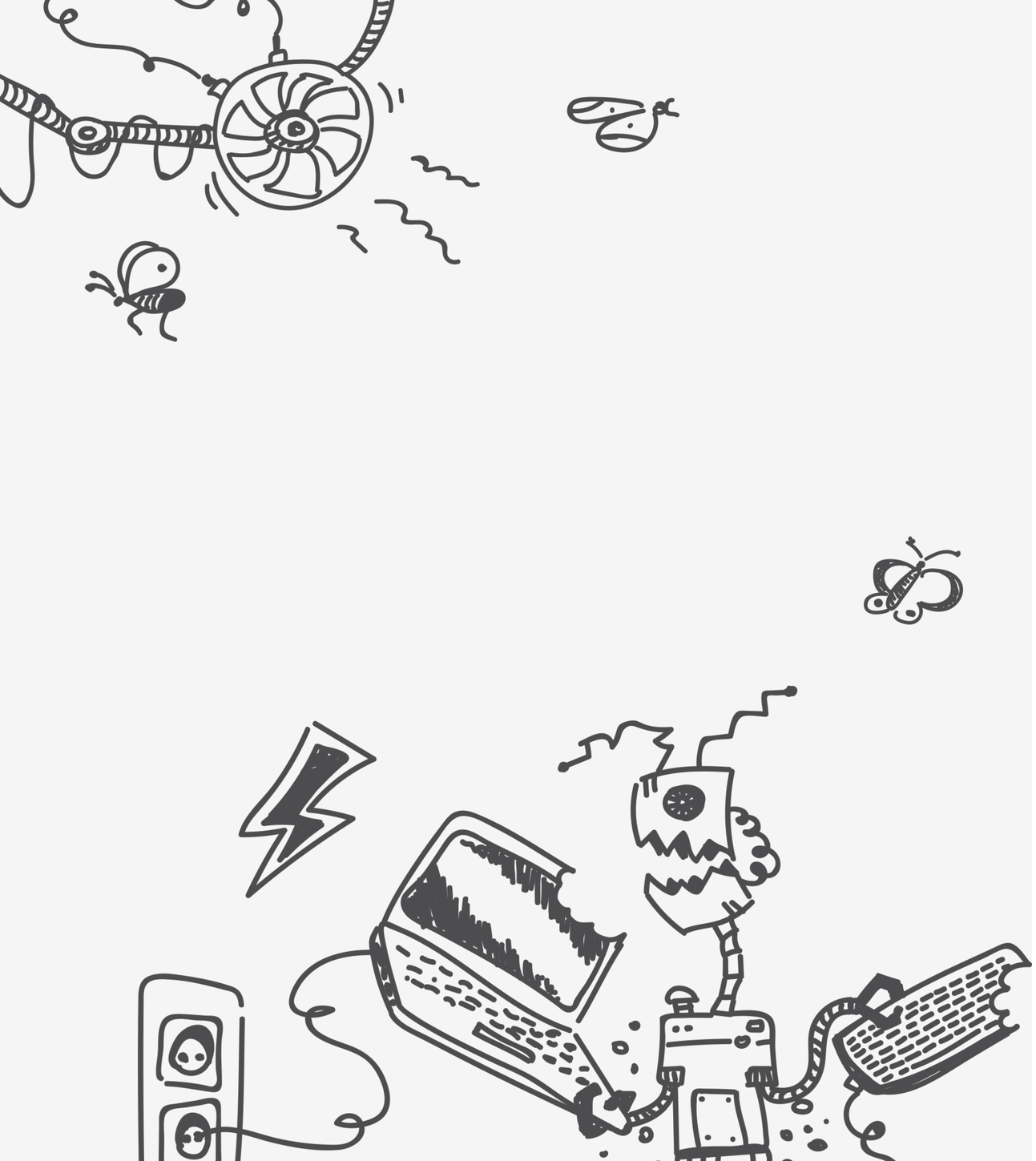
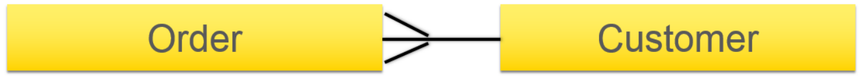


Relacje między tabelami - jeden do wielu

Relacja jeden do wielu

Relacja, w której jeden element z danej tabeli, może być połączony z wieloma elementami z innej tabeli.

Klient może mieć wiele zamówień.
Zamówienie musi mieć tylko jednego klienta.



