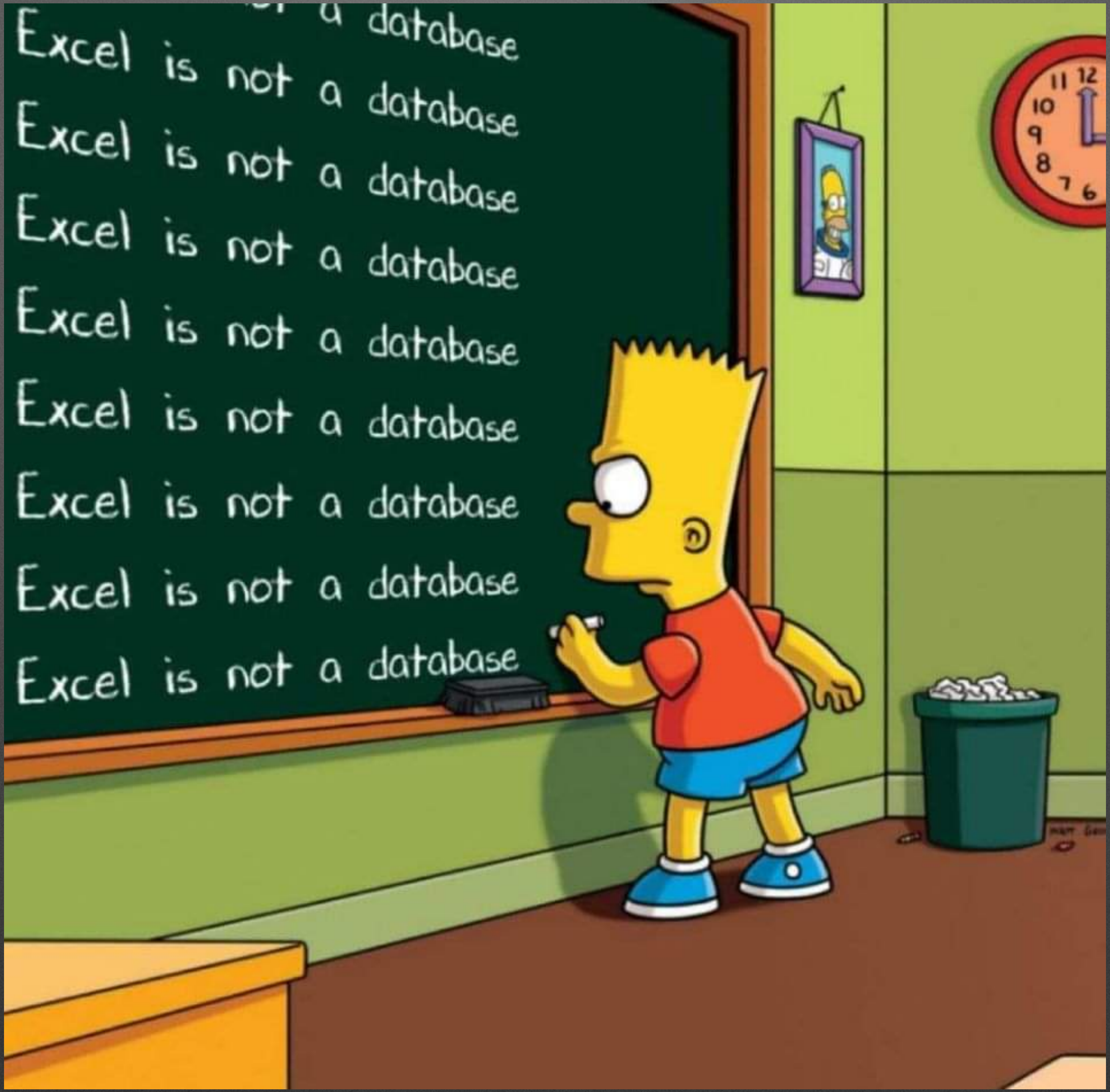
Welcome To

Practical and Awesome Database Design

T. L. Ford
www.Cattail.Nu

MS Access
SQLite3
MySQL
Tap Forms
SQL Server
Oracle

SQL



Except When It Is

Workshop Topics

- Databases are Fun!
- Relational Database Design
- Inner, Left, Right, Outer Joins

- WITH
  - Efficiency
  - Security
  - Functionality

Part 1
Database Tables

Sorting Information

- Data is grouped by topic into things called tables.

