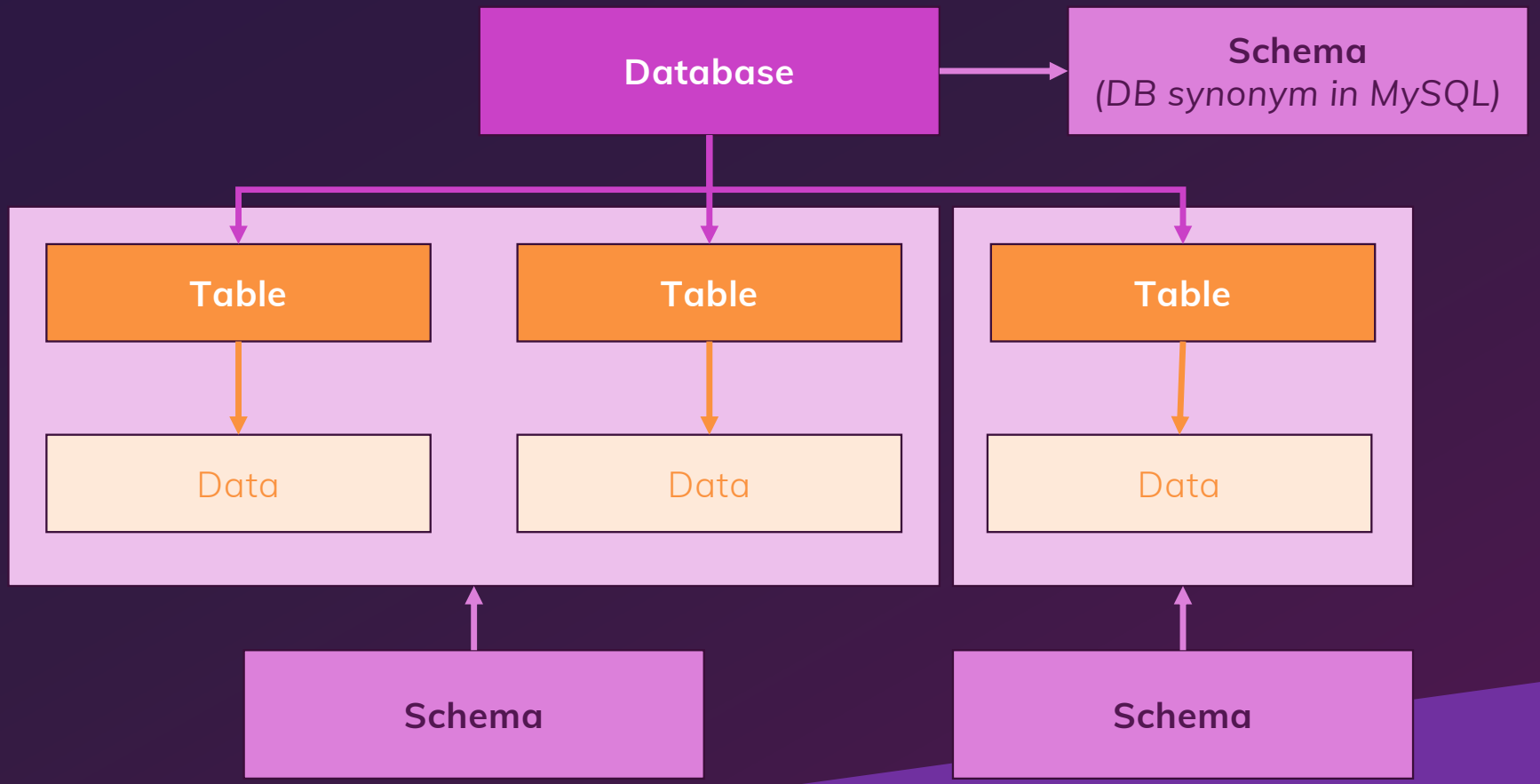
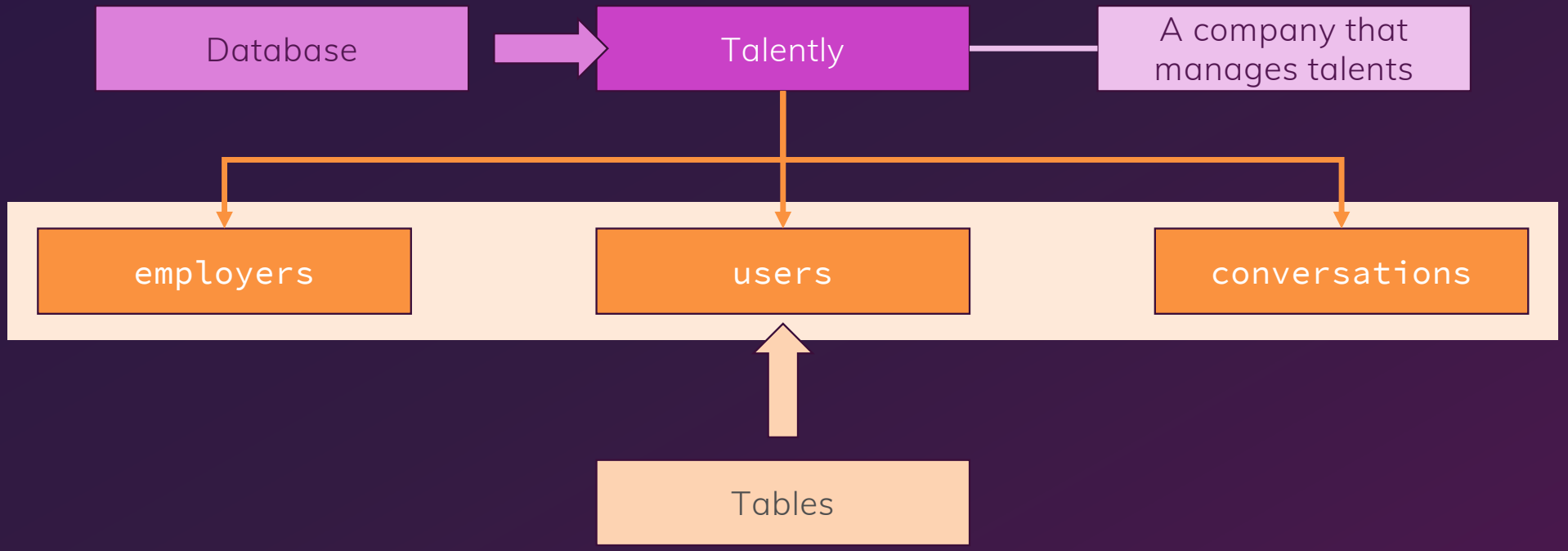
