Databázy používajú predovšetkým binárne formáty, pretože umožňujú efektívne ukladanie, vyhľadávanie a manipuláciu s údajmi. Hoci existujú textové databázy, v porovnaní s binárnymi databázami majú obmedzenia z hľadiska výkonu, typov údajov, indexovania a škálovateľnosti. Textové databázy sú jednoduchšie, prenosné a dobre spolupracujú so systémami riadenia verzií, ale sú pomalšie, majú obmedzené typy údajov, chýba im indexovanie a majú problémy s veľkými objemami údajov.

V analytickej časti som sa venoval oblastiam, ako je parsovanie SQL príkazov, spracovanie údajov, ľudsky čitateľné formáty, a transakcie v rôznych databázových systémoch.

Výsledkom mojej práce je ovládač JDBC, ktorý parsuje SQL príkazy pomocou bezkontextovej gramatiky, spracováva údaje v pamäti a podporuje dva najpoužívanejšie formáty JSON a XML, do ktorých perzistuje údaje. Ovládač používa na overovanie údajov schémy JSON a XSD. Aby bolo jasné, v našom prípade je databáza priečinok, v ktorom sú tabuľky uložené v súboroch XML alebo JSON. Metadáta o tabuľkách sú uložené v schémach. Ako nástroj na verzovanie databázy sme implementovali GIT, ktorý možno voliteľne použiť v ovládači. Viac o tom budem hovoriť neskôr.

Začal by som parsovaním SQL príkazov, ktoré sme spracovali pomocou nástroja ANTLR, ktorý dokáže generovať parser danej gramatiky. Ovládač používa odvodenú syntax SQLite, inými slovami, implementovali sme menšiu množinu gramatiky SQLite. Na obrázku je znázornené, ako ANTLR vygeneroval strom parsovania z príkazu SQL. Zo stromu extrahujeme tieto údaje, vytvoríme z nich objekt prenosu údajov a odošleme ho na spracovanie.

Ovládač využíva princíp databáza v pamäti, čo znamená, že údaje budeme spracovávať v pamäti, kým sa nezapíšu na disk. Tento princíp spracovania údajov nám umožnil rýchle spracovanie údajov pre stredne veľkú databázu. Tabuľky sa načítavajú do pamäte len v prípade potreby, takzvaným "lenivým" spôsobom. Je dôležité spomenúť, že tento spôsob spracovania údajov nie je optimálny, ak máme veľmi veľkú databázu, nad ktorou treba iterovať, pretože to znamená, že do pamäte treba načítať celú databázu. Na zlepšenie tohto problému by sa v budúcnosti mohol implementovať mechanizmus stránkovania.

Náš prístup k ukladaniu údajov spočíva v ukladaní údajov do pamäte, kým sa neodovzdajú. To nám umožňuje uchovávať nezapisované údaje v objektoch a zapísať ich všetky naraz, keď sa zavolá metóda commit. V prípade, že je potrebné použiť metódu rollback, údaje sa môžu jednoducho znovu načítať z perzistovaných súborov do pamäte. Existuje však riziko poškodenia údajov, preto sme sa rozhodli implementovať git ako nástroj na verzovanie databázy, aby sme mohli databázu vrátiť do platného bodu. Rozhodli sme sa pre úroveň izolácie transakcií "READ COMMITTED", aby sme zabezpečili konzistenciu medzi súbežnými operáciami. Toto nemenné 30 nastavenie zabezpečuje, že všetky údaje načítané zo systému už boli zapísané do súborov, čím predchádza potenciálnym nekonzistentnostiam spôsobeným súbežnými transakciami. Aby sme ešte viac posilnili konzistenciu nášho ovládača a zabránili konfliktom z viacvláknových transakcií, rozhodli sme sa implementovať pesimistické zamykanie ako náš preferovaný mechanizmus zamykania. Ten zahŕňa získanie zámkov na údaje pred povolením prístupu, čím sa zabezpečí, že údaje môže v danom čase meniť len jedno vlákno.

Na overenie funkčnosti ovládača sme použili niekoľko metód. Po prvé, pre každú zložku nášho ovládača sme vytvorili jednotkové testy, ktoré testovali správnosť danej časti kódu. Testy boli napísané v testovacom rámci JUnit5 v kombinácii s Mockito. V jednotkových testoch sme dosiahli pokrytie kódu na úrovni 85 percent pre ovládač a 65 percent pre analyzátor. Potom sme náš ovládač porovnali z hľadiska výkonu s ovládačom SQLite, pričom sme merali celkový čas behu operácií CRUD, v ktorom sa nám v niektorých prípadoch podarilo prekonať SQLite. Tieto testy nám tiež ukázali, že GIT nám síce prináša tieto výhody, ale má veľký negatívny výkonnostný odpor. Preto sme sa rozhodli umožniť výber medzi predvolenou možnosťou a gitom pomocou konfigurácie.   
Ako integračný test sme vytvorili jednoduchú aplikáciu, do ktorej sme importovali náš ovládač, ktorý by som teraz rád ukázal.

V aplikácií je možné vykonávať jednoduché CRUD operácie, respektíve vytvoriť si novú úlohu, označiť ju ako hotové alebo ju vymazať.