Who Cares About Relational Database Design?

Louis Davidson

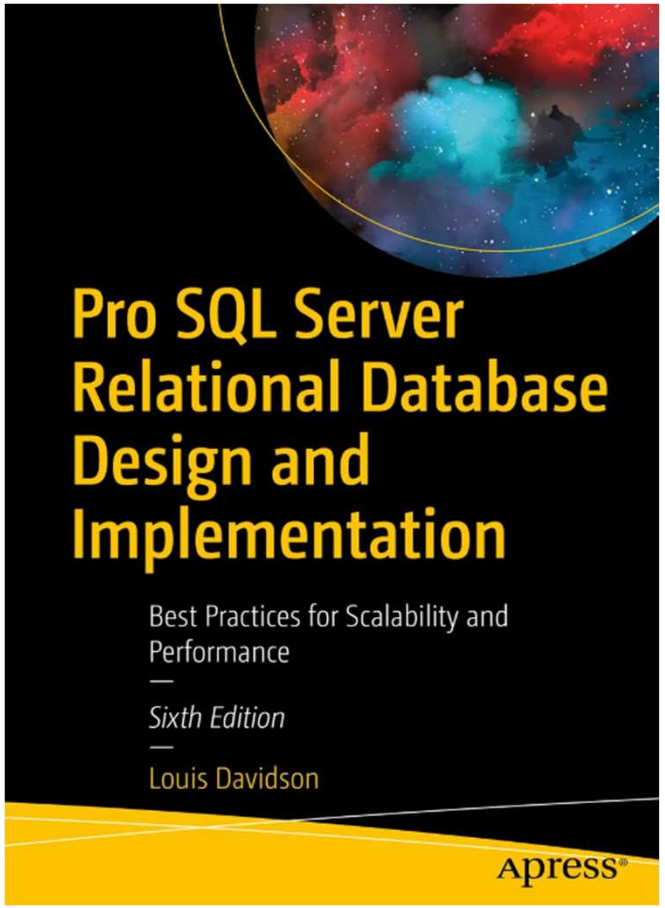
Editor of 

I do, but who am I?

- Database programmer/architect > 20 years
- Microsoft MVP 19 Times
- Simple-Talk.com Editor
- Written 6 books on database design
  - They were technically all versions of the same book.
  - They at least had slightly different titles each time
  - Seventh Edition? Who knows

- Super Brief Contact Info: DRSQL
  - Twitter, Website (.org), Email (@hotmail.com)

- Article writing interest?
  - editor@simple-talk.com



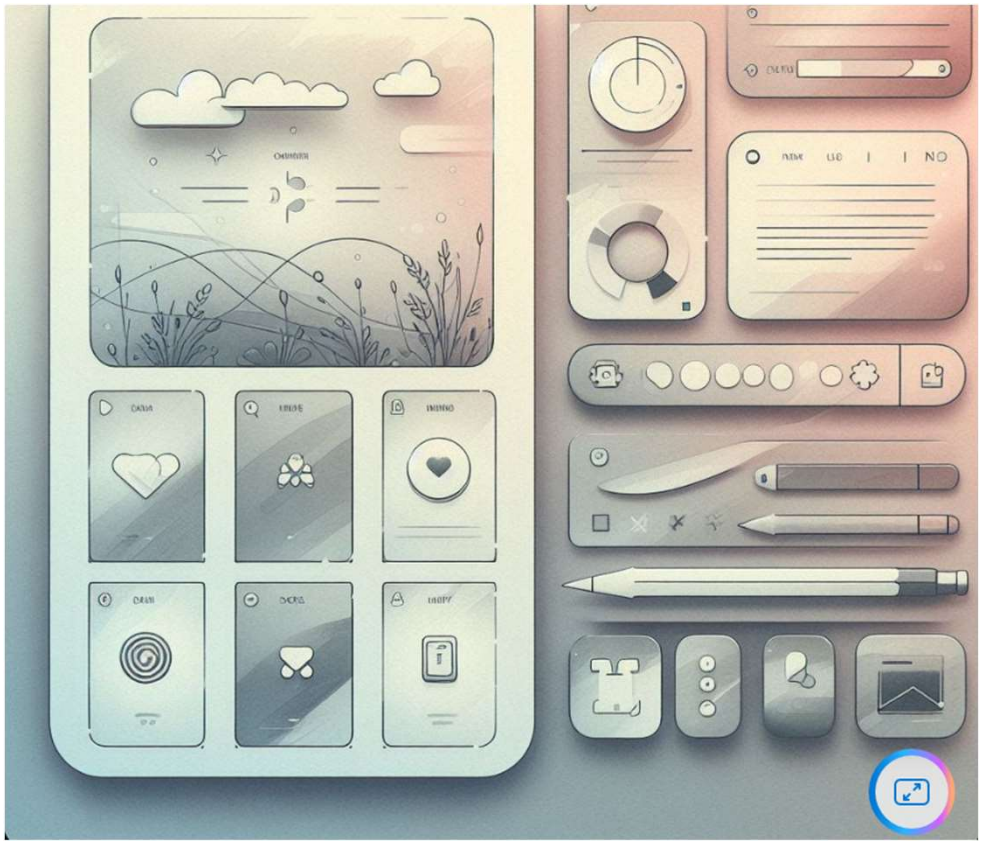
Well? Who cares?

What about people who create application databases?