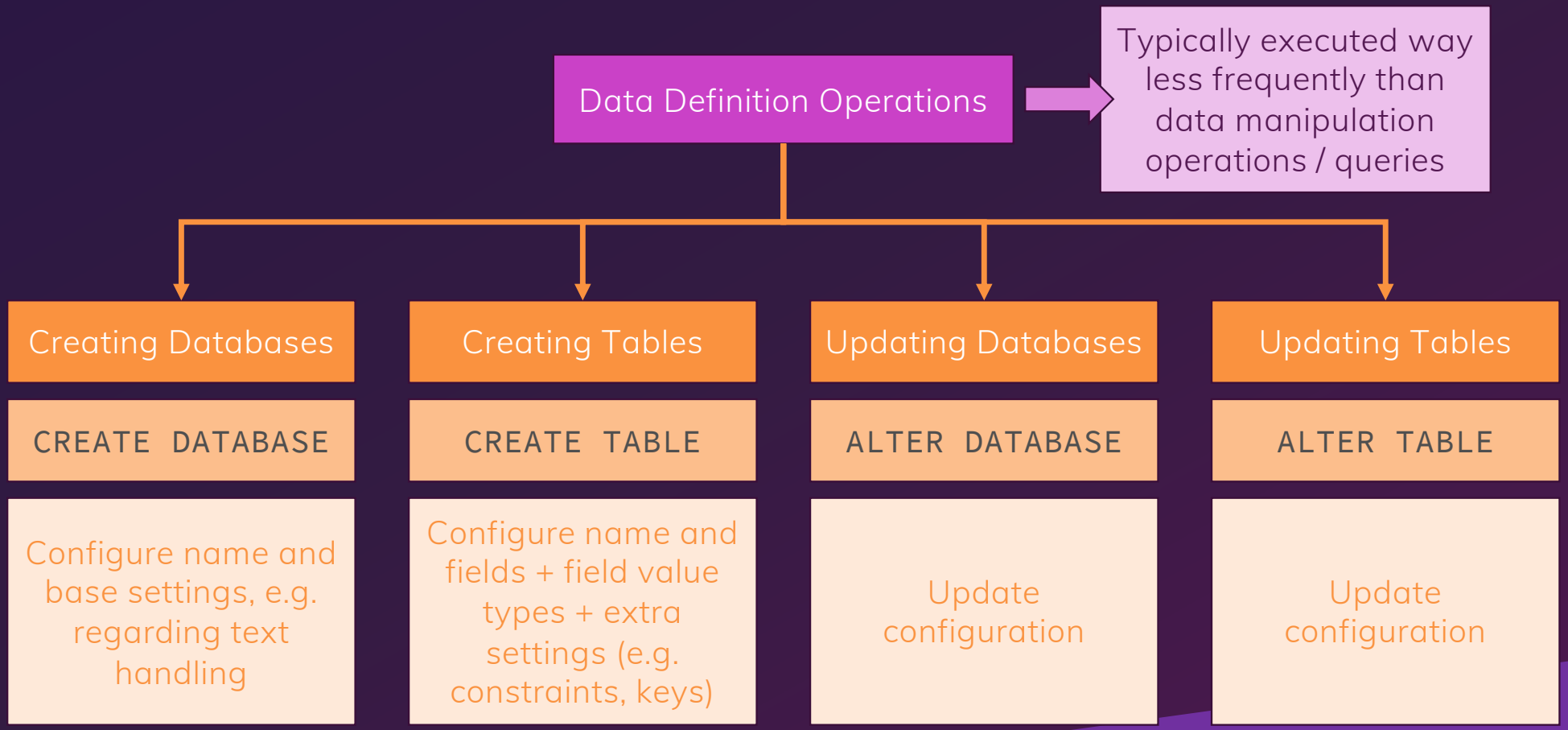
Database & Table Hierarchy



