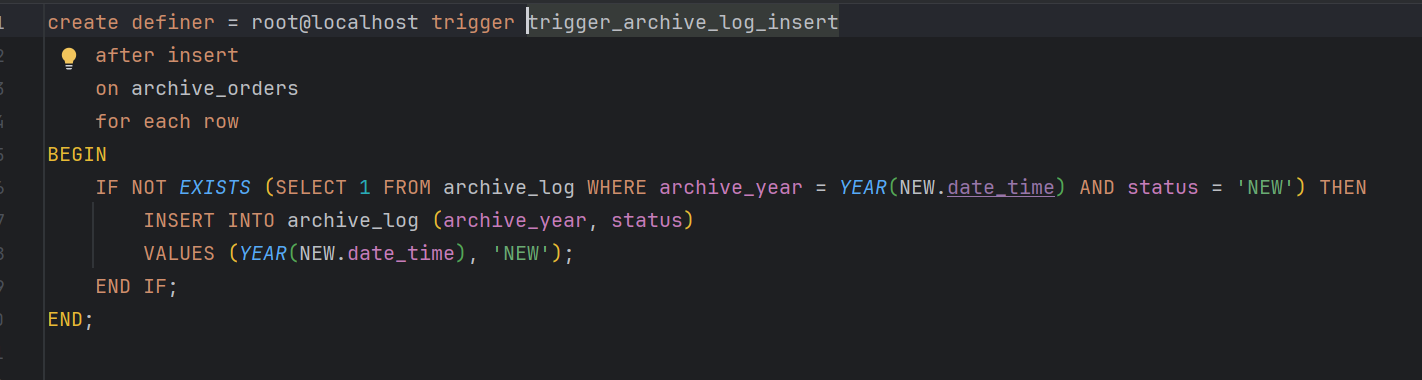
AI-generated content may be incorrect.

# Archiválás:

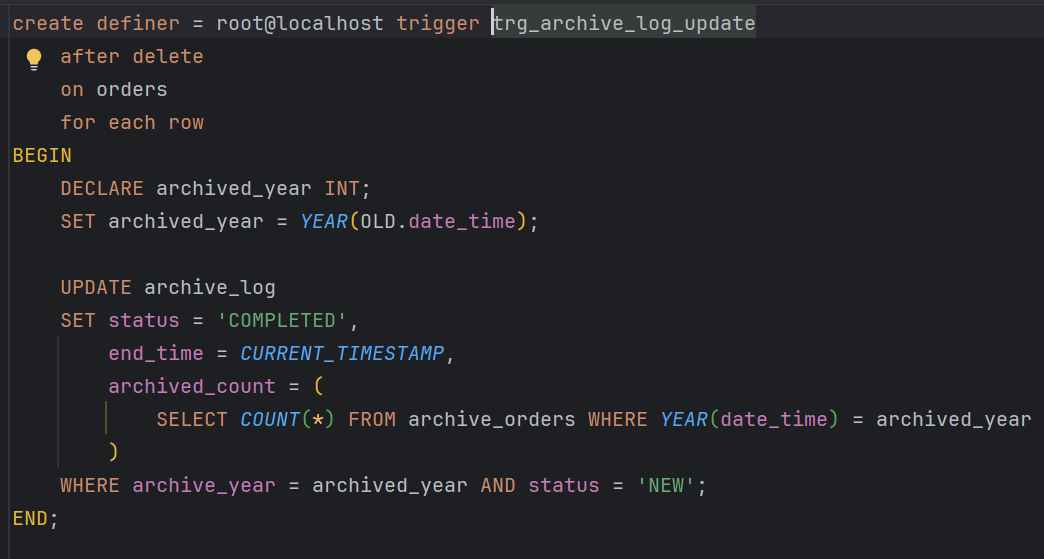
Az archiválás rendkívül elterjedt módszer nagyvállalati szoftvereknél. A célja, hogy azokat a táblákat, amelyekben nagyon sok rekord található, és amelyeket gyakran használnak (például listázásra vagy szűrésre), méretük csökkentésével optimalizáljuk. A méret csökkentést azzal érjük el, hogy ezeknek a tábláknak a régebbi rekordjait egy másik táblába rakjuk át, ez lesz az archív tábla. Ennek köszönhetően a tábla mérete jóval kisebb lesz, ezzel növelve a teljesítményt és a régebbi adatok is megmaradnak egy másik táblában. Ezekre a régebbi adatokra sokszor jogszabályi előírás van, hogy mennyi ideig kötelező őket eltárolni.

A projektemben az orders tábla archiválását valósítottam meg. Mivel ez nem egy éles rendszer, így nem lenne feltétlenül szükség az archiválásra, azonban ennek a technikának a segítségével jól lehet demonstrálni a kurzuson tanult eszközöket. Az archiváláshoz létrehoztam egy archive\_orders táblát, ami struktúrában megegyezik az orders-el. Az alkalmazásomban az orders táblához logikailag tartozik egy ordered\_book tábla is, így ehhez is készítettem egy archív táblát. Ezenfelül kászítettem egy archive\_log táblát, amibe logolni fogom, ha történt már archiválás.

Az archiválási logikát el lehetett volna végezni backend oldalon is kódban, viszont mivel a kurzus fókusza az adatbázis technikákon volt, így az archiválásért egy tárolt eljárás felelős. A tárolt eljárás egy év paramétert vár és az orders, ordered\_book táblából inserteli a megfelelő rekordokat archív táblába, majd ezeket törli is az eredeti táblából.

  
A screen shot of a computer program

AI-generated content may be incorrect.Az archive\_log tábla töltéséhez pedig készítettem kettő triggert. Az egyik az archive\_order táblába beszúrás pillanatában fut le. Ez a trigger felelős azért, hogy a log-ba beírja elkezdődött az archiválás. Illetve létrehoztam egy másik triggert, ami akkor fut le, ha az order táblából törlünk, ekkor befrissíti a log táblában lévő státuszt COMPLETED-re. A képen a két trigger látható:



A screenshot of a computer

AI-generated content may be incorrect.Az arhciváláshoz létrehoztam kettő endpointot. Az egyik meghívja a tárolt eljárást, míg a másik lekéri az archivált éveket. Ezekhez az endpoitnokhoz angularban készítettem két kis egyszerű képernyőt:

A screenshot of a computer

AI-generated content may be incorrect.Ennek a két képernyőnek hála az adminisztrátor egyszerűen tudja archiválni a rendeléseket és ennek az eredményéről a napló képernyőn is könnyen tud tájékozódni.

A projekt forráskódját az alábbi git repository tartalmazza: