## Fundamenty Analityki Danych: Od Procesu ETL do Języka DAX

### Część 1: Jak Powstaje i Jak Wygląda Hurtownia Danych

Zanim zaczniemy projektować model czy pisać formuły, musimy zrozumieć, czym jest hurtownia danych i, co ważniejsze, *jak* powstaje z surowych systemów transakcyjnych.

## Wprowadzenie: Czym jest Hurtownia Danych?

Hurtownia danych (ang. *Data Warehouse*, DWH) to centralny system repozytorium danych, który gromadzi i integruje dane z wielu różnych źródeł w organizacji (takich jak systemy sprzedaży, finanse, marketing, HR).

Głównym celem hurtowni danych **nie jest** obsługa bieżących transakcji (jak w sklepie internetowym – tym zajmują się bazy OLTP), ale umożliwienie **analizy i raportowania** (OLAP). Hurtownie są zoptymalizowane pod kątem szybkiego odczytu i agregacji ogromnych ilości danych historycznych, aby wspierać podejmowanie decyzji biznesowych (Business Intelligence, BI).

Fundamentem większości nowoczesnych hurtowni danych jest **modelowanie wymiarowe**, którego kluczowymi komponentami są tabele faktów i tabele wymiarów. Ale jak się do nich dochodzi?

## Proces Transformacji: Jak Baza Danych Staje się Hurtownią (ETL/ELT)

Zanim będziemy mogli analizować dane w schemacie gwiazdy, musimy je *stworzyć*. Dane w bazie sklepu internetowego nie są hurtownią – trzeba je przebudować w specjalnym procesie.

### Problem Dwóch Światów (OLTP vs. OLAP)

Baza danych (np. sklepu internetowego) i hurtownia danych to dwa zupełnie różne światy, zaprojektowane do przeciwnych celów:

1. **Baza Transakcyjna (OLTP - Online Transaction Processing):**
   * **Cel:** Szybkie *zapisywanie* danych (np. nowe zamówienie).
   * **Struktura:** Wysoce **znormalizowana** (rozbita na dziesiątki małych tabel), aby uniknąć powtarzania danych. Analiza jest tu ekstremalnie wolna.
2. **Hurtownia Analityczna (OLAP - Online Analytical Processing):**
   * **Cel:** Szybkie *odczytywanie* i agregowanie danych (np. "Jaka była sprzedaż w regionach?").
   * **Struktura:** Wysoce **zdenormalizowana** (Schemat Gwiazdy), aby zminimalizować liczbę połączeń i zmaksymalizować prędkość odczytu.

Nie możemy po prostu "skopiować" bazy OLTP i nazwać jej hurtownią. Musimy ją całkowicie **przebudować**. Ten proces przebudowy nazywa się **ETL**.

### Rozwiązanie: Proces ETL – Most Między Bazą a Hurtownią

Proces transformacji bazy w hurtownię to zbiór procedur inżynierii danych, określany klasycznym akronimem **ETL (Extract, Transform, Load)**.

To jest "silnik", który zasysa surowe dane transakcyjne i wypluwa czyste, piękne tabele faktów i wymiarów, gotowe do analizy.

#### 1. E = Extract (Ekstrakcja): Pobieranie Surowca

To pierwszy etap, w którym pobieramy dane z ich systemów źródłowych.

* **Co robimy?** Łączymy się z różnymi źródłami w firmie: główną bazą danych SQL (OLTP), systemem CRM, plikami Excel z budżetami, danymi z API (np. Google Analytics).
* **Wyzwanie:** Musimy pobrać te dane, nie "zabijając" systemów produkcyjnych. Zazwyczaj robi się to w nocy lub pobiera tylko zmiany (Delta Load).

#### 2. T = Transform (Transformacja): Serce Procesu

To jest najważniejszy i najbardziej złożony etap. To tutaj surowa baza danych **staje się** hurtownią danych. Surowe dane lądują w obszarze przejściowym (Staging Area), gdzie dzieje się cała magia.

