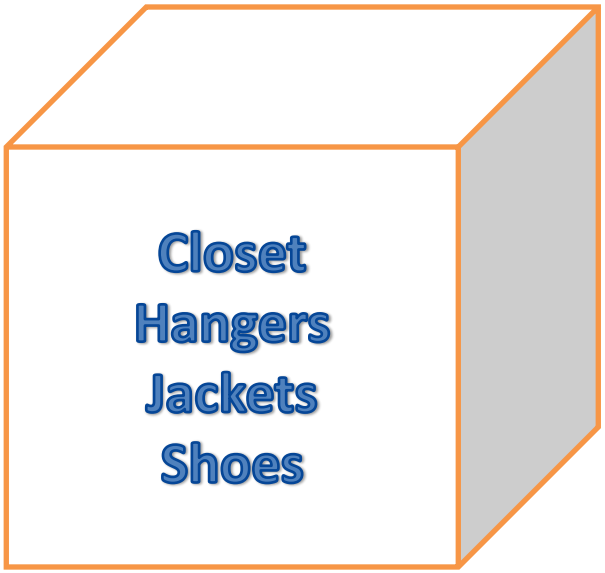
Introduction to Databases and SQL

(Structured Query Language)

Why Have a Database?

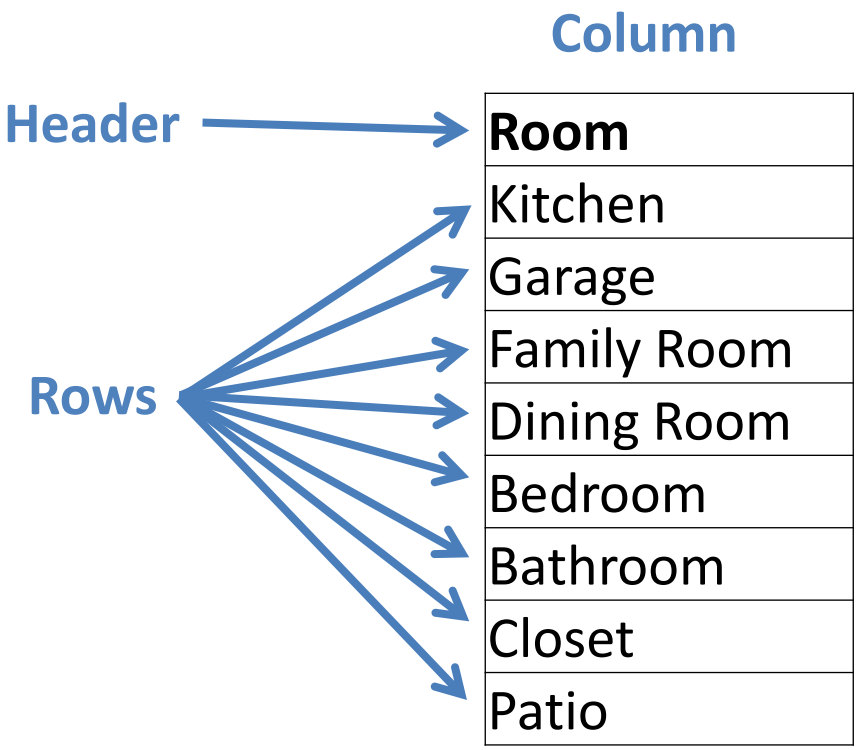
- A moving story



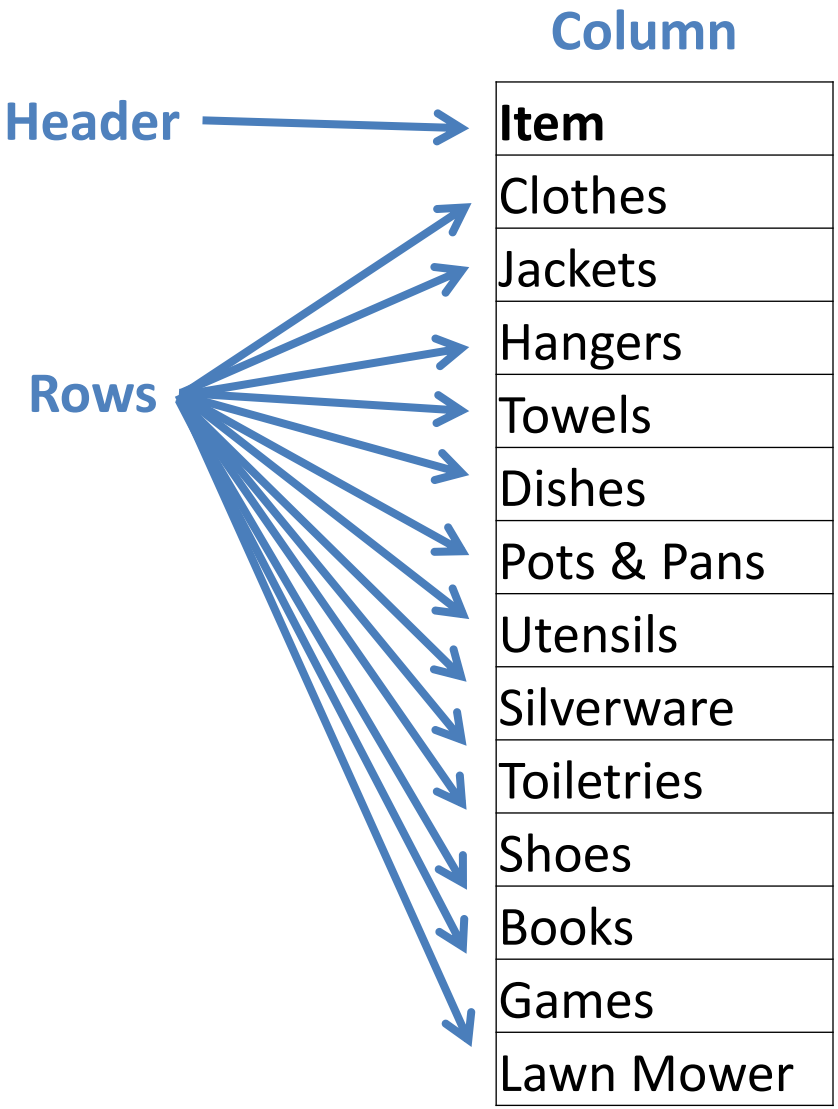
Moving Lists

- Rooms
  - Kitchen
  - Garage
  - Family Room
  - Dining Room
  - Bedroom
  - Bathroom
  - Closet
  - Patio
- Items
  - Clothes
  - Jackets
  - Hangers
  - Towels
  - Dishes
  - Pots & Pans
  - Utensils
  - Silverware
  - Toiletries
  - Shoes
  - Books
  - Games
  - Lawn Mower

