mmsql 1.1 – Handbuch
SQLite-Verwaltung für den Colour Maximite 2

© 2026 Matthias Hollatz – MIT-Lizenz (siehe LICENSE.txt)

Dieses Handbuch beschreibt mmsql 1.1 für den Colour Maximite 2 mit Firmware 8. Es richtet sich an alle, die mit mmsql arbeiten, vom ersten Ausprobieren bis zum Austausch von Datenbanken mit dem PC. Wer SQL noch gar nicht kennt, fängt am besten mit dem Tutorial an (TUTORIAL_mmsql.pdf) und nimmt dieses Handbuch zum Nachschlagen.

Inhalt

1. Einführung
2. Voraussetzungen und Installation
3. Erste Schritte
4. Der Zeileneditor und die Tasten
5. SQL mit mmsql
6. Befehlsübersicht
7. Optionen (.set)
8. Ausgabe, Export und Import
9. Grenzen und Höchstwerte
10. Meldungen und Hinweise
11. Hinweise zur Firmware
12. Anhang

1. Einführung

mmsql ist ein Verwaltungswerkzeug für SQLite-Datenbanken, geschrieben in MMBasic für den Colour Maximite 2 (CMM2). Seit der Firmware 7 enthält der CMM2 eine SQLite-Datenbank, die BASIC-Programme mit den Befehlen `SQL OPEN`, `SQL EXEC`, `SQL QUERY` usw. nutzen können. mmsql macht diese Datenbanken direkt am CMM2 zugänglich: ansehen, abfragen, ändern, prüfen, reparieren, exportieren und importieren.

mmsql arbeitet wie das Kommandozeilen-Werkzeug `sqlite3` am PC:

- Befehle für mmsql selbst beginnen mit einem **Punkt**, z. B. `.tables` oder `.help`.
- Alles andere ist **SQL** und geht direkt an die Datenbank, z. B. `SELECT * FROM computer`.

mmsql läuft gleichzeitig auf dem Bildschirm des CMM2 und auf einem seriellen Terminal (z. B. Tera Term). Beide zeigen dasselbe, man kann an beiden tippen.

Datenbanken, die mmsql anlegt oder ändert, sind normale SQLite-Dateien. Sie lassen sich am PC mit `sqlite3`, dem *DB Browser for SQLite* oder jedem anderen SQLite-Werkzeug öffnen, und umgekehrt.

Lizenz: mmsql steht unter der MIT-Lizenz. SQLite selbst ist gemeinfrei (public domain).

2. Voraussetzungen und Installation

Voraussetzungen

Rechner	Colour Maximite 2, Generation 1, 2 oder 3
Firmware	MMBasic **8.00.00b1** oder neuer
SQLite-Modul	`A:/sql/sql.mod` aus dem Firmware-Paket
Für Firmware 7.00.00RC6	mmsql **1.0.1** (eigene Version, ohne die Neuerungen von 1.1)

Startet mmsql 1.1 auf einer älteren Firmware, meldet es sich mit einem Hinweis auf mmsql 1.0.1 und beendet sich.

Dateien

Datei	Zweck
`mmsql.bas`	das Programm
`mmsql_en.hlp`	Hilfetexte, Englisch
`mmsql_de.hlp`	Hilfetexte, Deutsch
`LICENSE.txt`	MIT-Lizenz
`README.md`, `TUTORIAL_mmsql.pdf`, `demo.sql`	Kurzbeschreibung, Tutorial, Beispieldaten

Alle Dateien kommen in denselben Ordner, z. B. `A:/sql/`. mmsql legt in diesem Ordner zwei weitere Dateien an:

Datei	Inhalt
`mmsql.cfg`	die Optionen (`.set`), als lesbare Textdatei
`mmsql.his`	der Verlauf der eingegebenen Befehle

Beide dürfen gelöscht werden; mmsql beginnt dann mit den Standardwerten bzw. einem leeren Verlauf.

Übertragen auf den CMM2

Am einfachsten mit XMODEM über das serielle Kabel, z. B. in Tera Term (*Datei → Übertragung → XMODEM → Senden*), am CMM2:

```
chdir "a:/sql"
xmodem receive "mmsql.bas"
```

Für jede Datei wiederholen. Alternativ die SD-Karte am PC beschreiben.

3. Erste Schritte

Starten

```
RUN "mmsql"
RUN "mmsql", "A:/sql/computers.db"
```

Ohne Datei startet mmsql ohne Datenbank; mit Datei wird sie gleich geöffnet. mmsql zeigt beim Start eine Kopfzeile:

```
mmsql V1.1 - SQLite manager for the Colour Maximite 2 (G2)
(c) 2026 Matthias Hollatz - MIT License
Type .help for help
mmsql>
```

In Klammern steht die Generation des CMM2 (G1, G2 oder G3; unter `OPTION PICOMITE` „G2/G3“, weil die Firmware die G3 dann nicht meldet).

