Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна робота № 4

з курсу:

Організація баз даних та знань

Виконав:
студент групи КН-207
Гірняк Т. О.

Прийняла:
Мельникова Н.І.

Львів – 2019 р.

Тема:

Запити на додавання, зміну та вилучення даних

Мета:

Розробити SQL-запити для внесення нових значень в таблиці в режимі одиничного та групового доповнення; розробити SQL-запити для внесення змін в рядки таблиць; розробити SQL-запити для вилучення вибраних рядків.

Хід виконання:

Створення DAO файлів створених раніше таблиць для виконання команд над БД:

DAO для таблиці Workout:

```
@Dao
interface WorkoutDAO {

    @get:Query("Select * from WorkoutTable")
    val allWorkoutsMaybe: Maybe<List<WorkoutTable>>

    @get:Query("Select * from WorkoutTable")
    val allWorkouts: List<WorkoutTable>

    @get:Query("SELECT * FROM WorkoutTable ORDER BY workoutId DESC LIMIT 1;")
    val lastItemId: Int

    @Query("SELECT * FROM WorkoutTable WHERE workoutId = :workoutId")
    fun getLiveWorkoutById(workoutId: Int): LiveData<WorkoutTable>

    @Query("SELECT * FROM WorkoutTable WHERE workoutId = :workoutId")
    fun getWorkoutByIdMaybe(workoutId: Int): Maybe<WorkoutTable>

    @Query("SELECT * FROM WorkoutTable WHERE workoutId = :workoutId")
    fun getWorkoutById(workoutId: Int): WorkoutTable

    @Query("Select * From WorkoutTable Where startTime = :date")
    fun getWorkoutByStartTime(date: Long?):  Maybe<WorkoutTable>

    @Query("Select * From WorkoutTable Where dateOfCreation = :date")
    fun getWorkoutSingleByDateOfCreationTime(date: String): Single<WorkoutTable>

    @Query("Select * From WorkoutTable Where isCompleted = :isCompleted")
    fun getAllCompletedWorkouts(isCompleted: Boolean):  Maybe<WorkoutTable>?

    @Insert(onConflict = REPLACE)
    fun insertWorkout(workout: WorkoutTable)

    @Query("DELETE FROM WorkoutTable")
    fun nukeTable()

    @Query("DELETE FROM WorkoutTable WHERE workoutId = :workoutId")
    fun deleteWorkoutById(workoutId: Int)
```

```
    @Delete
    fun deleteWorkoutItem(workout: WorkoutTable)

    @Update
    fun updateData(workout: WorkoutTable)

    @Query("UPDATE WorkoutTable SET workoutVolume = :volume WHERE workoutId = :workoutId")
    fun updateWorkoutVolume(volume: Double, workoutId: Int)

    @Query("UPDATE WorkoutTable SET isCompleted = :isCompleted WHERE workoutId = :workoutId")
    fun updateWorkoutCompleteness(isCompleted: Boolean, workoutId: Int)

    @Insert
    fun insertAll(vararg dataEntities: WorkoutTable)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(workouts: List<WorkoutTable>)
}
```

DAO для таблиці Exercise:

```
@Dao
interface ExercisesDAO {

    @get:Query("SELECT * FROM ExercisesTable ORDER BY itemId DESC LIMIT 1;")
    val lastItemId: Int

    @Insert
    fun insertAll(vararg dataEntities: ExercisesTable)

    @Query("SELECT * FROM ExercisesTable WHERE itemId = :itemId")
    fun getLiveItemById(itemId: Int): LiveData<ExercisesTable>

    @Query("SELECT * FROM ExercisesTable WHERE itemId = :itemId")
    fun getExerciseById(itemId: Int): Maybe<ExercisesTable>

    @Query("Select exerciseName From ExercisesTable Where itemId = :itemId")
    fun getExerciseNameById(itemId: Int): Maybe<String>

    @Query("Select * from ExercisesTable Where exerciseName = :exerciseName")
    fun getExerciseByNameMaybe(exerciseName: String): Maybe<ExercisesTable>

    @Query("Select * from ExercisesTable Where exerciseName = :exerciseName")
    fun getExerciseByName(exerciseName: String): ExercisesTable

    @Query("Select * from ExercisesTable")
    fun readExercises(): List<ExercisesTable>

    @Query("SELECT COUNT(itemId) FROM ExercisesTable")
    fun getCount(): Maybe<Int>

    @Query("Select exerciseName from ExercisesTable")
    fun getAllExerciseNames(): Maybe<List<String>>

    @Query("Select * from ExercisesTable")
    fun readExercisesMaybe(): Maybe<List<ExercisesTable>>

    @Insert(onConflict = REPLACE)
    fun insertItem(exercise: ExercisesTable): Long?
```

```
    @Query("DELETE FROM ExercisesTable")
    fun nukeTable()

    @Query("DELETE FROM ExercisesTable WHERE itemId = :itemId")
    fun deleteItemById(itemId: Int)

    @Delete
    fun deleteListItem(exercise: ExercisesTable)

    @Update
    fun updateData(exercise: ExercisesTable)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(exercisesTables: List<ExercisesTable>)

    @Query("Select * From ExercisesTable Where muscleGroup = :muscleGroup")
    fun getExercisesByMuscleGroup(muscleGroup: String): Maybe<List<ExercisesTable>>
}
```

Виконання команд DAO для занесення даних в БД, оновлення та видалення:

1) Отримання усіх врав з таблиці Exercise з використанням реактивного програмування (RxJava):

```
private fun refreshSearchAdapter(database: GymDiaryDatabase, searchAdapter:
SearchExerciseAdapter){
    database.exercisesDAO().readExercisesMaybe().subscribeOn(Schedulers.io())
            ?.observeOn(AndroidSchedulers.mainThread())?.subscribe { exercises ->
                run {
                    searchAdapter.refresh(ArrayList(exercises))
                }
            }
}
```

2) Оновлення даних в таблиці Workout:

```
@Query("UPDATE WorkoutTable SET workoutVolume = :volume WHERE workoutId = :workoutId")
fun updateWorkoutVolume(volume: Double, workoutId: Int)


database.workoutDAO().updateWorkoutVolume(totalVolume, currentWorkout.workoutId)
```

3) Видалення всіх даних з таблиці:

```
@Query("DELETE FROM WorkoutTable")
fun nukeTable()
```

4) Видалення поля з таблиці Workout за параметром id:

```
@Query("DELETE FROM WorkoutTable WHERE workoutId = :workoutId")
fun deleteWorkoutById(workoutId: Int)
```

Висновок:

На даній лабораторній роботі я виконав команди оновлення, видалення та отримання даних з БД використовуючи SQLite з мовою Kotlin.