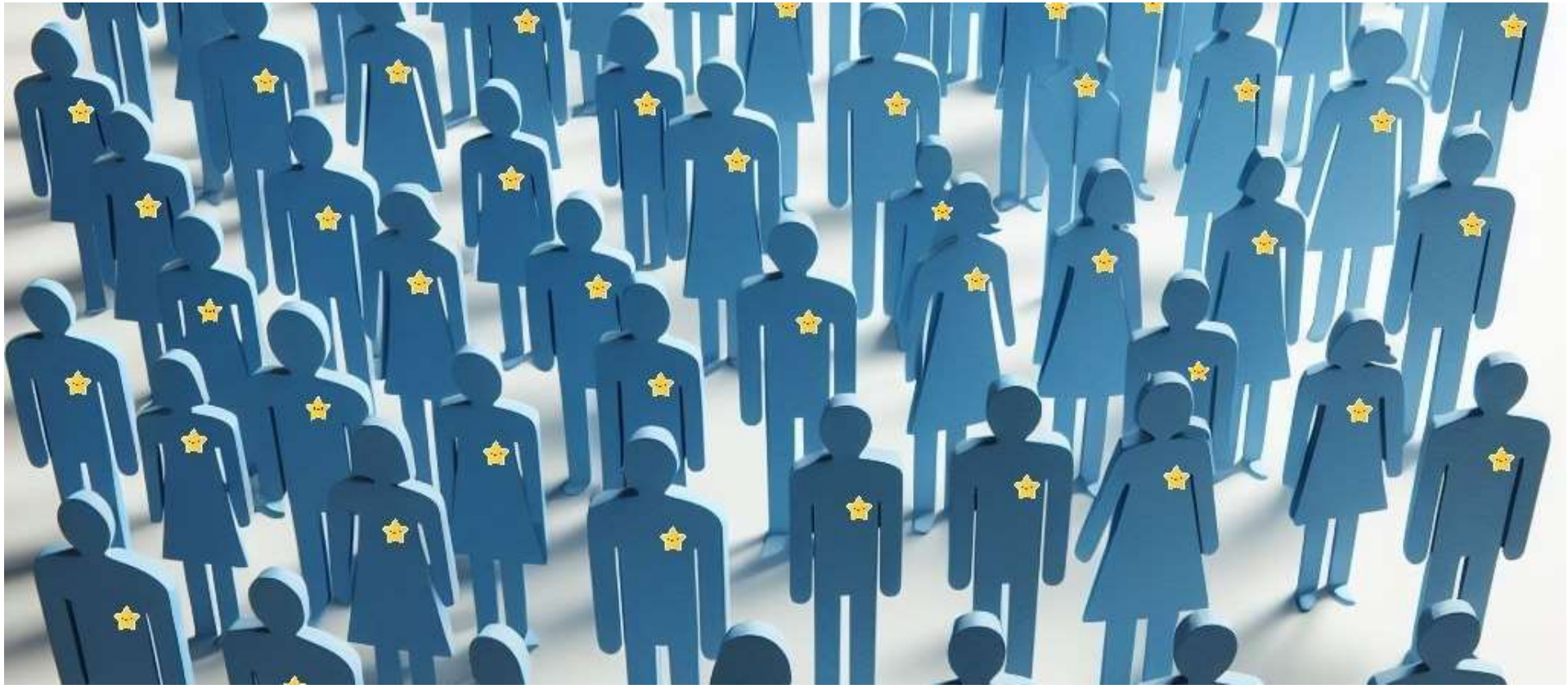
Bad interfaces do suck

We all like a nice UI

- We just want it to do what we told it to do…
- And more importantly store the data we expect it to.
- In ONE TRY.



So, who is affected by poorly designed databases?



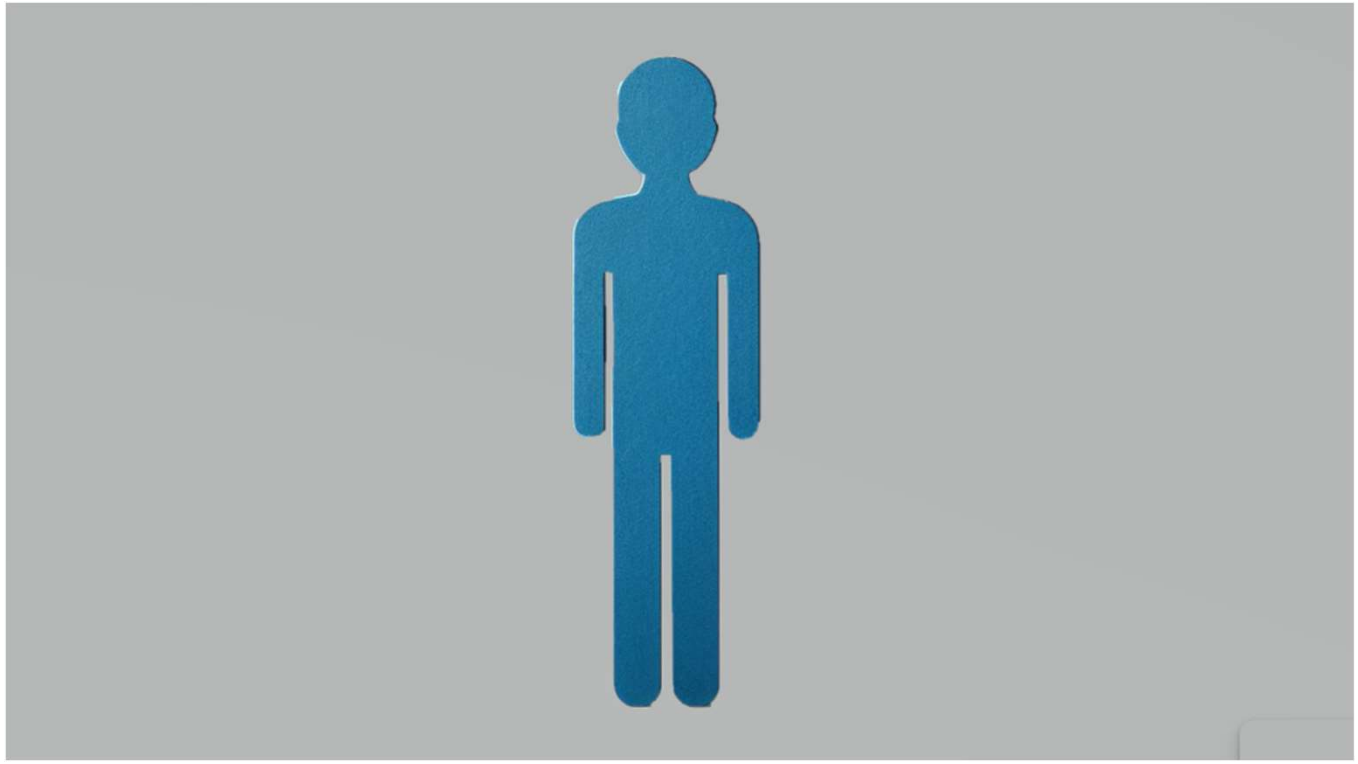
In short

The Big Why

So like I said… all of us should care about db design when we are building a new solution that stores data..