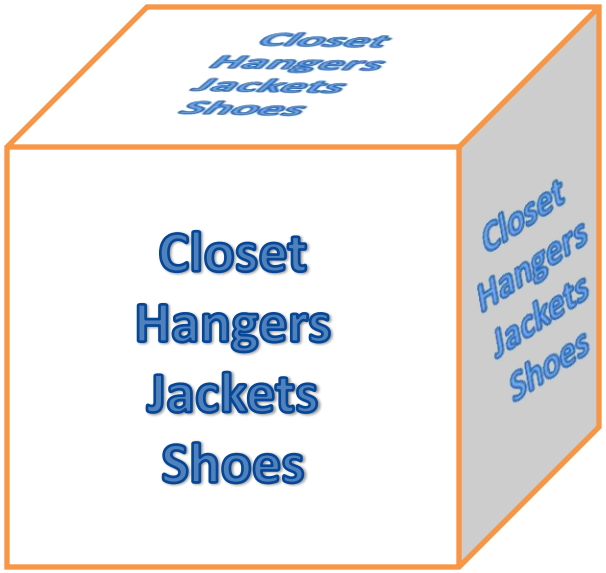
Database Table Rooms



Database Table Items



Box Contents



Database Table Boxes

Columns

Header

Rows

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware

Database Table Boxes

But wait! Box #3 also has Utensils!

Columns

Header

Rows

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware

Database Table Boxes

Columns

Header

Rows

Box #	Room	Item 1	Item 2	Item 3	Item 4
1	Closet	Hangers	Jackets	Shoes	
2	Bathroom	Towels	Toiletries		
3	Kitchen	Towels	Dishes	Silverware	Utensils

Database Table Boxes

Columns

Header

Rows

Box #	Room	Item 1	Item 2	Item 3	Item 4
1	Closet	Hangers	Jackets	Shoes	
2	Bathroom	Towels	Toiletries		
3	Kitchen	Towels	Dishes	Silverware	Utensils
4	Family Room	Books	Games		

Database Table Boxes

But wait! Box #4 has Books from two rooms!

Columns

Header

Rows

Box #	Room	Item 1	Item 2	Item 3
1	Closet	Hangers	Jackets	Shoes
2	Bathroom	Towels	Toiletries	
3	Kitchen	Towels	Dishes	Silverware
4	Family Room	Books	Games	

Database Table Boxes

Columns

Box #	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		

Database Table Boxes

But wait! Box #5 has multiple items from multiple rooms!

Box #	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5						

Miscellaneous

There Must be a Better Way!

Let’s talk about relationships.

- A Box can have:
  - One item in it (books)
  - Many items in it (hangers, jackets, shoes)
- Likewise a Box can have items from:
  - One room (kitchen)
  - Many rooms (family room, bedroom)

Relationships and Cardinality

- Relationship
  - How 2 or more things are connected
- Cardinality
  - The number of things in a set

Cardinality

- Cardinality examples
  - Zero things
  - One thing
  - Many things

How many is Many?

Relationships and Cardinality

- A Box can have:
  - One item in it (books)
  - Many items in it (hangers, jackets, shoes)
- Likewise a Box can have items from:
  - One room (kitchen)
  - Many rooms (family room, bedroom)

Large Items?

- Items that will not be boxed
  - Patio
    - Lawn Mower
  - Family Room
    - Couch
    - Chair
    - Tables
    - Etc.

Relationships and Cardinality

- An Item belongs to One Room
  - Clothes
- An Item may go into One Box, Many Boxes , or No Boxes
  - Silverware, Shoes, Lawn mower

Empty Boxes?

- Do we care about empty boxes?
  - Yes
    - We'll keep them and move them too.
  - No
    - We'll sell them back or trash them.

Zero Cardinality or Optionality

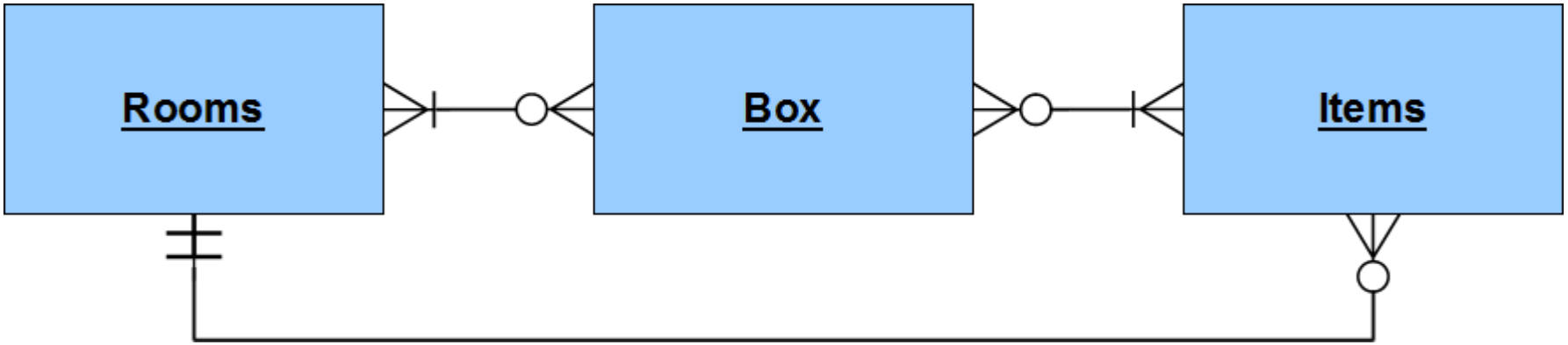
- Zero Cardinality (Optionality) examples
  - An Item may not go in One Box
    - Lawn mower, couch, table, etc.
  - A Room may not have anything that goes into a Box
  - A Box may not have anything in it.

