
First Steps in SQL
365 DataScience


Creating a Database – Part I
365 DataScience

Creating a Database – Part I

So far:

Creating a Database – Part I

So far:

- Theory of Relational Databases

Creating a Database – Part I

So far:

- Theory of Relational Databases
- SQL Theory

Creating a Database – Part I

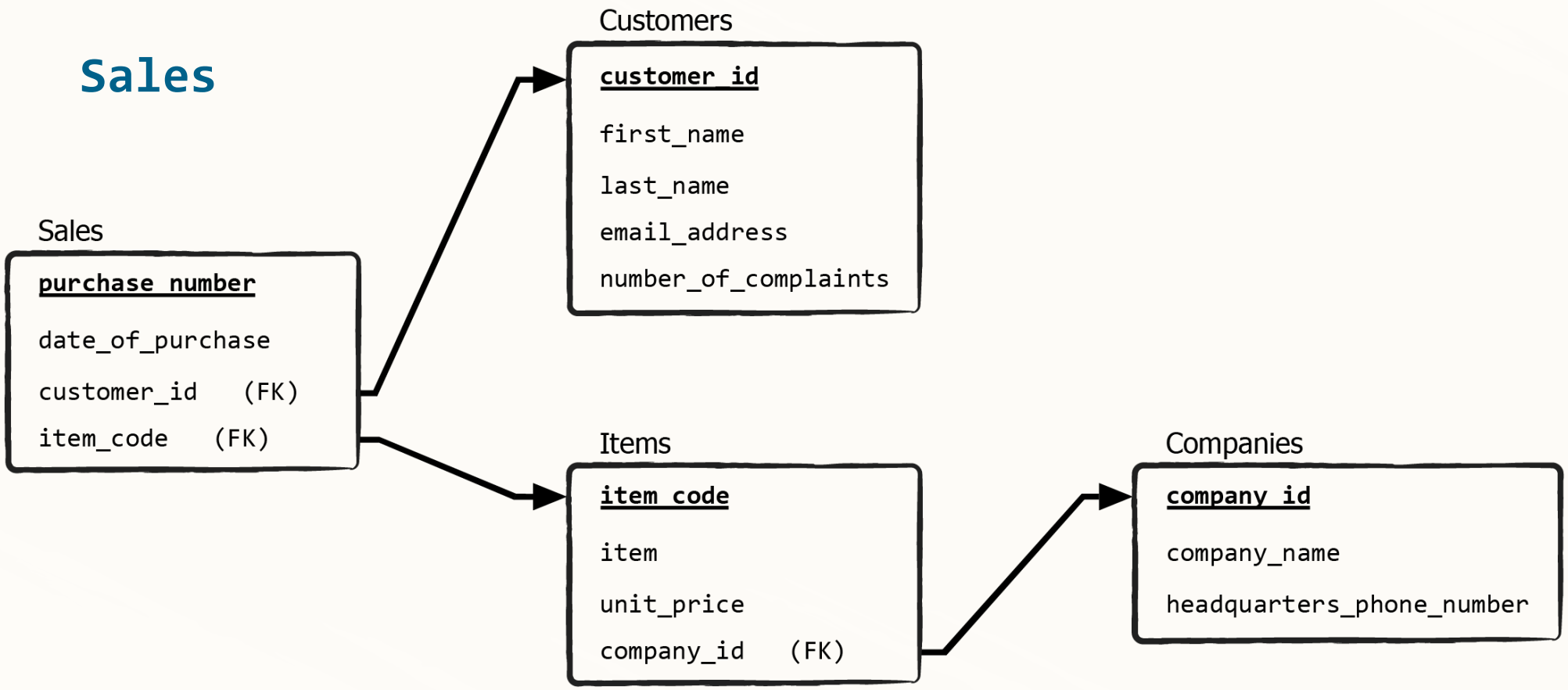
So far:

- Theory of Relational Databases
- SQL Theory
- Download and Installation of MySQL Workbench (provided by ORACLE®)

Creating a Database – Part I

Sales

Creating a Database – Part I



Creating a Database – Part I


```SQL
CREATE DATABASE [IF NOT EXISTS] database_name;
```

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- CREATE DATABASE

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **CREATE DATABASE**

  creates a database as an abstract unit

Creating a Database – Part I

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- [IF NOT EXISTS]

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **[IF NOT EXISTS]**

  verifies if a database with the same name exists already

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **[IF NOT EXISTS]**

  verifies if a database with the same name exists already

  - the brackets around mean the statement is *optional* (you could either type or omit the statement)

Creating a Database – Part I


```SQL
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- database_name

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **database_name**

  give a name that is short but at the same time as related to the content of the data as possible

Creating a Database – Part I

Sales

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **database_name**

  give a name that is short but at the same time as related to the content of the data as possible

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **database_name**

  give a name that is short but at the same time as related to the content of the data as possible

  - the SQL code is not case sensitive

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **database_name**

  give a name that is short but at the same time as related to the content of the data as possible

  - the SQL code is not case sensitive
  - in this element the quotes are optional

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- ; (the semicolon character)

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

- **; (the semicolon character)**

  it functions as a *statement terminator*

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

; (the semicolon character)

it functions as a *statement terminator*

- when your code contains more than a single statement, ; is indispensable

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

; (the semicolon character)

it functions as a *statement terminator*

- when your code contains more than a single statement, ; is indispensable
- will help you avoid errors sometimes

Creating a Database – Part I


```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

; (the semicolon character)

it functions as a *statement terminator*

- when your code contains more than a single statement, ; is indispensable
- will help you avoid errors sometimes
- will improve the readability of your code

Introduction to Data Types

365 DataScience

Introduction to Data Types

- We must *always* specify the *type* of data that will be inserted in *each* column of the table

Different data types represent different *types of information* that can be contained in a specific column

Introduction to Data Types

	Surname of a person:	
string	‘James’	

Introduction to Data Types

	Surname of a person:	
string	‘James’	

length

a measure used to indicate how many symbols a certain string has

Introduction to Data Types

	Surname of a person:	length	
string	‘James’	5 symbols	
	‘Jackson’	7 symbols	

length

a measure used to indicate how many symbols a certain string has

Introduction to Data Types

- Digits, symbols, or blank spaces can also be used in the string format

they will only convey text information

e.g. addresses: ‘Hugh Street 45’

Introduction to Data Types

size

indicates the memory space used by a data type

- measured in bytes

- 1 byte ~ 1 symbol

Introduction to Data Types

	Surname of a person:	length	size
string	‘James’	5 symbols	5 bytes

Introduction to Data Types

storage

the physical space in the computer drive's memory, where the data is being saved or stored


String Data Types
365 DataScience

String Data Types

- String – the text format in SQL

‘James’ – a variable of the string data type

= a variable of the alphanumeric data type

String Data Types

string data type		Storage	Example
character	CHAR	fixed	CHAR(5)

String Data Types

string data type		Storage	Example	*length* (symbols)	*size* (bytes)
character	CHAR	fixed	CHAR(5)		
			‘James’	5	5
			‘Bob’	3	5

CHAR(5)

5 represents the maximum number of symbols you are allowed to use in writing a value in this format

String Data Types

string data type		Storage	Example	*length* (symbols)	*size* (bytes)
character	CHAR	fixed	CHAR(5)		
			'James'	5	5
			'Bob'	3	5
variable character	VARCHAR	variable	VARCHAR(5)		
			'James'	5	5
			'Bob'	3	3

String Data Types

string data type		Maximum size (bytes)
character	CHAR	255
variable character	VARCHAR	65,535

String Data Types

string data type		Maximum size (bytes)	
character	CHAR	255	50% faster
variable character	VARCHAR	65,535	a lot more responsive to the data value inserted

String Data Types

Companies		
company_id	headquarters_phone_number	company
1	+1 (202) 555-0196	COA
2	+1 (202) 555-0152	COB
3	+1 (229) 853-9913	COC
4	+1 (618) 369-7392	COD

company `CHAR(3)`

Password:

the symbols cannot be more than 10 characters

password `VARCHAR(10)`

String Data Types

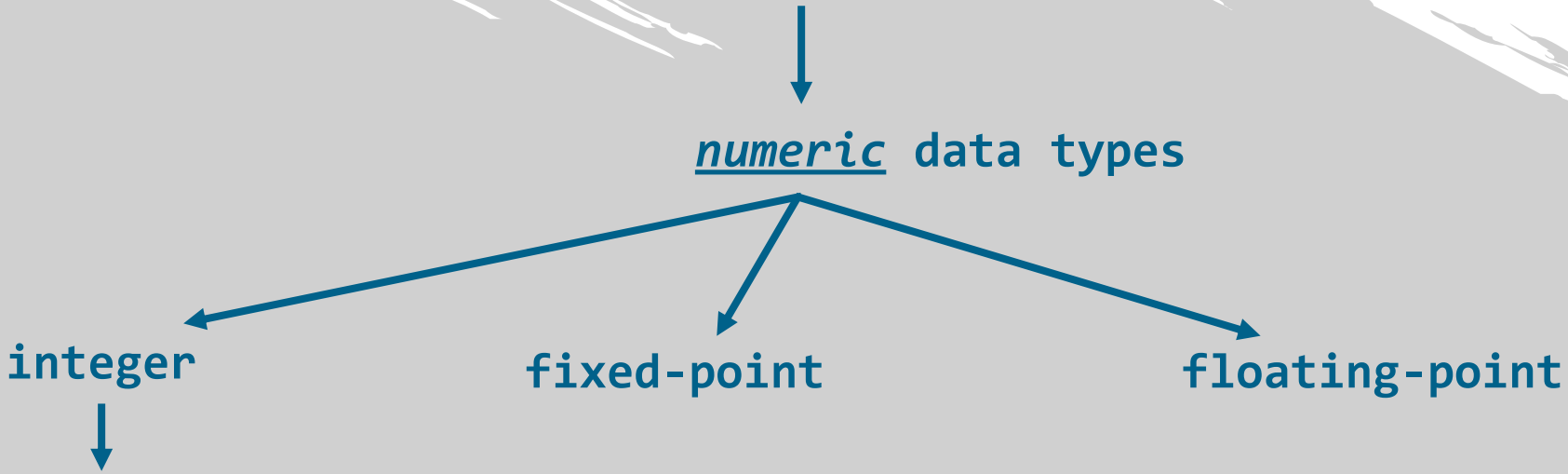
string data type		Example
character	CHAR	CHAR(5)
variable character	VARCHAR	VARCHAR(5)
ENUM ("enumerate")	ENUM	ENUM('M','F')

MySQL will show an error if you attempt to insert any value different from "M" or "F".


Integers
365 DataScience

Integers

text ≠ numbers



integers

whole numbers with no decimal point

e.g. 5; 15; -200; 1,000

INTEGER **INT**

Integers

numeric data type	*size* (bytes)	minimum value (signed/unsigned)	maximum value (signed/unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32,768	32,767
		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
		0	16,777,215
INT	4	-2,147,483,648	2,147,483,647
		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Integers

signed ≠ unsigned

if the encompassed range includes
both positive and negative values

Integers

numeric data type	*size* (bytes)	minimum value (signed/unsigned)	maximum value (signed/unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32,768	32,767
		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
		0	16,777,215
INT	4	-2,147,483,648	2,147,483,647
		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Integers

signed ≠ **unsigned**

if the encompassed range includes both positive and negative values

if integers are allowed to be only positive

Integers

numeric data type	*size* (bytes)	minimum value (signed/unsigned)	maximum value (signed/unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32,768	32,767
		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
		0	16,777,215
INT	4	-2,147,483,648	2,147,483,647
		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Integers

- *integer* data types are ‘signed’ by default

Integers

numeric data type	*size* (bytes)	minimum value (signed/unsigned)	maximum value (signed/unsigned)
TINYINT	1	-128	127
		0	256
SMALLINT	2	-32,768	32,767
		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
		0	16,777,215
INT	4	-2,147,483,648	2,147,483,647
		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Integers

- *integer* data types are ‘signed’ by default
- if you want to use a range containing only positive, ‘unsigned’ values, you would have to specify this in your query

Integers

numeric data type	*size* (bytes)	minimum value (signed/unsigned)	maximum value (signed/unsigned)
TINYINT	1	-128	127
		0	256
SMALLINT	2	-32,768	32,767
		0	65,535
MEDIUMINT	3	-8,388,608	8,388,607
		0	16,777,215
INT	4	-2,147,483,648	2,147,483,647
		0	4,294,967,295
BIGINT	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Integers

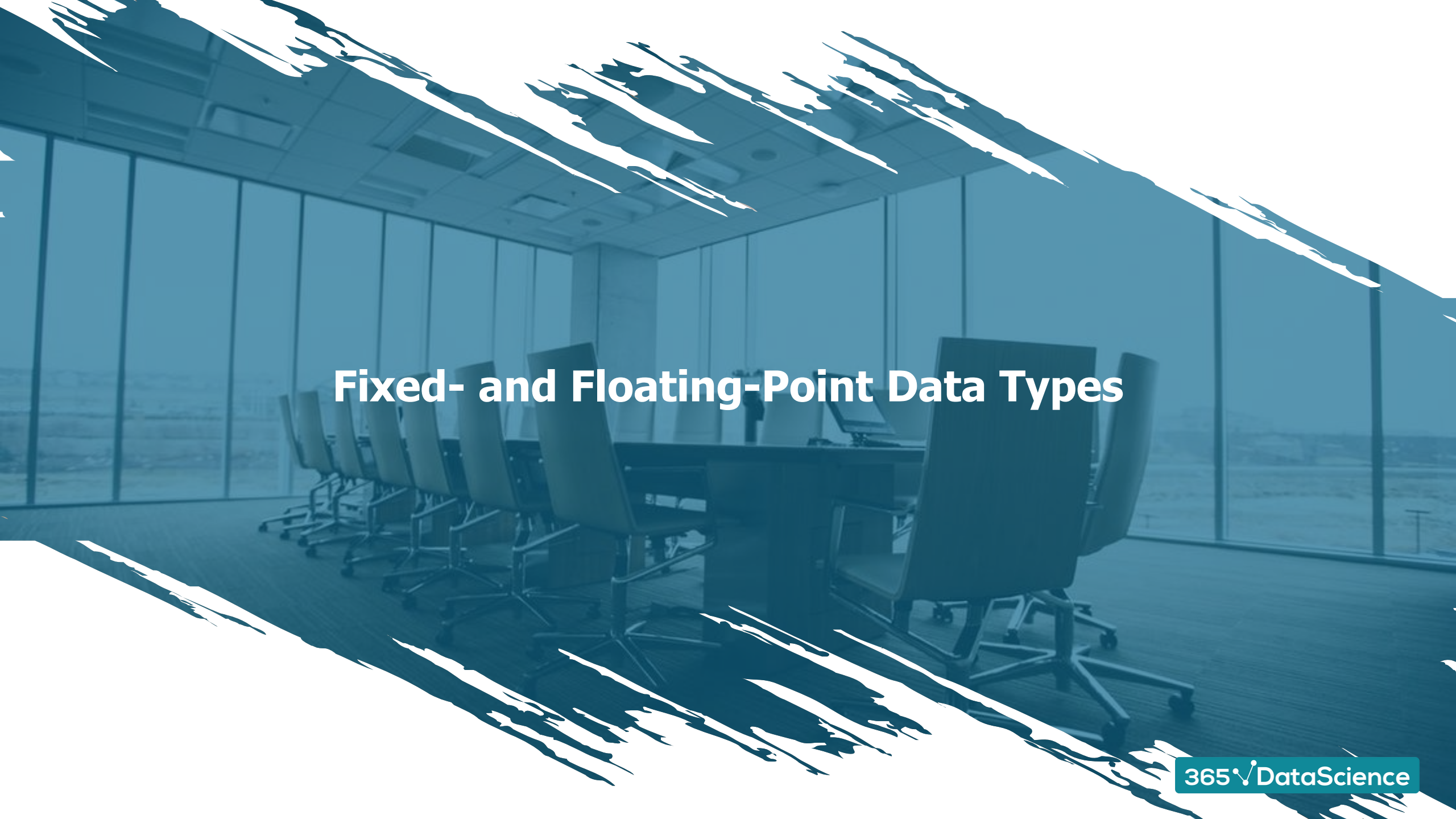
- Why not just use `BIGINT` all the time?

