Kotlin

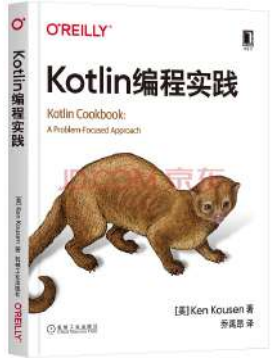
使用 DSL + KSP 打造跨平台的 Kotlin SQLite 框架

乔禹昂

qiaoyuang2012@gmail.com

November 26, 2022

- 携程机票资深移动开发工程师
- KUG Shanghai Organizer
- Kotlin Cookbook (O’Reilly, Kotlin 编程实践) 译者
- Podcast：《Kotlin 炉边漫谈》创办人之一



Kotlin Multiplatform 近況如何？

- KMM 1.7.20 起进入 beta 阶段
- Kotlin/Native new memory management 1.7.20 起默认启用
- Android Jetpack 部分 library 开始向 Kotlin Multiplatform 迁移

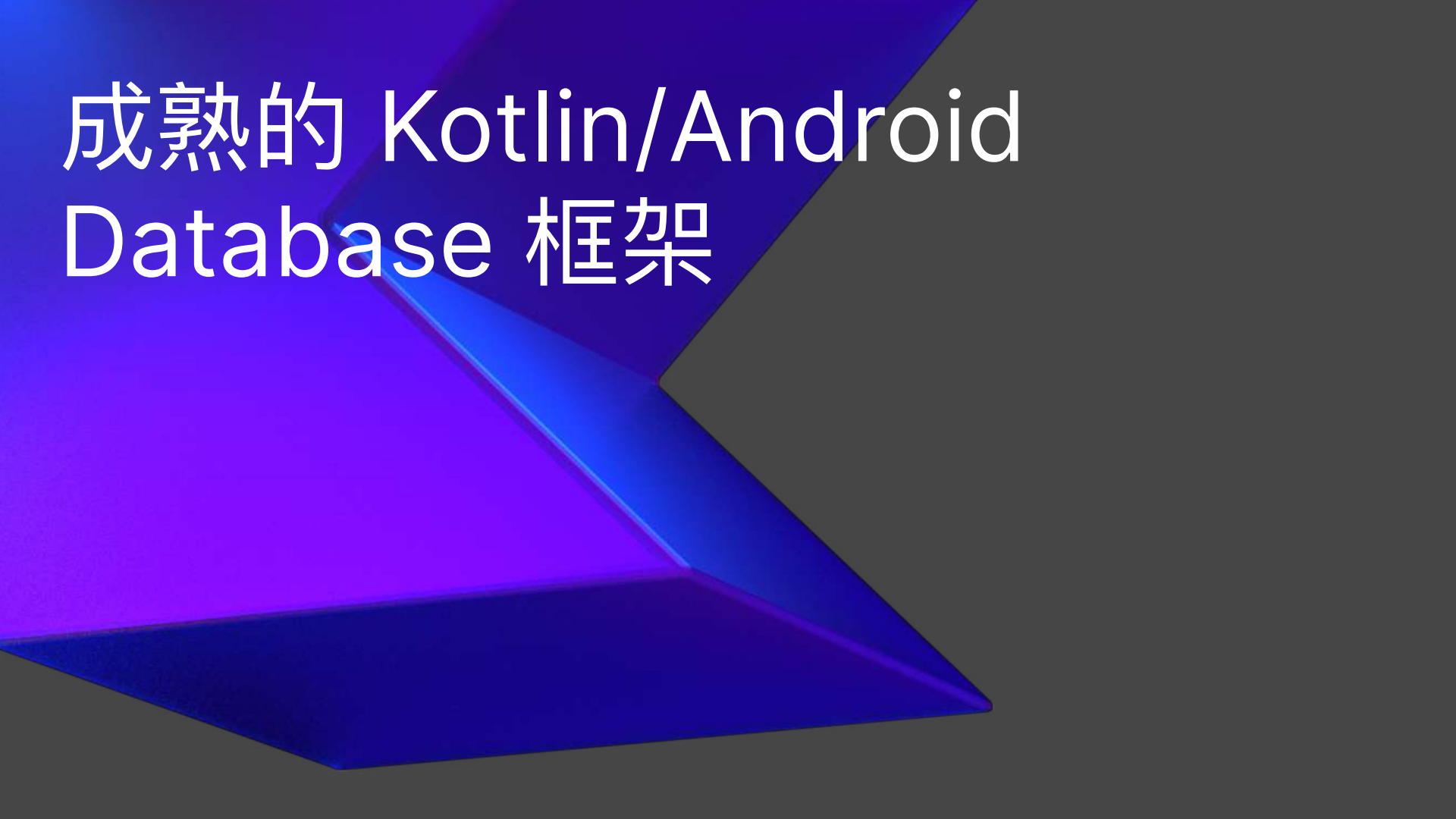
愈发丰富的生态

图片来源：https://kotlinlang.org/lp/mobile/static/multiplatform-mobile-app-wizard-63ba67b440498ffb774bdb0e112b1ce6.svg