Or altering an existing app.

Or adding new features to an existing app.



So what can I do? I am just one person!

Start caring early…

- **Think**: What problem are we really trying to solve?
- **Communicate**: Get agreement from all involved that you are solving the right problem
  - Users
  - Management
    - Project
    - Client
  - Programmers
  - Anyone else who might disagree with you and cause your design harm later. (other than your significant other, unless you work together.)
- **Document**: Write it down so it cannot be disputed in a few weeks (or years)

- The common term for what you need is Requirements

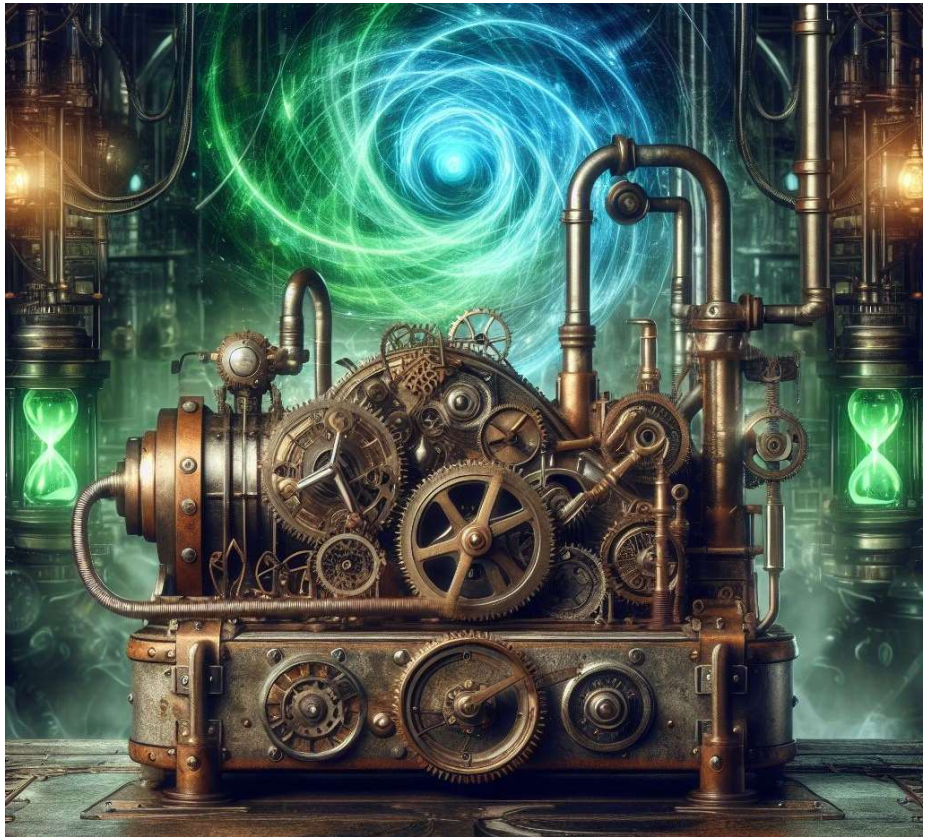
Requirements are Family Vacation Plans

If everyone decided on Lake Eerie (instead of Erie), then everyone is to blame

Design goal

- A database that works well, and meets your needs, and doesn't lie to you more than you expect it to…
- And you go from there…
  - Chose a data store that meets your needs
  - All your needs…

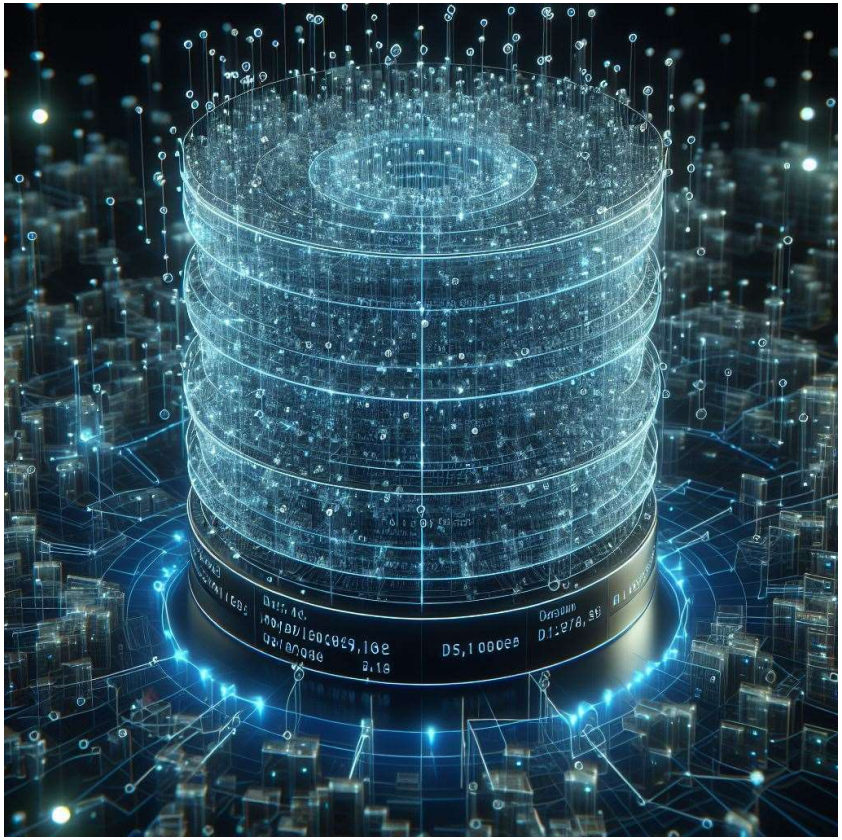
Skipping ahead in time



- Pretend you have chosen a database management system…
  - Which means you have requirements, a budget, meetings scheduled, your whiteboards covered in ink and marked “do not erase” on every corner, lunch ordered in for the next month, and you basically have your ducks all sitting in a row ready to get quacking

- *And I don’t care which one*
- *And I don’t care why you chose it*
  - *I do hope you can change your mind if necessary.*

Now what?



Controversial opinion time…

- It is time to iteratively design…
- It doesn't matter if your system requires structured, semi-structured, or unstructured data
- You want to know the structure…
  - Even if the structure is strictly "no structure"
  - I can't imagine a true.. Zero structure database

Structure because… you have to use the data

And trust the data…

Normal Forms/Normalization

- A process to understand the shape of your data
  - Probably to work with a relational engine
  - And even if not, to help you understand how your data is used
- Specified as a series of forms that signify compliance
- Practically speaking:
  - A set of standards to work towards along the way
  - Somewhat done instinctively
  - Important to understand even if you cannot achieve perfection

  - Utter nonsense if you do not understand the requirements!

Why I am talking about normalization?

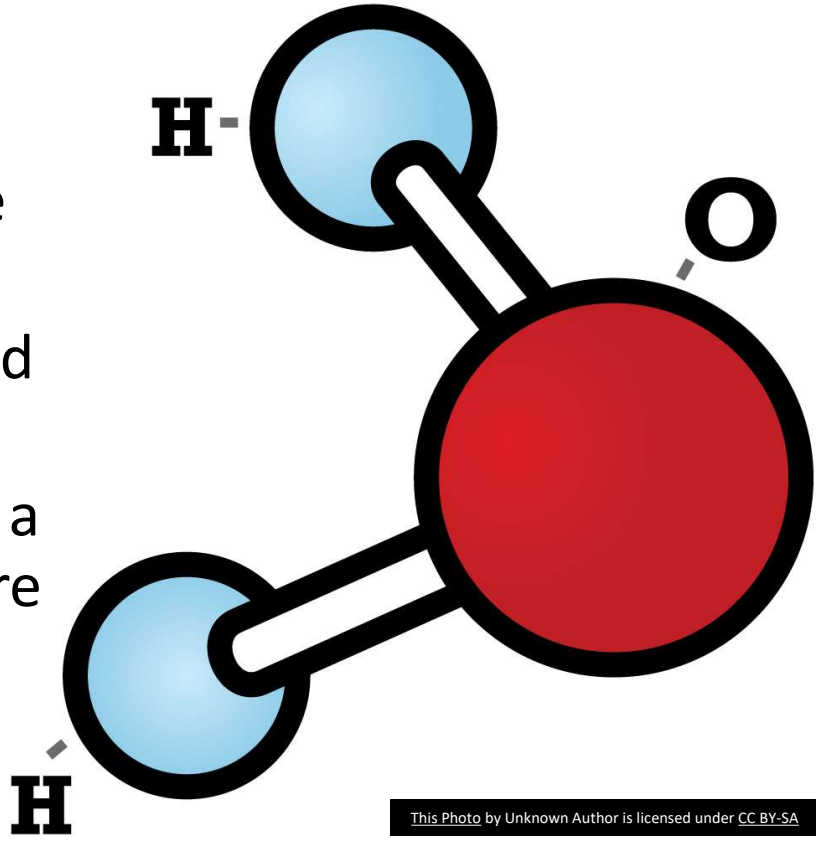
Because I used to do reporting ETL for a living…

Normalization Main Programming Purpose

- Eliminate DML modification anomalies and partial column operations
- Meaning, you need to update one piece of information:
  - One entire piece of data is changed
  - No other piece of data *needs* changing
- Normalization is focused around protecting the integrity of your data by
  - Making every column and table an *atomic* unit
  - Making modifications *atomic* operations

Atomicity

- At the lowest level possible without losing the original characteristics
  - Similar to context of physics as we know it in the 21st century
  - Break down H^20 into Hydrogen and Oxygen, no big deal
  - Break down Hydrogen it becomes a different sort of matter and you are going to need a new laboratory



Example: Go get Aunt Granny's Mud Pie Recipe

You know to go to the Recipe Box!

Mud Pie

A classic twist on a timeless classic.

Often referred to as sweet revenge pie, this pie is awful and inedible, but at least not lethal. Easy to make, using ingredients you almost certainly have around your house... around the outside of it anyhow.

Ingredients

2 cups dirt
2 cups water
1 cup sand
1 frozen pie crust
1 cup chopped nuts (acorns will do)

Directions

1. Put the 2 cups of the dirt and 1 cup of the water in a bowl and stir (be certain that the dirt is actually dirt. You don't want to make that mistake)

2. Add some of the sand, again making sure you have pure sand.

3. Add some more of the water, and the chopped nuts to the mixture and continue to stir.

4. Once the mixture has a wet firm feel to it, pack it into the pie crust

5. Put on campfire until it is a little singed on the top. Let it cool and serve it to friend who broke the head off your Han Solo action figure just to see how easy it would be and now you have a headless Han and it makes you so mad.

There is a reason we don’t use scrolls anymore…

Mud Pie

A classic twist on a timeless classic.

Often referred to as sweet revenge pie, this pie is awful and inedible, but at least not lethal. Easy to make, using ingredients you almost certainly have around your house... around the outside of it anyhow.