Integers

- Why not just use `BIGINT` all the time?
- e.g. if you are sure that, in a certain column, you won't need an integer smaller than 0 or greater than 100, `TINYINT` would do the job perfectly and you would not need more storage space per data point

Integers

- Why not just use `BIGINT` all the time?
- e.g. if you are sure that, in a certain column, you won't need an integer smaller than 0 or greater than 100, `TINYINT` would do the job perfectly and you would not need more storage space per data point
- a smaller integer type may increase the processing speed



Fixed- and Floating-Point Data Types

365 DataScience

Fixed- and Floating-Point Data Types

number:	precision	
10.523	5	

precision

refers to the number of digits in a number

Fixed- and Floating-Point Data Types

number:	precision	scale
10.523	5	3

scale

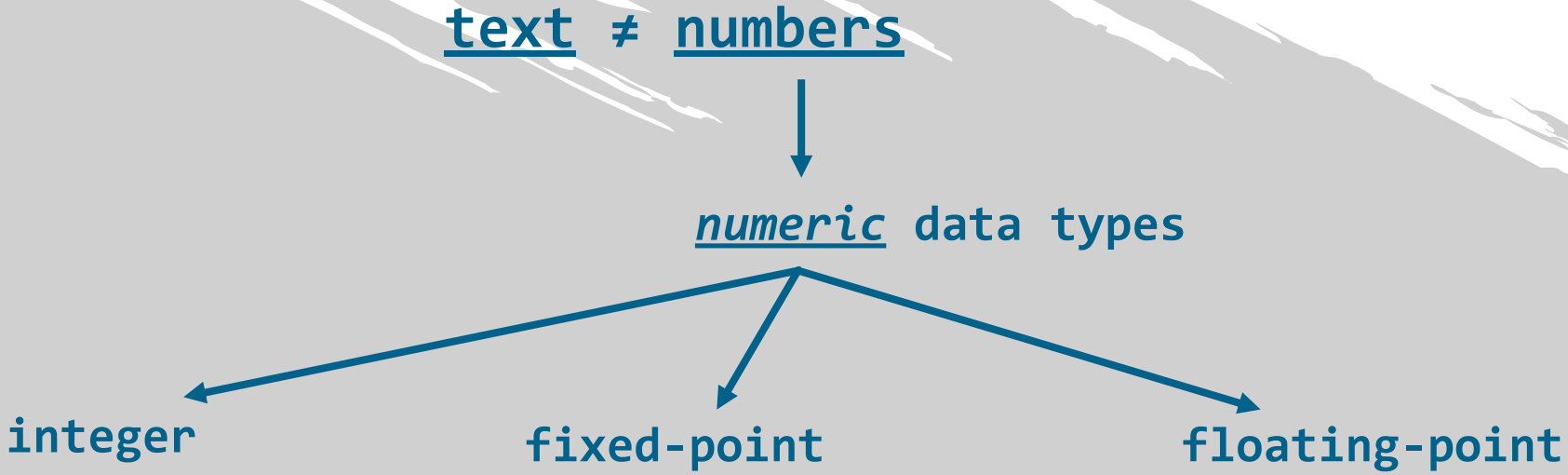
refers to the number of digits to the right of the decimal point in a number

Fixed- and Floating-Point Data Types

number:	precision	scale	
10.523	5	3	
36.875	5	3	

e.g. `DECIMAL ( 5 , 3 )`

Fixed- and Floating-Point Data Types



Fixed- and Floating-Point Data Types

- fixed-point data represent exact values

DECIMAL (5 , 3)	10.523
10.5	10.500
10.5236789	10.524 

Fixed- and Floating-Point Data Types

- fixed-point data represent exact values

when only one digit is specified within the parentheses, it will be treated as the precision of the data type

```
DECIMAL ( 7 )
```