Data Model Notation

Thing A has (is related to) One or More of Thing B
Thing B has (is related to) One and Only One Thing A

Notation	Zero or One	One and Only One	Zero or Many	One or Many
Crow's Feet (IDEF1X)				

Moving Data Model



Notation	Zero or One	One and Only One	Zero or Many	One or Many
Crow's Feet (IDEF1X)				

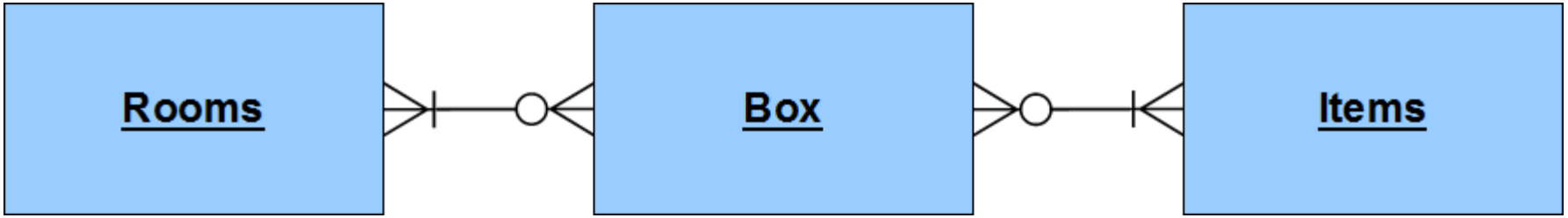
Database Table Boxes

Back to our problem

Box	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5						

Miscellaneous

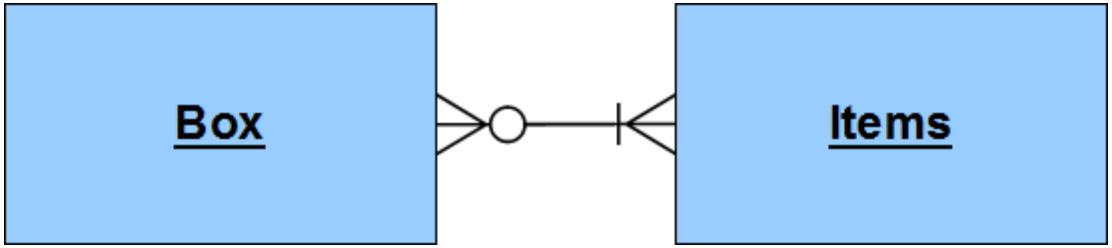
Database Table Boxes

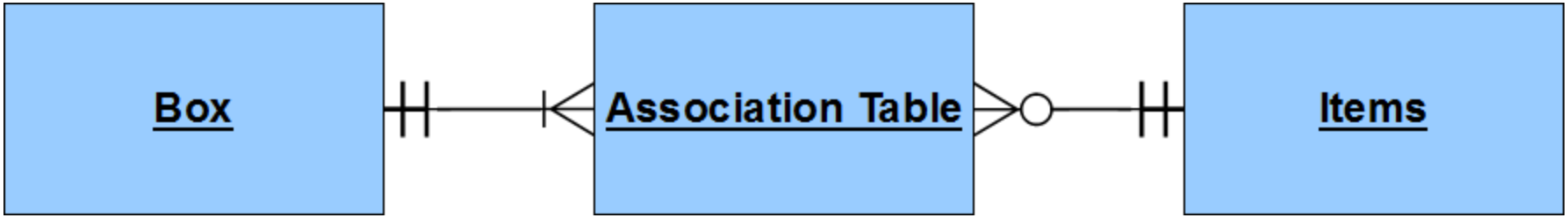


Box	Room 1	Room 2	Item 1	Item 2	Item 3	Item 4
1	Closet		Hangers	Jackets	Shoes	
2	Bathroom		Towels	Toiletries		
3	Kitchen		Towels	Dishes	Silverware	Utensils
4	Family Room	Bedroom	Books	Games		
5						

Miscellaneous

Association Table





Association Table Solution

Box #	Room	Item
1	Closet	Hangers
1	Closet	Jackets
1	Closet	Shoes
2	Barthroom	Towels
2	Bathroom	Toiletries
4	Family Room	Games
4	Bedroom	Books
5	Closet	hanger
5	Closet	Shoes
5	Garbage	Shoes
5	Kitchen	Books

But Wait… Typos!

Box #	Room	Item
1	Closet	Hangers
1	Closet	Jackets
1	Closet	Shoes
2	Barthroom	Towels
2	Bathroom	Toiletries
4	Family Room	Games
4	Bedroom	Books
5	Closet	hanger
5	Closet	Shoes
5	Garbage	Shoes
5	Kitchen	Books

Database Table Rooms - Updated

Columns

Header

Rows

Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

Database Table Items - Updated

Columns

Header

Rows

Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

Association Table - Updated

No Typos!

Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

But what's in the Box?

SQL Statements

- SELECT – Retrieve (SELECT) data from a table.
- INSERT – Add (INSERT) data into a table.
- DELETE – Remove (DELETE) data from a table.
- CREATE – Add (CREATE) a table, or other object.
- DROP – Remove (DROP) a table, or other object.
- EXEC – Run (EXECUTE) a Stored Procedure.

