‘rules’ of table design...

know your data and what you are going to do with them

- What kind of data do you have?
- What kind of data will you add later?
- What queries are you going to need?
- What queries are you going to run often?
- Who else will use it?

do not feel limited by the format data come in

do not make relational tables that you do not need

...‘rules’ of table design

make column values independent from all other columns

- eliminate as much duplication as possible
- eliminate as much polymorphism as possible
- eliminate as many NULL values as possible

aim for ‘two’ column tables: ID and value(s)

- make column values atomic

break the ‘rules’

- worst case: the database is slow and difficult to maintain...
- you can always improve it later

MariaDB: numbers...

DEC: numbers with a decimal place (with or without signs)

FLOAT, DOUBLE (SQL standard)

differ in the number of bytes (4, 8) and significant digits (6–7, 15–16)

DECIMAL (not SQL standard, but more useful)

‘fixed’ precision floating point number

INT: integer numbers (with or without signs)

5 different sizes (pick the smallest useful)

differ in the number of bytes (1, 2, 3, 4, 8)

AUTO_INCREMENT

MariaDB: ...numbers

			default (signed)		UNSIGNED	
type	bits	bytes	min	max	min	max
TINYINT	8	1	-128	127	0	255
SMALLINT	16	2	-32,768	32,767	0	65,535
MEDIUMINT	24	3	-8,388,608	8,388,607	0	16,777,215
INT	32	4	-2,147,483,648	2,147,483,647	0	4,294,967,295
BIGINT	64	8	9.2×10^{-18}	9.2×10^{18}	0	1.8×10^{19}

MariaDB: binary

BIT (not SQL standard)

- a fixed–length array of bits
- does not always work with connectors/libraries
- use HEX, OCT, or BIN to see the values in the database

BINARY

- a fixed–length array of bytes
- data are displayed as CHAR

VARBINARY

- a variable–length array of bytes
- data are displayed as CHAR

MariaDB: dates

useful for record keeping, date calculations

‘0’ values allowed for incomplete records

DATETIME (YYYY-MM-DD HH:MM:SS)

DATE (YYYY-MM-DD)

TIMESTAMP (YYYY-MM-DD HH:MM:SS)

automatically updates

TIME (HH:MM:SS)

YEAR (YYYY)

MariaDB: text

type of text specified by CREATE TABLE

- CHARSET=UTF8 | CHARSET=ascii

determine (or guess) how much you will need

CHAR

- fixed 1–65,535 (set by CREATE TABLE)
- takes more memory

VARCHAR

- variable 1–65,535 (max set by CREATE TABLE)
- takes less memory

MariaDB: ‘documents’

best used in ‘dedicated’ tables for greater speed

not (usually) fully indexed (use hashes to force unique)

BLOB (TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB)

- good for storing binary files (e.g. images)
- often not practical or portable

TEXT (TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT)

- good for storing long text strings
- sorting uses only the first X characters

MariaDB: predefined...

know (or guess) all possible values for a field

ENUM (not SQL standard)

- up to 65,535 values
- 1 or 2 bytes for storage
- can be NULL
- fully indexed without separate index
- sorts based on order of values in CREATE TABLE
- no polymorphism
- best to pretend ENUM is text when writing queries

MariaDB: ...predefined

SET (not SQL standard)

- up to 64 values
- 1, 2, 3, 4, or 8 bytes for storage
- polymorphism
- can be NULL
- sorts based on order of values in CREATE TABLE
- best to pretend SET it is text when writing queries

MariaDB: geospatial

Well–Known Text (WKT) Format (also in binary)

POINT	X Y
LINESTRING	X1 Y1, X2 Y2, …
POLYGON	X1 Y1, X2 Y2, X3 Y3, …

...and many more

great for queries (e.g. is point X inside polygon Y)

MariaDB: PRIMARY KEY

must be UNIQUE and NOT NULL

usually the ‘ID’ field

synthetic (surrogate) keys: arbitrary integer numbers

- faster, fewer problems over time
- usually set to AUTO_INCREMENT

natural keys: derived from the data (one or more columns)

- may cause problems when data are modified, or unforeseen duplicates entered

influences how data are stored on disk (for most storage engines)

- OPTIMIZE TABLE

MariaDB: FOREIGN KEY

references a column in another table

- usually the primary key
- must have the same datatype (usually an INT)
  - not all data types are supported (e.g. LONGTEXT)

makes relations explicit to the storage engine

- ON UPDATE CASCADE | SET NULL | DELETE

used by the storage engine to optimize queries

queries can be less wordy (i.e. relations can be left unwritten)

MariaDB: KEY

also called INDEX

used for faster searches of non–ID fields

should have few (no) duplicate values

UNIQUE KEY can be used to constrain the data

do not use KEY on fields you will never search or sort

precompute KEY for a table rather than for each search

EXPLAIN => possible_keys, key

MariaDB: …keys

SPATIAL INDEX

works like KEY, but for geospatial data

MariaDB: storage engines…

the ‘real’ database program

MyISAM (original, deprecated) and Aria (the new MyISAM)

- fast and efficient
- not designed for data consistency
  - Aria has (optional) transaction support
  - no automated foreign key support
- table–level locking (good for read–heavy applications)
- cannot be used with Galera Cluster

MariaDB: …storage engines…

InnoDB

- slower, but consistent (*c*. 4× slower than Aria)
- designed for data consistency (ACID)
  - transaction support
  - automated foreign key support
- row–level locking (good for read/write performance)
- can be used with Galera Cluster