1234567

```
DECIMAL ( 7, 0 )
```

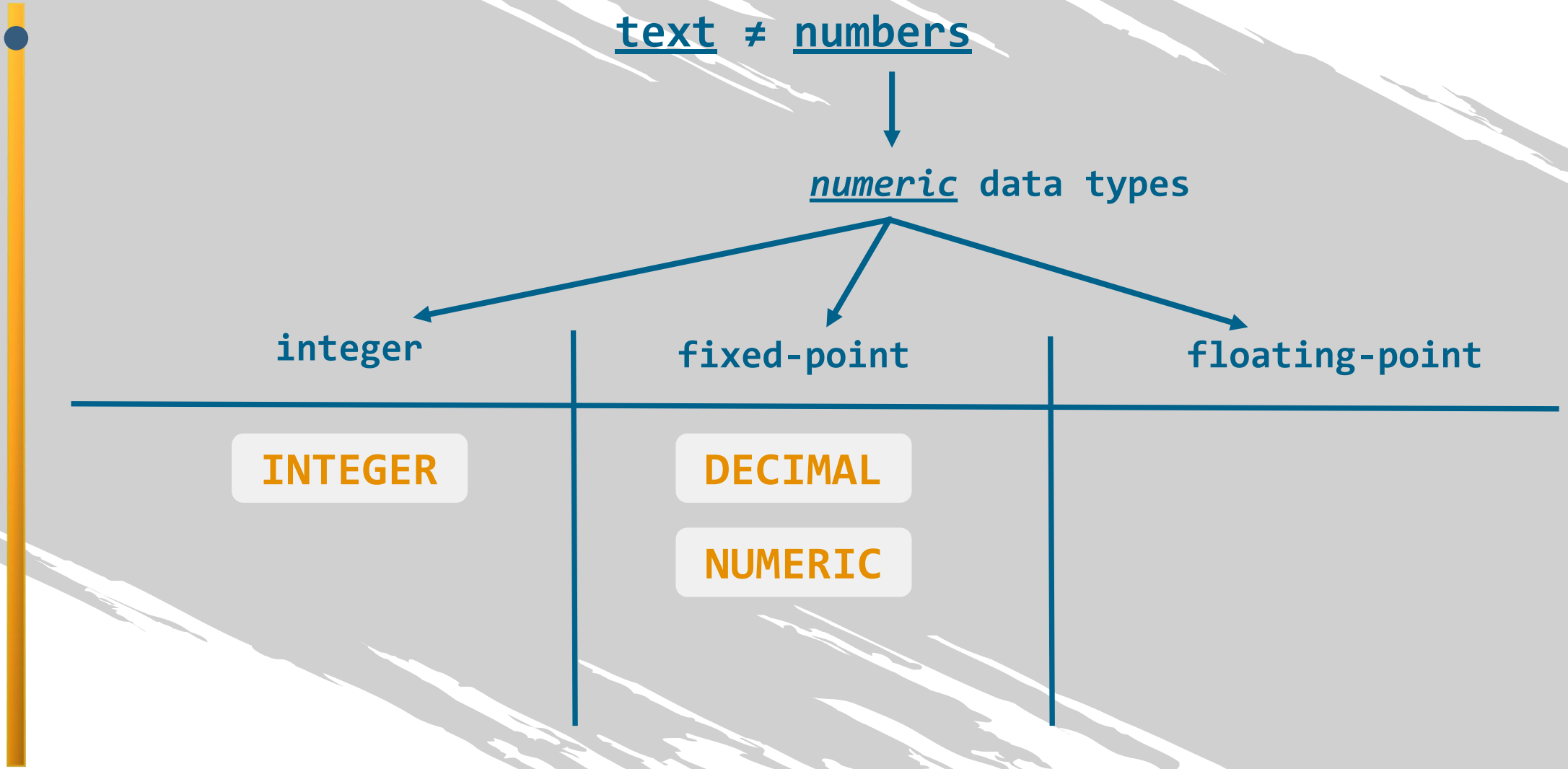
Fixed- and Floating-Point Data Types

- fixed-point data represent exact values

DECIMAL has a synonymous data type. It is called NUMERIC.

`DECIMAL` = `NUMERIC`

Fixed- and Floating-Point Data Types



Fixed- and Floating-Point Data Types

`DECIMAL` = `NUMERIC`

e.g. salaries

`NUMERIC ( p , s )`

precision: `p` = 7

scale: `s` = 2

---

e.g. `NUMERIC (7,2)` $ 75,000.50

Fixed- and Floating-Point Data Types

floating-point data type

- used for approximate values only
- aims to balance between range and precision (=> “floating”)

FLOAT (5 , 3)	10.523
10.5236789	10.524 
	(10.524 is an approximate value)

Fixed- and Floating-Point Data Types

the main difference between the fixed- and the floating-point type is in the way the value is represented in the memory of the computer

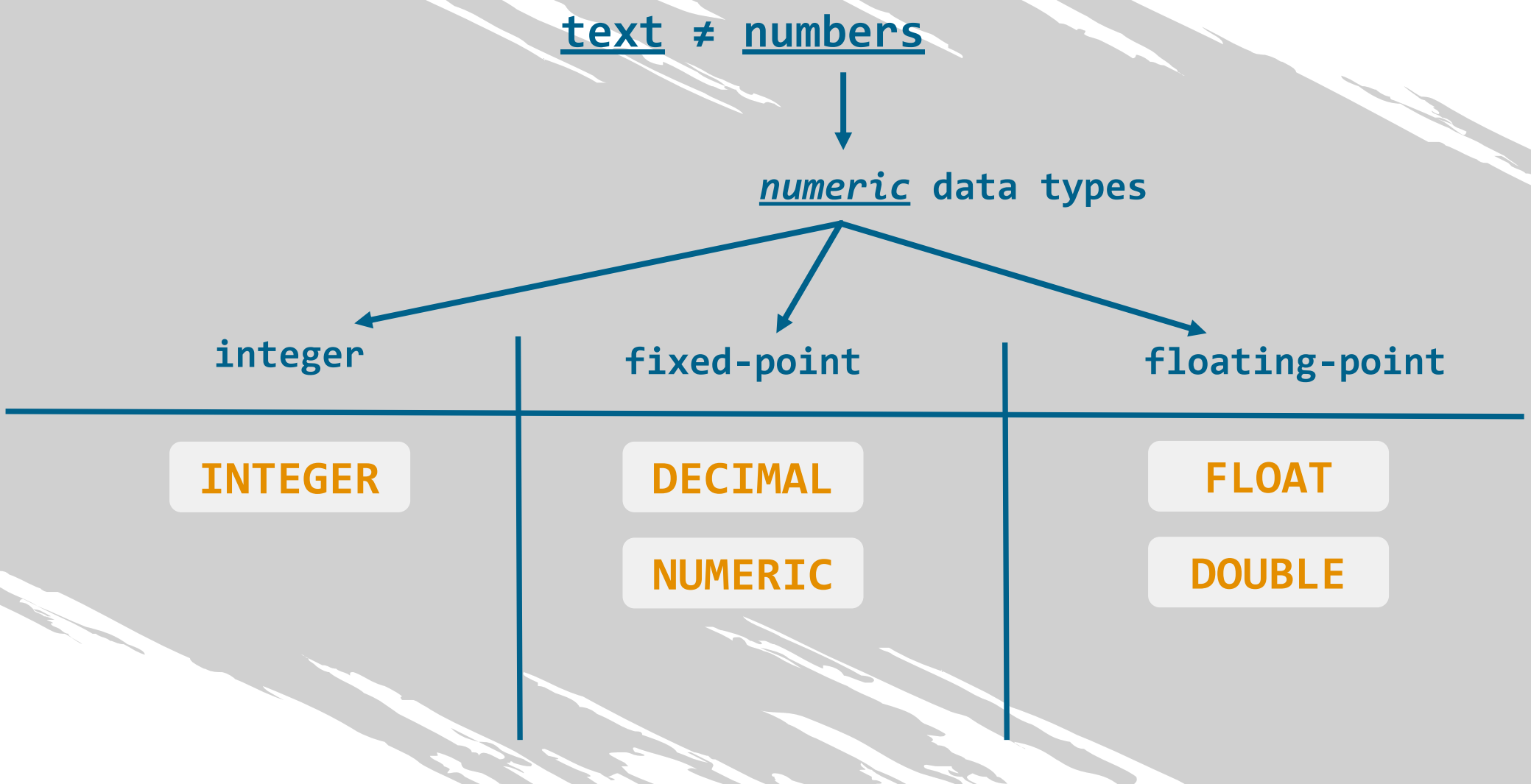
DECIMAL (5 , 3)