Der Prompt

Prompt	Bedeutung
`mmsql>`	keine Datenbank offen
`mmsql[club.db]>`	die Datenbank `club.db` ist offen
`mmsql[club.db]*>`	eine Transaktion ist offen (nach `BEGIN`), siehe Kapitel 5

Befehle und SQL

```
mmsql> .open club.db
Opened: club.db
mmsql[club.db]> .tables
name             type  ncol
-------------------------
member           table 4
members_per_city view  2
owns             table 3
 3 row(s)
mmsql[club.db]> SELECT name, city FROM member WHERE joined < 2021
```

Vergisst man den Punkt vor einem mmsql-Befehl, hilft mmsql weiter:

```
mmsql> open computers.db
Did you mean .open? Commands start with a dot.
```

Das gilt auch für ein Komma statt des Punktes (`,help`).

Hilfe

Eingabe	zeigt
`.help`	die Übersicht aller Befehle nach Gruppen
`.help befehl`	Einzelheiten zu einem Befehl, z. B. `.help import`
`.help keys`	die Tasten des Zeileneditors
`.help sql`	Hinweise zu SQL in mmsql
`.help utf8`	die Anzeige von Umlauten und anderen UTF-8-Zeichen

Die Hilfetexte stehen in `mmsql_en.hlp` bzw. `mmsql_de.hlp`. Umschalten mit `.set helpfile mmsql_de.hlp`.

Beenden

`.quit` (oder `.q`, `.exit`) schließt die Datenbank und beendet mmsql. Ist eine Transaktion offen, fragt mmsql vorher nach (siehe `.close`).

4. Der Zeileneditor und die Tasten

Jede Eingabe wird in einem Zeileneditor bearbeitet, am Bildschirm und am Terminal gleich.

Taste	Wirkung
← →	Cursor bewegen
Pos1 / Ende	an den Anfang / das Ende der Zeile
Backspace	Zeichen links vom Cursor löschen
Entf	Zeichen am Cursor löschen
Esc	Zeile leeren
↑ ↓	frühere Eingaben aus dem Verlauf holen
Enter	Eingabe ausführen

Zeilen, die länger als die Bildschirmbreite sind, scrollen seitlich. Eine Eingabe ist höchstens **255 Zeichen** lang (die Länge eines MMBasic-Strings).

Verlauf: Die letzten Eingaben (Standard 50, Option `history`) werden in `mmsql.his` gespeichert und stehen auch nach einem Neustart mit ↑ zur Verfügung.

Blättern in Ergebnissen (`.browse`, Abfragen, lange Ausgaben):

Taste	Wirkung
Enter oder N	nächste Seite
B	vorige Seite (bei `.browse` und Abfragen)
Q oder Esc	aufhören

Ctrl-C

- **Während eine SQL-Anweisung läuft**, bricht Ctrl-C sie ab. mmsql läuft weiter und sagt genau, was SQLite rückgängig gemacht hat:

```
Interrupted (Ctrl-C).
The statement was undone. Your transaction is still open (COMMIT or ROLLBACK).
```

War die abgebrochene Anweisung eine Änderung (INSERT, UPDATE, DELETE …) innerhalb einer Transaktion, macht SQLite die **ganze Transaktion** rückgängig; mmsql meldet dann „The whole transaction since BEGIN was undone.“ Eine abgebrochene Abfrage (SELECT) lässt die Transaktion offen.

- **An allen anderen Stellen** (am Prompt, beim Schreiben einer Exportdatei) beendet Ctrl-C mmsql wie jedes BASIC-Programm. Eine offene Transaktion wird dann rückgängig gemacht; nichts davon wird geschrieben. Eine unvollständige Exportdatei bleibt liegen.
- Mit `OPTION BREAK 0` lässt sich eine laufende SQL-Anweisung nicht abbrechen; mmsql weist beim Start darauf hin. Mit `OPTION BREAK 4` übernimmt Ctrl-D die Rolle von Ctrl-C.

Umlaute und UTF-8

SQLite speichert Text als UTF-8. Die Schrift des CMM2 kennt keine Umlaute, deshalb zeigt mmsql jedes UTF-8-Zeichen (z. B. ä, ß, €) als **ein** Platzhalterzeichen an: Standard ist Code 250, ein mittiger Punkt (im Tera Term erscheint es als `ú`). So bleiben Spalten gerade, und der Zeileneditor behandelt das Zeichen als eines.

Einstellung	Wirkung
`.set utf8 dot`	Platzhalter (Standard)
`.set utf8 raw`	Bytes unverändert senden – sinnvoll, wenn das Terminal auf UTF-8 steht
`.set dotchar 46`	anderer Platzhalter, hier der Punkt (Code 33 bis 255)

Dateien (`.once`, `.output`, `.dump`) und die Datenbank enthalten immer echtes UTF-8. Umlaute lassen sich in mmsql derzeit nicht eintippen.

5. SQL mit mmsql

Anweisungen

Alles, was nicht mit einem Punkt beginnt, ist SQL. Ein Semikolon am Ende ist erlaubt, aber nicht nötig.