* **Kluczowe działania:**
  + **Czyszczenie (Cleansing):** Poprawiamy błędy ("Polska", "PL" -> "Polska"), usuwamy duplikaty, uzupełniamy braki.
  + **Integracja (Integration):** Łączymy dane z różnych źródeł (np. dane klienta z SQL z jego aktywnością z Google Analytics).
  + **Modelowanie i Denormalizacja (TO JEST KLUCZOWE):**
    - **Tworzenie Wymiarów:** Bierzemy 5 znormalizowanych tabel z bazy OLTP (np. Produkty, Kategorie, Producenci) i **łączymy je (JOIN) w jedną, szeroką tabelę** DimProduct.
    - **Tworzenie Faktów:** Bierzemy tabele transakcyjne (np. Zamowienia i SzczegolyZamowienia), agregujemy je i wyciągamy z nich tylko miary (Ilosc, Cena) oraz klucze obce.
  + **Tworzenie Kluczy Zastępczych (Surrogate Keys):** W bazie źródłowej produkt może mieć ID "SKU-123XYZ". W naszej hurtowni tworzymy dla niego nowy, czysty klucz ProductID = 1.

#### 3. L = Load (Ładowanie): Serwowanie Gotowego Dania

To ostatni krok, w którym nasze nowo utworzone, czyste tabele faktów i wymiarów są fizycznie ładowane do docelowej hurtowni danych (systemu OLAP), skąd będą je czytać analitycy i Power BI.

* **Co robimy?** Wkładamy gotowe tabele DimProduct, DimCustomer, FactSales do naszej hurtowni.
* **Rodzaje ładowania:** Najczęściej jest to **ładowanie przyrostowe (Incremental Load)**, czyli codzienne dokładanie tylko nowych danych (np. sprzedaży z wczoraj).

### Nowoczesna Alternatywa: ELT (Extract, Load, Transform)

W erze chmury proces ten często się odwraca:

1. **E (Extract):** Pobieramy surowe dane.
2. **L (Load):** Natychmiast **ładujemy** je w surowej, "brudnej" formie do hurtowni chmurowej.
3. **T (Transform):** Używamy ogromnej mocy obliczeniowej samej hurtowni, aby wykonać wszystkie transformacje (czyszczenie, modelowanie) **już wewnątrz niej**.

## Modelowanie Wymiarowe: Schemat Gwiazdy

Teraz, gdy wiemy *jak* powstały nasze dane, zobaczmy, jak wygląda ich docelowa struktura. Najpopularniejszym podejściem jest **schemat gwiazdy** (ang. *Star Schema*). Nazwa pochodzi od jego wizualnej reprezentacji:

* **W centrum** znajduje się jedna lub więcej **tabel faktów**.
* **Wokół niej** (na ramionach gwiazdy) znajdują się **tabele wymiarów**.

Ten prosty i przejrzysty model jest wysoce zoptymalizowany pod kątem zapytań analitycznych, ponieważ minimalizuje liczbę skomplikowanych połączeń (JOIN) potrzebnych do uzyskania odpowiedzi.

## 🎯 Tabele Faktów: Serce Hurtowni Danych

Tabela faktów przechowuje **zdarzenia biznesowe i miary**. Odpowiada na pytanie: "**Co się wydarzyło i w jakiej ilości?**".

Jest to centralny punkt schematu gwiazdy. To zazwyczaj największa tabela w hurtowni, często zawierająca miliardy wierszy.