Ingredients

2 cups dirt
2 cups water
1 cup sand
1 frozen pie crust
1 cup chopped nuts (acorns will do)

Directions

1. Put the 2 cups of the dirt and 1 cup of the water in a bowl and stir (be certain that the dirt is actually dirt. You don't want to make that mistake)

2. Add some of the sand, again making sure you have pure sand.

3. Add some more of the water, and the chopped nuts to the mixture and continue to stir.

4. Once the mixture has a wet firm feel to it, pack it into the pie crust

5. Put on campfire until it is a little singed on the top. Let it cool and serve it to friend who broke the head off your Han Solo action figure just to see how easy it would be and now you have a headless Han and it makes you so mad.

Mud Pie

A classic twist on a timeless classic.

Often referred to as sweet revenge pie, this pie is awful and inedible, but at least not lethal. Easy to make, using ingredients you almost certainly have around your house... around the outside of it anyhow.

Ingredients

2 cups dirt

2 cups water

1 cup sand

1 frozen pie crust

1 cup chopped nuts (acorns will do)

Directions

1. Put the 2 cups of the dirt and 1 cup of the water in a bowl and stir (be certain that the dirt is actually dirt. You don't want to make that mistake)

2. Add some of the sand, again making sure you have pure sand.

3. Add some more of the water, and the chopped nuts to the mixture and continue to stir.

4. Once the mixture has a wet firm feel to it, pack it into the pie crust

5. Put on campfire until it is a little singed on the top. Let it cool and serve it to friend who broke the head off your Han Solo action figure just to see how easy it would be and now you have a headless Han and it makes you so mad.

And then as you are making one piece of paper per ingredient, you ask the question…

Does this make any sense for what my database needs?

Keep Your Design Simple

- Meet the user's requirements to the Normal Form's requirements
  - One table per user's concept
  - One column per user's attributes of that concept
- Things it is not:
  - Religion
  - Complicated
  - Easy
- Proper designs answer all user queries in straightforward SQL

Normal Forms Overview - 1NF

- Basic shaping of data for the engine
- Data broken down to its lowest form
  - Column Values are atomic
  - No duplicate rows
  - All rows must represent the same number of values (Sometimes referenced as “no repeating groups”)

First Normal Form Example 1

- Requirement: Table of school mascots

MascotId	Name	Color	School
1	Smokey	Black/Brown	UT
112	Smokey	Black/White	Central High
4567	Smokey	Smoky	Less Central High
979796	Smokey	Brown	Southwest Middle

- To truly be in the spirit of 1NF, some manner of uniqueness constraint needs to be on a column that has meaning
- It is a good idea to unit test your structures by putting in data that looks really wrong and see if it stops you, warns you, or something!

First Normal Form Example 2

- Requirement: Store information about books

BookISBN	BookTitle	BookPublisher	Author
111111111	Normalization	Apress	Louis
222222222	T-SQL	Apress	Michael
333333333	Indexing	Microsoft	Kim
444444444	DB Design	Apress	Jessica, & Louis
444444444-1	DB Design	Apress	Louis

- What is wrong with this table?
  - Lots of books have > 1 Author.
- What are common way users would "solve" the problem?
  - Any way they think of!
- What's a common programmer way to fix this?

First Normal Form Example 2

- Add a repeating group?

```
BookISBN    BookTitle       BookPublisher   …
==========  ------------    --------------
111111111   Normalization   Apress          …
222222222   T-SQL           Apress          …
333333333   Indexing        Microsoft       …
444444444   Design          Apress          …

Author1     Author2     Author3
----------- ----------- -----------
Louis
Michael
Kim
Jessica     Louis
```

- What is so wrong?

First Normal Form Example 2

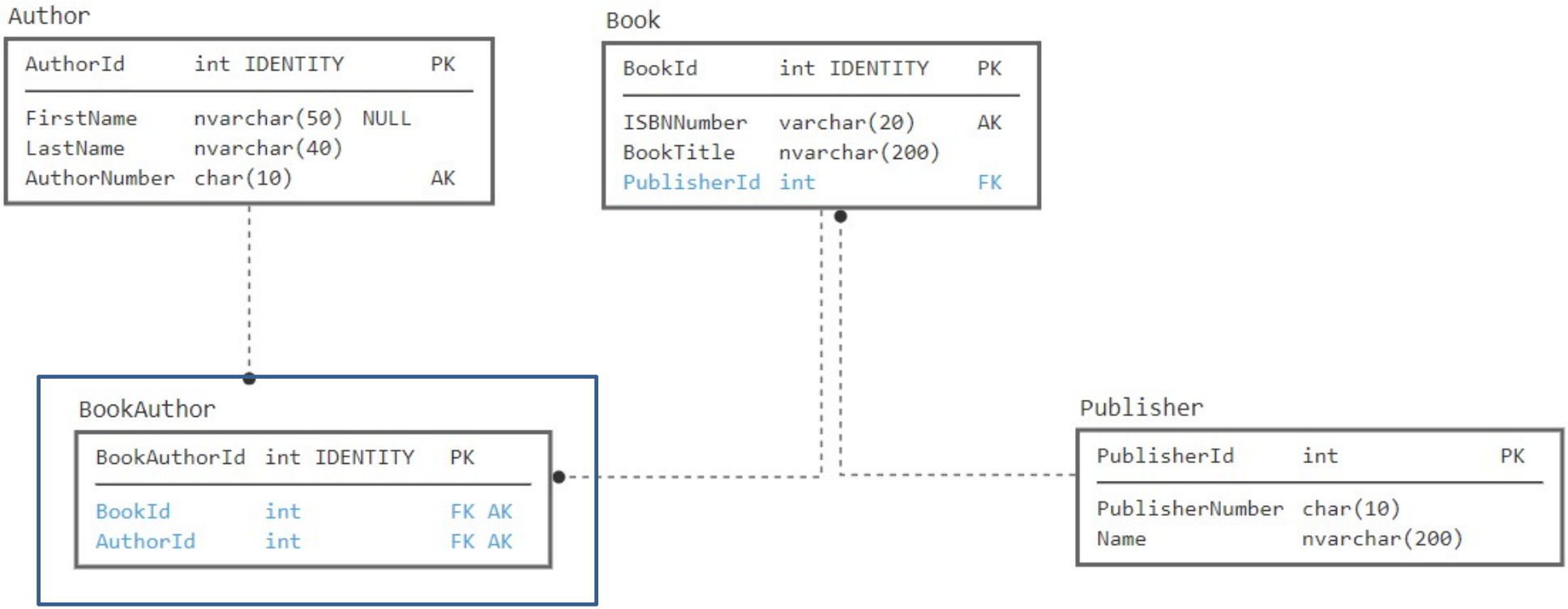
- Two tables!