为什么需要 SQLite 框架？

- SQLite 不支持 Kotlin Multiplatform 开发

- 直接使用 SQLite 的原始 API 容易出错
- SQL 语句以 String 的形式存在于代码中，不受 compiler 检查
- SQLite 不支持直接存取 object


成熟的 Kotlin/Android
Database 框架

Jetpack Room

- Database: SQLite
- Platform: Android
- API Style: Annotation + DAO

```
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    fun loadAllByIds(userIds: IntArray): List<User>

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    fun findByName(first: String, last: String): User

    @Insert
    fun insertAll(vararg users: User)

    @Delete
    fun delete(user: User)
}
```

Exposed

- Database: Postgres, MySQL, MariaDB, SQLite, H2, Oracle, SQL Server
- Platform: Kotlin/JVM
- API Style: DSL (key-value)

```
Users.insert {
    it[id] = "smth"
    it[name] = "Something"
    it[Users.cityId] = null
}

Users.update({ Users.id eq "alex"}) {
    it[name] = "Alexey"
}

Users.deleteWhere{ Users.name like "%thing"}
```

Ktorm

Ktorm
KOTLIN ORM LIB WITH SQL DSL

- Database: JDBC
- Platform: Kotlin/JVM
- API Style: DSL (chain call)

```
val query = database
    .from(Employees)
    .select(Employees.name)
    .whereWithConditions {
        if (someCondition) {
            it += Employees.managerId.isNull()
        }
        if (otherCondition) {
            it += Employees.departmentId eq 1
        }
    }
```

SQLDelight

- Database: SQLite, MySQL(JVM), PostgreSQL(JVM), HSQL/H2(JVM)
- Platform: Kotlin Multiplatform(Android, iOS, macOS, Windows, JVM, JavaScript)
- API Style: DSL (Code generation)

.sq

```
playerForNumber:
SELECT *
FROM hockey_player
WHERE number = ?;
```

.kt

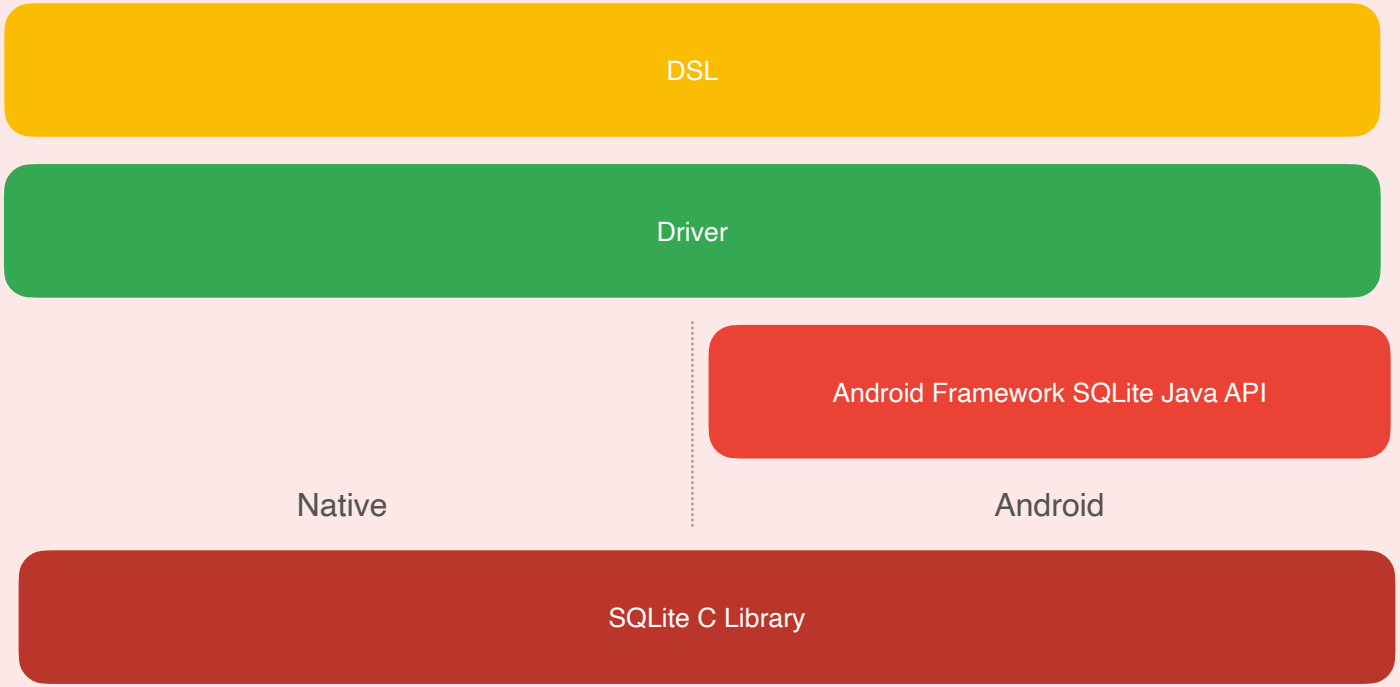
```
package com.example

import com.invest.core.PlayerQueries

class Players(val queries: PlayerQueries) {
  fun favouritePlayer(): String {
    return queries.playerForNumber( number: 10).executeAsOne().name
  }
}
```