### Kluczowe Cechy Tabeli Faktów:

1. **Zawiera Miary (Metrics/Measures):**
   * Są to dane liczbowe, które można agregować (sumować, uśredniać, liczyć). Są to "fakty" dotyczące zdarzenia biznesowego.
   * *Przykłady miar:* KwotaSprzedazy, IloscSztuk, Koszt, CzasTrwaniaRozmowy.
2. **Zawiera Klucze Obce (Foreign Keys):**
   * Tabela faktów nie przechowuje opisów (np. nazwy produktu czy miasta klienta). Zamiast tego zawiera **klucze (identyfikatory)**, które łączą ją z tabelami wymiarów.
   * *Przykłady kluczy:* ProductID\_FK, CustomerID\_FK, DateID\_FK, StoreID\_FK.
3. **Jest "Wąska" i "Długa":**
   * **Wąska:** Zazwyczaj ma stosunkowo mało kolumn (głównie klucze i miary).
   * **Długa:** Zawiera ogromną liczbę wierszy, ponieważ każdy wiersz reprezentuje pojedyncze zdarzenie (np. jedną sprzedaż, jedno kliknięcie na stronie).
4. **Ziarnistość (Granularity):**
   * To kluczowa koncepcja projektowa. Ziarnistość określa, co reprezentuje pojedynczy wiersz w tabeli faktów.
   * *Przykład:* Czy jeden wiersz to jedna sprzedana pozycja na paragonie, jeden cały paragon, czy jedna dzienna suma sprzedaży dla sklepu? Im niższa ziarnistość (np. pozycja na paragonie), tym bardziej szczegółowe analizy są możliwe, ale tabela rośnie szybciej.

### Przykład: FactSales (Fakty Sprzedaży)

| **SalesID (PK)** | **DateID\_FK** | **ProductID\_FK** | **CustomerID\_FK** | **StoreID\_FK** | **QuantitySold** | **SalesAmount** | **CostAmount** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 101 | 550 | 1002 | 12 | **2** | **199.98** | **120.00** |
| 2 | 101 | 610 | 4510 | 15 | **1** | **49.50** | **30.00** |
| 3 | 102 | 550 | 8112 | 12 | **1** | **99.99** | **60.00** |

## 📚 Tabele Wymiarów: Kontekst Biznesowy

Tabela wymiarów przechowuje **atrybuty opisowe**, które nadają kontekst liczbom z tabeli faktów. Odpowiada na pytania: "**Kto, Co, Gdzie, Kiedy i Dlaczego?**".

Każda tabela wymiarów jest połączona z tabelą faktów za pomocą klucza.

### Kluczowe Cechy Tabeli Wymiarów:

1. **Zawiera Atrybuty (Attributes):**
   * Są to głównie dane tekstowe, opisowe, które służą do filtrowania, grupowania i etykietowania danych.
   * *Przykłady atrybutów:* NazwaProduktu, Kategoria, Kolor, NazwiskoKlienta, Miasto, Region, NazwaSklepu.
2. **Zawiera Klucz Główny (Primary Key):**
   * Każda tabela wymiaru ma unikalny klucz główny (np. ProductID, CustomerID), który jest używany w tabeli faktów jako klucz obcy. Często jest to klucz zastępczy (surrogate key) – prosty numer (np. 1, 2, 3...) niezależny od systemów źródłowych.
3. **Jest "Szeroka" i "Krótka":**
   * **Szeroka:** Może mieć wiele kolumn (atrybutów) – nie jest niczym dziwnym wymiar klienta ze 100 kolumnami (wiek, płeć, adres, status itp.).
   * **Krótka:** Zawiera znacznie mniej wierszy niż tabela faktów. Liczba wierszy odpowiada liczbie unikalnych bytów (np. liczba wszystkich produktów, liczba wszystkich klientów).

### Zaawansowany Koncept: Jak śledzić zmiany w wymiarach? (SCD)

Co się stanie, gdy klient zmieni adres? Albo gdy produkt zostanie przeniesiony do innej kategorii? Dane w wymiarach nie są statyczne. Sposób, w jaki radzimy sobie z tymi zmianami, nazywa się **Slowly Changing Dimensions (SCD)**, czyli "Wolno Zmieniające się Wymiary".

