SQLite
CS582 Project Phase 1

Alex Robic and Matt Bundas

SQLite - An Overview

- SQL-type RDBMS, known for being minimalist, concise, local, yet powerful

”SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.” - sqlite.org

- Used when you are dealing with local, non-enormous, single-writer data

- Not open source, but in the public domain

SQLite - Development and History

- Developed in 2000

- Original developer/founder wrote SQLite to support one of his applications, it blew up from there

- Just 3 developers working on SQLite

- Much of its development since its beginnings is bug fixing, small features

- Lots of testing, ~2 million tests needed to attain 100% branch coverage

SQLite - Unique Characteristics

- Local - Stores data in a single ".db" file, is serverless

- Deployability - Very lightweight (~700 kB), in one C file, cross platform, zero configuration, in public domain

- Much more embedded in applications than MySQL and others, removing overhead, can be faster than reading straight from disk.

- Only one writer at a time, makes use of file-system locks

- Data Typing

SQLite - Drawbacks

- Only one writer at a time, file locking
- Serverless
- Data stored in one file, harder to use parallel disks, limits on max file size
- Less features
- Scalability
- Security Issues due to being in public domain
- Less irregularity checking, similar to C in general

SQLite - Competitors

- Major RDBMS like Postgre, MySQL, Oracle

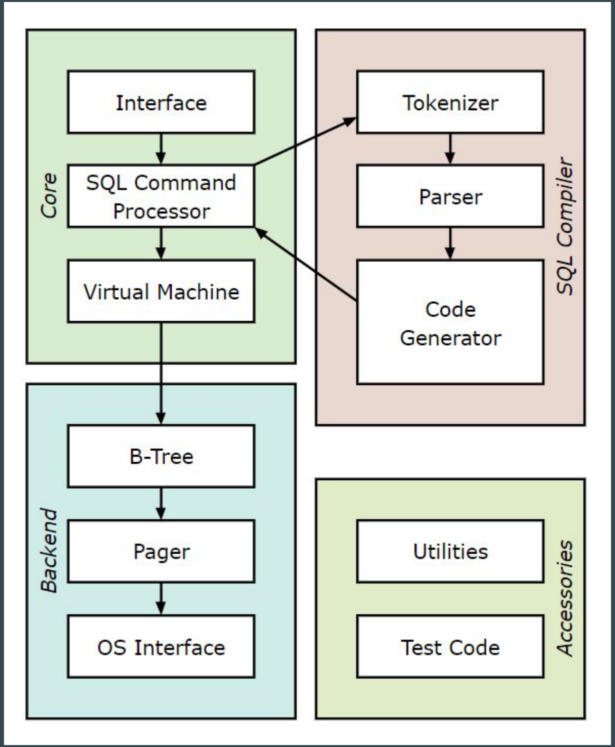
- “Lite” DBMS like OrmLite (SQL), Couchbase Lite

- Exists in a precise, yet expansive niche

- Founder claims its direct competitor is fopen()

SQLite - Architecture

- SQLite uses bytecode running in a VM
- Parser calls tokenizer (as opposed to YACC/BISON)
- Lemon parser generator (maintained with SQLite)
- Query planner
- B-Tree implementation for Indexes and Tables



SQLite - Indexing

- B-Trees:
    - Balanced tree
    - Minimize complexity when doing insertions, deletions and searches
    - Complexity for each operation logarithmic
- Can use algebraic expressions as Indexes
- Each B-Tree stored in the same file
- Can use function calls but only from deterministic functions

SQLite - Supported Queries

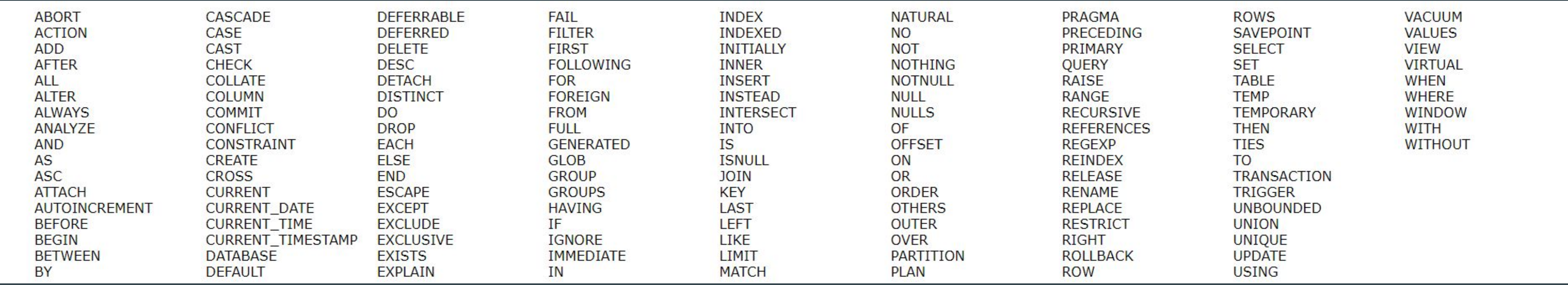
- Supports SQL language (but not all functionalities)

- Some functions in the Alter table family of commands NOT supported

- Right outer join and full outer join are omitted

- Cross and left join are implemented

- Doesn’t support GRANT and REVOKE commands (not a big database system)