10.5236789 → 10.524 

FLOAT (5 , 3)

10.5236789 → 10.524 

Fixed- and Floating-Point Data Types



Fixed- and Floating-Point Data Types

Floating-point data type	size (bytes)	precision	maximum number of digits
FLOAT	4	*single*	23
DOUBLE	8	*double*	53

Other Useful Data Types

365 DataScience

Other Useful Data Types

DATE

used to represent a date in the format YYYY-MM-DD

1st of January 1000 - 31st of December 9999

---

e.g. 25th of July 2018: `'2018-07-25'`

Other Useful Data Types

DATE +  = DATETIME

next to the date, we could save the time:

YYYY-MM-DD HH:MM:SS[.fraction]

0 - 23:59:59.999999

---

e.g. 25th of July 2018 9:30 a.m.: ‘2018-07-25 9:30:00’

Other Useful Data Types

DATETIME

represents the date shown on the calendar and the time shown on the clock

vs.

TIMESTAMP

used for *a well-defined, exact point in time*

Other Useful Data Types

TIMESTAMP

used for *a well-defined, exact point in time*

1st of January 1970 UTC – 19th of January 2038, 03:14:07 UTC

- records the moment in time as the number of seconds passed after the 1st of January 1970 00:00:00 UTC

---

e.g. 25th of July 2018: `1,535,155,200`

Other Useful Data Types

TIMESTAMP

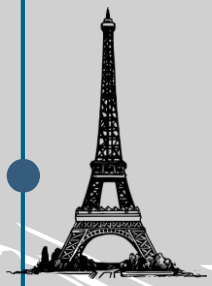
- representing a moment in time as a number allows you to easily obtain the difference between two TIMESTAMP values

e.g. end time:	‘2018-07-25 10:30:00’ UTC	TIMESTAMP
-		
start time:	‘2018-07-25 09:00:00’ UTC	TIMESTAMP
	5,400	TIMESTAMP

Other Useful Data Types

TIMESTAMP is appropriate if you need to handle time zones

London  = 1:00 a.m

Paris  = 2:00 a.m

‘1970-01-01 01:00:00’ UTC

Other Useful Data Types

string, *date*, and *time* data types	*numeric* data types
CHAR	INTEGER
VARCHAR	DECIMAL
DATE	NUMERIC
DATETIME	FLOAT
TIMESTAMP	DOUBLE

data must be written within quotes

only numeric values are written without quotes

Other Useful Data Types

BLOB

Binary Large OBject

- refers to a file of binary data – data with 1s and 0s
- involves saving *files* in a record

Other Useful Data Types

Customers					
customer_id	first_name	last_name	email_address	number_of_complaints	photo
1	John	McKinley	john.mackinley@365careers.com	0	
2	Elizabeth	McFarlane	e.mcfarlane@365careers.com	2	
3	Kevin	Lawrence	kevin.lawrence@365careers.com	1	
4	Catherine	Winnfield	c.winnfield@365careers.com	0	*.jpg

365 DataScience

Other Useful Data Types

string, *date*, and *time* data types	*numeric* data types
CHAR	INTEGER
VARCHAR	DECIMAL
DATE	NUMERIC
DATETIME	FLOAT
TIMESTAMP	DOUBLE

365 DataScience

Creating a Table

365 DataScience

Creating a Table


```sql
CREATE DATABASE [IF NOT EXISTS] sales;
```

Creating a Table

SQL

```sql
CREATE DATABASE [IF NOT EXISTS] sales;

CREATE TABLE table_name ( );
```

Creating a Table

SQL

```
CREATE DATABASE [IF NOT EXISTS] sales;
CREATE TABLE table_name ( );
```

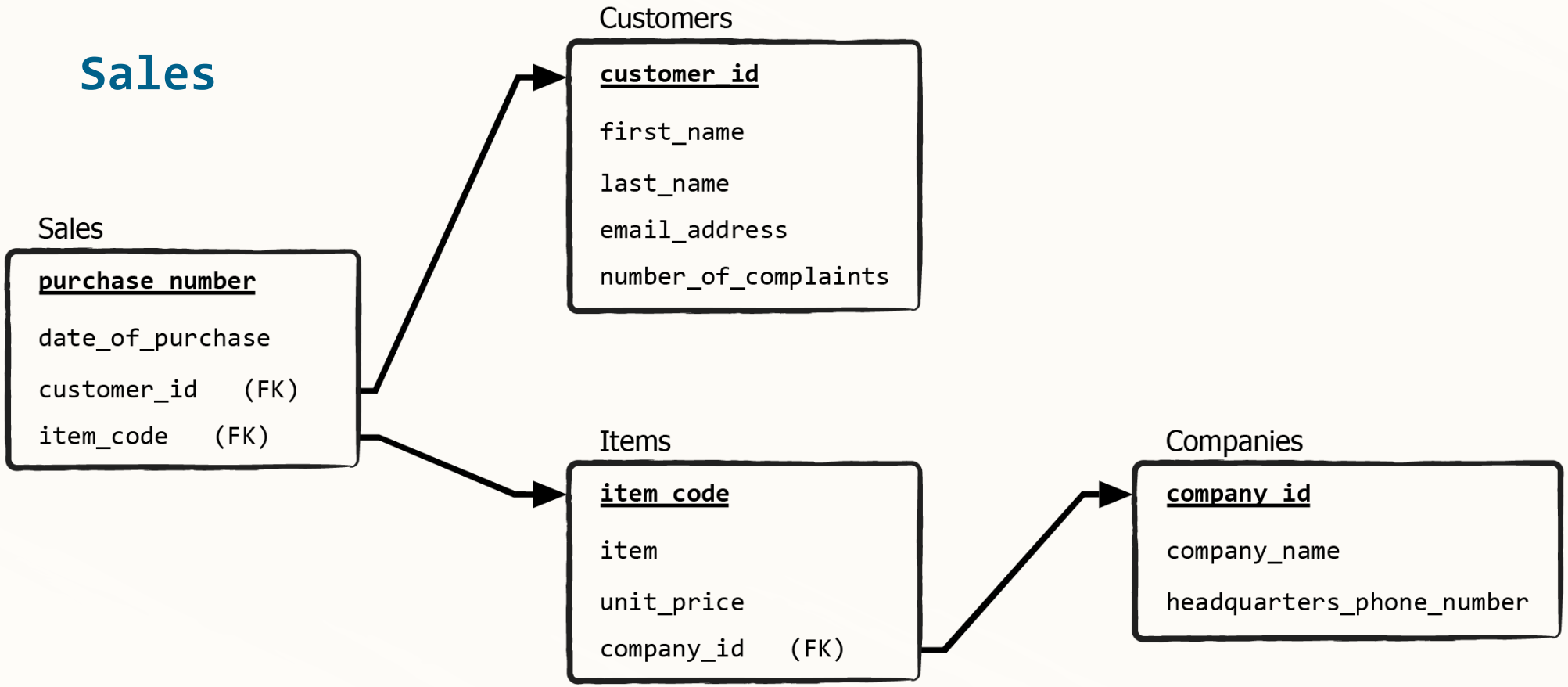
- compulsory requirement: add *at least* one column