Istnieje kilka typów, ale najważniejsze to:

* **Typ 1 (SCD 1): Nadpisz stare dane.**
  + **Co robimy?** Po prostu aktualizujemy stary wiersz nowymi danymi (np. zmieniamy stary adres klienta na nowy).
  + **Zaleta:** Proste w implementacji.
  + **Wada:** \*\* tracimy historię\*\*. Nie wiemy już, gdzie klient mieszkał wcześniej, a jego *przeszła* sprzedaż jest teraz błędnie przypisana do *nowego* adresu.
* **Typ 2 (SCD 2): Dodaj nowy wiersz.**
  + **Co robimy?** To najpopularniejsze rozwiązanie w BI. Zamiast nadpisywać, dodajemy **nowy wiersz** dla tego samego klienta, ale z nowymi danymi. Stary wiersz oznaczamy jako "nieaktualny" (np. za pomocą daty ważności DataDo lub flagi CzyAktywny).
  + **Zaleta:** **Zachowujemy pełną historię**. Możemy analizować sprzedaż zarówno pod kątem starego, jak i nowego adresu.
  + **Wada:** Bardziej skomplikowane w implementacji (wymaga m.in. kluczy zastępczych).
* **Typ 3 (SCD 3): Dodaj nową kolumnę.**
  + **Co robimy?** Dodajemy nową kolumnę, aby przechować *poprzednią* wartość (np. PoprzedniAdres).
  + **Zaleta:** Prostsze niż Typ 2.
  + **Wada:** Śledzi tylko jedną poprzednią zmianę. Rzadko stosowane.

## Podsumowanie: Fakty vs. Wymiary

Poniższa tabela zestawia kluczowe różnice między tymi dwoma typami tabel:

| **Cecha** | **Tabela Wymiarów (Dimension Table)** | **Tabela Faktów (Fact Table)** |
| --- | --- | --- |
| **Przeznaczenie** | Dostarcza **kontekstu** (Kto, Co, Gdzie, Kiedy) | Dostarcza **miar** (Ile, Jak dużo) |
| **Zawartość** | Atrybuty opisowe, tekstowe (Nazwa, Kategoria) | Miary liczbowe, transakcyjne (Kwota, Ilość) |
| **Klucze** | Zawiera Klucz Główny (np. ProductID) | Zawiera Klucze Obce do wymiarów |
| **Struktura** | "Szeroka" (wiele kolumn) i "Krótka" (mało wierszy) | "Wąska" (mało kolumn) i "Długa" (miliardy wierszy) |
| **Połączenia** | Łączy się z tabelą faktów (1:N) | Łączy się z wieloma wymiarami (N:1) |
| **Przykład zapytania** | WHERE Kategoria = 'Rowery' | SUM(KwotaSprzedazy) |
| **Analogia** | **Rzeczowniki** w zdaniu biznesowym | **Czasowniki i liczby** w zdaniu |

## Wniosek (Modelowanie)

Właściwe rozdzielenie danych na **miary (fakty)** i **kontekst (wymiary)** jest absolutną podstawą projektowania wydajnej i użytecznej hurtowni danych. Tabela faktów bez wymiarów to tylko niezrozumiałe liczby. Tabele wymiarów bez faktów to tylko listy, które niczego nie mierzą.

Dopiero ich połączenie w **schemat gwiazdy** pozwala analitykom zadawać pytania typu: "Pokaż mi **sumę sprzedaży (fakt)** dla **klientów (wymiar)** z **Warszawy (atrybut wymiaru)** w **zeszłym kwartale (atrybut wymiaru)**".

### Część 2: Język Analityczny DAX

Gdy mamy już solidny model danych (Część 1), musimy nauczyć się, jak zadawać mu pytania. Do tego służy język DAX. Zrozumienie poniższych koncepcji to ten moment "Aha!", po którym DAX staje się logiczny.