Anatomy of a SELECT Statement

```
SELECT
        <column,…>
FROM <table>
WHERE <condition>
ORDER BY <column,…>
```

Anatomy of a SELECT Statement

SELECT

<column,…>

*A list of columns or just use * for all columns*

SELECT

*

Get all columns

SELECT

Name, Zip, Phone

Get only 3 columns

Anatomy of a SELECT Statement

FROM <table>

Name of a table

FROM Rooms	FROM Items	FROM Boxes
Rooms table	*Items table*	*Boxes table*

Anatomy of a SELECT Statement

WHERE <condition>

Criteria or filter conditions

WHERE Item = ’Books’

Only get rows where the Item is Books

WHERE ItemID >= 4

Only get rows where the value of the Item ID is greater than or equal to

Anatomy of a SELECT Statement

ORDER BY <column,…>

Sort by a column or columns,
Ascending (ASC) is the default,
descending (DESC) is an option.

ORDER BY Item

Sort by the Item (ascending)

ORDER BY ItemID DESC

Sort by the ItemID descending

Time to Play!

Open sqlite3

C:\Sharky\Sqlite\sqlite3.exe

```
SQLite version 3.8.10.1 2015-05-09 12:14:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Sqlite housekeeping

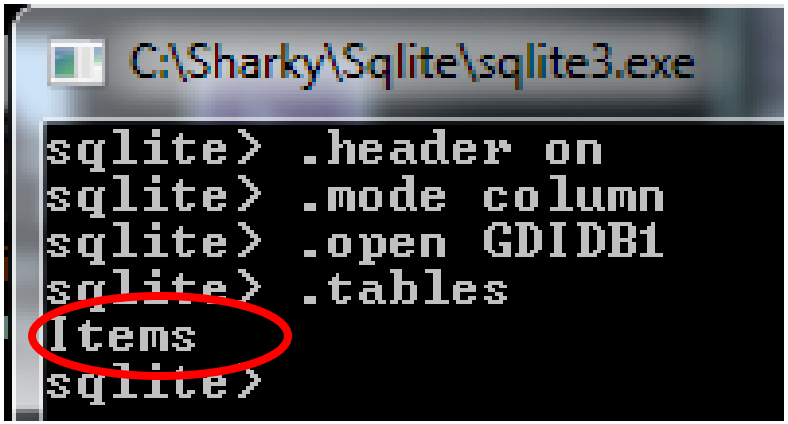
sqlite> **.header on**

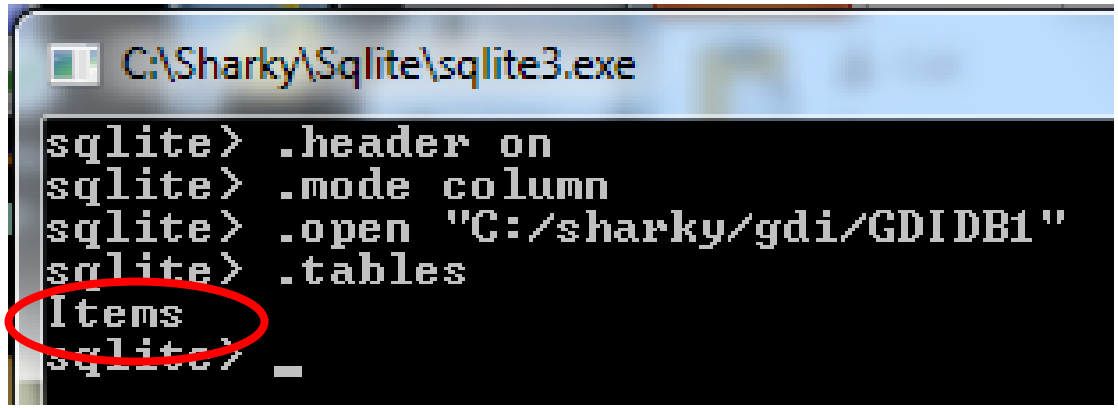
sqlite> **.mode column**

sqlite> **.open GDIDB1**

sqlite> **.tables**

To use full path - use '/' and must be in quotes!
.open "C:/sharky/gdi/GDIDB1"





Basic SELECT Statement

SELECT

<column,…> *(use * for all columns)*