Class Notes

Study Groups

Grades

http://www.cattail.nu/database/01_sort_fields.html

Exercise
1

Tables

have a Definition

which is a list of Fields *each which has a*

Data Type

and may have properties:

max length
allow/prevent null, empty string
default value
max value
auto increment

and may have Indexes *for single/groups of* Fields

and store Data

in Records (rows/tuples)

each which is a Set of Values

matching the Field Definitions

Definition

tblPeople

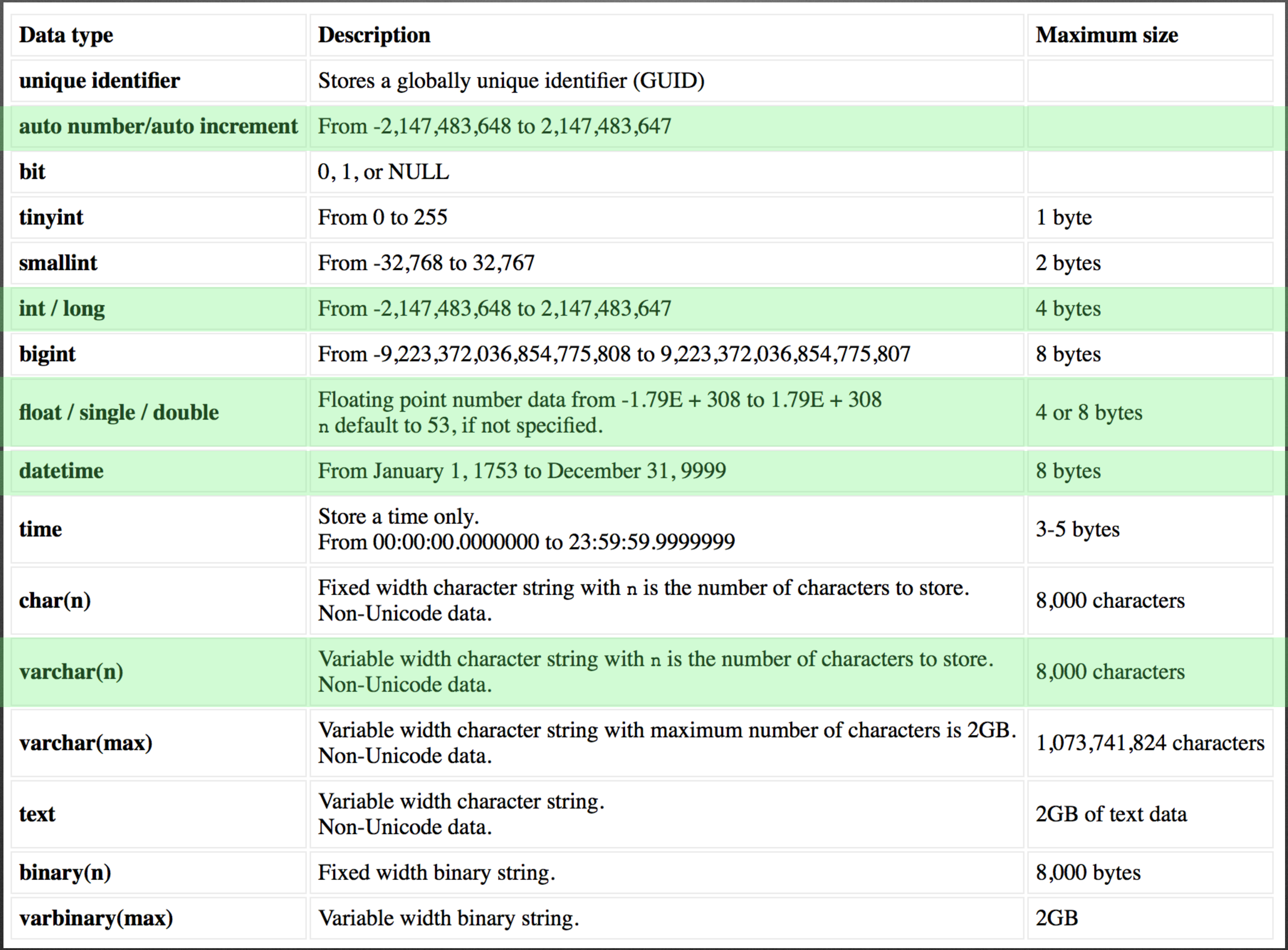
Field	Data Type
txtName	text / varchar
intAge	integer
dtmBirthday	date / time

schema

Data

txtName	intAge	dtmBirthday
"Milo"	40	#Oct 23 21:18:45 EDT 1980#
"Mario"	30	#Oct 23 21:18:45 EDT 1990#
"Lucas"	20	#Oct 23 21:18:45 EDT 2000#