- Anweisungen, die Daten liefern (`SELECT`, `WITH`, `VALUES`, `PRAGMA`), zeigen das Ergebnis im eingestellten Ausgabemodus (Standard: Tabelle, seitenweise).
- Alle anderen melden `OK`; `INSERT`, `UPDATE`, `DELETE` und `REPLACE` zusätzlich die Zahl der geänderten Zeilen und bei `INSERT` die neue rowid:

```
mmsql[test.db]> INSERT INTO t VALUES(1)
OK, 1 row(s) changed, last ID 1
```

Mehrere Anweisungen in einer Zeile

Mehrere Anweisungen werden mit `;` getrennt und **einzeln nacheinander** ausgeführt, jede mit ihrem eigenen Ergebnis – wie bei `sqlite3`:

```
mmsql[test.db]> CREATE TABLE t(a); INSERT INTO t VALUES(1); SELECT * FROM t
OK
OK, 1 row(s) changed, last ID 1
a
-
1
 1 row(s)
```

- Der erste Fehler (oder Ctrl-C) stoppt den Rest der Zeile: „Not run: n more statement(s) of this line.“
- Ein `;` in Texten (`'a;b'`), in Namen in Anführungszeichen (`"a;b"`, `` `a;b` ``, `[a;b]`) und in Kommentaren (`/* … */`) trennt nicht. Ab `--` gilt der Rest der Zeile als Kommentar.
- `CREATE TRIGGER … BEGIN …; …; END` bleibt in einem Stück, weil der Rumpf selbst Semikolons enthält.
- Höchstens 32 Anweisungen pro Zeile.

Transaktionen

Eine Transaktion fasst mehrere Änderungen zusammen: entweder alle oder keine. Bei vielen INSERTs ist sie außerdem viel schneller, weil die SD-Karte nur einmal beschrieben wird.

```
BEGIN
INSERT INTO ...
INSERT INTO ...
COMMIT
```

`COMMIT` (oder `END`) schreibt die Änderungen, `ROLLBACK` verwirft sie. `SAVEPOINT`, `RELEASE` und `ROLLBACK TO` funktionieren ebenfalls.

Solange eine Transaktion offen ist, zeigt der Prompt einen **Stern**: `mmsql[test.db]*>`. mmsql verfolgt dazu `BEGIN`, `SAVEPOINT`, `COMMIT`, `END` und `ROLLBACK`. Nach Fehlern, Ctrl-C und `RELEASE` fragt mmsql SQLite, ob die Transaktion noch offen ist, und meldet es, wenn SQLite sie selbst rückgängig gemacht hat („Note: SQLite has undone the open transaction.“).

`.quit`, `.close` und `.open` fragen bei offener Transaktion: `[C] commit  [R] rollback  [Esc] cancel`.

Mehrere Datenbanken: ATTACH

Neben der offenen Datenbank (`main`) lassen sich weitere Dateien anhängen, bis zu 10:

```
ATTACH 'A:/sql/computers.db' AS retro
SELECT m.name, k.name FROM owns o JOIN member m ON m.id = o.member_id JOIN retro.computer k ON k.id =
o.computer_id
CREATE TABLE computer AS SELECT * FROM retro.computer
DETACH retro
```

`.databases` zeigt die angehängten Dateien.

Was das SQLite-Modul nicht enthält

Das SQLite-Modul des CMM2 (SQLite 3.47.0) ist für den Speicher des Rechners verkleinert. Es fehlen:

fehlt	Folge
`EXPLAIN`, `ANALYZE`	keine Abfragepläne, keine Statistik
WAL-Modus	Journal immer im Modus `delete`; Dateien im WAL-Modus vom PC lassen sich nicht öffnen
Mathe-Funktionen	kein `sqrt()`, `sin()`, `power()` usw.
Erweiterungen	kein `load_extension()`
Zeitzonen	`datetime('now')` liefert die Uhrzeit der CMM2-Uhr (`TIME$`); die Modifikatoren `localtime` und `utc` fehlen

Vorhanden sind u. a. JSON-Funktionen, Fensterfunktionen, Views, Trigger, Fremdschlüssel, `VACUUM INTO` und `ATTACH`. Datenbanken in UTF-8 und UTF-16 lassen sich öffnen; Text kommt immer als UTF-8 an.

6. Befehlsübersicht

Die Befehle sind alphabetisch geordnet. Eckige Klammern `[ ]` bezeichnen Angaben, die man weglassen kann; `a|b` heißt „a oder b“. Kurzformen stehen in Klammern hinter dem Befehl. Dateinamen ohne Laufwerk und Ordner beziehen sich auf den aktuellen Ordner von MMBasic.

Übersicht nach Gruppen

Gruppe	Befehle
Datenbank	`.open` `.close` `.info` `.databases` `.backup` `.vacuum` `.files` `.kill`
Ansehen	`.tables` `.indexes` `.triggers` `.schema` `.columns` `.browse`
Daten	`.insert` `.edit` `.delete` `.import` `.read`
Struktur	`.create` `.alter` `.reindex`
Prüfen	`.check` `.fkcheck` `.fk`
Ausgabe	`.mode` `.headers` `.separator` `.nullvalue` `.once` `.output` `.dump`
Optionen	`.set` `.timer` `.bail` `.echo` `.cls` `.help` `.quit`

.alter tabelle

Ändert die Struktur einer Tabelle. Zeigt die Spalten und bietet an:

Taste	Aktion
A	Spalte hinzufügen (Name, Typ, NOT NULL, Default werden abgefragt)
R	Spalte umbenennen
D	Spalte löschen
T	Tabelle umbenennen

Vor dem Ausführen wird die SQL-Anweisung gezeigt. Beispiel: `.alter member`

.backup datei.db

Schreibt eine saubere, kompakte Kopie der offenen Datenbank (`VACUUM INTO`). Fragt, bevor eine vorhandene Datei überschrieben wird. Auch der beste Weg, aus einer beschädigten Datenbank zu retten, was noch lesbar ist. Beispiel: `.backup club_copy.db`

.bail on|off

`on`: `.read` stoppt beim ersten Fehler. `off` (Standard): `.read` macht weiter und stoppt erst nach 10 Fehlern.