需求确定

- 支持 KMM (Android、iOS)
- SQL 支持被 compiler 校验且 API 现代化
- 支持自动 serialization 与 deserialization
- Size 不能过大

设计与实现

初步设计


DSL
Driver
Android Framework SQLite Java API
Native
Android
SQLite C Library

Calling SQLite’s C functions in Kotlin

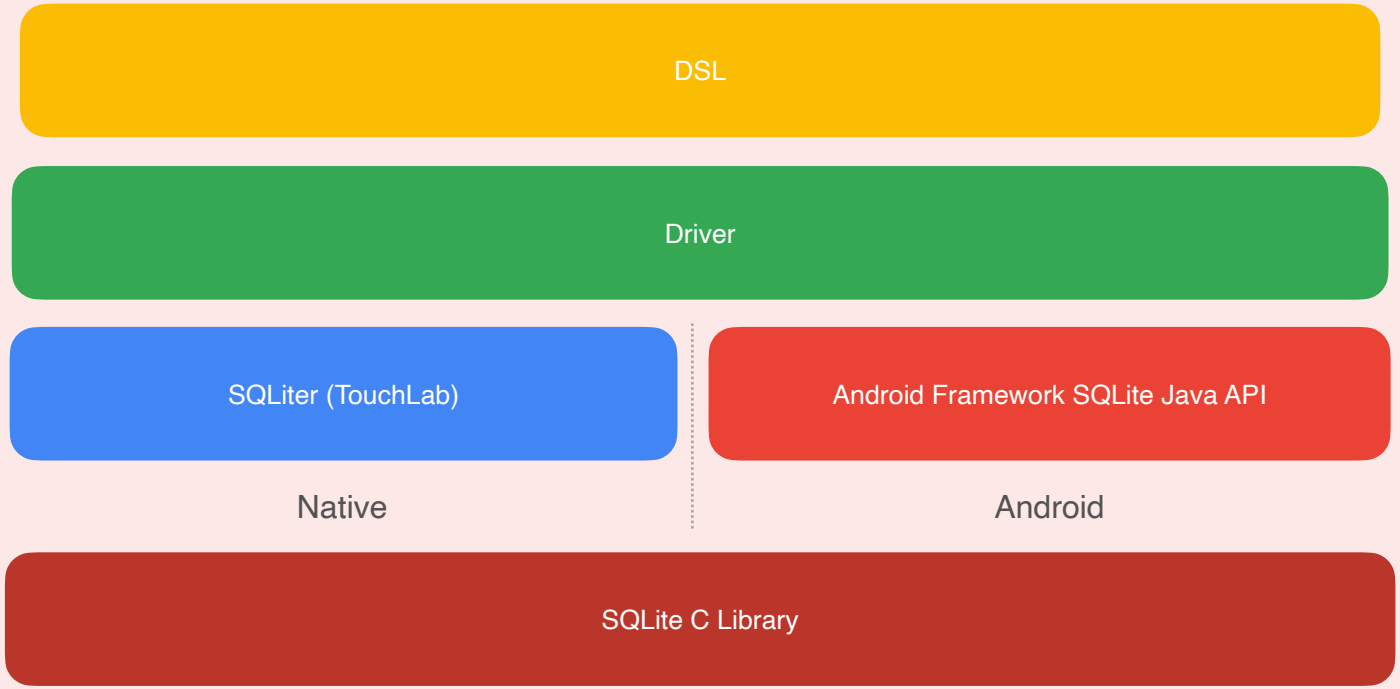
```
fun openDatabase() = memScoped {
    val dbPtr = alloc<CPointerVar<sqlite3>>()
    val openResult = sqlite3_open_v2(path, dbPtr.ptr, sqliteFlags, null)
    if (openResult != SQLITE_OK) {
        throw IllegalStateExcepetion(sqlite3_errmsg(dbPtr.value)?.toKString() ?: “”)
    }
}
```

SQLiter(TouchLab)