Data Types



Data type	Description	Maximum size
unique identifier	Stores a globally unique identifier (GUID)	
auto number/auto increment	From -2,147,483,648 to 2,147,483,647	
bit	0, 1, or NULL	
tinyint	From 0 to 255	1 byte
smallint	From -32,768 to 32,767	2 bytes
int / long	From -2,147,483,648 to 2,147,483,647	4 bytes
bigint	From -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
float / single / double	Floating point number data from -1.79E + 308 to 1.79E + 308 `n` default to 53, if not specified.	4 or 8 bytes
datetime	From January 1, 1753 to December 31, 9999	8 bytes
time	Store a time only. From 00:00:00.0000000 to 23:59:59.9999999	3-5 bytes
char(n)	Fixed width character string with `n` is the number of characters to store. Non-Unicode data.	8,000 characters
varchar(n)	Variable width character string with `n` is the number of characters to store. Non-Unicode data.	8,000 characters
varchar(max)	Variable width character string with maximum number of characters is 2GB. Non-Unicode data.	1,073,741,824 characters
text	Variable width character string. Non-Unicode data.	2GB of text data
binary(n)	Fixed width binary string.	8,000 bytes
varbinary(max)	Variable width binary string.	2GB

Primary and Foreign Keys

Primary Key:
1 per table.
Uniquely identifies the row.
Usually auto-generated.
Don't let users enter it.

Foreign Key:
As many as needed.
Identifies a row in another table using that table's primary key value.
Usually automated or entered by a pick-list.

Naming Conventions (1 of 2)
Good Practices

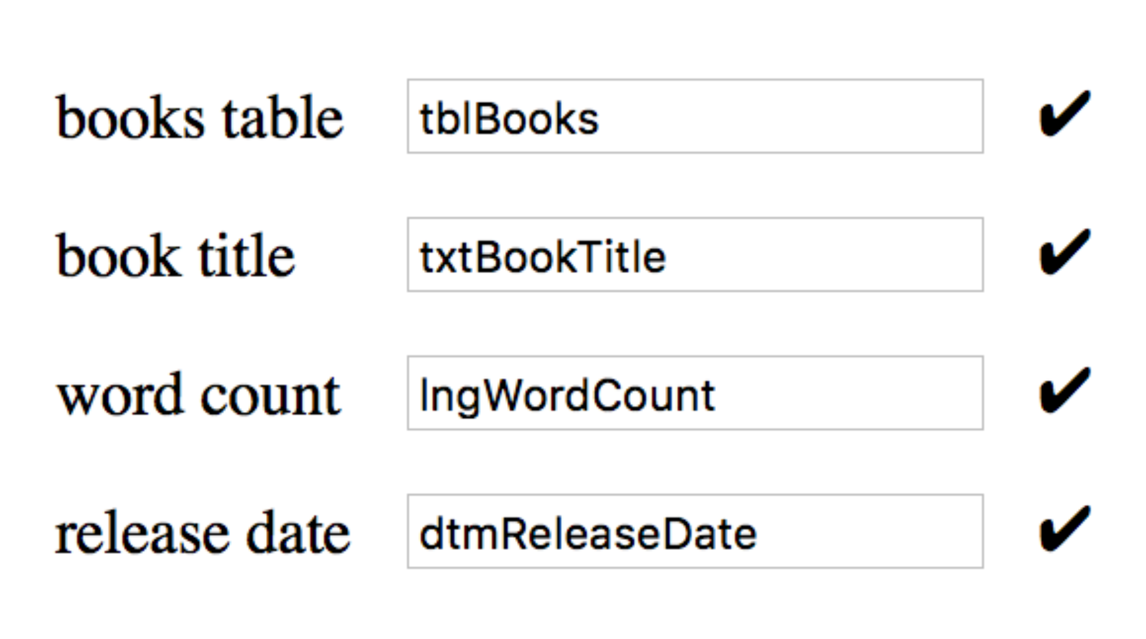
- Hungarian Notation, Camel Case
- Avoids keywords by default
- Avoids spaces by default
- No symbol characters (i.e. no fields you have to escape when writing sql [this is a bad name]
- Easily know if you have to escape in SQL where clauses.
- WHERE txtField='quoted' AND lngField=3 AND dtsField < #10/9/2020#