.browse tabelle [WHERE …] [ORDER BY …] (`.b`)

Zeigt die Zeilen einer Tabelle seitenweise (Enter/N, B, Q). Die Anzahl Zeilen pro Seite stellt die Option `rows` ein.

```
.browse computer
.browse items WHERE city='Berlin' ORDER BY price DESC
```

Spalten werden höchstens 24 Zeichen breit. Längere Werte enden mit `~` (gekürzt), Werte über `maxtext` erscheinen als `<n bytes>~!`. Kam eine der Markierungen vor, steht hinter der Zeilenzahl eine Legende. Passen nicht alle Spalten auf den Bildschirm, meldet mmsql „(+n column(s) not shown)“; `.mode line` zeigt dann alle.

.check [full]

`.check`	schnelle Prüfung (`PRAGMA quick_check`)
`.check full`	vollständige Prüfung (`PRAGMA integrity_check`)

Zeigt „OK – no problems found.“ oder die gefundenen Probleme. Große Datenbanken dauern, auf Generation 1 sogar Minuten. Bei Problemen mit `.backup` retten, was geht.

.close

Schließt die Datenbank und meldet „Closed: datei“. Bei offener Transaktion fragt mmsql `[C] commit [R] rollback [Esc] cancel`; mit Esc bleibt die Datenbank offen.

.cls

Löscht Bildschirm und Terminal.

.columns tabelle (`.c`)

Zeigt die Spalten einer Tabelle: Nummer, Name, Typ, NOT NULL, Default und ob die Spalte (Teil des) Primärschlüssels ist.

.create tabelle

Legt eine Tabelle Schritt für Schritt an: je Spalte Name, Typ, Primärschlüssel, NOT NULL und Default. Ein leerer Name beendet die Liste. Die fertige Anweisung wird vor dem Ausführen gezeigt; sie darf höchstens 255 Zeichen lang sein (sonst die Tabelle per SQL anlegen).

.databases (`.db`)

Listet die offene Datenbank (`main`), `temp` und angehängte Datenbanken mit ihren Dateien.

.delete tabelle schlüssel

Zeigt die Zeile und löscht sie nach einer Rückfrage. `schlüssel` ist der Wert des Primärschlüssels; hat die Tabelle keinen einspaltigen Primärschlüssel, ist es die `rowid` (mit `SELECT rowid, * FROM tabelle` sichtbar). Lange Werte erscheinen kurz als `(long text, n bytes)` bzw. `(BLOB, n bytes)` . Beispiel: `.delete computer 19`

.dump [tabelle]

Gibt die ganze Datenbank (oder eine Tabelle) als SQL-Text aus: `CREATE TABLE` , ein `INSERT` für jede Zeile, dann Indizes, Views und Trigger, eingerahmt von `BEGIN TRANSACTION;` und `COMMIT;` . Mit `.once` oder `.output` in eine Datei; mit `.read` wieder einlesbar oder am PC mit `sqlite3` verwendbar. Format siehe Kapitel 8.

```
.once club_backup.sql
.dump
```

.echo on|off

`on` : `.read` zeigt jede Anweisung vor dem Ausführen (lange Zeilen gekürzt mit Längenangabe).

.edit tabelle schlüssel

Ändert eine Zeile. `schlüssel` wie bei `.delete` . Jedes Feld erscheint mit seinem Wert im Zeileneditor:

Eingabe	Wirkung
Enter	Wert behalten
neuer Wert	ändern
`NULL` oder `\N`	NULL
`''`	leerer Text
`\NULL`	der Text „NULL“

Gespeichert wird nach „Save n change(s)? (y/n)“ in einer Transaktion; nur geänderte Spalten werden geschrieben. Texte über 255 Zeichen und BLOBs erscheinen als `(long text, n bytes – kept)` bzw. `(BLOB, n bytes – kept)` und bleiben unverändert; ändern lassen sie sich per SQL (`UPDATE`). Beispiel: `.edit computer 5`

.files [muster]

Listet Dateien mit Größe und Datum, nach Namen sortiert, mit dem Ordner darüber. Ohne Muster: `*.db` . Größen unter 10 000 Bytes erscheinen genau in Bytes, darüber in KB.

```
.files *.sql
.files *
.files A:/sql/*.db
```

.fk [on|off]

Zeigt oder schaltet die Prüfung von Fremdschlüsseln für diese Sitzung (`PRAGMA foreign_keys`). SQLite startet mit ausgeschalteter Prüfung.

.fkcheck [tabelle]

Findet Zeilen, die einen Fremdschlüssel verletzen (`PRAGMA foreign_key_check`).

.headers on|off

Spaltennamen als erste Zeile bei `csv` und `list` (Standard: on).

.help [befehl] (`.h`)

Zeigt die Übersicht bzw. die Hilfe zu einem Befehl (siehe Kapitel 3).

.import [–skip N] datei.csv tabelle

Importiert eine CSV-Datei nach RFC 4180 (Felder in Anführungszeichen, `""` für ein Anführungszeichen, Zeilenumbrüche in Feldern).