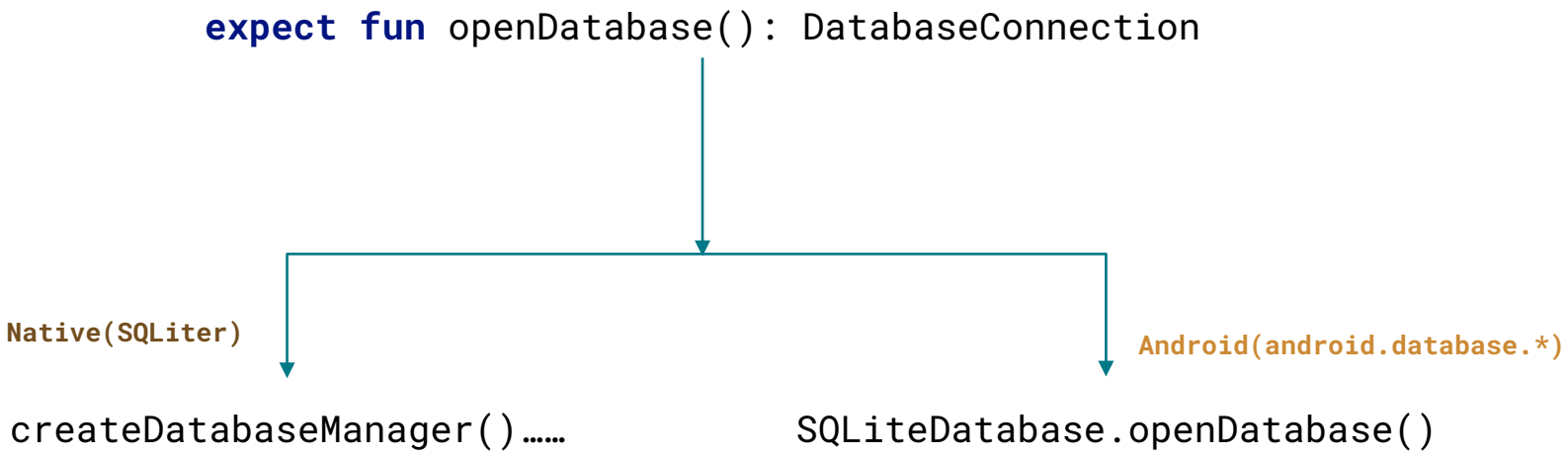
```
fun openDatabase() {
    val config = DatabaseConfiguration(name = “Person.db”, version = “1”……)
    val manager = createDatabaseManager(config)
    //……
}
```

支持平台：iOS、macOS、tvOS、watchOS、Linux、Windows

架构设计


DSL
Driver
SQLiter (TouchLab)
Android Framework SQLite Java API
Native
Android
SQLite C Library