BookISBN	BookTitle	BookPublisher
111111111	Normalization	Apress
222222222	T-SQL	Apress
333333333	Indexing	Microsoft
444444444	DMV Book	Simple Talk

BookISBN	Author	*ContributionType*
111111111	Louis	***Principal Author***
222222222	Michael	***Principal Author***
333333333	Kim	***Principal Author***
444444444	Jessica	***Contributor***
444444444	Louis	***Principal Author***

- And it gives you easy expansion

First Normal Form Example 2



First Normal Form Example 2.1

```
Email1    Email2    Email3
--------- --------- -----------
```

But inevitably, requirements change, and the structures are inadequate… and unchangeable… So you do something like this (and yes, I had to do this recent enough it still makes me a bit ill…that db has more columns than this!):

```
Email1Status Email1Type   Email1PrivateFlag
------------ ------------ --------------------

Email2Status Email2Type   Email2PrivateFlag
------------ ------------ --------------------

Email3Status Email3Type   Email3PrivateFlag
------------ ------------ --------------------
```

First Normal Form Example 3

- Requirement: Store users and their names

```
UserId      UserName        PersonName
=========== ~~~~~~~~~~~~~~~ ---------------
1           Drsql           Louis Davidson
2           Kekline         Kevin Kline
3           Datachix2       Audrey Hammonds
4           PaulNielsen     Paul Nielsen
```

- How would you search for someone with a last name of Nielsen? David? Davidson?
- What if the name were more realistic with Suffix, Prefix, Middle names?

First Normal Form Example 3

- Break the person's name into individual parts

UserId	UserName	PersonFirstName	PersonLastName
1	Drsql	Louis	Davidson
2	Kekline	Kevin	Kline
3	Datachix2	Audrey	Hammonds
4	PaulNielsen	Paul	Nielsen

- This optimizes the most common search operations
- It isn't a "sin" to do partial searches on occasion, just don't make it habitual:
  - I know the last name ended in "son" or "sen"
- If you also commonly need the full name available, let the engine manage this using a calculated column:
  - PersonFullName AS CONCAT(PersonFirstName + ' ', PersonLastName)

Normal Forms Overview – 2NF, 3NF and Boyce-Codd (BCNF) Normal Forms

- Eliminate incorrect data dependencies in your tables
  - All attributes are either a key, or fully dependent on a key (the whole key, and nothing but the key)
  - Violations usually manifest themselves as multiple column, row-wise repeating groups
- In other words…
  - All keys for a table are identified
  - All columns describe that “thing” that the table is modeling

Intrarow Dependency

- If you can determine the value of one attribute X given a different attribute Y, then Y is functionally dependent on X. X is considered the determinant.

  Example:

X	Y	Z
1	1	2
2	2	4
3	2	4

- Assuming this is the entire known universe. X is unique key:
  - Y and Z are functionally dependent on X
  - But, is Z is functionally dependent on Y (or vice versa)?

Boyce Codd NF Example 1

- Requirement: Defines the types of car(s) that a driver likes

Driver	Car Style	Height	EyeColor	MaxWeight
Louis	Station Wagon	6′0″	Blue	2900
Louis	Hatchback	6′0″	Blue	2500
Ted	Coupe	5′8″	Brown	2200

- Check the attributes against the meaning of the table
  - Height and EyeColor define the attributes of the driver alone
  - MaxWeight? The weight of vehicle for that style it is acceptable for the style of car? Or the driver? Naming is important!
    - Defined as: Vehicle weight for car style

Boyce Codd NF Example 1

- Solution: 3 independent tables, 1 for driver, 1 for driver's car style preference, 1 for driver and car style

Driver	Car Style
Louis	Station Wagon
Louis	Hatchback
Ted	Coupe

Driver	Height	EyeColor
Louis	6′0″	Blue
Ted	5′8″	Brown

Car Style	MaxWeight
Station Wagon	2900
Hatchback	2500
Coupe	2200

Boyce Codd NF Example 2

- Requirement: Driver registration for rental car company

Driver	VehicleOwned	Height	EyeColor	WheelCount
Louis	Hatchback	6′0″	Blue	4
Ted	Coupe	5′8″	Brown	4
Rob	Tractor trailer	6′8″	NULL	18

- Column Dependencies
  - Height and EyeColor, check
  - VehicleOwned, check
  - WheelCount, <buzz>, driver's do not have wheelcounts

Boyce Codd NF Example 2

- Two tables, one for driver, one for type of vehicles and their characteristics

Driver	VehicleOwned (FK)	Height	EyeColor
Louis	Hatchback	6′0″	Blue
Ted	Coupe	5′8″	Brown
Rob	Tractor trailer	6′8″	NULL

VehicleOwned	WheelCount
Hatchback	4
Coupe	4
Tractor trailer	18

Natural Keys

```
create table driversLicense
(
        driversLicenseStateCode char(2) not null,
        driversLicenceNumber varchar(40) not null,
        classCode char(2) not null,
        PRIMARY KEY (driversLicenseStateCode, driversLicenseNumber)
)
create table employee
(
        employeeNumber char(10) not null primary key
        driversLicenseStateCode char(2) null,
        driversLicenceNumber varchar(40) null,
        firstName varchar(30) not null,
        middleName varchar(30) null,
        lastName varchar(30) not null,
        UNIQUE (driversLicenseStateCode, driversLicenseNumber)
        FOREIGN KEY(driversLicenseStateCode, driversLicenseNumber)
                REFERENCES driversLicense(driversLicenseStateCode, driversLicenseNumber)
)
```