- **Neue Tabelle:** Die erste Zeile liefert die Spaltennamen, alle Spalten bekommen den Typ TEXT.
- **Vorhandene Tabelle:** Alle Zeilen sind Daten; `--skip 1` überspringt eine Kopfzeile.
- Trennzeichen aus `.separator` (ein Zeichen).
- Höchstens 20 Spalten. Ein Feld pro Datensatz darf länger als 255 Zeichen sein (bis `maxtext`).
- Der Import läuft in einer Transaktion. Nach 10 fehlerhaften Datensätzen (falsche Feldzahl, zu lang …) wird **nichts** importiert. Bei Ctrl-C ebenfalls nichts.

```
mmsql[club.db]> .import members.csv member_copy
Created table member_copy with 4 column(s)
 8 row(s) imported into member_copy
```

Siehe auch Kapitel 8.

.indexes [tabelle] (`.ix`, `.indices`)

Listet die Indizes aller Tabellen oder einer Tabelle mit ihren Spalten.

.info

Zeigt Datei, SQLite-Version, Seitengröße, Zahl der Seiten und Größe, freie Seiten (`VACUUM` gibt sie frei), Journal-Modus, Encoding, Auto-Vacuum, `user_version` und die Anzahl der Tabellen, Views, Indizes und Trigger.

.insert tabelle

Fügt eine Zeile ein. Fragt jede Spalte ab und zeigt dabei Typ, NOT NULL, Default und Primärschlüssel:

Eingabe	Wirkung
Enter	Default-Wert bzw. neuer Schlüssel bei INTEGER PRIMARY KEY
`NULL` oder `\N`	NULL
`''`	leerer Text
`\NULL`	der Text „NULL“

Am Ende „Insert? (y/n)“ und die neue rowid. Höchstens 20 Spalten und 255 Zeichen je Wert; für breitere Tabellen `INSERT` per SQL verwenden.

.kill datei

Löscht genau eine Datei nach einer Rückfrage (keine Platzhalter wie `*`). Die offene Datenbank lässt sich nicht löschen (erst `.close`). Liegt eine `datei.db-journal` daneben (Rest einer abgebrochenen Transaktion), bietet mmsql an, sie mitzulöschen.

```
mmsql> .kill test.db
Delete test.db (8192 B)? (y/n) y
Deleted: test.db
```

.mode table|csv|list|insert [tabelle]|line

Stellt die Ausgabe von Abfragen ein (wird in `mmsql.cfg` gespeichert). Einzelheiten in Kapitel 8.

Modus	Ausgabe
`table`	ausgerichtete Spalten, seitenweise (Standard)
`csv`	durch Komma getrennt (RFC 4180)
`list`	Werte mit Trennzeichen, ohne Anführungszeichen
`insert`	`INSERT`-Anweisungen für die angegebene Tabelle (Standard „table“)
`line`	ein Feld pro Zeile, gut für breite Zeilen und lange Texte

---

.nullvalue text

Wie NULL in `table`, `list`, `csv` und `line` erscheint, z. B. `.nullvalue NULL`. `.nullvalue ''` zeigt NULL leer (Standard). Gespeichert in `mmsql.cfg`.

.once datei

Die **nächste** Abfrage (oder `.dump`) geht in die Datei statt auf den Bildschirm. Eine vorhandene Datei wird überschrieben.

```
.mode csv
.once items.csv
SELECT * FROM items
```

.open datei.db (`.o`)

Öffnet eine Datenbank. Fehlt die Datei, fragt mmsql, bevor es eine neue, leere anlegt. Es ist immer nur eine Datenbank offen (weitere mit `ATTACH`); eine bereits offene wird vorher geschlossen (mit Rückfrage bei offener Transaktion). Beispiel: `.open A:/daten/verein.db`

.output [datei]

Alle folgenden Abfragen gehen in die Datei, bis `.output` ohne Datei wieder auf den Bildschirm schaltet.

.quit (`.q`, `.exit`)

Schließt die Datenbank (siehe `.close`) und beendet mmsql.

.read datei.sql

Führt SQL aus einer Datei aus.

- Eine oder mehrere Anweisungen pro Zeile (getrennt mit `;`, wie am Prompt).
- Zeilen über 255 Zeichen (z. B. aus `.dump`) werden als Ganzes ausgeführt, bis `maxtext`; längere werden übersprungen.
- Leere Zeilen und Zeilen, die mit `--` beginnen, werden übersprungen; Punkt-Befehle werden nicht ausgeführt.
- Stoppt nach 10 Fehlern, mit `.bail on` beim ersten.
- Ctrl-Z oder NUL am Ende (Füllbytes von XMODEM) beenden die Datei.
- Am Ende: „n statement(s) OK, n error(s), n line(s) read“, und ein Hinweis, falls eine Transaktion offen geblieben ist.

Eine Anweisung muss in **einer** Zeile stehen; über mehrere Zeilen verteilte Anweisungen versteht `.read` noch nicht (siehe Anhang A).

.reindex [name]

Baut alle Indizes neu auf, die einer Tabelle oder einen einzelnen Index.

.schema [name] (`.s`)

Zeigt die `CREATE`-Anweisungen aller Objekte oder einer Tabelle, eines Index, einer View oder eines Triggers.