Creating a Table


```sql
CREATE TABLE table_name
(
    column_1 data_type constraints,
    column_2 data_type constraints,
    …
    column_n data_type constraints
);
```

Creating a Table

Sales



Creating a Table

AUTO_INCREMENT

frees you from having to insert all purchase numbers manually through the INSERT command at a later stage

Creating a Table

AUTO_INCREMENT

frees you from having to insert all purchase numbers manually through the INSERT command at a later stage

- assigns 1 to the first record of the table and automatically *increments* by 1 for every subsequent row

Creating a Table

AUTO_INCREMENT

sales

purchase_number
1
2
3
4
…
n

Creating a Table

AUTO_INCREMENT

sales

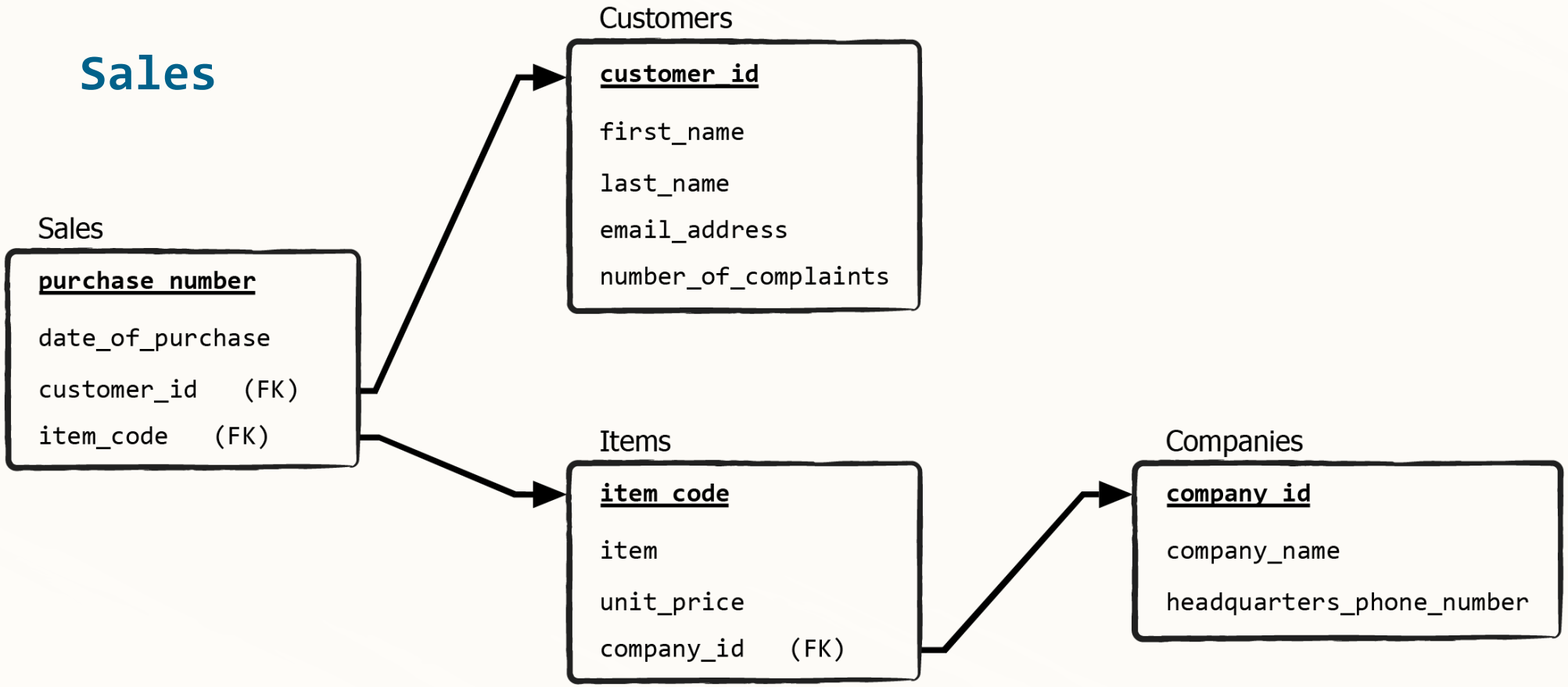
purchase_number
1
2
3
4
…
n

SALES

purchase_number	date_of_purchase	customer_id	item_code
1	03/09/2016	1	A_1
2	02/12/2016	2	C_1
3	15/04/2017	3	D_1
4	24/05/2017	1	B_2
5	25/05/2017	4	B_2
6	06/06/2017	2	B_1
7	10/06/2017	4	A_2
8	13/06/2017	3	C_1
9	20/07/2017	1	A_1
10	11/08/2017	2	B_1

