Temat: *Normalizacja bazy danych.*

Przykładowa nieznormalizowana baza danych

id_zamowienia	nazwa_klienta	adres_klienta	nazwa_towaru	cena_towaru
1	Jan Kowalski	Kwiatowa 12	Bułka	0,35
2	Jan Kowalski	Kwiatowa 12	Bagietka	1,15
3	Jan Kowalski	Kwiatowa 12	Pączek	1
4	Jan Nowak	Liściasta 21	Bułka	0,35
5	Paweł Kowalczyk	Liściasta 37	Bułka	0,35

Anomalie przy usuwaniu
Kasując zamówienie o ID 5, jednocześnie kasujemy dane o kliencie Pawle Kowalczyku. Chcieliśmy tylko usunąć zamówienie, a jednocześnie tracimy dane klienta.
Anomalie przy wstawianiu
Jak dodać do bazy danych klienta, który jeszcze nic nie kupił? Jak dodać do bazy danych produkt, który jeszcze nie został przez nikogo zakupiony?
Anomalie przy modyfikacji
Gdyby bułka w sklepie podrożała o 10 groszy, trzeba by edytować jej cenę w aż 3 miejscach. A co, gdy się pomylimy? Lub zapomnimy edytować cenę w jednym miejscu?
Redundancja
Dane Jana Kowalskiego powtarzają się w aż 3 wierszach.

Naprawa bazy – tak aby nie występowały w niej podobne anomalie to **normalizacja**.

Normalizacja
Normalizacja to zespół czynności, które mają doprowadzić bazę do postaci normalnej. Robi się to między innymi poprzez wyeliminowanie redundancji czy wprowadzenie dodatkowych tabel i relacji między nimi.

Postać normalna
Postać normalna bazy danych to taka, w której nie występują anomalie (takie jak redundancja).

Ta sama baza danych po normalizacji

id_klienta (k. gł.)	nazwa_klienta	adres_klienta
1	Jan Kowalski	Kwiatowa 12
2	Jan Nowak	Liściasta 21
3	Paweł Kowalczyk	Liściasta 37

id_towaru (k. gł.)	nazwa_towaru	cena_towaru
1	Bułka	0,35
2	Bagietka	1,15
3	Pączek	1

id_zamówienia (klucz główny)	id_klienta (klucz obcy)	id_towaru (klucz obcy)
1	1	1
2	1	2
3	1	3
4	2	1
5	3	1

Korzyści wynikające z normalizacji bazy danych

- Zmniejszamy ogólną liczbę przechowywanych danych
- Tabele są łatwiejsze do edycji – wystarczy podmienić wartość w jednym miejscu
- Wprowadzenie normalizacji pozwala oszczędzić miejsce na dysku
- Rozwiązujemy problemy z anomaliami dodawania, modyfikacji i usuwania informacji
- Spowalniamy niektóre operacje (te związane z koniecznością łączenia kilku tabel), ale niektóre przyśpieszamy (te związane z operowaniem na pojedynczych tabelach)

Aby baza danych mogła być nazywana znormalizowaną, musi spełniać dwa główne warunki.

Warunek 1
Pierwszy warunek mówi o prawidłowym projektowaniu tabel. Aby warunek ten był spełniony, każda tabela musi:

a) opisywać tylko jeden obiekt (jeżeli tabela przechowuje dane o produktach, to nie może jednocześnie przechowywać adresów klientów ani informacji o pracownikach sklepu)
b) wartości poszczególnych kolumn muszą być elementarne, atomowe, niepodzielne, nie mogą zawierać kolekcji danych (jeżeli kolumna tabeli przechowuje nazwę ulicy przy której mieszka klient, to nie może po przecinku przechowywać jeszcze województwa i kraju)
c) kolejność wierszy nie może mieć znaczenia dla działania bazy

Baza nieznormalizowana

id_plci	nazwa_plci	imiona
1	mężczyzna	Tomek, Kacper
2	kobieta	Ala, Ela

Powyższa tabela łamie podpunkt b) - kolumna „imiona" nie spełnia warunku niepodzielności. Baza nie jest więc znormalizowana. Tak zaprojektowana baza byłaby trudna i nieefektywna w użyciu. Dla przykładu – mamy za zadanie policzyć, ile jest imion kobiecych. Nie jest to niewykonalne, ale jest umiarkowanie trudne – kosztowne w czas i zasoby komputera. Kwerenda która miałaby to zrobić byłaby dość złożona.

Baza po normalizacji

id_płci (klucz główny)	nazwa_płci
1	mężczyzna
2	kobieta

id_imienia (klucz główny)	id_płci (klucz obcy)	imie
1	1	Tomek
2	1	Kacper
3	2	Ala
4	2	Ela

Warunek 2
Drugi warunek mówi o tym, że w jednej tabeli jedna kolumna nie może wynikać bezpośrednio z drugiej.

Baza nieznormalizowana

id_pracownika	Stanowisko	pensja_netto	pensja_brutto	pracownik
1	kierownik	2780	3450	Jan Kowalski
2	kierownik	2780	3450	Adam Nowak
3	prezes	6850	7750	Paweł Zwyczajny

Po pierwsze – kolumna „pensja_brutto” może być wyliczona na podstawie kolumny „pensja_netto”. To jest duży błąd, ponieważ podczas zmieniania pensji ktoś mógłby edytować pensję netto, a zapomnieć edytować pensję brutto.
Drugi błąd jest równie poważny. Zakładając, że osoby na tym samym stanowisku zarabiają tyle samo, powinniśmy wydzielić stawki przypisane do danego stanowiska do osobnej tabeli. Chcąc podwyższyć pensję wszystkim kierownikom, robilibyśmy to wtedy w jednym miejscu, a nie wielu naraz.

Baza po normalizacji

id_pracownika (klucz główny)	id_stanowiska (klucz obcy)	pracownik
1	1	Jan Kowalski
2	1	Adam Nowak
3	2	Paweł Zwyczajny

id_stanowiska (klucz główny)	stanowisko	pensja_netto
1	kierownik	2780
2	prezes	6850

Ponieważ normalizacja zakłada podział bazy na wiele mniejszych tabel, aby pobrać informacje z kilku tabel naraz trzeba wykonywać kosztowne operacje łączenia. Są to tak zwane operacje *JOIN* języka *SQL*.
Operacje *JOIN* są dość kosztowne, używając dużo pamięci operacyjnej (RAM). Zwykle nie są to jednak różnice aż tak duże, by miało to być uciążliwe. W przypadku, gdy dla projektanta bazy ważne są nawet ułamki milisekund, czasami umyślnie wprowadza się anomalie takie jak redundancja, by oszczędzić moc obliczeniową komputera.

Taką operację – umyślne wprowadzanie anomalii – nazywa się **denormalizacją**.