.separator x

Trennzeichen für `csv` und `list`, 1 bis 3 Zeichen; `\t` steht für Tab. `.import` braucht ein Trennzeichen aus einem Zeichen. Gespeichert in `mmsql.cfg`.

.set [name wert]

`.set` allein listet alle Optionen mit Wert und Erklärung; `.set name wert` ändert eine Option, zeigt die geänderte Zeile und speichert sie in `mmsql.cfg`. Alle Optionen in Kapitel 7.

.tables (`.t`)

Listet die Tabellen und Views mit der Anzahl ihrer Spalten.

.timer on|off

Zeigt nach SQL und längeren Befehlen die Laufzeit. Gespeichert in `mmsql.cfg`.

.triggers

Listet die Trigger und ihre Tabellen.

.vacuum

Baut die Datei ohne freie Seiten neu auf und zeigt die Größe vorher und nachher. Braucht kurzzeitig etwa so viel freien Platz auf der Karte wie die Datenbank selbst.

7. Optionen (.set)

Alle Optionen werden mit `.set name wert` geändert und sofort in `mmsql.cfg` (neben dem Programm) gespeichert.

Option	Standard	erlaubt	Bedeutung
`rows`	20	5 bis 100	Zeilen pro Seite beim Blättern (gilt nach Neustart)
`width`	Bildschirmbreite	40 bis 250	Ausgabebreite in Zeichen
`history`	50	0 bis 200	Zeilen im Verlauf (gilt nach Neustart)
`timer`	0	0, 1	Laufzeit anzeigen
`mode`	table	table, csv, list, insert, line	Ausgabemodus
`headers`	1	on/off bzw. 1/0	Spaltennamen bei csv und list
`separator`	`,`	1 bis 3 Zeichen oder `\t`	Trennzeichen für csv und list
`nullvalue`	(leer)	beliebiger Text	Anzeige von NULL
`utf8`	dot	dot, raw	Anzeige von UTF-8-Zeichen
`dotchar`	250	Code 33 bis 255 oder ein Zeichen	Platzhalter für `utf8 dot`
`maxtext`	32768	1024 bis 1048576	längster Text bzw. BLOB, den mmsql liest (Bytes)
`helpfile`	mmsql_en.hlp	Dateiname	Hilfedatei, z. B. mmsql_de.hlp

width: Der Standard ist die Zeichenbreite des Bildschirms, höchstens die mit `OPTION WIDTH` eingestellte Terminalbreite.

maxtext: mmsql legt dafür einen Speicherpuffer dieser Größe an. Auf Generation 2 und 3 ist auch 1 MB kein Problem; auf Generation 1 mit wenig freiem Speicher besser kleiner bleiben. Reicht der Speicher nicht, setzt mmsql den Wert auf 32768 zurück.

`mmsql.cfg` ist eine einfache Textdatei mit Zeilen `name=wert` und kann auch am PC bearbeitet werden.

8. Ausgabe, Export und Import

Ausgabemodi

table – ausgerichtete Spalten mit Kopfzeile, seitenweise:

```
id name             maker       year cpu   ramkb
-----------------------------------------------
1  Apple II         Apple       1977 6502  4
2  PET 2001         Commodore   1977 6502  4
```

- Eine Spalte ist höchstens 24 Zeichen breit. Längere Werte enden mit `~`.
- Texte über `maxtext` erscheinen als `<n bytes>~!`.
- Passen nicht alle Spalten in die Breite, werden die übrigen weggelassen: „(+n column(s) not shown)“.
- Hinter der Zeilenzahl erklärt eine Legende `~` und `~!`, wenn sie vorkamen.

line – ein Feld pro Zeile; zeigt lange Texte ganz, mit Zeilenumbrüchen eingerückt:

```
id   = 9
name = Commodore 64
note = erste Zeile
       zweite Zeile
```

csv – Werte durch das Trennzeichen getrennt (Standard Komma), nach RFC 4180: Ein Feld kommt in Anführungszeichen, wenn es das Trennzeichen, ein Anführungszeichen oder einen Zeilenumbruch enthält; Anführungszeichen werden verdoppelt. Mit `.headers on` steht die Kopfzeile vorn.

list – wie csv, aber ohne Anführungszeichen.

insert – je Zeile ein `INSERT INTO "tabelle" VALUES(...);` für die bei `.mode insert tabelle` angegebene Tabelle.

Werte in Exporten

Wert	csv / list	insert / .dump
NULL	Text aus `.nullvalue` (Standard leer)	`NULL`
ganze Zahl	wie gespeichert (64 Bit)	wie gespeichert
Kommazahl	15 gültige Stellen	17 gültige Stellen (verlustfrei)
Text	unverändert (UTF-8)	`'Text'`, `'` verdoppelt
Text mit Zeilenumbruch	in Anführungszeichen	`'a'||char(10)||'b'` – alles in einer Zeile
BLOB	rohe Bytes	`X'0102A0…'`

Dateien

- Exportdateien (`.once`, `.output`, `.dump`) bekommen das Zeilenende **CRLF**.
- Lange Texte und BLOBs werden **vollständig** geschrieben. Ist ein Wert länger als `maxtext`, bricht der Export mit einer Meldung ab (Zeile, Spalte, nötige Größe), und die Datei wird als unvollständig gemeldet. Abhilfe: `.set maxtext` erhöhen.

```
Stopped: a text of 40000 bytes (table lt, row 3, column note) is longer than
maxtext (32768). Use .set maxtext 40000 and try again.
lt_dump.sql is incomplete.
```