FROM <table>

SQL!

Single line

sqlite> **SELECT * FROM Items;**

Multi-line

sqlite> **SELECT ***

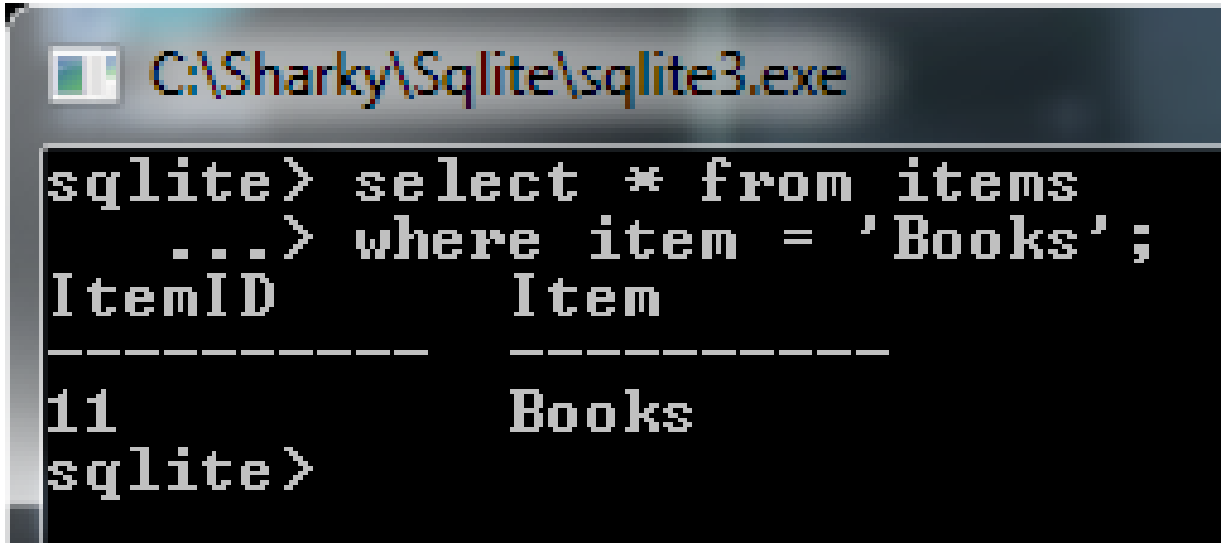
...> **FROM Items**

...> **;**

SQL Where Clause

sqlite> **SELECT * FROM Items**

...> **WHERE Item = 'books';**



SQL Where Clause

sqlite> **SELECT * FROM Items**

...> **WHERE ItemID >= 4;**

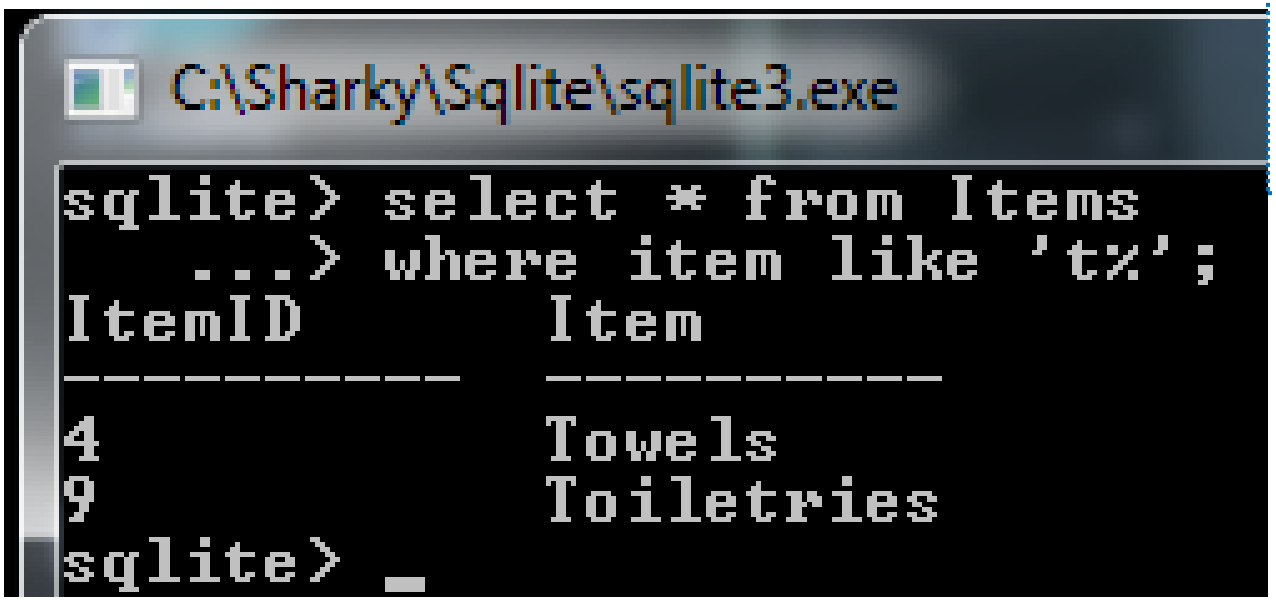
```
C:\Sharky\Sqlite\sqlite3.exe
sqlite> select * from Items
   ...> where itemID >= 4;
ItemID       Item
-----------  -----------
4            Towels
5            Dishes
6            Pots & Pan
7            Utensils
8            Silverware
9            Toiletries
10           Shoes
11           Books
12           Games
13           Lawn Mower
sqlite>
```

SQL Where Clause

sqlite> **SELECT * FROM Items**

...> **WHERE Item like 't%';**

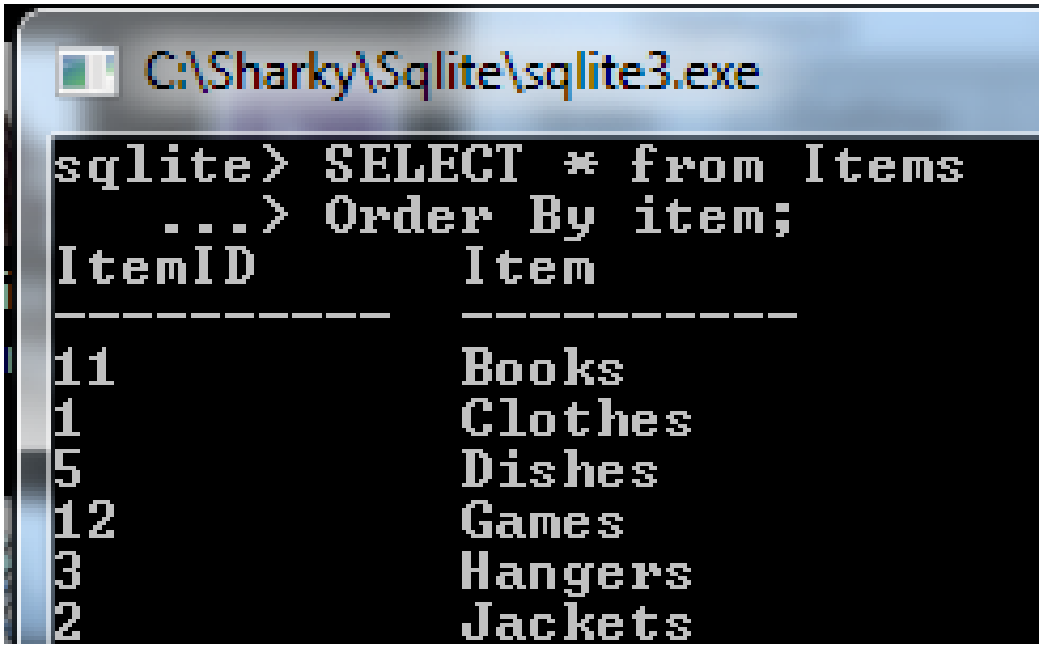
% is a wildcard
't%' – starts with 't'
'%t' – ends with 't'
'%t% - has a 't' somewhere



SQL Order By Clause

sqlite> **SELECT * FROM Items**

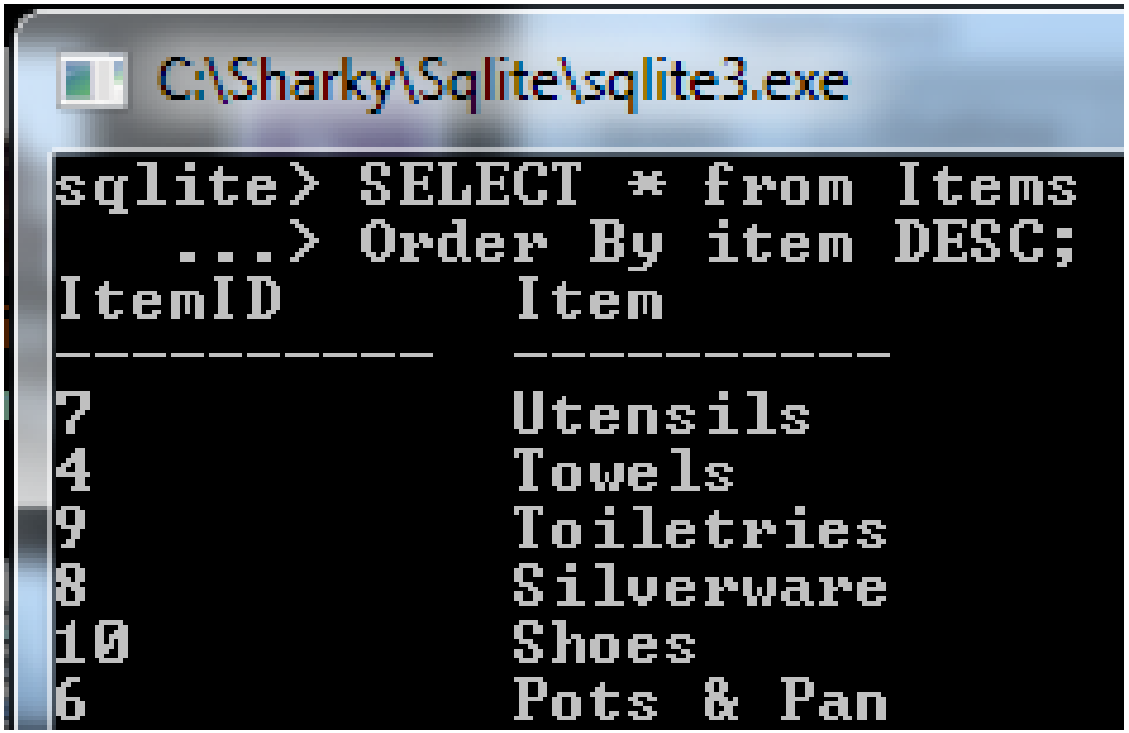
...> **Order By item;**



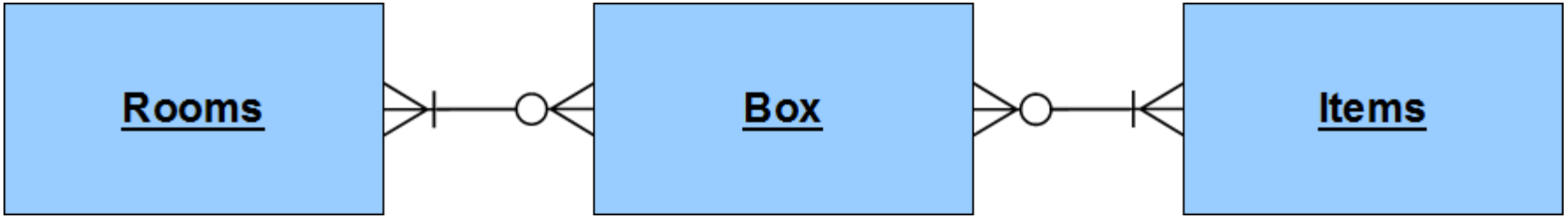
SQL Order By Clause

sqlite> **SELECT * FROM Items**

...> **Order By item DESC;**



Multiple Related Tables



Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

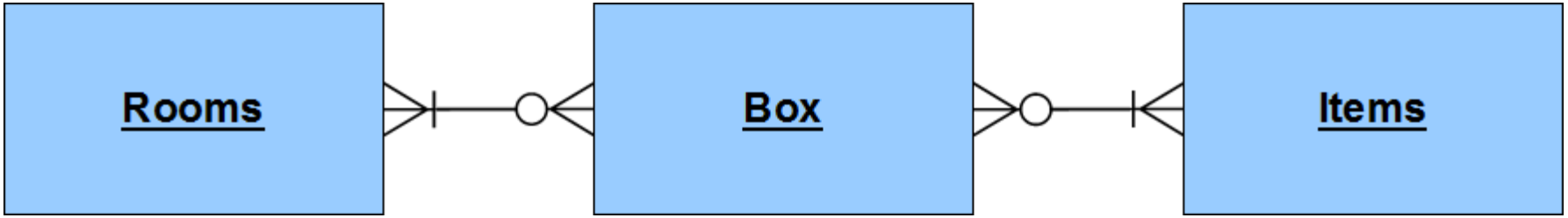
Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

Anatomy of a SELECT JOIN Statement

