POWERING POTENTIAL

Data Design

Module 2: Databases

Databases

Module 2: Contents

- Data types.
- Constraints and Keys.

Objectives

Data design

Describe the different data types used within databases

- How are they used and why do we need them?

Explore how constraints are used to refine information

- What's the value of using constraints on our data?

Discuss using primary and foreign keys in a relational database

- How are they used to link information together?

Data types

Databases: Module 2

Data Design

Data types

There are three main data types, all of which have their own sub-types

- **Numeric:** Data which you can use for calculations.
- **Text:** Text used for values you wouldn't calculate, like postcodes, email addresses, and so on.
- **Date/Time:** Handy for rubber-stamping and organising data.

These data types are assigned to the fields within database tables

- From here, we can construct a baseline constraint for the type of data we store in each table.

Data types: Numeric

TYPE	MIN	MAX	EXAMPLE
BIT	1	64	63
BOOL(EAN)	-128	127	0 (FALSE) !0 (TRUE)
TINYINT	-128	127	47
SMALLINT	-32768	32767	31337
MEDIUMINT	-8388608	8388607	7145531
INT	-2147483748	2147483747	12
BIGINT	-92233720368547758082^16	92233720368547507	625494294624
DEC(IMAL)/NUMERIC/FIXED	VERY LOW	VERY HIGH	35D.30D
FLOAT/DOUBLE	HARDWARE-DEPENDENT	HARDWARE-DEPENDENT	BIT

Data types: Text

TYPE	FORMAT	STYLE	EXAMPLE
CHAR	0-255 CHARACTERS	FIXED-LENGTH STRING RIGHT-PADDED WITH SPACES	HENLO FRENS
VARCHAR	0-65535 CHARACTERS	VARIABLE-LENGTH STRING ROW-SIZE DEPENDENT CHARACTER-SHEET DEPENDENT	HENLO FRENS

There are other string types that are stored in binary format, instead of characters:

- These are more efficient due to less overhead, so they don't need to store data in the character set.
- They only use 1 and 0, so you don't need to rely on whatever character format you're using.

Data types: Date/Time

TYPE	FORMAT	MIN	MAX	EXAMPLE
DATE	YYYY-MM-DD	1000-01-01	9999-12-31	1995-02-09
DATETIME	YYYY-MM-DD HH:MM:SS.SSSSSS	1001-01-01 00:00:00.000000	9999-12-31 23:59:59.999999	2112-12-21 16:20:00.000001
TIMESTAMP	SECONDS SINCE EPOCH IN S	1970-01-01 00:00:00.000000	2038-01-19 03:14:07.999999	2019-08-02 09:52:58.000005
TIME	HHH:MM:SS.SSSSSS	-838:59:59.000000	838:59:59.000000	444:44:44.444444
YEAR	YYYY	1901	2155	2155

Practice: GAME Database

Outcome:

- Think about how data might be presented in a large database system and how we can apply it to the things we make here.

Steps:

10 minutes, in pairs

- Based on the tables you drew earlier, discuss what data type each field would be.
- Each pair will add their ideas to the table I will draw the board.

Setting up fields

customers

CUSTOMER ID	NAME	ADDRESS	EMAIL	PASSWORD
1	SIMON	256 BYTE STREET	SI@MAIL.CO.UK	**********
2	MARKUS	47 RED TIE ROAD	MARKUS47@POST.COM	**********
3	EMMA	63 NUMBER LANE	EM@LETTER.BOX	*******

- Customer ID: int
- Name: varchar(100)
- Address: varchar(100)
- Email: varchar(50)
- Password: varchar(30)

games

PRODUCT ID	TITLE	QUANTITY	PRICE	AGE RATING
1	SHOOT THE COOL GUN 9	8965	79.99	18
2	GUNBLADERS XXII	546	64.99	15
3	PAINT DRYING SIMULATOR 2012	35	37.99	3
4	SITAR HERO	456	45.99	12