## Kluczowa Decyzja: Kiedy stosować Kolumny Obliczeniowe, a kiedy Miary?

To fundamentalna decyzja, którą podejmujesz, pisząc formułę DAX. Zły wybór prowadzi do wolnych raportów i błędnych wyników.

| **Cecha** | **Kolumna Obliczeniowa (Calculated Column)** | **Miara (Measure)** |
| --- | --- | --- |
| **Kiedy obliczana?** | Przy **odświeżaniu danych**. Wartość jest obliczana i fizycznie zapisywana w tabeli. | **W locie**, w momencie użycia w wizualizacji. Nic nie jest zapisywane z góry. |
| **Gdzie działa?** | W **Kontekście Wiersza** (obliczana wiersz po wierszu). | W **Kontekście Filtru** (obliczana na przefiltrowanym zestawie danych). |
| **Co robi z pamięcią?** | **Zajmuje pamięć RAM i miejsce na dysku**. Każdy wiersz przechowuje wynik. | **Prawie nie zajmuje miejsca** (tylko definicję formuły). Obliczenia obciążają CPU. |
| **Kiedy używać?** | ✅ Gdy chcesz **filtrować** lub **grupować** dane (np. kategoryzacja). | ✅ Gdy chcesz **agregować** dane (sumować, uśredniać, liczyć). |
| **Przykład** | PrzedziałCenowy = IF(Produkt[Cena] > 100, "Drogi", "Tani") | [Total Sales] = SUM(Sales[SalesAmount]) |
| **Złota Zasada** | Użyj jej, jeśli wynik ma być **statyczną etykietą** dla wiersza. | Użyj jej, jeśli wynik ma być **dynamiczną agregacją** reagującą na filtry. |

**Wniosek:** Zawsze preferuj **Miary**. Są znacznie bardziej elastyczne i wydajne dla modelu. Kolumn obliczeniowych używaj tylko wtedy, gdy absolutnie musisz stworzyć nową kategorię do filtrowania (np. na osi wykresu lub we fragmentatorze) i nie możesz tego zrobić wcześniej w Power Query.

## Klucz do DAX: Jak naprawdę działają Kontekst Filtru i Kontekst Wiersza

Cześć! W tym module zrobimy sobie przerwę od klikania i zajmiemy się sercem i duszą języka DAX.

Jeśli czułeś się kiedyś zdezorientowany, dlaczego ta sama formuła SUM() działa w jednym miejscu, a w innym nie, albo dlaczego Power BI "nie wie", co chcesz policzyć – to jest moduł dla Ciebie.

Problem leży w tym, że w DAX istnieją **dwa równoległe światy**: Kontekst Wiersza i Kontekst Filtru. Twój sukces zależy od tego, czy wiesz, w którym z nich się w danym momencie znajdujesz.

### 1. 🚶 Świat Pierwszy: Kontekst Wiersza (Row Context)

Wyobraź sobie, że masz przed sobą tabelę w Excelu i wykonujesz obliczenia **wiersz po wierszu**.

* **Gdzie go spotkasz?** Głównie w **Kolumnach Obliczeniowych** (Calculated Columns).
* **Jak działa?** Kiedy tworzysz kolumnę obliczeniową, jej formuła jest wykonywana dla **każdego pojedynczego wiersza** w tabeli, jeden po drugim.
* **Co "widzi"?** W danym momencie formuła widzi **tylko i wyłącznie wartości z bieżącego wiersza**. Nie wie nic o wierszach powyżej, poniżej, ani o żadnych filtrach na raporcie.
* **Metafora:** To jakbyś spacerował po tabeli z klapkami na oczach, patrząc tylko na jeden wiersz na raz.

Przykład:

Tworzysz w tabeli Sales (Sprzedaż) nową kolumnę obliczeniową:

LiniaSprzedazy = Sales[Ilosc] \* Sales[CenaJednostkowa]

To działa idealnie. Dlaczego? Bo dla pierwszego wiersza bierze Ilosc (np. 2) i CenaJednostkowa (np. 100) z *tego* wiersza i daje wynik (200). Potem przechodzi do drugiego wiersza i robi to samo dla *jego* wartości.