```
SELECT
        <column,…>
FROM <table1>
        JOIN <table2>
                ON <table1.column> = <table2.column>
        …
WHERE <condition>
ORDER BY <column,…>
```

Multiple Related Tables = SQL JOIN



Box	Room ID	Item ID
1	7	3
1	7	2
1	7	10
2	6	4
2	6	9
4	3	12
4	5	11
5	7	3
5	7	10
5	2	10
5	1	11

Room ID	Room
1	Kitchen
2	Garage
3	Family Room
4	Dining Room
5	Bedroom
6	Bathroom
7	Closet
8	Patio

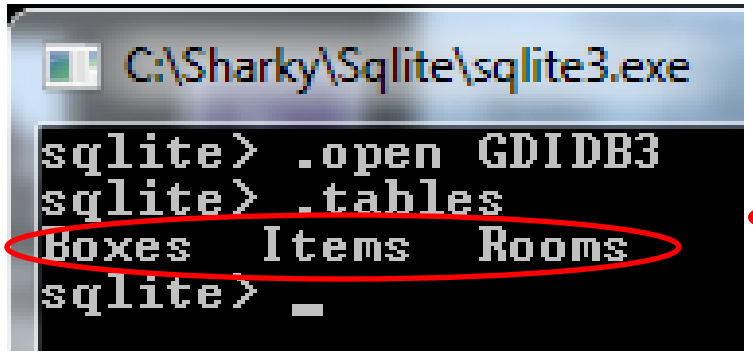
Item ID	Item
1	Clothes
2	Jackets
3	Hangers
4	Towels
5	Dishes
6	Pots & Pans
7	Utensils
8	Silverware
9	Toiletries
10	Shoes
11	Books
12	Games
13	Lawn Mower

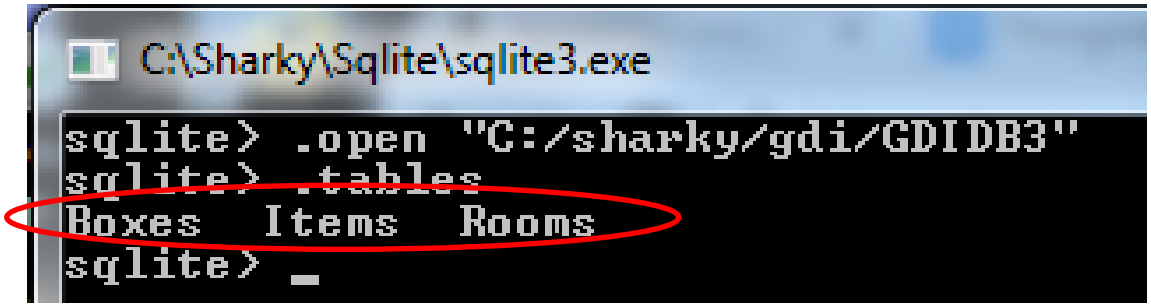
Sqlite housekeeping

sqlite> **.open <path>GDIDB3**

sqlite> **.tables**

To use full path - use '/' and must be in quotes!
.open "C:/sharky/gdi/GDIDB3"





SQL JOIN Clause

sqlite> **SELECT * FROM Boxes b**

...> **JOIN Rooms r ON b.roomID = r.roomID;**

C:\Sharky\Sqlite\sqlite3.exe