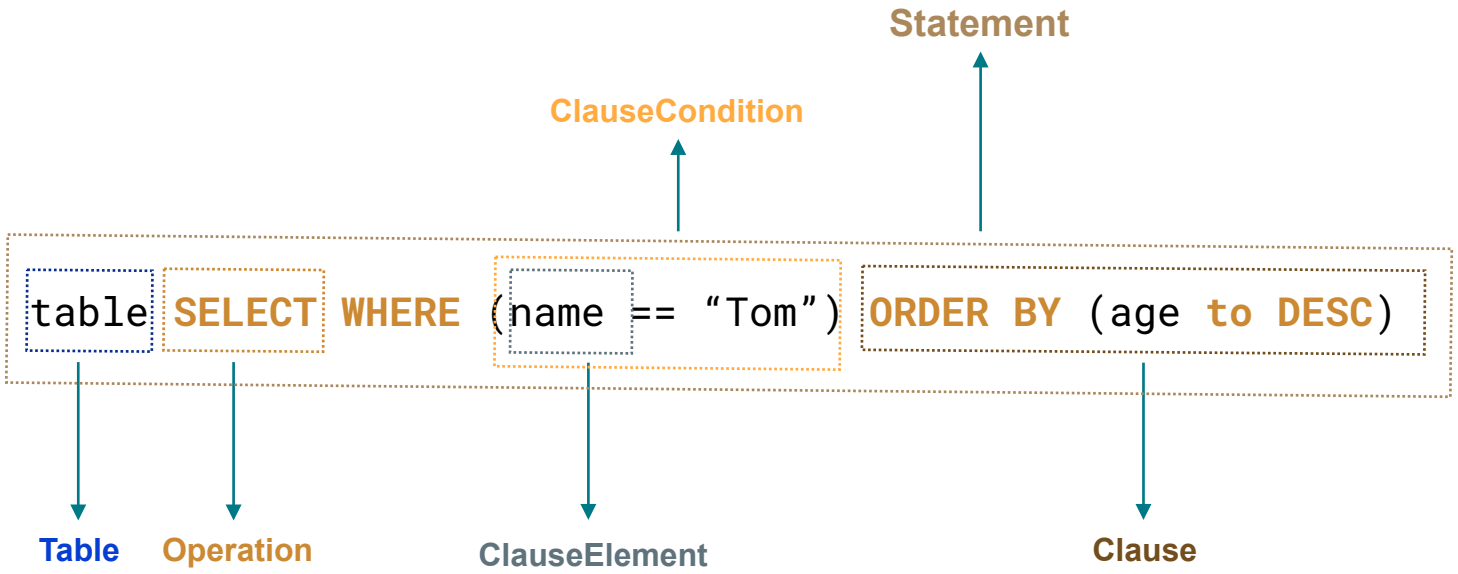
Driver


expect fun openDatabase(): DatabaseConnection
Native(SQLiter)
Android(android.database.*)
createDatabaseManager()……
SQLiteDatabase.openDatabase()

DSL 的 API 长什么样？

```
fun sample() {
    lateinit var statement: SelectStatement<Person>
    database {
        val table = Table<Person>(“person”)
        table INSERT listOf(tom, jerry, nick)
        table DELETE WHERE (name == “Jerry”)
        table UPDATE SET (age = 27) WHERE (name == “Nick”)
        statement = table SELECT WHERE (name == “Tom”) ORDER BY (age to DESC)
    }
    val result: List<Person> = statement.getResult()
}
```

DSL 类型设计


Statement
ClauseCondition
table SELECT WHERE (name == “Tom”) ORDER BY (age to DESC)
Table
Operation
ClauseElement
Clause