Relacja jeden do wielu

Relację jeden do wielu tworzymy przez dodanie dodatkowej kolumny, w której trzymamy wartość jednoznacznie identyfikującą obiekt z drugiej tabeli. W większości przypadków jest to **klucz główny** drugiej tabeli, ale jeśli tabela ma ustawiony modyfikator `UNIQUE` na którejś z kolumn, to ta kolumna również może posłużyć do założenia relacji.

```
CREATE TABLE orders(
  order_id serial NOT NULL,
  customer_id int NOT NULL,
  order_details varchar(255),
  PRIMARY KEY(order_id),
  FOREIGN KEY(customer_id)
  REFERENCES customers(customer_id)
);
```

Relacja jeden do wielu

```
INSERT INTO orders(customer_id, order_details)
VALUES (3, 'Zamowienie1'), (3, 'Zamowienie2'), (1, 'Zamowienie3');

SELECT * FROM customers JOIN orders
ON customers.customer_id=orders.customer_id
WHERE customers.customer_id=3;
```

```
+-------------+--------+----------+------------+--------------+
| customer_id | name   | order_id | customer_id|order_details |
+-------------+--------+----------+------------+--------------+
|           3 | Wojtek |        1 |           3|Zamówienie1   |
|           3 | Wojtek |        2 |           3|Zamówienie2   |
+-------------+--------+----------+------------+--------------+
```

FOREIGN KEY

- Atrybut `FOREIGN KEY` dopisany do jakiejś kolumny mówi po prostu, że ta kolumna wskazuje na klucz główny (lub kolumnę z modyfikatorem `UNIQUE`) innej tabelki.
- Zabezpiecza on przed wprowadzeniem niepoprawnych danych, np. nie pozwoli wpisać wartości klucza, która nie występuje w drugiej tabeli. Przyspiesza też pracę bazy danych.

FOREIGN KEY – dodawanie elementu

Mamy dwie tabele: `customers` i `orders` połączone relacją jeden do wielu:

```
SELECT * FROM customers;
```

```
+-------------+-------+
| customer_id | name  |
+-------------+-------+
|           1 | Jacek |
|           3 | Paweł |
+-------------+-------+
```

```
holonet=# INSERT INTO orders (customer_id, order_details) VALUES (10, 'zamówienie 100');
ERROR:  insert or update on table "orders" violates foreign key constraint "orders_customer_i
DETAIL:  Key (customer_id)=(10) is not present in table "customers".
```

FOREIGN KEY – dodawanie elementu

Mamy dwie tabele: `customers` i `orders` połączone relacją jeden do wielu:

```
SELECT * FROM customers;
```

```
+-------------+-------+
| customer_id | name  |
+-------------+-------+
|           1 | Jacek |
|           3 | Paweł |
+-------------+-------+
```

```
holonet=# INSERT INTO orders (customer_id, order_details) VALUES (10, 'zamówienie 100');
ERROR:  insert or update on table "orders" violates foreign key constraint "orders_customer_i
DETAIL:  Key (customer_id)=(10) is not present in table "customers".
```

→ Jeżeli w drugiej tabeli nie ma klucza głównego, do którego chcemy się odnieść przez klucz zewnętrzny, to SQL zwróci nam błąd.

FOREIGN KEY – usuwanie elementu

```
SELECT * FROM orders;
```

```
+-------------+--------+
| customer_id | street |
+-------------+--------+
|           3 | xxx    |
+-------------+--------+
```

```
DELETE FROM customers
WHERE customer_id = 3;
```

FOREIGN KEY – usuwanie elementu

```
ERROR:  update or delete on table "customers" violates foreign key
    constraint "orders_customer_id_fkey" on table "orders"
    DETAIL:  Key (customer_id)=(3) is still referenced from table "orders".x
```

Co się stało?

Baza danych nie pozwoliła usunąć rekordu z tabeli `customers`, ponieważ w tabeli `orders`, powiązanej relacją z tabelą klientów, znajduje się wpis, który odnosi się do tego rekordu.

Co robić?!

→ można najpierw usunąć odpowiednie wpisy z tabeli `orders`, potem dopiero usunąć wpisy tabeli `customers`,

→ albo użyć `ON DELETE CASCADE` podczas definiowania tabeli. Jak to zrobić – dowiecie się przy okazji omawiania relacji jeden-do-jednego.