```
sqlite> SELECT * from Boxes b
   ...> JOIN Rooms r on b.roomID = r.roomID;
Box         RoomID      ItemID      RoomID      Room
----------  ----------  ----------  ----------  ----------
1           7           3           7           Closet
1           7           2           7           Closet
1           7           10          7           Closet
2           6           4           6           Bathroom
2           6           9           6           Bathroom
4           3           12          3           Family Roo
4           5           11          5           Bedroom
5           7           3           7           Closet
5           7           10          7           Closet
5           2           10          2           Garage
5           1           11          1           Kitchen
sqlite> _
```

SQL JOIN Clause

```
sqlite> SELECT * FROM Boxes b
   ...> JOIN Rooms r ON b.roomID = r.roomID
   ...> JOIN Items i ON b.itemID = i.itemID;
```

C:\Sharky\Sqlite\sqlite3.exe

```
sqlite> SELECT * from Boxes b
   ...> JOIN Rooms r on b.roomID = r.roomID
   ...> JOIN Items i on b.itemID = i.itemID;
Box         RoomID      ItemID      RoomID      Room        ItemID      Item
----------  ----------  ----------  ----------  ----------  ----------  --------
--
1           7           3           7           Closet      3           Hangers
1           7           2           7           Closet      2           Jackets
1           7           10          7           Closet      10          Shoes
2           6           4           6           Bathroom    4           Towels
2           6           9           6           Bathroom    9           Toiletri
es
```

SQL JOIN Clause

```
sqlite> SELECT Box, Room, Item
   ...> FROM Boxes b
   ...> JOIN Rooms r ON b.roomID = r.roomID
   ...> JOIN Items i ON b.itemID = i.itemID;
```

C:\Sharky\Sqlite\sqlite3.exe

