

# SQLite3:

I got your data right  
here! And Here! And  
**HERE! AND HERE!**

## Dan Hibbitts

- 👁 consultant - Target Your Hand
  - 👁 iPhone and Android solution development
  - 👁 web and desktop development
  - 👁 database solutions
- 👁 over 20 years of experience
- 👁 contact info
  - 👁 twitter: @MobilityMatters
  - 👁 email: dhibbitts@TargetYourHand.com

## agenda

- SQLite overview
- SQLite API
- hands-on SQLite
- Freshmeat DB Library for Objective-C
- resources

# what is SQLite

- single file database
- portable/cross-platform file format
- SQL db engine library small footprint (~265k-765k depending on options)
- serverless (zero administration)
- public domain license
- ACID (atomic, constancy, isolation, durability)
- thread and process safe (if hardware works)
- UTF8 and UTF16 support

# how does it store data

- file based, backward compatible to 3.0.0
- forward compatible
- 32bit / 64bit
- big-endian / little-endian (internally big-endian)
- IEEE floating point support
- no naming conventions on files
- can attach to multiple files to do cross database queries

# where is it used

- desktops - all shipping Macs, Solaris, available for Windows and others
- mobile devices - iPhone, iPod, iPad, Android, Symbian, and others
- Used by CoreData
- embedded hardware
  - zero malloc option
- embedded in applications
  - Firefox, Chrome, Safari, LightRoom, AIR, ...
- source available

## where does it fit?

- not intended to compete with client/server db systems
- intended to compete with fopen()
  - csv, xml, home grown file formats, etc.

## so what

- custom file formats suck
- XML is overly verbose
- time spent writing code to manage files could be better spent elsewhere
- solves the multi platform issue in advance

## API

- simple API
  - sqlite3 - object used to connect to db
  - sqlite3\_stmt - prepared statement object
- 8 interfaces (or families)
- app defined functions and collating sequences

## APIs (contd.)

- `sqlite3_open` // open the db file
- `sqlite3_prepare` // create prepared statement
- `sqlite3_bind` // bind columns to data
- `sqlite3_step` // step through rows
- `sqlite3_column` // access columns
- `sqlite3_reset` // reuse the prepared statement
- `sqlite3_finalize` // destroy the prepared statement
- `sqlite3_close` // close the db file

## accessing data

- DML (inserts/updates using SQL strings)
  - don't use DML
- data binding
  - binds variables in your code to columns in your database and allows you to iterate over rowsets

## why use data binding



<http://xkcd.com/327/>

`Robert'); DROP TABLE Students; --`

## pseudo code

```
sqlite3_open();
stmt=sqlite3_prepare("INSERT INTO tb VALUES
(?,?,?)" );
for ( i = 0; i < 10; i++ ) {
  sqlite3_bind( stmt, 1, yourFirstValue );
  sqlite3_bind( stmt, 2, yourSecondValue );
  sqlite3_bind( stmt, 3, yourThirdValue );
  sqlite3_step( stmt );
  sqlite3_reset( stmt );
}
sqlite3_finalize( stmt );
sqlite3_close();
```

## dynamic typing

datatypes are only suggestions

```
CREATE TABLE tab( x INTEGER );
INSERT INTO tab VALUES ( 8675309 );
INSERT INTO tab VALUES ( '8675309' );
INSERT INTO tab VALUES ( 'Jenny' );
SELECT * FROM tab;
8675309
8675309
Jenny
```

## check constraints exist

(there is no need to panic)

```
CREATE TABLE tab1 ( z INTEGER
CHECK( typeof(z) = 'INTEGER' ));

CREATE TABLE tab2 (
x VARCHAR(20)
CHECK( typeof(x) = 'text'
AND length(x) <= 20 ) );
```

# sqlite3 app

- character mode app allowing you to open and manipulate sqlite databases
- .help - displays commands
- .tables - displays tables in the open database
- .schema - shows create statement for a table
- DML or DDL SQL commands (terminated by ;)

## hands-on: SQLite



source: Ben Stanger

“but I don’t want to use  
a C API or CoreData!”

- FMDB
  - a thin Objective-C library that wraps the C API
  - easy to use objects that allow you to manipulate the database objects

# FMDB classes

- FMDatabase - wraps the database functions
  - executeUpdate
  - executeQuery
  - beginTransaction, rollback and commit
  - hadError

# FMDB classes (cont.)

- FMResultSet - wrap the result of a query (table, view, etc.) usually obtained using the result of the database object's executeQuery
  - intForColumnIndex
  - stringForColumnIndex
  - next
  - hasAnotherRow

## hands-on: FMDB



# take-aways

- SQLite3 is a powerful and well designed database library
- while being well supported on Mac OS X and iPhone OS is also portable and well supported on other platforms
- SQLite is "easy" to use
- if you are looking for a book check out "The Definitive Guide to SQLite" (available on Safari)

the end



thank you!

Photo: Frank Ruffini / iStockphoto.com

[dhibbitts@TargetYourHand.com](mailto:dhibbitts@TargetYourHand.com)

twitter: [@MobilityMatters](https://twitter.com/MobilityMatters)

you can download these slides -  
<http://tinyurl.com/mmPDF20100513>