Our Example For This Course Section



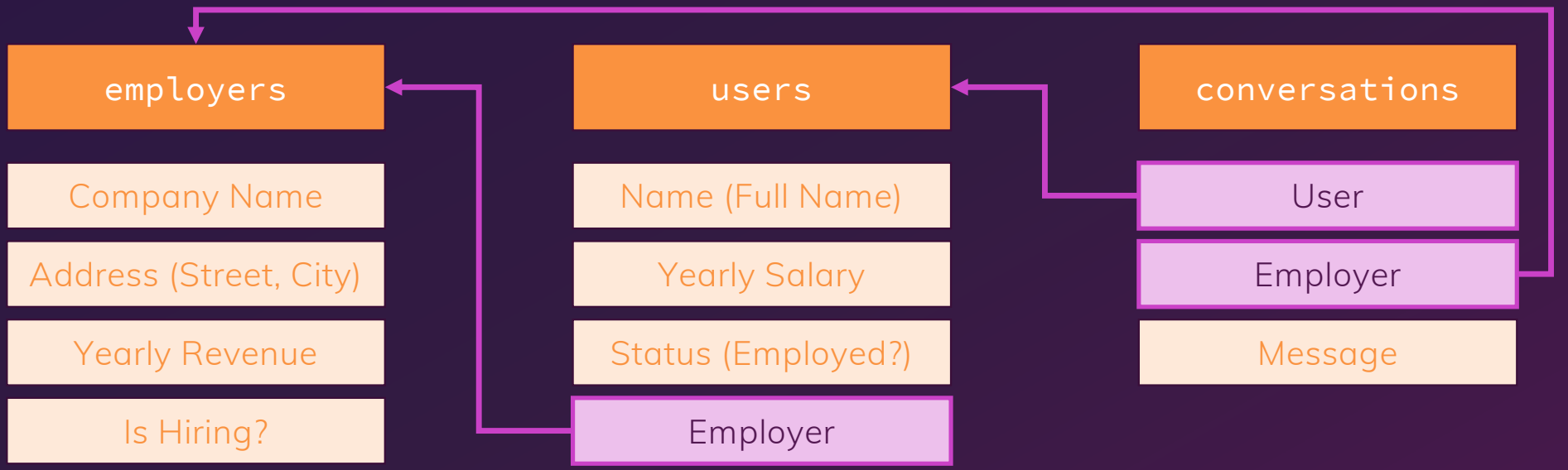
Managing Databases & Tables



Talently: Which Data Should Be Stored?

employers	users	conversations
Company Name	Name (Full Name)	User
Address (Street, City)	Yearly Salary	Employer
Yearly Revenue	Status (Employed?)	Message
Is Hiring?	Employer	Date Sent

Talently: Which Data Should Be Stored?



ACADEMIND

Key Data Types / Value Types

Text	Numeric	Date	Other
`CHAR(X)`	`INT, SMALLINT, …`	`DATE`	`BOOLEAN`
Store text up to X characters; shorter text will be space padded	Integer numbers (between min and max boundaries) are allowed	A value like 1986-10-20 (i.e. no hours or minutes)	True or false (0 or 1)
`VARCHAR(X)`	`DECIMAL, NUMERIC`	`DATETIME, TIMESTAMP`	`JSON`
Store text up to X characters; shorter strings will not be changed	Decimal numbers with a fixed precision (exact values)	A value like 1986-10-20 14:39:05 (i.e. with hours, minutes etc.)	JSON-formatted text data
`TEXT, LONGTEXT, …`	`FLOAT, REAL`		`SERIAL`
Text of any size can be stored without specifiy a max size first	Decimal numbers with floating points (approximated values)		An auto-incrementing integer number
`ENUM`			
Only values from a predefined set of allowed values are accepted			

Not all types are part of the official standard – and not all database systems support all types

ACADEMIND

Integer Values

5 10 -20 ...

ACADEMIND

Number Values With Decimal Places

3.14

5.58

-10.999

...

CHAR vs VARCHAR vs TEXT (vs LONGTEXT ...)

Pre-defined maximum length

CHAR(X)	VARCHAR(X)
	Typically used!
Text with max. length of X bytes	
One byte can be one character ➔ Depends on encoding	
Shorter text is space-padded	Shorter text is not changed
CHAR(4)	VARCHAR(4)
Inserted 'hi'	Inserted 'hi'
Stored 'hi'	Stored 'hi'

No maximum length
(database system limits apply)

TEXT	LONGTEXT, …
Typically used!	
Text with no user-defined max. length (max. length depends on data type)	
One byte can be one character ➔ Depends on encoding	
Max. size is 1GB in Postgres, 65,535 characters in MySQL	Not supported in Postgres, different types with different sizes in MySQL

Not part of the SQL standard but supported by many database systems

ACADEMIND

A Closer Look At Numeric Value Types

Integer ("Whole") Numbers	Exact Decimal Point Numbers	Approximate Decimal Point Numbers
`INT, SMALLINT, …`	`DECIMAL, NUMERIC`	`FLOAT, REAL, …`
`3, -10, -1831, 9418125`	`724.12, -8.195, 51413.1`	`724.12, -8.195, 51413.1`
Numbers without any decimal places	Numbers with decimal places and exact precision	Numbers with decimal places and approximate precision
Inserted numbers with decimal places are rounded	Inserted numbers are stored exactly (no data loss)	Stored approximately (data loss is possible)
Great for mathematical calculations	Great for data that requires exactness (e.g. monetary)	Great for numeric data where exactness is not required
Great performance	Slow performance	Great performance
Different types of integers occupy different amounts of space	Precision can be set when the table is created	Different types of numbers occupy different amounts of space

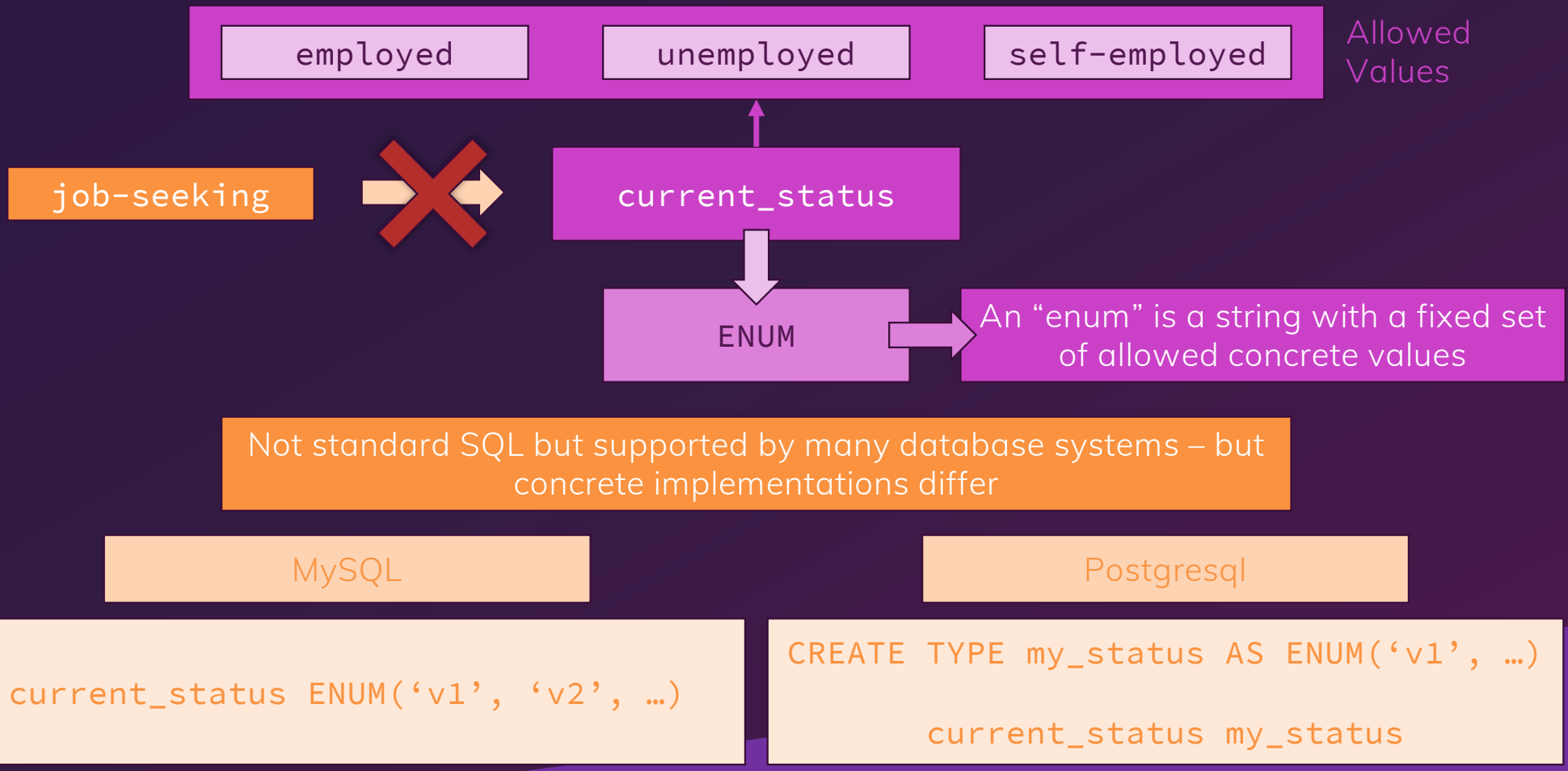
Storing The Salary

Storing The Salary Exactly	Storing The Salary Approximately	Storing The Salary As An Integer ("Whole") Number
`19,000.12`	`19,000.12 vs 19,000.13`	`19,000`
Stored exactly, so that it can be used in calculations without data loss / inaccuracy	Stored approximately as exact values might not matter	The decimal values might not matter at all when talking about salaries
Comes at a performance penalty	Good performance!	Good performance!