Naming Conventions (2 of 2)
Hungarian Notation

- tbl for Table
- dtm for Date Time
- bln for Boolean
- txt for Char/Text (fixed length)
- mem for Memo/Varchar (variable length)
- int for Integer (0-255)
- lng for Long (0-255+)
- sng for Single (floating point)
- dbl for Double (big floating point)

http://www.cattail.nu/database/02_naming.html

Exercise
2



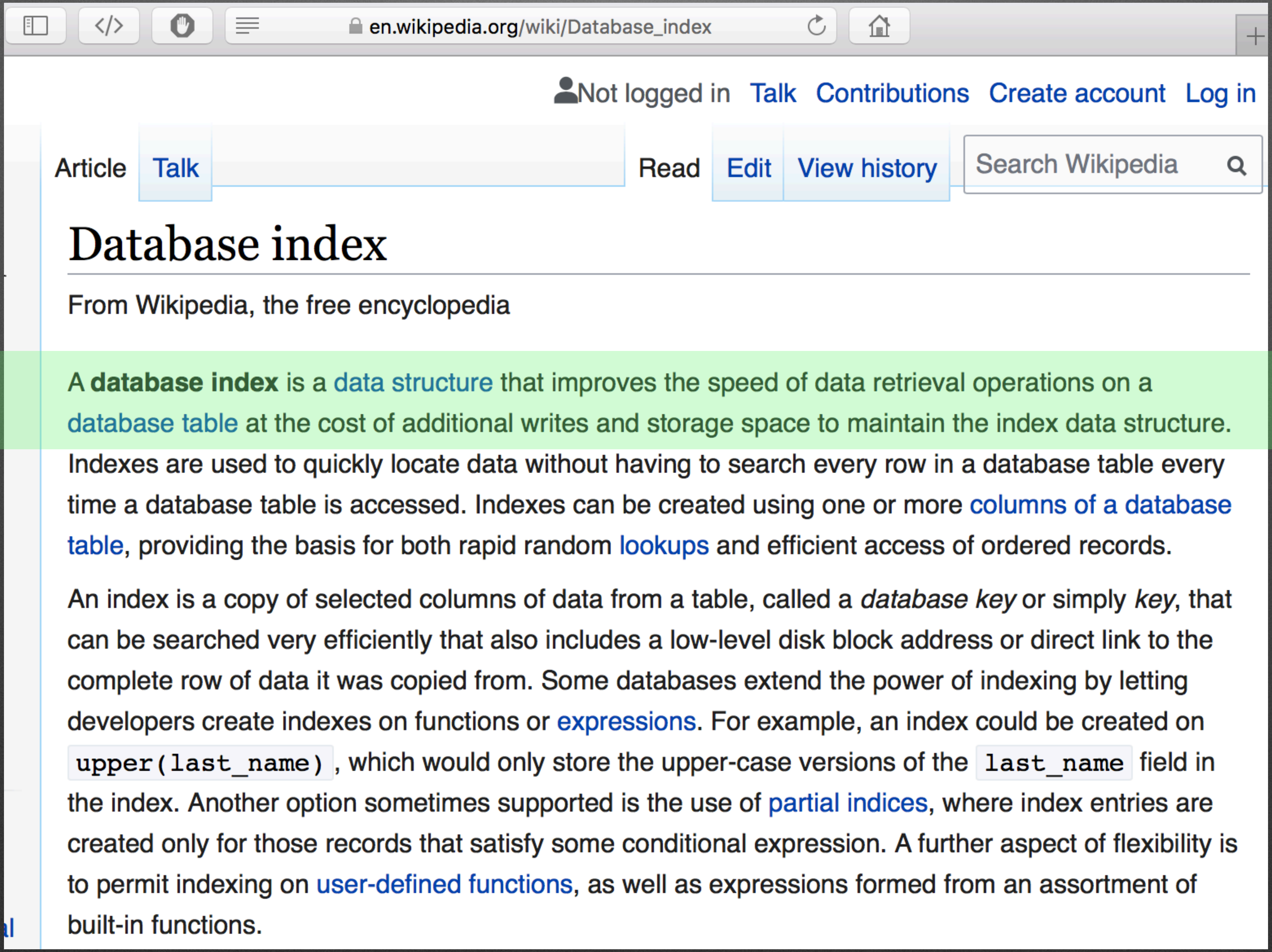
books table	tblBooks	✔
book title	txtBookTitle	✔
word count	lngWordCount	✔
release date	dtmReleaseDate	✔

Sorting Fields in Table Definition

- Primary key
- Foreign keys
- Fixed length, non-null (bln, int, lng, sng, dbl, txt)
- Fixed length, null
- Variable length, non-null
- Variable length, null

Stores and retrieves data most efficiently.
SQL will let you order fields for use/display user-friendly.

