




Room Migrations

I need to add a new column to my existing table.

I have users using the app. After this change, they should be able to update the app without losing data. Database migration or die, what do you choose?

A scene from the animated movie 'Madagascar'. On the left, a large, purple, tentacle-like alien with a wide, toothy grin looms over a group of four penguins. The penguins are standing in a line, looking up at the alien with expressions of concern. The background shows a green wall with some control panels and a wooden floor.

Kowalsky,
analysis!?

Database migration
is the better
option, Sir.




The app is using
Room 2.5.1. We
have a plan.

First, we need to
define our
migration...

```
val MIGRATION_1_2 = object : Migration(1, 2) {  
    override fun migrate(database: SupportSQLiteDatabase) {  
        val sql = "ALTER TABLE `favorite_movies` ADD COLUMN `movieId` INTEGER NOT NULL DEFAULT 0"  
        database.execSQL(sql)  
    }  
}
```


Table name

Column
name



It is very important
we define a default
value to existing
entries

```
val MIGRATION_1_2 = object : Migration(1, 2) {  
    override fun migrate(database: SupportSQLiteDatabase) {  
        val sql = "ALTER TABLE `favorite_movies` ADD COLUMN `movieId` INTEGER NOT NULL DEFAULT 0"  
        database.execSQL(sql)  
    }  
}
```



We already define
the start version
and the end version...

```
val MIGRATION_1_2 = object : Migration(1, 2) {  
    override fun migrate(database: SupportSQLiteDatabase) {  
        val sql = "ALTER TABLE `favorite_movies` ADD COLUMN `movieId` INTEGER NOT NULL DEFAULT 0"  
        database.execSQL(sql)  
    }  
}
```

Great, let's do it!



Aye sir





```
Room.databaseBuilder(  
    androidContext(),  
    DataBaseFactory::class.java,  
    name: "MovieDBAppDataBase"  
) RoomDatabase.Builder<DataBaseFactory>  
    .addMigrations(MIGRATION_1_2)  
    .fallbackToDestructiveMigration()  
    .build() DataBaseFactory  
    .favoriteMoviesDAO()
```

```
Room.databaseBuilder(  
    androidContext(),  
    DataBaseFactory::class.java,  
    name: "MovieDBAppDataBase"  
) RoomDatabase.Builder<DataBaseFactory>  
    .addMigrations(MIGRATION_1_2)  
    .fallbackToDestructiveMigration()  
    .build() DataBaseFactory  
    .favoriteMoviesDAO()
```

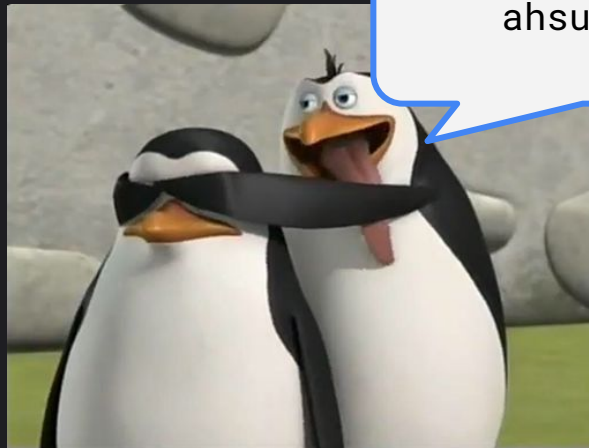
Database version
updated!



```
@Database(entities = [FavoriteMovieDTO::class], version = 2, exportSchema = false)  
abstract class DataBaseFactory : RoomDatabase() {  
  
    ▲ Gabriel Moro  
    abstract fun favoriteMoviesDAO(): FavoriteMoviesDAO  
  
}
```



```
@Entity(tableName = "favorite_movies")
data class FavoriteMovieDTO(
    @PrimaryKey(autoGenerate = true)
    val id: Int? = null,
    val votesAverage: Float,
    val title: String,
    val posterImageUrl: String?,
    val backdropImageUrl: String?,
    val overview: String,
    val releaseDate: String,
    val language: String,
    val popularity: Float,
    val movieId: Long,
) {
```



ahsusqegvqh

Great Rico, so now
we have the entity
updated too!