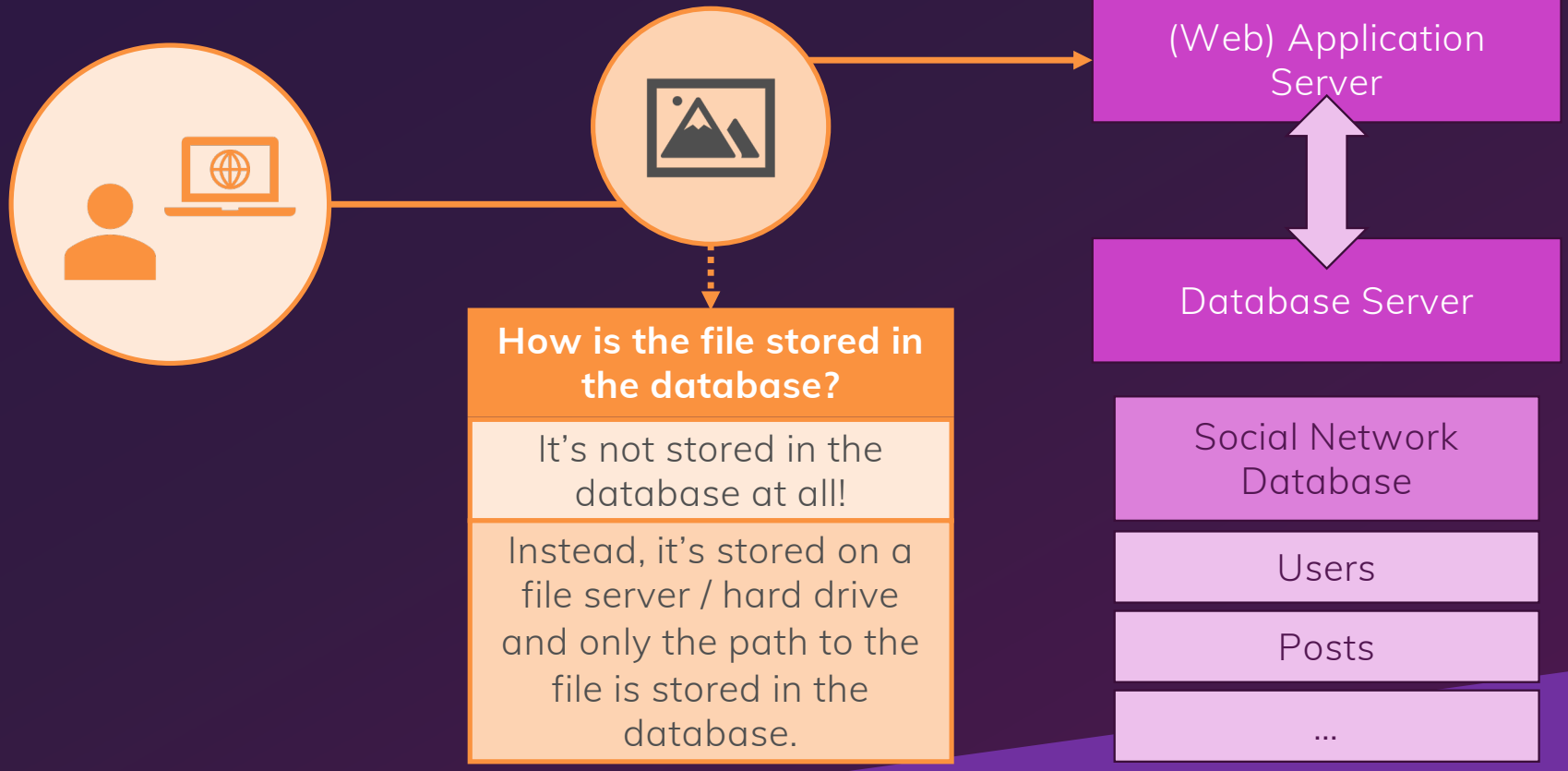
ACADEMIND

Floating Point Numbers / Fractional Numbers Are Challenging For Computers

Enums


employed
unemployed
self-employed
Allowed Values
job-seeking
current_status
ENUM
An “enum” is a string with a fixed set of allowed concrete values
Not standard SQL but supported by many database systems – but concrete implementations differ
MySQL
Postgresql
current_status ENUM(‘v1’, ‘v2’, …)
CREATE TYPE my_status AS ENUM(‘v1’, …)
current_status my_status

What About Files?