```
sqlite> SELECT Box, Room, Item
   ...> FROM Boxes b
   ...> Join Rooms r on b.roomID = r.roomID
   ...> Join Items i on b.itemID = i.itemID;
Box         Room        Item
----------  ----------  ----------
1           Closet      Hangers
1           Closet      Jackets
1           Closet      Shoes
2           Bathroom    Towels
2           Bathroom    Toiletries
```

Anatomy of a CREATE TABLE Statement

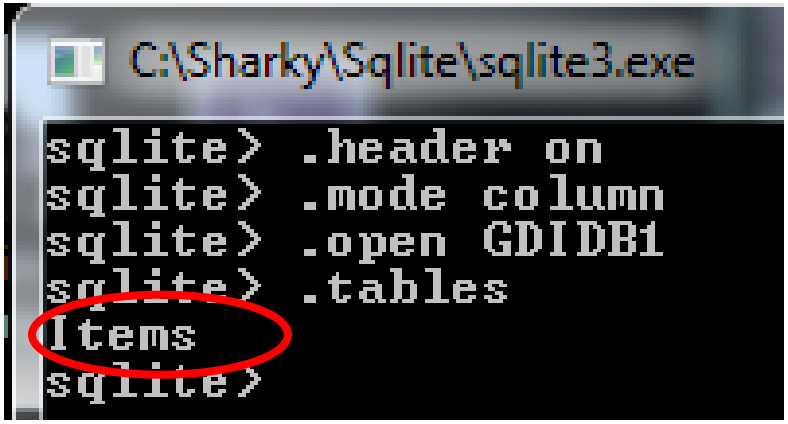
```
CREATE TABLE <table name> (
        <column name> <data type> <constraint>,
        …
);
```

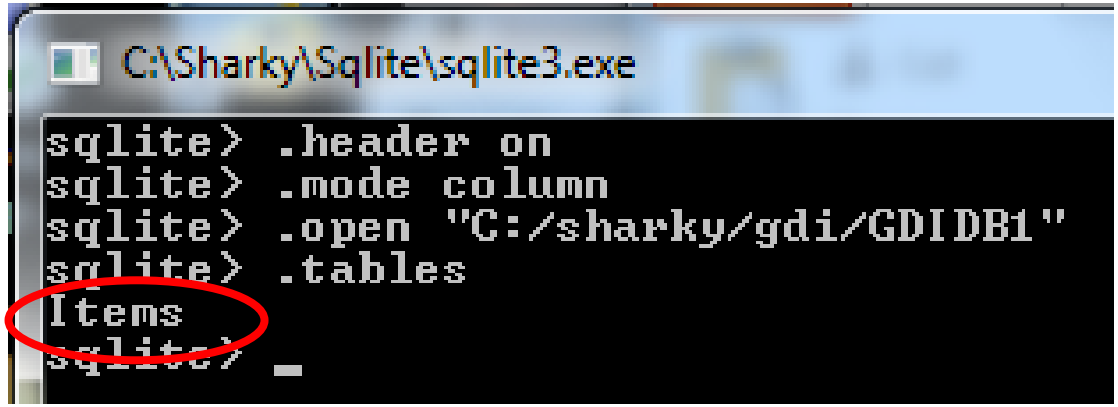
Sqlite housekeeping

sqlite> **.open GDIDB1**

sqlite> **.tables**

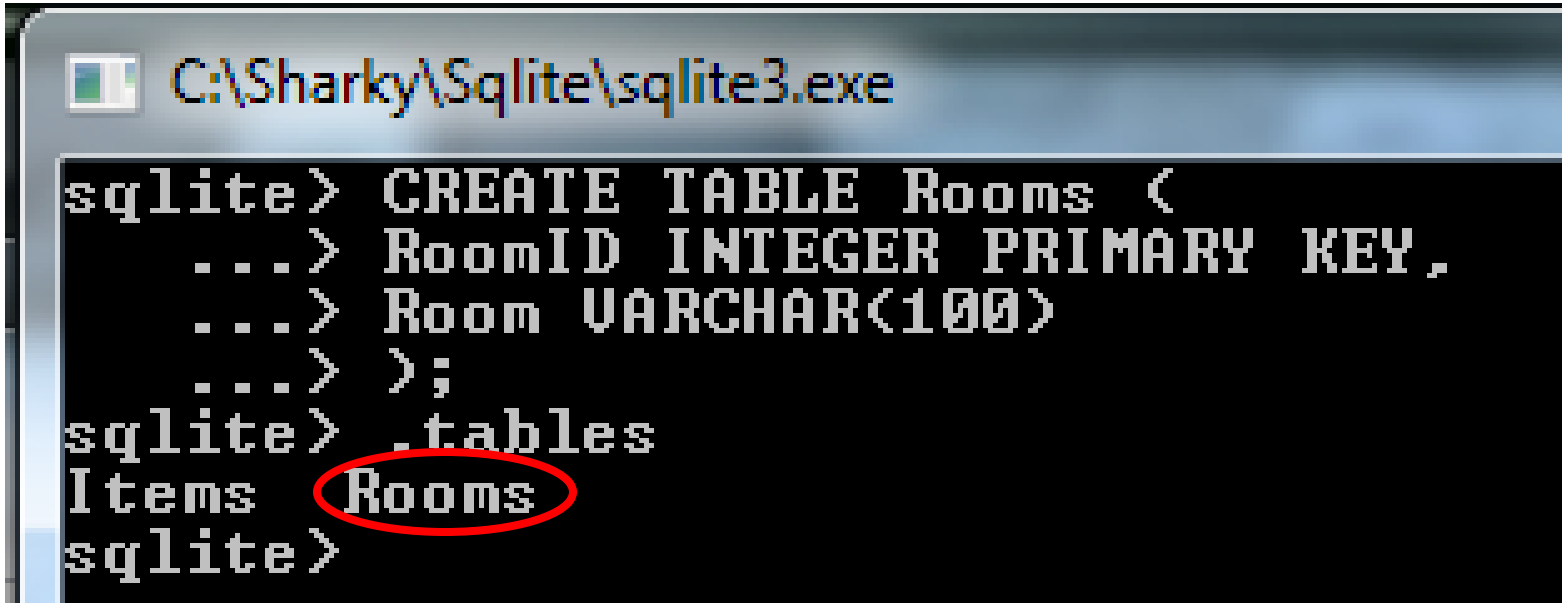
To use full path - use '/' and must be in quotes!
.open "C:/sharky/gdi/GDIDB1"





CREATE TABLE Statement

```
sqlite> CREATE TABLE Rooms (
   ...> RoomID INTEGER PRIMARY KEY,
   ...> Room VARCHAR(100)
   ...> );
```



Anatomy of an INSERT Statement

```
INSERT INTO <table name> (column,…)
        VALUES (values,…);
```

Data for a single row only

INSERT Statement

sqlite> **INSERT INTO Rooms (Room)**

...> **VALUES ('Kitchen');**

sqlite> **SELECT * FROM Rooms;**

SQL Statements

- ✓ SELECT – Retrieve (SELECT) data from a table.
- ✓ INSERT – Add (INSERT) data into a table.
- DELETE – Remove (DELETE) data from a table.
- ✓ CREATE – Add (CREATE) a table, or other object.
- DROP – Remove (DROP) a table, or other object.
- EXEC – Run (EXECUTE) a Stored Procedure.


LOONEY TUNES
"That's all Folks!"