Composite Keys – The Surrogate Effect

This is a mathematically equivalent representation

```
create table driversLicense
(
        driversLicenseId int primary key --might be identity or even GUID
        driversLicenseStateCode char(2),
        driversLicenceNumber varchar(40),
        classCode char(2),
        unique (driversLicenseStateCode, driversLicenseNumber)
)
create table employee
(
        employeeId int identity primary key, -might be identity or even GUID
        employeeNumber char(10) not null UNIQUE,
        driversLicenseId int null UNIQUE
                                REFERENCES driversLicense(driversLicenseId)
        firstName varchar(30) not null,
        middleName varchar(30) null,
        lastName varchar(30) not null
)
```

Fourth Normal Form

- The key of every table should represent no more than one independent multi-valued relationship
- In other words, the combination of key attributes should represent one thing only

Fourth Normal Form Example

- Requirement: define the classes offered with teacher and book

Trainer	Class	Book
Louis	Normalization	DB Design & Implementation
Chuck	Normalization	DB Design & Implementation
Fred	Implementation	DB Design & Implementation
Fred	Golf	Topics for the Non-Technical

- Dependencies
  - Class determines Trainer (Based on qualification)
  - Class determines Book (Based on applicability)
  - Trainer does not determine Book (or vice versa)
- If trainer and book are related (like if teachers had their own specific text,) then this table is in 4NF

Fourth Normal Form Example

Trainer	Class	Book
Louis	Normalization	DB Design & Implementation
Chuck	Normalization	DB Design & Implementation
Fred	Implementation	DB Design & Implementation
Fred	Golf	Topics for the Non-Technical

Question: What classes do we have available and what books do they use?

```
SELECT DISTINCT Class, Book
FROM   TrainerClassBook
```

Class	Book
Normalization	DB Design & Implementation
Implementation	DB Design & Implementation
Golf	Topics for the Non-Technical

Doing a very slow operation, sorting your data unnecessarily… please wait

Fourth Normal Form Example

- Break Trainer and Book into independent relationship tables to Class

Class	Trainer
Normalization	Louis
Normalization	Chuck
Implementation	Fred
Golf	Fred

Class	Book
Normalization	DB Design & Implementation
Implementation	DB Design & Implementation
Golf	Topics for the Non-Technical

Can you over-normalize?

- Short answer: sort of
- Long answer: no
  - Breaking objects down beyond user needs is not productive
  - Lots of joins are not always that costly
  - Over-normalization is usually over-engineering, ignoring what the user needs

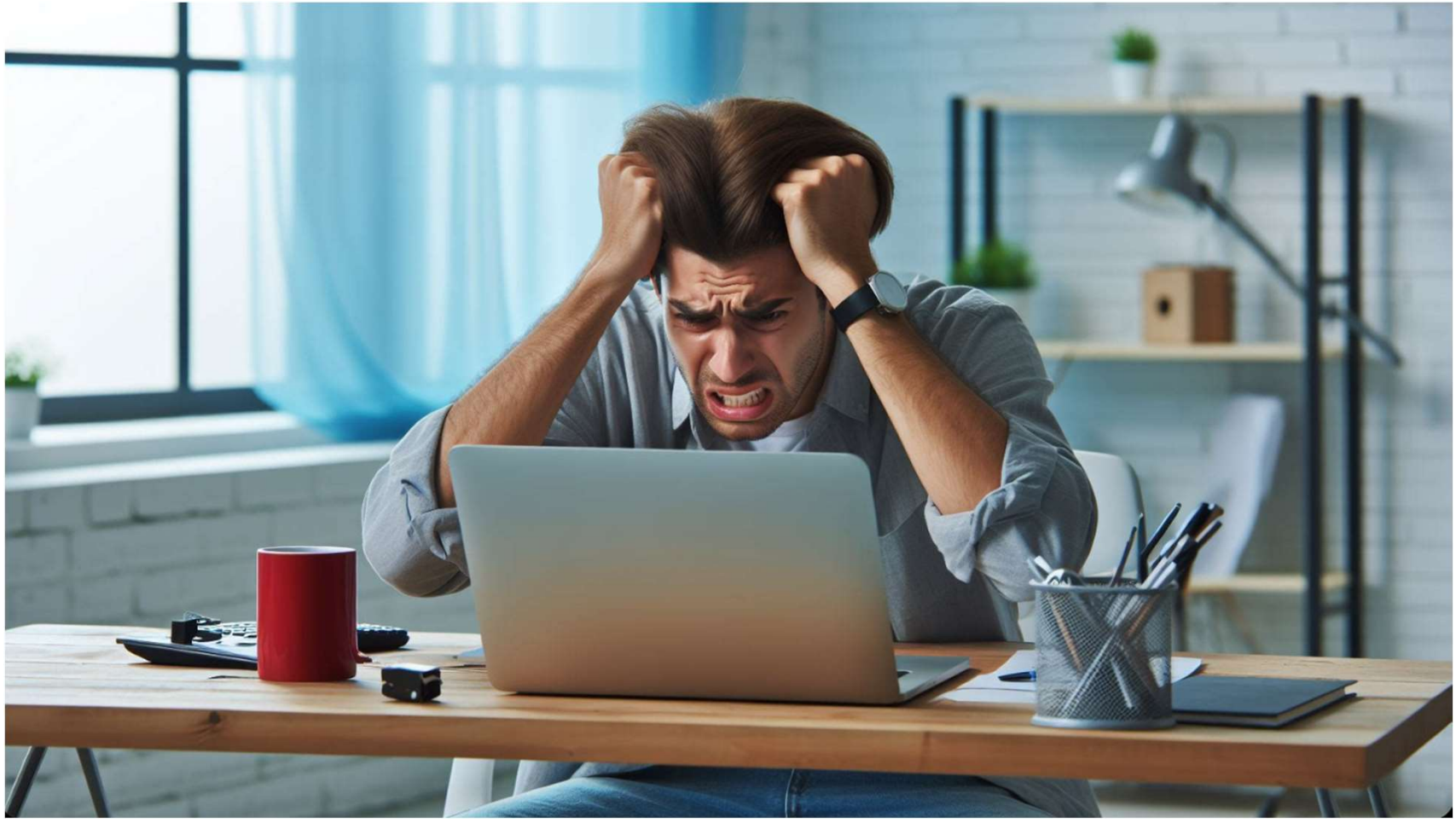
Denormalization

- 100% an implementation concept
- Stepping back from strict normalization for a good reason
- Common reasons
  - Performance
  - Usability
  - But mostly
    - FUD (Fear, uncertainty, doubt)
    - Try the normalized version first if possible
    - Learn how to tune your data platform
- Not knowing what normalization is not denormalization