(Not) Storing Files In Databases

```
CREATE TABLE users (
    user_name VARCHAR(255),
    image_path VARCHAR(200),
);
```

A column that should hold the image file paths

```
INSERT INTO users (user_name, image_path)
VALUES ('DBMax', 'uploads/images/db_max.jpg');
```

The path to the image (which is stored on some file server)

The Problem With "No Data"

first_name	last_name	salary	status
Max	Schwarz	15000	self-employed
Julie	Barnes	19000	employed
Michael	Smith	0	unemployed

Might distort analyses

Average: 11,333.33

The NULL Value

first_name	last_name	salary	status
Max	Schwarz	15000	self-employed
Julie	Barnes	19000	employed
Michael	Smith	[NULL]	unemployed

Shouldn't be null!

Is ignored

Average: 17,000

Allowing Or Forbidding NULL Values

```
CREATE TABLE users (
    full_name VARCHAR(255) NOT NULL,
    salary INT -- NULL is allowed because it's not forbidden
);
```

`NOT NULL` is a "Constraint"

This column must contain a (valid) value – omitting it is not possible

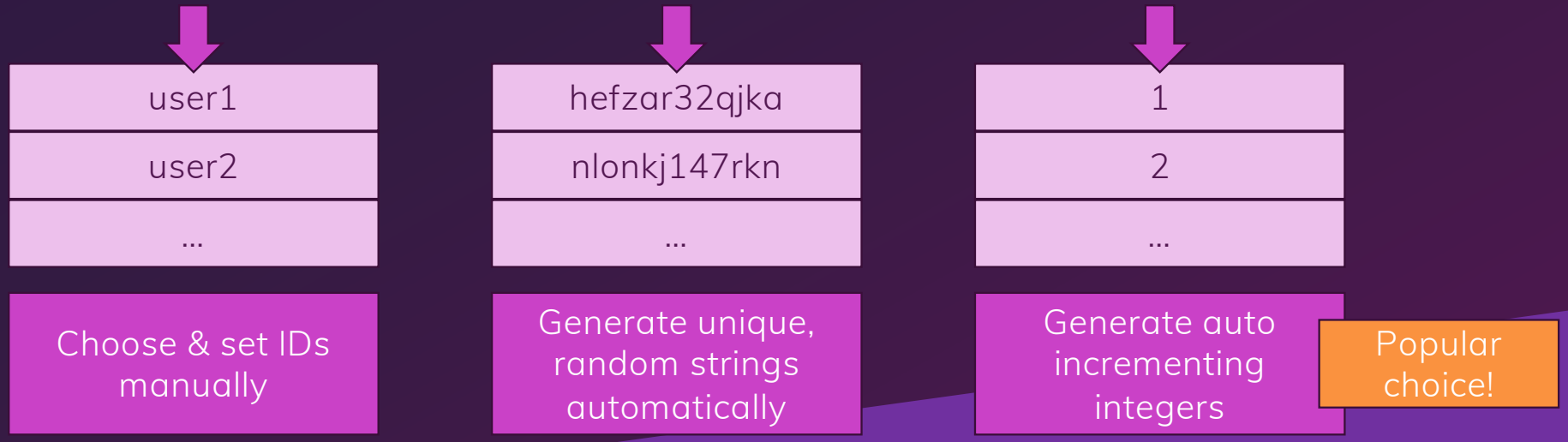
ACADEMIND

What If Multiple Users Have The Same Name?

The Role & Importance Of Unique IDs

When storing data, each data entry should have at least one unique value (for identifying the record)

Unique ID



Setting Unique IDs & Primary Keys

```
CREATE TABLE users (
    id INT NOT NULL UNIQUE,
    full_name VARCHAR(255) NOT NULL,
    salary INT
);
```

`UNIQUE` is a "Constraint"

This column must not contain duplicate values

Setting Unique IDs & Primary Keys

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(255) NOT NULL,
    salary INT
);
```

`AUTO_INCREMENT` is an "Attribute"

The db system will automatically insert an incrementing value

`PRIMARY KEY` is a "Constraint"

This column must be unique and `NOT NULL`; Only one `PRIMARY KEY` per table is allowed!

In most SQL databases / environments (e.g. Postgresql), `AUTO_INCREMENT` is not supported.

Use `SERIAL PRIMARY KEY` instead of `INT PRIMARY KEY AUTO_INCREMENT`

Setting Unique IDs & Primary Keys (Postgres)

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    salary INT
);
```