Das Dump-Format

```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE member(id INTEGER PRIMARY KEY, name TEXT NOT NULL, city TEXT, joined INTEGER);
INSERT INTO "member" VALUES(1,'Ada','London',2019);
...
CREATE INDEX owns_member ON owns(member_id);
CREATE VIEW members_per_city AS SELECT city, count(*) AS members FROM member GROUP BY city;
COMMIT;
```

Jede Anweisung steht in **einer** Zeile; Zeilenumbrüche in Texten werden mit `char(10)` bzw. `char(13)` geschrieben. So lässt sich ein Dump mit `.read` wieder einlesen und auch mit `sqlite3` am PC verwenden.

Import

CSV mit .import: Siehe Befehl `.import`. Zu beachten:

- NULL ist in CSV nicht von einem leeren Feld zu unterscheiden; ein leeres Feld wird als leerer Text importiert.

- Neue Tabellen bekommen nur Spalten vom Typ TEXT. Für Zahlen die Tabelle vorher mit den gewünschten Typen anlegen (`CREATE TABLE`) und dann in die vorhandene Tabelle importieren (`--skip 1` bei Kopfzeile).
- Höchstens 20 Spalten, ein Feld pro Datensatz über 255 Zeichen.

SQL mit .read: Dumps von mmsql und einfache SQL-Skripte mit einer Anweisung pro Zeile. Ein Dump von `sqlite3` am PC funktioniert, solange keine Anweisung über mehrere Zeilen geht (bei `CREATE TABLE` aus Werkzeugen und bei Triggern oft der Fall).

Austausch mit dem PC

- Dateien per XMODEM übertragen oder die SD-Karte am PC lesen.
- XMODEM füllt den letzten Block auf (Tera Term mit Ctrl-Z, der CMM2 mit NUL). `.read` und `.import` erkennen das als Dateiende; am PC kann man die Füllbytes am Dateiende ignorieren oder abschneiden.
- Datenbanken vom PC müssen im Journal-Modus `delete` gespeichert sein, nicht `wal` (am PC: `PRAGMA journal_mode=delete;`).

Alle Exporte und Importe wurden Byte für Byte mit SQLite am PC verglichen: CSV, insert und Dump, Rundreisen CSV → Datenbank → CSV und Dump → Datenbank → Dump, mit Anführungszeichen, Zeilenumbrüchen, Tabs, UTF-8, 64-Bit-Zahlen, Texten bis 40 000 Zeichen und BLOBs mit NUL-Bytes.

9. Grenzen und Höchstwerte

Was	Grenze	Grund / Hinweis
Eingabe am Prompt	255 Zeichen	Länge eines MMBasic-Strings
Anweisungen pro Zeile	32	
Zeile in `.read`	bis `maxtext`	als Ganzes ausgeführt
Text oder BLOB lesen und schreiben	bis `maxtext` (1024 bis 1 048 576 Bytes)	Speicherpuffer
Wert im Zeileneditor (`.edit`, `.insert`)	255 Zeichen	längere: per SQL
Spalten bei `.insert` und `.import`	20	Firmware: höchstens 20 Werte pro Befehl
Spalten bei `.edit`	32 (G1) / 64 (G2, G3)	Spaltenpuffer
Lange Felder pro Datensatz bei `.import`	1	
Fehler bis zum Abbruch (`.read`, `.import`)	10	
Spalten einer Abfrage im Speicher	32 (G1) / 64 (G2, G3)	weitere werden nicht gezeigt
Spaltenbreite in `table`	24 Zeichen	längere enden mit `~`
Spaltennamen	64 Zeichen	
Namen in Listen (`.schema`, `.tables`, `.files`)	200	
`.create`-Anweisung	255 Zeichen	sonst per SQL
Zeilen pro Seite (`rows`)	5 bis 100	
Ausgabebreite (`width`)	40 bis 250	
Verlauf (`history`)	0 bis 200 Einträge	
Trennzeichen	1 bis 3 Zeichen (`.import`: 1)	
angehängte Datenbanken (`ATTACH`)	10	SQLite-Modul
Datenbankgröße	durch die SD-Karte begrenzt	FAT32 / exFAT

Von mmsql belegte Dateinummern: #7 (`.read`, `.import`), #8 (`.once`, `.output`, `.dump`), #9 (`mmsql.cfg`, `mmsql.his`, Hilfedatei). Wer mmsql-Teile in eigene Programme übernimmt, sollte diese Nummern nicht gleichzeitig verwenden.