Creating a Table

Sales



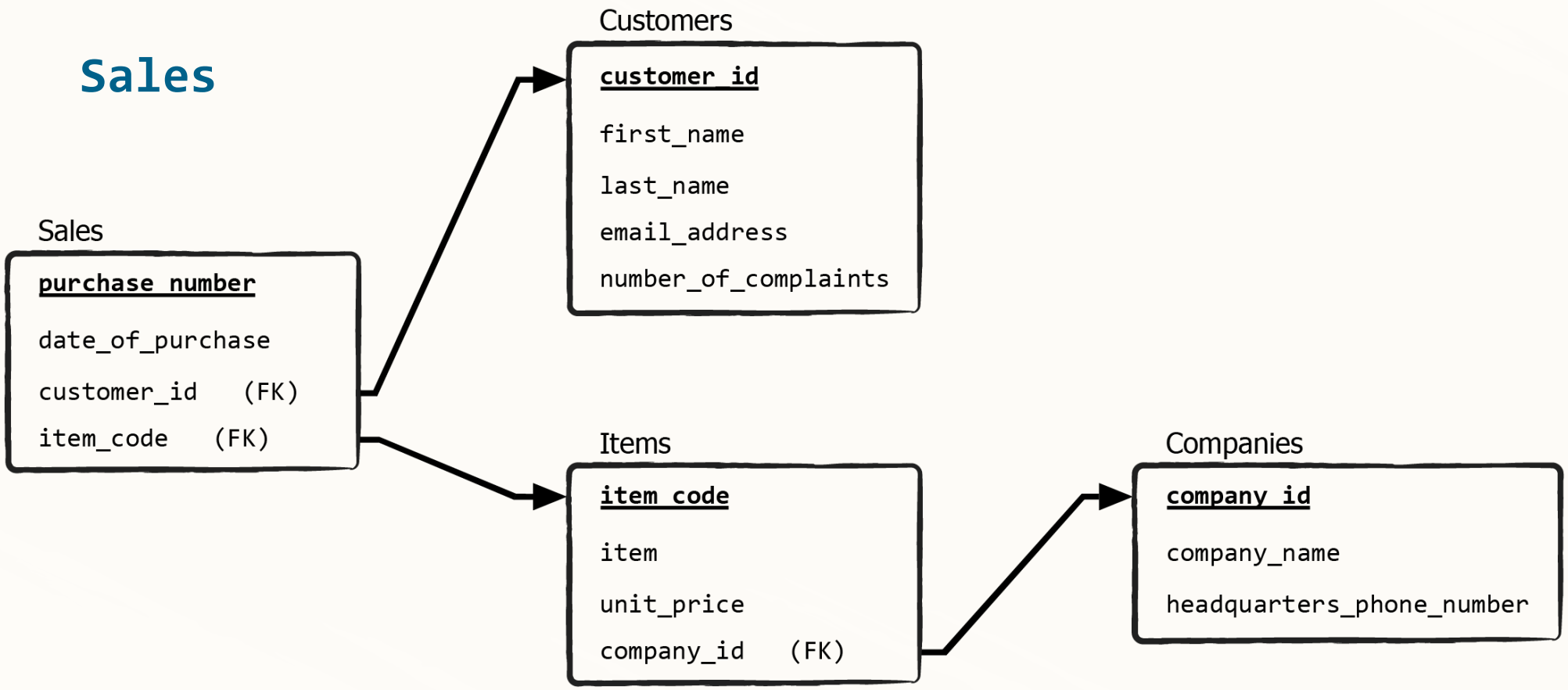

Using Databases and Tables
365 DataScience

Using Databases and Tables

- **queries**

  one of their main features is to manipulate data within a database

Using Databases and Tables



Using Databases and Tables

queries

one of their main features is to manipulate data within a database

e.g.


```
SELECT * FROM customers;
```

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

---

SQL Objects:

- SQL table

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

---

SQL Objects:

- SQL table
- views
- stored procedures
- functions

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

1) set a default database

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

1) set a default database


```
USE sales;

SELECT * FROM customers;
```

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

1) set a default database

2) call a table from a certain database

Using Databases and Tables

- Whenever you would like to refer to an *SQL object* in your queries, you must specify the database to which it is applied

1) set a default database

2) call a table from a certain database


```
database_object . sql_object
```

Using Databases and Tables


```
database_object . sql_object
```

. – “dot operator”

signals the existence of a connection between the two object types

Using Databases and Tables


```
database_object . sql_object

SELECT * FROM sales.customers;
```

. – “dot operator”

signals the existence of a connection between the two object types

Additional Notes on Using Tables

365 DataScience

Additional Notes on Using Tables

query

a command you write in SQL with the idea of either *retrieving information* from the database on which you are working, or, alternatively, to *insert*, *update*, or *delete* data from it

Additional Notes on Using Tables

query

a command you write in SQL with the idea of either *retrieving information* from the database on which you are working, or, alternatively, to *insert*, *update*, or *delete* data from it

- it is a representation of a complete logical thought

Additional Notes on Using Tables

- **the DROP statement**

  used for deleting an SQL object

Additional Notes on Using Tables

the DROP statement

used for deleting an SQL object


```
DROP TABLE table_name;
```

Additional Notes on Using Tables

the DROP statement

used for deleting an SQL object


```sql
DROP TABLE table_name;

DROP TABLE sales;
```