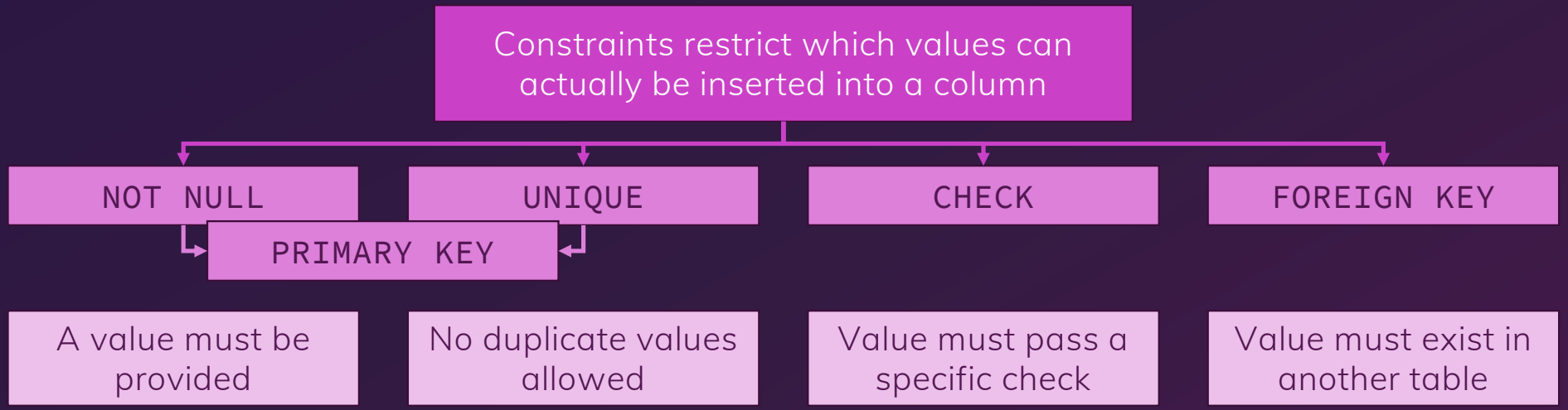
`SERIAL` is a Special Data Type

Creates an auto-incrementing integer

(not supported in MySQL)

ACADEMIND

More On Constraints



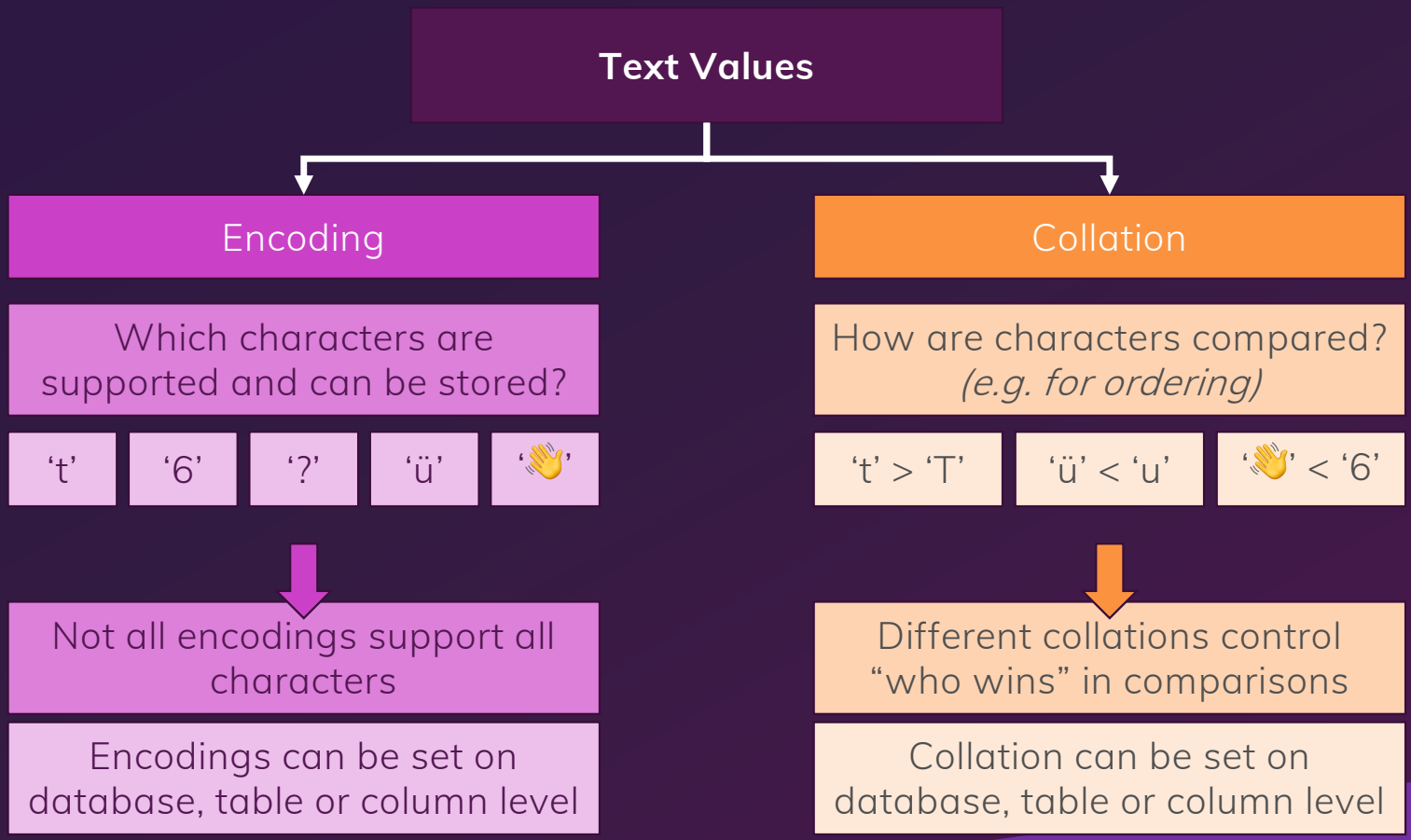
Column Constraints

```
salary INT CHECK (salary > 0)
```