Indexes (1 of 2)



Database index

From Wikipedia, the free encyclopedia

A **database index** is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

An index is a copy of selected columns of data from a table, called a *database key* or simply *key*, that can be searched very efficiently that also includes a low-level disk block address or direct link to the complete row of data it was copied from. Some databases extend the power of indexing by letting developers create indexes on functions or expressions. For example, an index could be created on `upper(last_name)`, which would only store the upper-case versions of the `last_name` field in the index. Another option sometimes supported is the use of partial indices, where index entries are created only for those records that satisfy some conditional expression. A further aspect of flexibility is to permit indexing on user-defined functions, as well as expressions formed from an assortment of built-in functions.

https://en.wikipedia.org/wiki/Database_index

Indexes (2 of 2)

Index these:

- Primary Keys
- Foreign Keys
- Often used search / sort fields

More indexes: Slow on add/update.
Less indexes: Slow if needed for search/sort.

Part 2
Table Relationships

Relationships (1 of 2)

- Because tables store information about a single topic, we need a way to link tables to get information from multiple tables.

- Link tables on keys (primary key / foreign key)

- *Normalization* - don't duplicate data unnecessarily.

Relationships (2 of 2)

As Designed

Enforced Referential Integrity

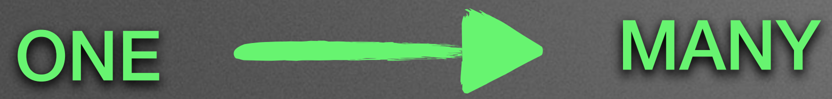
ONE

MANY

tblAuthors → tblBooks

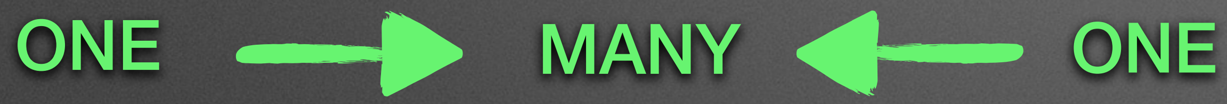
One author can write many books.
** excluding anthologies*

Common Designs



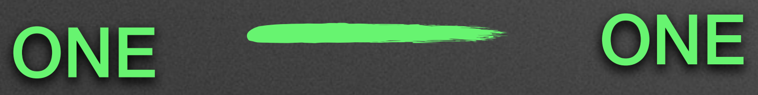
ONE → MANY

Lookup tables
Simple relationships



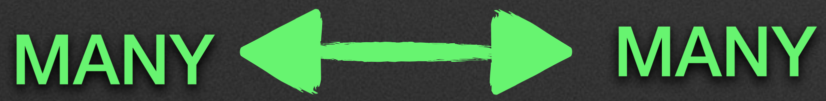
ONE → MANY ← ONE

Combine Tables as Unique Entity
Actual Items to Lookup Tables



ONE — ONE

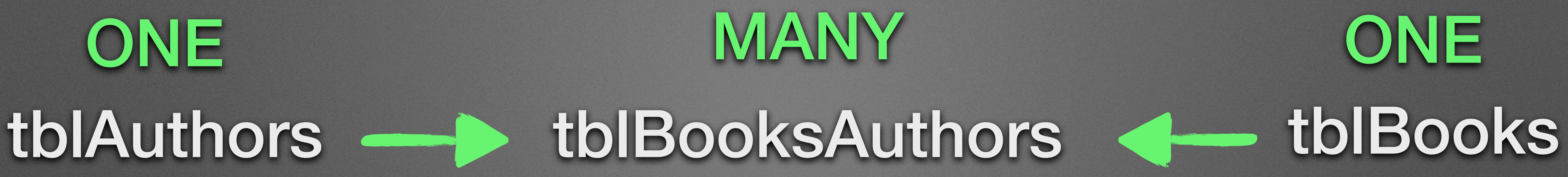
Useful for tables with
a lot of fields
OR
to optimize queries for large BLOB data



MANY ↔ MANY

Don't use this. (use one -> many <- one)

Example



ONE

tblAuthors

MANY

tblBooksAuthors

ONE

tblBooks

One book can have many authors.
Supports anthologies.

