什麼是 SQLite

SQLite 是一個輕量級、嵌入式關係型資料庫管理系統（RDBMS），廣泛應用於行動應用程式、桌面軟體和小規模專案。以下是對其功能、特性及限制的詳細說明，以條列式呈現，方便理解。

SQLite 的功能

SQLite 提供基本的關係型資料庫功能，適合簡單且高效的數據管理需求：

1. **標準 SQL 支援**
   - 支援大部分 SQL-92 標準，如 `SELECT`、`INSERT`、`UPDATE`、`DELETE`、`JOIN` 等。
   - 提供創建表格、索引、視圖和觸發器的能力。
2. **嵌入式儲存**
   - 不需要獨立的伺服器進程，直接嵌入應用程式中。
   - 數據儲存在單一檔案中（通常為 `.db` 或 `.sqlite` 格式）。
3. **事務處理**
   - 支援 ACID（原子性、一致性、隔離性、持久性）事務。
   - 提供 WAL（Write-Ahead Logging）模式，提升並發寫入性能。
4. **跨平台支援**
   - 可運行於 Windows、Linux、macOS、Android、iOS 等多種平台。
   - 不依賴特定硬體或作業系統。
5. **動態類型**
   - 支援靈活的資料類型（如 INTEGER、TEXT、REAL、BLOB），欄位可儲存任何類型數據。
6. **擴充功能**
   - 支援自訂函數（UDF）和虛擬表。
   - 可透過擴充模組（如 FTS5）實現全文搜尋。

SQLite 的特性

SQLite 的設計使其在特定場景中獨具優勢：

1. **輕量級**
   - 核心程式庫大小約 700 KB，資源佔用極低。
   - 適合記憶體和儲存空間有限的設備（如手機、嵌入式系統）。
2. **無伺服器架構**
   - 不需要獨立伺服器，直接由應用程式存取資料庫檔案。
   - 減少配置與維護成本。
3. **單檔案儲存**
   - 所有數據（表格、索引等）儲存在單一檔案中，便於備份與遷移。

4. **高可靠性**
    - 經過廣泛測試，穩定性高，適合長期運行。
    - 支援資料庫檔案加密（透過 SQLite Encryption Extension，需付費）。
5. **開源與免費**
    - 遵循公共領域許可（Public Domain），無版權限制。
    - 可自由使用、修改與分發。
6. **零配置**
    - 無需安裝或設定即可使用，開箱即用。

SQLite 的限制

雖然功能強大，但 SQLite 的設計也帶來一些限制：

1. **並發性能有限**
    - 僅支援有限的並發寫入（單一寫入者，WAL 模式下稍有改善）。
    - 不適合高並發應用（如大型網站後端）。
2. **資料量限制**
    - 理論上限：281 TB（檔案系統限制）。
    - 實務建議：資料庫大小超過 1-2 GB 時性能下降，適合中小型數據集。
3. **功能不完整**
    - 不支援某些進階 SQL 功能，如：
        - `RIGHT OUTER JOIN` 和 `FULL OUTER JOIN`（僅支援 `LEFT JOIN`）。
        - 無存儲過程（Stored Procedures）。
        - 無用戶管理或權限控制（因無伺服器設計）。
    - 觸發器功能有限，無法完全替代複雜邏輯。
4. **不支援分佈式系統**
    - 單機運行，無法像 MySQL 或 PostgreSQL 那樣實現分區或複製。
    - 不適合需要高可用性或負載平衡的場景。
5. **寫入密集應用受限**
    - 檔案鎖定機制可能導致寫入瓶頸，尤其在多執行緒環境中。
    - 不適合頻繁寫入的應用（如即時交易系統）。
6. **擴展性不足**
    - 無法輕鬆升級到企業級資料庫，需重構至其他系統（如 PostgreSQL）。

總結與適用場景

- **優勢總結**：輕量、簡單、高效，適合嵌入式應用、小型專案或單機環境。
- **適用場景**：

    - 行動應用（如 iOS/Android App 的本地資料庫）。
    - 桌面軟體（如瀏覽器、媒體播放器）。
    - 原型開發或教育用途。
- **不適用場景**：
    - 高並發 Web 應用。
    - 大規模企業資料庫。
    - 需要複雜權限管理的系統。

若您需要針對特定場景（如行動應用開發）深入分析 SQLite 的應用，或與其他資料庫（如 PostgreSQL）比較，請告訴我，我會進一步調整內容！

SQLite 的資料型別（Data Types）與傳統關係型資料庫（如 MySQL、PostgreSQL）有所不同，它採用**動態型別系統**（Dynamic Typing），這意味著欄位的資料型別並不嚴格綁定，而是根據儲存的值動態決定。以下是 SQLite 資料型別的詳細說明，包括其分類、特性與實際應用，呈現方式為條列式。

SQLite 資料型別的分類

SQLite 官方定義了五種主要的**儲存類別**（Storage Classes），這些類別決定了數據在資料庫中的儲存方式：

1. **NULL**
    - **描述**：表示空值或無數據。
    - **用途**：用於表示缺失或未知的值。
    - **範例**：`NULL`（如某欄位未填寫）。
2. **INTEGER**
    - **描述**：有符號整數，根據值大小使用 1、2、3、4、6 或 8 位元組儲存。
    - **範圍**：-2^63 到 +2^63-1（約 ±9.22 quintillion）。
    - **用途**：儲存整數數據，如 ID、年齡。
    - **範例**：`42`, `-15`, `0`。
3. **REAL**
    - **描述**：浮點數，使用 8 位元組儲存（IEEE 754 格式）。
    - **範圍**：精確度約 15-17 位有效數字。
    - **用途**：儲存帶小數的數值，如價格、溫度。
    - **範例**：`3.14`, `-0.001`, `2.0`。
4. **TEXT**
    - **描述**：文字字符串，支援 UTF-8、UTF-16BE 或 UTF-16LE 編碼。
    - **長度**：無嚴格限制（取決於資料庫檔案大小，上限約 281 TB）。
    - **用途**：儲存文字數據，如姓名、地址。
    - **範例**：`"Hello"`, `"John Doe"`, `""`（空字符串）。
5. **BLOB**
    - **描述**：二進位大型物件（Binary Large Object），儲存原始二進位數據。

- **長度**：無嚴格限制（同 TEXT）。
- **用途**：儲存圖片、文件或其他非結構化數據。
- **範例**：`x'89504E47'`（PNG 檔案開頭的十六進位數據）。

動態型別的特性

SQLite 的動態型別系統帶來以下獨特特性：

1. **型別親和性 (Type Affinity)**
   - SQLite 在創建表格時，可以指定欄位的「親和性」（Affinity），但不強制約束儲存值的型別。
   - 五種親和性：
     - **INTEGER**：偏好儲存整數。
     - **REAL**：偏好儲存浮點數。
     - **TEXT**：偏好儲存文字。
     - **BLOB**：偏好儲存二進位數據。
     - **NUMERIC**：偏好儲存數字（整數或浮點數），若無法轉換則儲存原始值。
   - **範例**：

     ```
     CREATE TABLE example (id INTEGER, value TEXT);
     INSERT INTO example VALUES ('123', 456); -- 允許插入不同型別
     ```

2. **靈活儲存**
   - 一個欄位可以儲存任何儲存類別的值，不會因宣告型別而拒絕。
   - 範例：宣告為 `INTEGER` 的欄位仍可儲存 `"abc"`，但查詢時會嘗試轉為整數（失敗則返回原始值）。
3. **型別轉換**
   - SQLite 在運算或比較時會自動嘗試轉換型別。
   - 範例：`"123" + 1` 結果為 `124`（文字轉整數）。

與 SQL 標準型別的對應

雖然 SQLite 使用儲存類別，但支援傳統 SQL 型別的語法，這些型別會映射到對應的親和性：

SQL 型別	映射到的親和性
INT, INTEGER, BIGINT	INTEGER
REAL, DOUBLE, FLOAT	REAL
CHAR, VARCHAR, TEXT	TEXT
BLOB, BINARY	BLOB
NUMERIC, DECIMAL	NUMERIC

- **注意**：SQLite 不強制執行長度限制（如 `VARCHAR(50)` 的 50 不生效），僅作為親和性參考。

Install SQLite

- 安裝 SQLite Download
- 安裝 DBeaver GUI Download
- 安裝 DBeaver GUI Download

實際應用範例

以下是一個簡單的表格定義與資料，展示型別使用：

```
CREATE TABLE students (
  student_id INTEGER PRIMARY KEY,  -- 整數主鍵
  name TEXT,                       -- 文字
  gpa REAL,                        -- 浮點數
  photo BLOB,                      -- 二進位數據
  notes TEXT                       -- 可儲存任何值
);

INSERT INTO students VALUES
  (1, 'Alice', 3.8, NULL, 'Good student'),
  (2, 'Bob', 3.2, x'89504E47', 123); -- BLOB 與數字混用
```

限制與注意事項

1. **無嚴格型別檢查**
   - 不像 MySQL 或 PostgreSQL，SQLite 不會阻止插入不匹配的型別，可能導致數據一致性問題。
2. **無日期/時間型別**
   - 無內建 `DATE` 或 `TIMESTAMP` 型別，需用 TEXT（如 "2023-10-15"）、INTEGER（Unix 時間戳）或 REAL（Julian 日）儲存。
   - 支援日期函數（如 `strftime`）處理。
3. **大小限制**
   - 單一值最大約 1 GB（由 `SQLITE_MAX_LENGTH` 控制），但建議避免過大數據以保持性能。

總結

- **功能**：SQLite 提供靈活的動態型別系統，支援五種儲存類別，滿足基本資料庫需求。
- **特性**：簡單、輕量，型別親和性允許高度自由度。
- **限制**：缺乏嚴格型別約束與進階型別（如日期），適合小型應用而非複雜系統。

若您需要針對某應用場景（如嵌入式系統）分析 SQLite 型別的應用，或想比較其與其他資料庫的型別系統，請告訴我，我會進一步深化說明！

SQLite 鍵（Key）類型說明

SQLite 支援多種類型的鍵（keys）用於資料表設計和關聯管理。以下是 SQLite 中主要的鍵類型：

主鍵（PRIMARY KEY）

```
-- 單一欄位作為主鍵
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY,
    username TEXT NOT NULL
);

-- 自動增長的主鍵
CREATE TABLE products (
    product_id INTEGER PRIMARY KEY AUTOINCREMENT,
    product_name TEXT NOT NULL
);
```

SQLite 的主鍵特性：

- 可以使用任何資料型別作為主鍵，但通常使用 INTEGER
- 當宣告為 `INTEGER PRIMARY KEY` 時，該欄位會自動成為別名為 `rowid` 的內部主鍵
- 使用 `AUTOINCREMENT` 關鍵字可以確保刪除記錄後，新增的記錄不會重複使用已刪除的 ID 值

複合主鍵（COMPOSITE PRIMARY KEY）

```
CREATE TABLE enrollments (
    student_id INTEGER,
    course_id INTEGER,
    enrollment_date TEXT,
    PRIMARY KEY (student_id, course_id)
);
```

複合主鍵特性：

- 由多個欄位組合而成
- 組合值在整個資料表中必須唯一
- 複合主鍵不能使用 `AUTOINCREMENT`

外鍵（FOREIGN KEY）

SQLite 從 3.6.19 版本開始支援外鍵約束，但預設是關閉的。需要使用 `PRAGMA foreign_keys = ON;` 啟用。

```
-- 啟用外鍵支援
PRAGMA foreign_keys = ON;

-- 單一外鍵
CREATE TABLE orders (
    order_id INTEGER PRIMARY KEY,
    customer_id INTEGER,
```

```
    order_date TEXT,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

-- 複合外鍵
CREATE TABLE order_items (
    order_id INTEGER,
    product_id INTEGER,
    quantity INTEGER,
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

外鍵特性：

- 用於建立表格之間的關聯
- 可以設定在更新或刪除參照記錄時的行為（CASCADE, SET NULL, SET DEFAULT, RESTRICT）
- 可以對單一欄位或多個欄位（複合外鍵）進行設定

唯一鍵（UNIQUE KEY）

```
CREATE TABLE customers (
    customer_id INTEGER PRIMARY KEY,
    email TEXT UNIQUE,
    phone TEXT UNIQUE
);

-- 複合唯一鍵
CREATE TABLE employee_projects (
    employee_id INTEGER,
    project_id INTEGER,
    role TEXT,
    UNIQUE (employee_id, project_id)
);
```

唯一鍵特性：

- 確保該欄位或欄位組合的值在表格中是唯一的
- 與主鍵不同，唯一鍵允許 NULL 值（SQLite 中，多個 NULL 值被視為不同值）

索引（INDEX）

雖然索引不是嚴格意義上的「鍵」，但它們在關聯資料庫中扮演重要角色：

```
-- 創建索引
CREATE INDEX idx_customers_name ON customers(last_name, first_name);

-- 唯一索引
CREATE UNIQUE INDEX idx_users_email ON users(email);
```

索引特性：

- 提升查詢效能
- 可以是單一欄位或多個欄位
- 可以設定為唯一索引（UNIQUE INDEX）

SQLite 的鍵結構雖然相比其他大型關聯式資料庫系統較為簡單，但仍提供了足夠的功能來確保資料的完整性和高效查詢。