Table Constraints

```
salary INT,
CHECK (salary > 0)
```

Text: Encoding & Collation



More Ways Of Creating Tables

Temporary Tables

```
CREATE TEMPORARY TABLE …
```

Tables that are only stored temporarily (in memory of the database server)

Useful for non-permanent data *(e.g. intermediate results)*

Tables Based On Other Tables / Data

```
CREATE TABLE … AS <query>
```

Creates a table and pre-populates it with data from a query result set

Useful if a subset of data from another table should be stored in a separate table

Generated Columns

first_name	last_name	full_name
'Max'	'Schwarz'	'Max Schwarz'
'Julie'	'Barnes'	'Julie Barnes'

Generated Columns

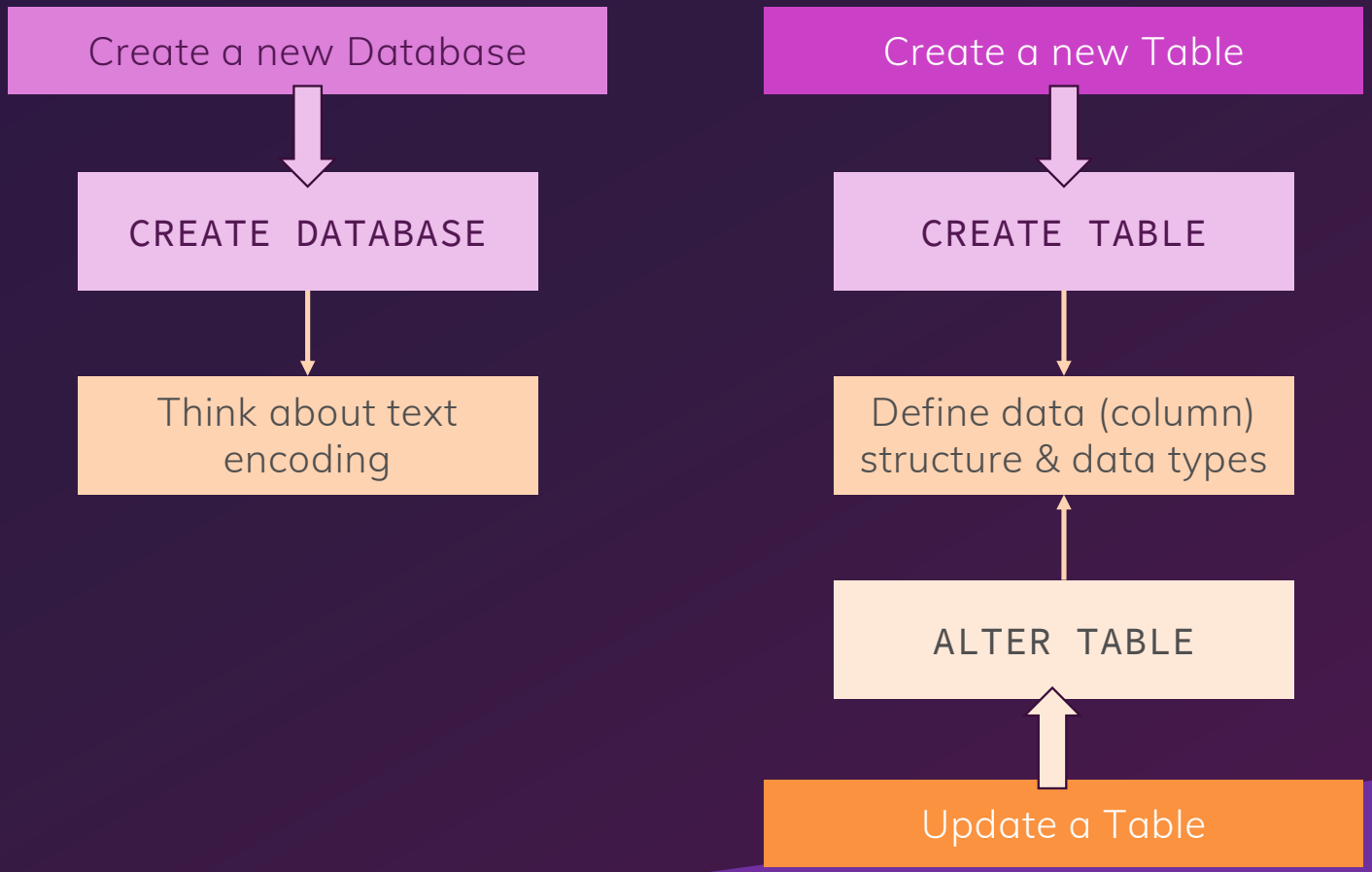
first_name	last_name	full_name
'Max'	'Schwarz'	'Max Schwarz'
'Julie'	'Barnes'	'Julie Barnes'

Could be created as a "Generated Column"

Value is derived automatically

Data doesn't have to be (and can't be) inserted manually

Module Summary



Exercise Time