10. Meldungen und Hinweise

Fehler von SQLite erscheinen als `Error: SQL: …` zusammen mit der Anweisung und ihrer Länge. Zu häufigen Fehlern gibt mmsql einen Hinweis:

Meldung	Bedeutung / Abhilfe
`no such table`	Tabellenname prüfen; `.tables` listet die Tabellen
`syntax error`	Tippfehler im SQL – oder ein Punkt-Befehl ohne Punkt; Eingabe max. 255 Zeichen
`file is not a database`	keine SQLite-Datei, verschlüsselt oder am PC im WAL-Modus gespeichert
`unable to open database file`	Dateiname, Ordner oder SD-Karte prüfen
`disk I/O error`	SD-Karte voll oder fehlerhaft (siehe auch Kapitel 11)
`attempt to write a readonly database`	Datei oder SD-Karte schreibgeschützt
`database is locked`	offene Transaktion? `COMMIT` oder `ROLLBACK`
`cannot commit - no transaction is active`	es läuft keine Transaktion (evtl. nach Ctrl-C schon rückgängig gemacht)
`stopped by Ctrl-C`	Anweisung abgebrochen, siehe Kapitel 4
`Did you mean .xxx?`	Punkt vergessen
`Not run: n more statement(s)`	Zeile nach Fehler abgebrochen
`Line n: longer than maxtext`	`.read`: Zeile zu lang, `.set maxtext` erhöhen
`Record n: … skipped`	`.import`: Datensatz fehlerhaft (Feldzahl, Länge)
`Too many errors`	`.read` / `.import` nach 10 Fehlern gestoppt
`Note: SQLite has undone the open transaction.`	nach einem Fehler hat SQLite die Transaktion beendet
`A transaction is still open`	nach `.read`: `COMMIT` oder `ROLLBACK` eingeben

11. Hinweise zur Firmware

OPTION CRLF: Ohne `OPTION PICOMITE` schickt die Enter-Taste in Firmware 8 standardmäßig CR und LF. mmsql setzt beim Start `OPTION CRLF CR`; die Einstellung gilt nur, solange mmsql läuft.

OPTION BREAK: Bestimmt die Abbruchtaste (Standard Ctrl-C). Mit `OPTION BREAK 0` lassen sich SQL-Anweisungen nicht abbrechen.

LGETSTR$ in 8.00.00b1: `LGETSTR$` schreibt bei einem Ergebnis von genau 255 Zeichen ein Byte zu viel in den Speicher. Das kann die offene Datenbank stören („disk I/O error“ beim COMMIT). mmsql ruft `LGETSTR$` deshalb nie mit mehr als 254 Zeichen auf. Gefunden beim Bau von mmsql 1.1, von Peter Mather für die nächste Firmware behoben. Wer eigene Programme mit LONGSTRING und SQL schreibt, sollte das ebenso halten.

LINE INPUT # und Tabs: `LINE INPUT #` wandelt Tabs in Leerzeichen um (gewolltes Verhalten von MMBasic). mmsql liest Dateien deshalb mit `INPUT$`, damit Tabs in Daten erhalten bleiben.

MM.DEVICE$ unter OPTION PICOMITE: liefert dann den PicoMite-Namen; die G3 lässt sich so nicht erkennen (mmsql zeigt „G2/G3“).

12. Anhang

A. Unterschiede zu sqlite3

mmsql hat die wichtigsten Befehle von `sqlite3`, dazu eigene für die Arbeit am CMM2 (`.browse`, `.insert`, `.edit`, `.delete`, `.create`, `.alter`, `.columns`, `.info`, `.check`, `.fkcheck`, `.files`, `.kill`, `.set`, `.cls`).

Noch nicht vorhanden, für spätere Versionen vorgesehen:

- Anweisungen über **mehrere Zeilen** am Prompt und in `.read` (z. B. Trigger und formatierte `CREATE TABLE` aus PC-Dumps); damit auch Anweisungen über 255 Zeichen am Prompt
- `.import` mit mehr als 20 Spalten
- weitere Ausgabemodi: `markdown`, `json`, `tabs`, `quote`
- `.tables` und `.schema` mit Muster, `.restore`, `.changes`, `.print`, `.width`

Auf dem CMM2 nicht möglich: `.recover` (fehlende Erweiterung), `.eqp` / `.expert` (kein EXPLAIN), `.load` (keine Erweiterungen), `.shell` / `.system`, `.excel`.

B. Neu in 1.1 gegenüber 1.0

- Braucht Firmware 8; für 7.00.00RC6 gibt es mmsql 1.0.1.
- Ctrl-C bricht SQL ab, mmsql läuft weiter und meldet genau, was rückgängig gemacht wurde.
- Transaktionen im Blick: Stern im Prompt, Rückfrage bei `.quit`, `.close`, `.open`.
- Mehrere Anweisungen pro Zeile, einzeln mit Ergebnis.
- Lange Texte und BLOBs über 255 Zeichen in Anzeige, Export, Dump, `.read` und `.import`; Option `maxtext`; Markierungen `~` und `~!`.
- Neue Befehle `.files` und `.kill`.
- `.info` zeigt das echte Encoding; Größen genau in Bytes bzw. KB; G1/G2/G3 im Kopf.
- Kein doppeltes Enter mehr (`OPTION CRLF CR`), Hinweis bei `OPTION BREAK 0`, Hinweis auch bei `,help`.
- Korrekturen aus 1.0.1: Absturz bei Werten mit genau 255 Zeichen und einem Anführungszeichen am Ende; Zeilenumbrüche in `.mode line`.

C. Dank

An Peter Mather für SQLite auf dem CMM2 und für die schnellen Antworten und Korrekturen (Ctrl-C, LONGSTRING, LGETSTR$), an Geoff Graham für MMBasic und an die Community auf TheBackShed.