ABORT	CASCADE	DEFERRABLE	FAIL	INDEX	NATURAL	PRAGMA	ROWS	VACUUM
ACTION	CASE	DEFERRED	FILTER	INDEXED	NO	PRECEDING	SAVEPOINT	VALUES
ADD	CAST	DELETE	FIRST	INITIALLY	NOT	PRIMARY	SELECT	VIEW
AFTER	CHECK	DESC	FOLLOWING	INNER	NOTHING	QUERY	SET	VIRTUAL
ALL	COLLATE	DETACH	FOR	INSERT	NOTNULL	RAISE	TABLE	WHEN
ALTER	COLUMN	DISTINCT	FOREIGN	INSTEAD	NULL	RANGE	TEMP	WHERE
ALWAYS	COMMIT	DO	FROM	INTERSECT	NULLS	RECURSIVE	TEMPORARY	WINDOW
ANALYZE	CONFLICT	DROP	FULL	INTO	OF	REFERENCES	THEN	WITH
AND	CONSTRAINT	EACH	GENERATED	IS	OFFSET	REGEXP	TIES	WITHOUT
AS	CREATE	ELSE	GLOB	ISNULL	ON	REINDEX	TO	
ASC	CROSS	END	GROUP	JOIN	OR	RELEASE	TRANSACTION	
ATTACH	CURRENT	ESCAPE	GROUPS	KEY	ORDER	RENAME	TRIGGER	
AUTOINCREMENT	CURRENT_DATE	EXCEPT	HAVING	LAST	OTHERS	REPLACE	UNBOUNDED	
BEFORE	CURRENT_TIME	EXCLUDE	IF	LEFT	OUTER	RESTRICT	UNION	
BEGIN	CURRENT_TIMESTAMP	EXCLUSIVE	IGNORE	LIKE	OVER	RIGHT	UNIQUE	
BETWEEN	DATABASE	EXISTS	IMMEDIATE	LIMIT	PARTITION	ROLLBACK	UPDATE	
BY	DEFAULT	EXPLAIN	IN	MATCH	PLAN	ROW	USING	

SQLite - Query Optimization

- embedded system → no querying from external databases (save time on networking)

- b tree implementation helps save time on insertion, deletion and searching (logarithmic complexity)

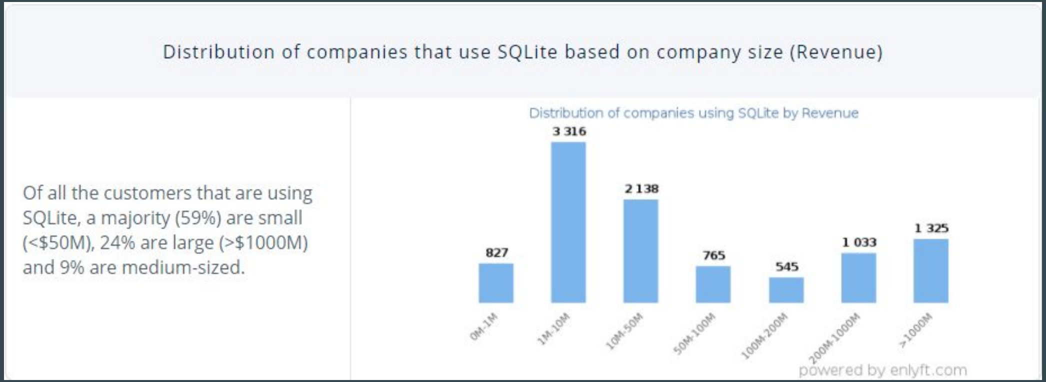
- very low footprint (700kB) → doesn’t interfere with other programs

SQLite - Use Cases

- Used when you are dealing with local, relatively small, often single user data
- Embedded into applications to handle data
- As a central hub to help multiple applications communicate
- Small to medium scale websites, < 100k hits a day
- In tandem with low-frequency updated data stored with a server RDBMS, like MySQL
- Transferring content across devices or operating systems
- Education and training, or in spaces where experts aren’t the primary user

SQLite - Users

- Used in essentially every smartphone, camera, game console, also used in web browsers and avionics
- Built into and used by the most popular operating systems, Windows and Mac
- Millions of other applications
- Over 1,300 individual billion dollar companies use it
- Perhaps the most widely deployed software in the world



Distribution of companies that use SQLite based on company size (Revenue)

Of all the customers that are using SQLite, a majority (59%) are small (<$50M), 24% are large (>$1000M) and 9% are medium-sized.

Resources

1 hr youtube video from creator - https://www.youtube.com/watch?v=Jib2AmRb_rk

wikipedia page - https://en.wikipedia.org/wiki/SQLite

tutorials - https://www.tutorialspoint.com/sqlite/sqlite_overview.htm

high level overview - https://www.codecademy.com/articles/what-is-sqlite

comparing mysql, others - https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

indexes - https://www.sqlitetutorial.net/sqlite-index/

official sqlite indexes - https://sqlite.org/expridx.html

indexes and performance - https://medium.com/@JasonWyatt/squeezing-performance-from-sqlite-indexes-indexes-c4e175f3c346

optimization from sqlite - https://www.sqlite.org/optoverview.html

official when to use sqlite - https://www.sqlite.org/whentouse.html

https://blog.trigent.com/five-of-the-most-popular-databases-for-mobile-apps/

market share - https://enlyft.com/tech/products/sqlite