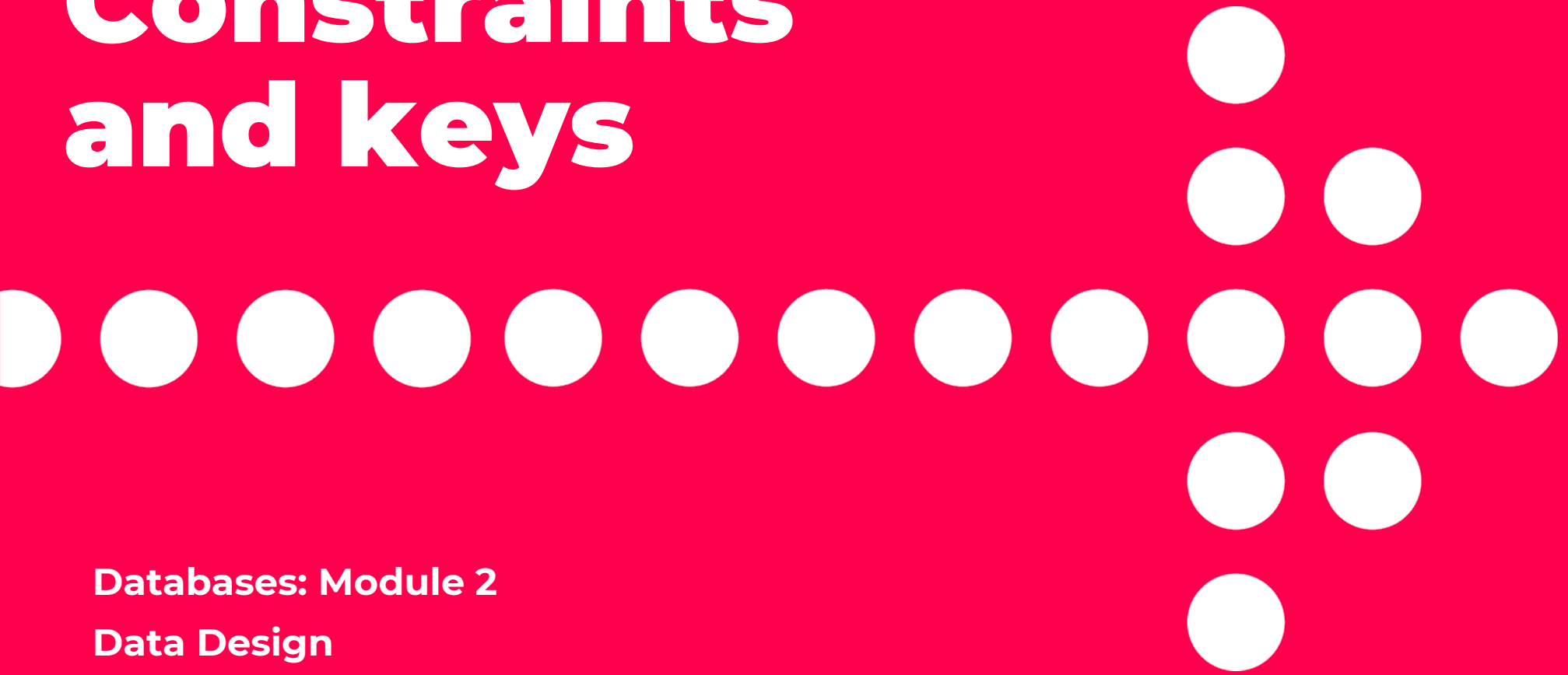
- Product ID: int
- Title: varchar(50)
- Quantity: int
- Price: dec(4,2)
- Age rating: int

orders

ORDER ID	CUSTOMER ID	PRODUCT ID	PLACED	TOTAL
1	1	4	2019-08-06	45.99
2	1	3	2019-08-14	37.99

- Order ID: int
- Customer ID: int
- Product ID: int
- Placed: date
- Total: dec(4,2)

Constraints and keys



Databases: Module 2

Data Design

Constraints

What if, in our customer table, we have two people named Markus?

- Based on the hundreds of customers GAME will have, this is inevitable!
- What other issues can you think of that might cause problems?

For this, we might have to use database constraints. This which will allow greater control over our data:

- **Unique:** All values entered in this field must not be the same as others.
- **Not null:** A field must be filled in. It cannot be left empty.
- **Default:** This field will have a certain ‘standard’ value assigned as a default
- **Keys:** A combination of unique and not null. The table can have a **primary key** to identify it.

Looking into the database

People can order the same product as other orders, but we wouldn't have an order without a customer – not null.

The date placed field could have a default value of today – the date it was created.

orders

ORDER ID	CUSTOMER ID	PRODUCT ID	PLACED	TOTAL
1	1	4	2019-08-06	45.99
2	2	3	2019-08-14	37.99

Each order should be unique to a record and each order requires an order number.

This field could also be not null, but we wouldn't want it to be unique as multiple orders may have the same total.

Keys

Within our database, we would want a key to uniquely identify a record quickly:

- We do this through a **primary key**.
- We can then associate records from different tables through a **foreign key**;

We can create a specific field to act as the primary key, or we can use a field that's already there:

- For instance, a customer table could have **customer id** as the primary key.
- We can then use the primary key from the **customer** table to link to the **order table**.

- In our case, we might use something like the customer ID.

customers

CUSTOMER ID	NAME	ADDRESS	EMAIL	PASSWORD
1	SIMON	256 BYTE STREET	SI@MAIL.CO.UK	**********
2	MARKUS	47 RED TIE ROAD	MARKUS47@POST.COM	**********
3	EMMA	63 NUMBER LANE	EM@LETTER.BOX	*******

primary key

orders

ORDER ID	CUSTOMER ID	PRODUCT ID	PLACED	TOTAL
1	1	4	2019-08-06	45.99
2	2	3	2019-08-14	37.99

foreign key

Summary

Databases: Module 2

Describe the different data types used within databases

- Numeric, date/time, and text data types all have specific sub-types which we can use to our advantage.

Discuss using primary and foreign keys in a relational database

- Primary and foreign keys are the building blocks used to construct relationships between entities, so that you can access information easier.


Thank you for listening
Any questions?