As Stored

tblAuthorsID	txtAuthorName
1	"T. L. Ford"
2	"Lois McMaster Bujold"
3	"Charles Stross"
4	"Dan Brown"

tblAuthorsID	tblBooksID
1	1
1	2
2	2
1	3
2	3
3	3

tblBooksID	txtBookName
1	"Colony One"
2	"Made Up Anthology"
3	"Another Made Up Anthology"

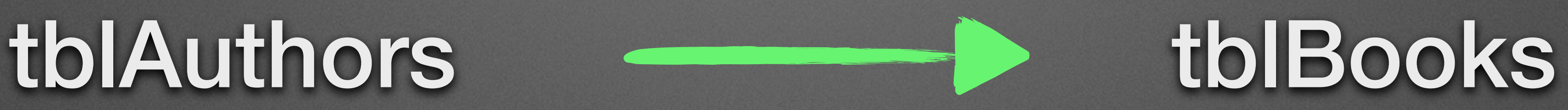
As Retrieved via SQL

txtAuthorName	tblAuthorsID	tblBooksID	txtBookName
"T. L. Ford"	1	1	"Colony One"
"T. L. Ford"	1	2	"Made Up Anthology"
"Lois McMaster Bujold"	2	2	"Made Up Anthology"
"T. L. Ford"	1	3	"Another Made Up Anthology"
"Lois McMaster Bujold"	2	3	"Another Made Up Anthology"
"Charles Stross"	3	3	"Another Made Up Anthology"

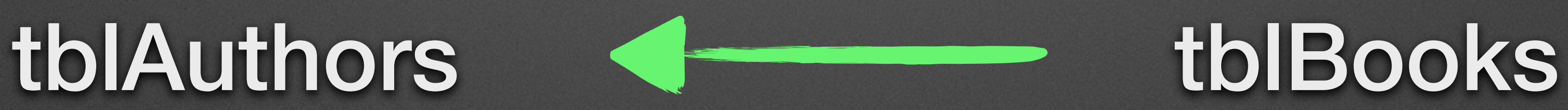
Design Arrows are Irrelevant to SQL

Beware the Arrows

As Used in SQL


tblAuthors → tblBooks

Get me the books this author has written.


tblAuthors ← tblBooks

Get me the author(s) that wrote this/these book(s).

SQL

```
SELECT tblA.*, tblB.*
 FROM
    tblA INNER JOIN tblB ON tblA.tblAID = tblB.tblAID
```

INNER JOIN

only retrieve data where both have values and both values match

```
    tblA LEFT JOIN tblB ON tblA.tblAID = tblB.tblAID
```

LEFT JOIN

retrieve all data on the first table (left) AND
nothing if no match OR data from the second (right) side if matched

```
    tblA RIGHT JOIN tblB ON tblA.tblAID = tblB.tblAID
```

RIGHT JOIN

retrieve all data on the second table (right) AND
nothing if no match OR data from the first (left) side if matched

```
    tblA OUTER JOIN tblB ON tblA.tblAID = tblB.tblAID
```

OUTER JOIN

retrieve all data from both tables AND
nothing if no match OR data if matched

SQL

```
SELECT tblA.*, tblB.*
 FROM

    tblA LEFT JOIN tblB ON tblA.tblAID = tblB.tblAID
```

LEFT JOIN

is the same as

```
    tblB RIGHT JOIN tblA ON tblB.tblAID = tblA.tblAID
```

RIGHT JOIN

Part 3
Wisdom

Personal Conventions (1 of 2)

CamelCase naming

Table tblTableName
Key field same as table name with ID
Primary key always listed first

tblTableName

tblTableNameID (autonumber)

Related tables include master table in name
Foreign keys listed after primary key
Foreign keys have same name as their associated primary key

tblTableNameAnother

tblTableNameAnotherID (autonumber)

tblTableNameID (foreign key)

Personal Conventions (2 of 2)

Fields added to every table

dtmCreatedOn

txtCreatedBy

dtmModifiedOn

txtModifiedBy

Field Lookup List Tables

Field name matches destination field

txtLookupField

txtLookupField (text)

Examples
and
Discussion

About Me

After a few wonderful decades programming databases, computer security, and doing general IT support, I left the windowless office to write, make art, and enjoy the magnificence of the outdoors. I program things for amusement now, including bots.

See my website: http://www.Cattail.Nu

Please consider reading or sharing one of my books: