SQLite

A brief introduction to black magic for Mobile Developers

Custom SQLite version

Custom SQLite version Installation

1. Obtain the code
2. Configure the native libraries.
3. Build the native libraries
4. Assemble the aar file

Custom SQLite version Installation

...or download compiled version and add is as a sub-project

```
dependencies {
        // Change "sqlite-android-3130000" to the name of the new module!
        compile project(':sqlite-android-3130000')
    }
```

Indexes

Indexes

An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently

Primary Key

Primary Key

```
CREATE TABLE countries (
 country_id INTEGER PRIMARY KEY,
 name text NOT NULL
);
```

Primary Key

```
CREATE TABLE countries (
 country_id INTEGER PRIMARY KEY,
 name text NOT NULL
);
```

Questions

Can PRIMARY KEY be null?

Question

Can PRIMARY KEY be null?

YES. Because of bug in early versions of SQLite

ROWID

ROWID

```
CREATE TABLE countries (
 country_id INTEGER,
 name text NOT NULL
);

....

SELECT * FROM countries WHERE rowid > 5
```

ROWID

```
CREATE TABLE countries (
 country_id INTEGER PRIMARY KEY,
 name text NOT NULL
);

....

SELECT * FROM countries WHERE rowid > 5
```

ROWID

```
CREATE TABLE countries (
 country_id INTEGER PRIMARY KEY, ← Alias for ROWID
 name text NOT NULL
);

....

SELECT * FROM countries WHERE rowid > 5
```

Question

Can we use ROWID as a logical row ID?

Question

Can we use ROWID as a logical row ID?

NO. ROWID could be changed in some circumstances (e.g. “VACUUM”)

Complex Primary Key

Primary Key

```
CREATE TABLE country_languages (
 country_id integer NOT NULL,
 language_id integer NOT NULL,
 PRIMARY KEY (country_id, language_id),
);
```

Primary Key

```
CREATE TABLE country_languages (
 country_id integer NOT NULL,
 language_id integer NOT NULL,
 PRIMARY KEY (country_id, language_id),
);
```

Question

What is faster - ROWID or Primary Key?

Question

What is faster - ROWID or Primary Key?

ROWID. It is used as a key in B-Tree

Autoincrement

Autoincrement

```
CREATE TABLE country_languages (
 country_id integer NOT NULL AUTOINCREMENT,
 language_id integer NOT NULL,
 PRIMARY KEY (country_id, language_id),
);
```

Question

Why we need Autoincrement if we already have ROWID?

Question

Why we need Autoincrement if we already have ROWID?

“the purpose of AUTOINCREMENT is to prevent the reuse of ROWIDs from previously deleted rows” docs

Indexes On Expressions

Indexes On Expressions

```
CREATE TABLE transactions(
  transaction_id INTEGER PRIMARY KEY,
  account_id INTEGER REFERENCES account,
  amount INTEGER,  -- in cents
...
);
CREATE INDEX transactions_magnitude ON transactions(account_id,
abs(amount));
```

Indexes On Expressions

```
SELECT * FROM transactions
WHERE account_id=...
AND
abs(amount)>=10000;
```

Question

Can we use Expression as a Primary Key?

Question

Can we use Expression as a Primary Key?

NO. They are not unique.

FTS

FTS

```
CREATE VIRTUAL TABLE email_fts USING fts5(sender, title, body)
```

FTS

```
SELECT * FROM email_fts WHERE email MATCH 'term' ORDER BY rank;

SELECT highlight(email, 2, '<b>', '</b>') FROM email_fts('term');

SELECT * FROM email('fts5') WHERE MATCH '"one two thr" * ';


SELECT * FROM email('fts5') WHERE MATCH 'NEAR(one two) three';
```

Question

Do we need a Primary Key in FTS table?

Question

Do we need a Primary Key in FTS table?

NO. FTS use ROWID + other indexes

R*Tree

R*Tree

```
CREATE VIRTUAL TABLE demo_index USING rtree(
   id,              -- Integer primary key
   minX, maxX,      -- Minimum and maximum X coordinate
   minY, maxY       -- Minimum and maximum Y coordinate
);
```

R*Tree

```
SELECT id FROM demo_index
 WHERE minX>=-81.08 AND maxX<=-80.58
   AND minY>=35.00  AND maxY<=35.44;
```

Question

What type SQLite use for minX, maxX, minY, maxY?

Question

What type SQLite use for minX, maxX, minY, maxY?

32bit float. Say “hello” to Roundoff Error

R*Tree

```
SELECT objname FROM demo_data, demo_index
 WHERE demo_data.id=demo_index.id
   AND contained_in(demo_data.boundary, :boundary)
   AND minX>=-81.0 AND maxX<=-79.6
   AND minY>=35.0 AND maxY>=36.2;
```

R*Tree

```
CREATE TABLE demo_data(
  id INTEGER PRIMARY KEY,  -- primary key
  objname TEXT,            -- name of the object
  objtype TEXT,            -- object type
  boundary BLOB            -- detailed boundary of object
);
```

JSON

JSON

```
CREATE TABLE `users` (
	`id` INTEGER,
	`name` TEXT,
	`body` JSON,
	PRIMARY KEY(`id`)
)
```

JSON

```
SELECT * FROM users WHERE json_extract(body, '$.data.phone') = 323
```

JSON

```
UPDATE user SET body = json_replace(body, '$.data.phone', 3) WHERE id = 1
```

JSON

```
CREATE TABLE `users` (
	`id` INTEGER,
	`name` TEXT,
	`phones` JSON, -- ['213-231-2', '555-5533', '12345']
	PRIMARY KEY(`id`)
)
```

JSON

```
SELECT DISTINCT user.name
  FROM user, json_each(user.phone)
 WHERE json_each.value LIKE '704-%';
```

JSON

```
SELECT DISTINCT user.name
  FROM user, json_each(user.phone)
 WHERE json_each.value LIKE '704-%';
```

JSON

```
CREATE TABLE json_tree(
    key ANY,            -- key for current element relative to its parent
    value ANY,          -- value for the current element
    type TEXT,          -- 'object','array','string','integer', etc.
    atom ANY,           -- value for primitive types, null for array & object
    id INTEGER          -- integer ID for this element
    parent INTEGER,     -- integer ID for the parent of this element
    fullkey TEXT,       -- full path describing the current element
    path TEXT,          -- path to the container of the current row
    json JSON HIDDEN,   -- 1st input parameter: the raw JSON
    root TEXT HIDDEN    -- 2nd input parameter: the PATH at which to start
);
```

Question

What’s the difference between JSON and TEXT fields?

Question

What’s the difference between JSON and TEXT fields?

No difference at all :(Because of backward compatibility

Recursive

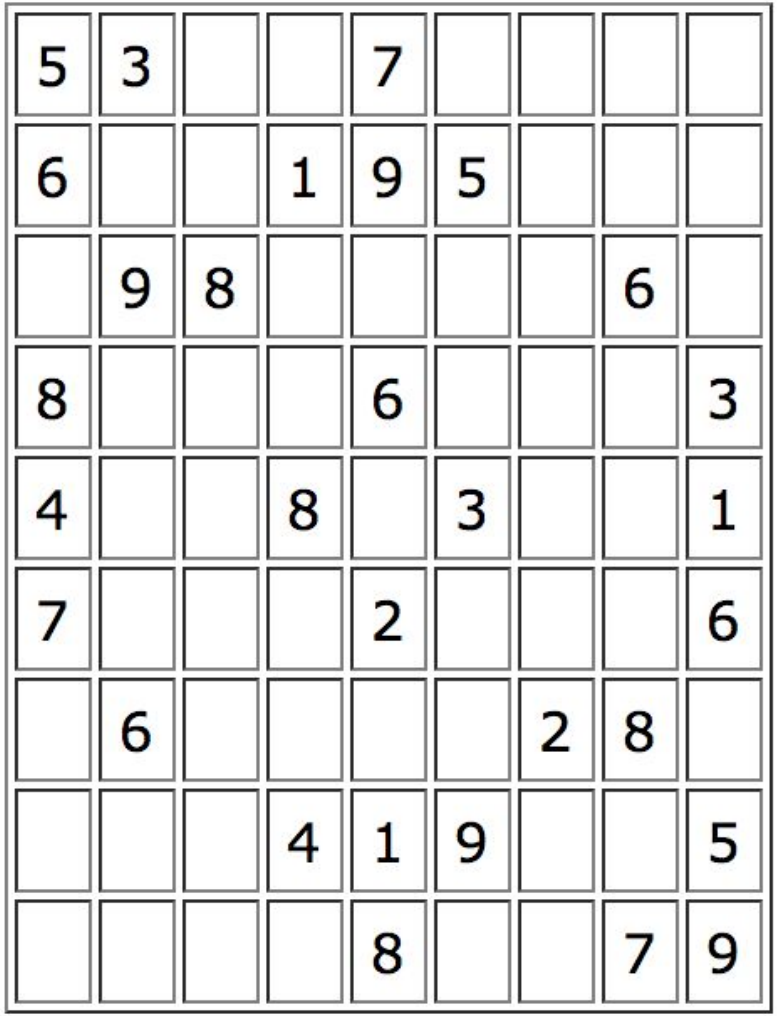
Recursive

```
CREATE TABLE staff(
  name TEXT PRIMARY KEY,
  boss TEXT REFERENCES org,
  salary INT, --cents
);
```

Recursive

```
WITH RECURSIVE
  works_for_vasya(n) AS (
    VALUES('Vasya')
    UNION
    SELECT name FROM staff, works_for_vasya
     WHERE org.boss=works_for_vasya.n
  )
SELECT avg(salary) FROM org
 WHERE org.name IN works_for_vasya;
```

DEMO


5 3 7
6 1 9 5
9 8 6
8 6 3
4 8 3 1
7 2 6
6 2 8
4 1 9 5
8 7 9

53..7....6..195....98....6.8...6...34..8.3..17.
..2...6.6....28....419..5....8..79


QUESTIONS?
ASK THEM MEOW
memegenerator.net