DSL 类型关系

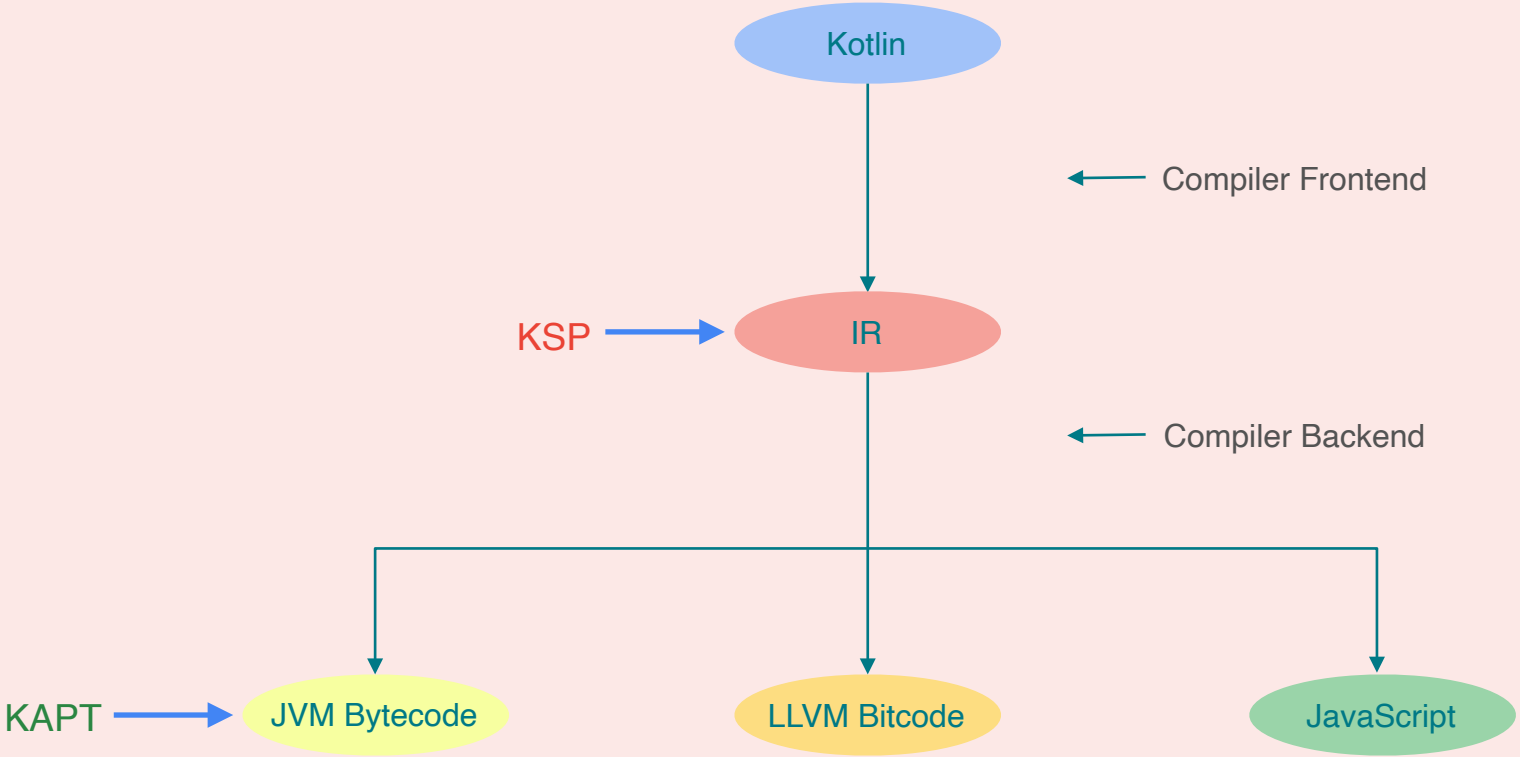
Table + Operation + Clause = Statement

Statement + Clause = Statement

ClauseElement + String/Number = ClauseCondition

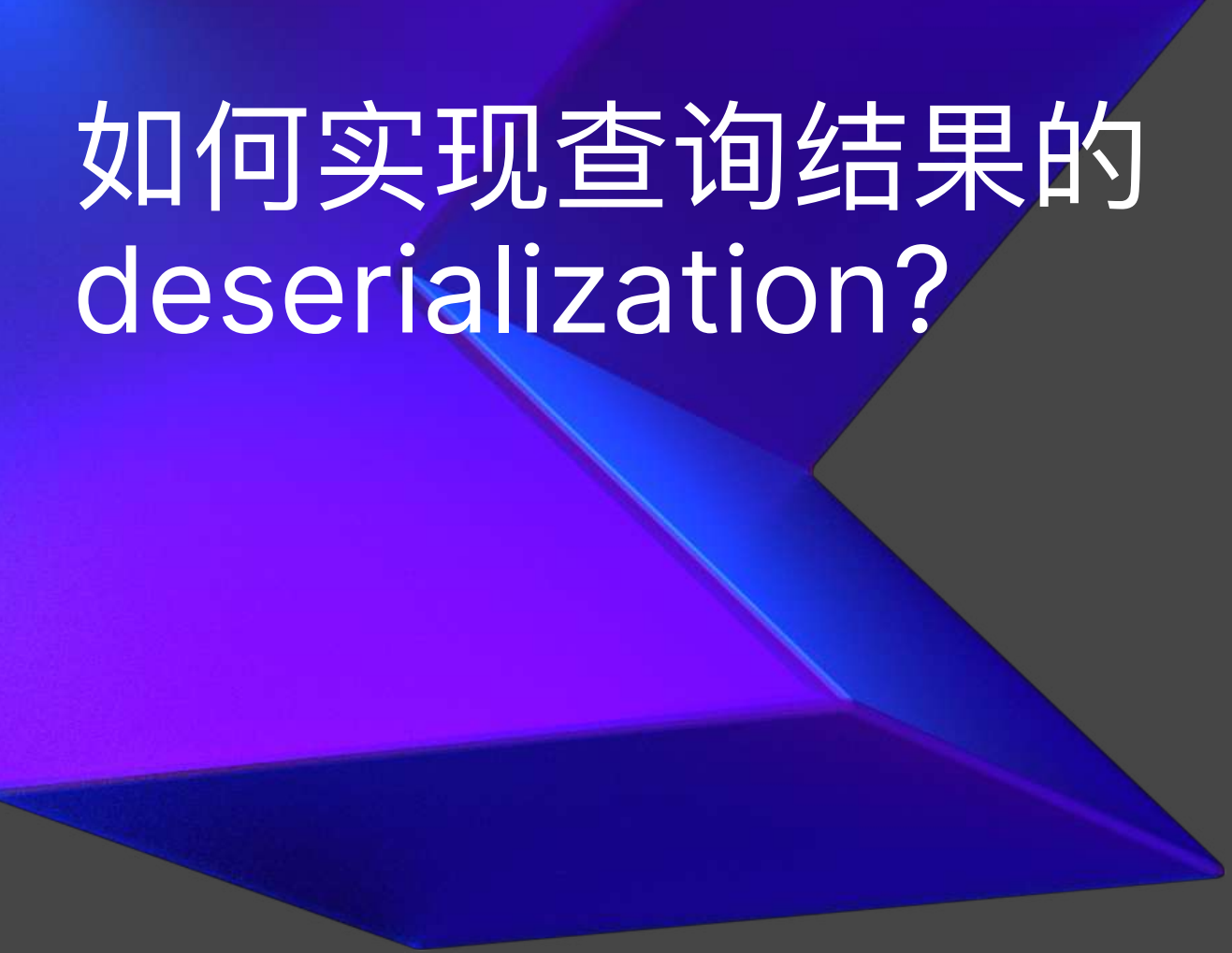
ClauseCondition + ClauseCondition = ClauseCondition

Kotlin Symbol Processor



KSP 如何处理 class？

```
@DBRow(“person”)
data class Person(
    val age: Int,
    val name: String,
)

// KSP generated:
object PersonTable : Table<Person>(“person”) {
    val name: ClauseString get() {…}
    val age: ClauseNumber get() {…}
    var SetClause<Person>.name: String set(value) {…}
    var SetClause<Person>.age: Int set(value) {…}
}
```


如何实现查询结果的
deserialization?