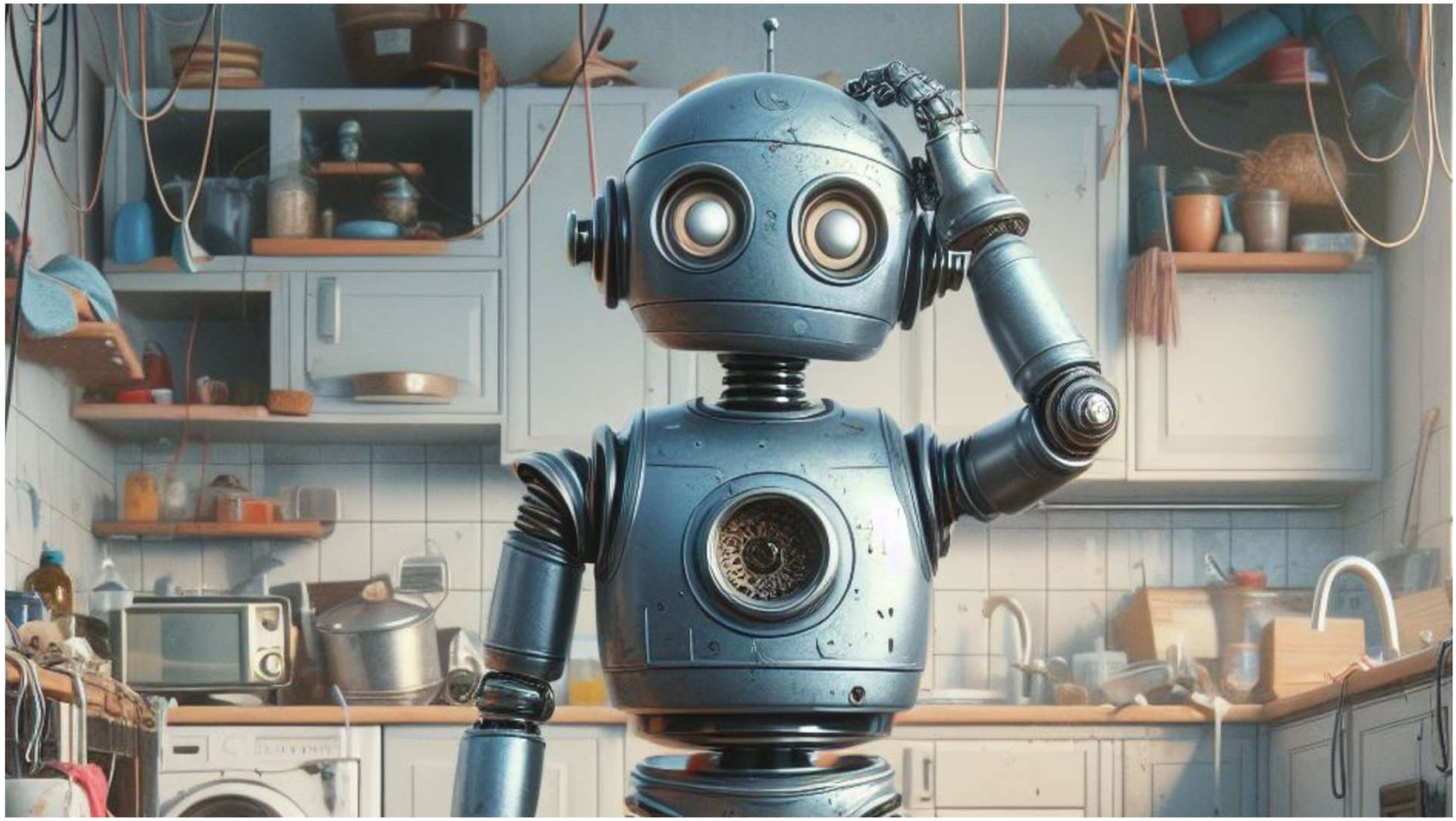
So you have a design…build!

- I usually build to a SQL Server that contains model databases
- Strongly consider using SQL Server (or any RDBMS) to implement
  - Data checks (check constraints)
  - Column domains (data types and check/null constraints)
  - Table relationships (foreign key constraints)
- Minimally, make certain that the data is protected at a level lower than a user can get access
- Ideally:
  - You should trust that your data is valid to the immutable business rules at all times
  - The only data cleansing you should ever do is to look for users doing dumb stuff

The better you design…the less your users make this face



Perhaps more importantly… nor does AI



Test…test…test

- Start building unit tests in during conceptual modeling that you can run to make sure that your design works
- Remember the requirements? Develop the test plan directly from them
- Throw tons of bad data at the design, especially places where you thought data should have been designed better
- Try to get an error for every constraint implemented
- Try to get an error for every constraint NOT implemented

Questions? Contact info..

- Louis Davidson - louis@drsql.org
- Website – http://drsql.org
  - Get slides here: https://drsql.org/presentations
- Twitter – http://twitter.com/drsql

- Editor/Blogger at Simple Talk https://www.red-gate.com/simple-talk/author/louis-davidson/
- (Always looking for new writers! editor@simple-talk.com)

Thank you!