Driver 的查询结果 Cursor

```
interface CommonCursor {
    fun getInt(columnIndex: Int): Int
    fun getString(columnIndex: Int): String?
    //……
    fun getColumnIndex(columnName: String): Int
    fun forEachRow(block: (Int) -> Unit)
    fun close()
}
```

使用 kotlinx.serialization 的自定义能力

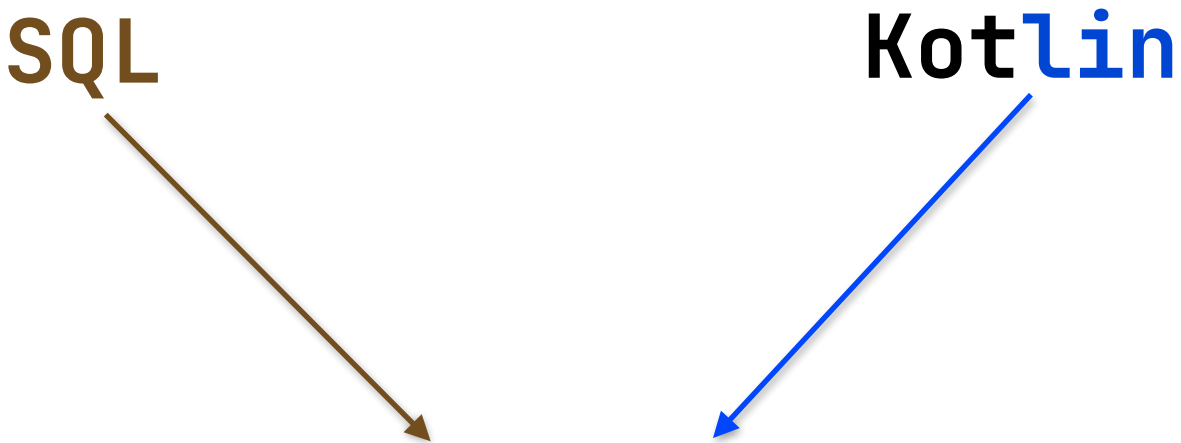
- 导入 kotlinx.serialization
- 给需要反序列化的 class 添加 @Serializable
- 自定义 Decoder

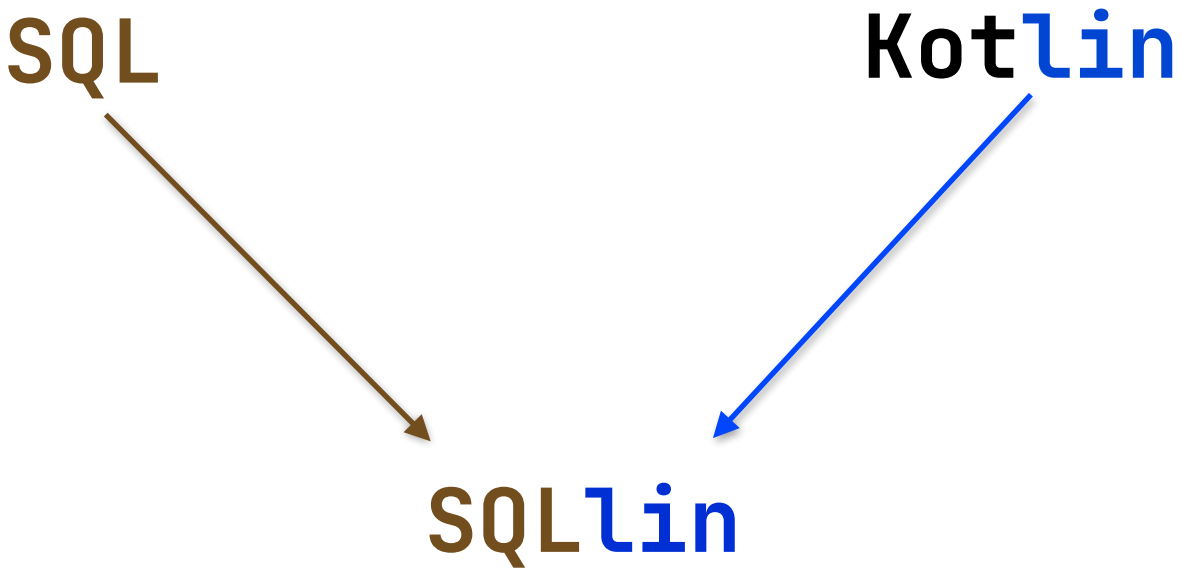
```
fun <T> getResult(
    cursor: CommonCursor,
    deserializer: DeserializerStrategy<T>): List<T> = buildList {
    val decoder = QueryDecoder(cursor)
    cursor.forEachRow {
        add(decoder.decodeSerializableValue(deserializer))
    }
}
```

最后，起个名字吧~

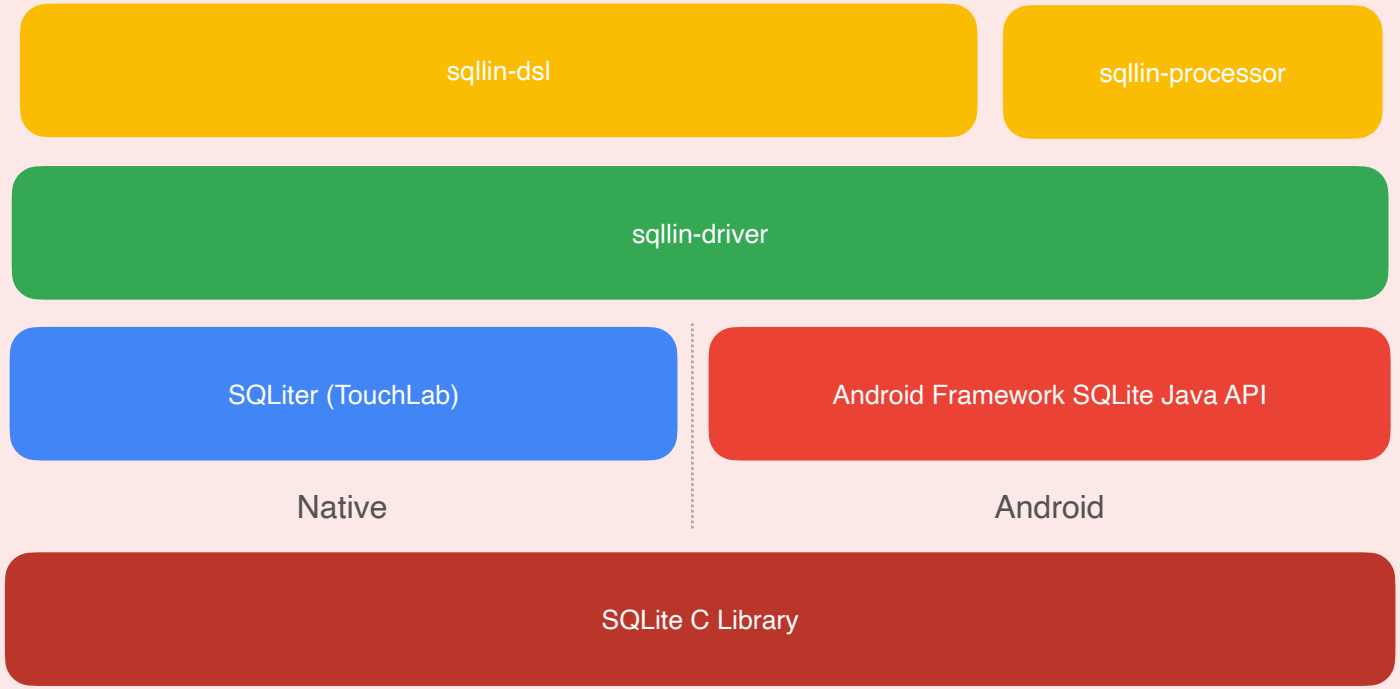
SQL

Kotlin


SQL
Kotlin


SQL
Kotlin
SQLlin

最终设计


sqllin-dsl
sqllin-processor
sqllin-driver
SQLiter (TouchLab)
Android Framework SQLite Java API
Native
Android
SQLite C Library

支持平台

- Android
- iOS (x64, arm32, arm64, simulatorArm64)
- macOS (x64, arm64)
- watchOS (x86, x64, arm32, arm64, simulatorArm64)
- tvOS (x64, arm64, simulatorArm64)
- Linux (x64)
- Windows (mingwX86, mingwX64) // Maybe in the future

最终效果

```
fun sample() {
    lateinit var statement: SelectStatement<Person>
    database {
        PersonTable { table ->
            table INSERT listOf(tom, jerry, nick)
            table DELETE WHERE (name EQ “Jerry”)
            table UPDATE SET {age = 27} WHERE (name NEQ “Nick”)
            statement = table SELECT WHERE (name EQ “Tom”) ORDER_BY (age to DESC)
        }
    }
    val result: List<Person> = statement.getResult()
}
```

SQLlin: https://github.com/ctripcorp/SQLlin

Thanks!
Have a nice Kotlin

qiaoyuang2012@gmail.com

微信群：国内 KMM 技术交流

微信群：上海 Kotlin 用户组

公众号：Kotlin 维修车间

稀土掘金：Kotlin 上海用户组

Podcast：Kotlin 炉边漫谈