**Problem:** Co, jeśli w kolumnie obliczeniowej spróbujesz napisać:

SumaSprzedazy = SUM(Sales[LiniaSprzedazy])

Otrzymasz tę samą, ogromną liczbę (sumę *całej* kolumny) w *każdym pojedynczym wierszu*. Kontekst wiersza nie ma tu sensu. DAX zsumuje całą kolumnę, a potem wpisze ten sam wynik do każdej komórki, bo nie potrafi inaczej.

### 2. 🔍 Świat Drugi: Kontekst Filtru (Filter Context)

To jest świat, w którym żyją Twoje **raporty i wizualizacje**.

* **Gdzie go spotkasz?** Głównie w **Miarach** (Measures).
* **Jak działa?** Zanim Twoja miara (np. SUM(Sales[Sprzedaz])) w ogóle zacznie liczyć, Kontekst Filtru **najpierw "przygotowuje" jej dane**.
* **Co "widzi"?** Kontekst Filtru to zestaw **wszystkich filtrów** aktywnych w danym momencie na Twoim raporcie.
  + Filtr z fragmentatora (np. Rok = 2024)
  + Filtr z innego wykresu (np. kliknąłeś na "Rowery")
  + Filtr z samej wizualizacji (np. wiersz w tabeli dla "Niemcy")
* **Metafora:** To jakbyś dał Power BI sito (albo zestaw sit). Zanim miara zacznie liczyć, Power BI przesiewa całą tabelę faktów przez te sita i daje mierze do policzenia **tylko te wiersze, które zostały**.

Przykład:

Tworzysz prostą miarę:

[Total Sales] = SUM(Sales[SalesAmount])

1. Wrzucasz tę miarę na **kartę KPI**. Pokazuje **10 000 000 zł**. Kontekst filtru jest pusty, więc miara sumuje wszystko.
2. Wrzucasz ją na **wykres słupkowy wg Kategorii**. Na słupku "Rowery" pokazuje **4 000 000 zł**. Dlaczego? Bo Kontekst Filtru dla tego słupka to "wszystkie dane, GDZIE Kategoria = 'Rowery'". Miara policzyła tylko przefiltrowane wiersze.
3. Dodatkowo klikasz na **fragmentator** Rok = 2024. Słupek "Rowery" pokazuje teraz **1 500 000 zł**. Dlaczego? Bo Kontekst Filtru to teraz "wszystkie dane, GDZIE Kategoria = 'Rowery' ORAZ Rok = 2024".

W każdym z tych przypadków miara jest **ta sama**, ale wynik jest inny, bo działał w innym **Kontekście Filtru**.

**Problem:** Co, jeśli w *mierze* spróbujesz napisać formułę tak jak w kolumnie?

MojaMiara = Sales[Ilosc] \* Sales[CenaJednostkowa]

To **nie zadziała**. Power BI zgłosi błąd. Dlaczego? Bo miara (działająca w Kontekście Filtru) nie wie, o który *konkretny wiersz* Ci chodzi. Widzi potencjalnie *tysiące* wierszy, które przeszły przez filtry i nie wie, którą Ilosc ma pomnożyć przez którą CeneJednostkowa.

### 3. 👑 CALCULATE: Władca i Tłumacz Kontekstów

Mamy więc dwa światy, które się nie rozumieją. Kontekst Wiersza (w kolumnach) nie wie o filtrach. Kontekst Filtru (w miarach) nie wie o pojedynczych wierszach.

I tu wchodzi **CALCULATE** – najważniejsza funkcja w całym DAX.

CALCULATE to potężny "modyfikator". Robi dwie rzeczy:

1. **Jest jak tarcza:** Potrafi zignorować istniejący Kontekst Filtru.
2. **Jest jak miecz:** Potrafi narzucić *nowy*, własny Kontekst Filtru.

**Jak działa CALCULATE (w uproszczeniu):**

CALCULATE( [Wyrażenie] ; [Nowy Filtr 1] ; [Nowy Filtr 2] ... )

1. **Krok 1:** Bierze wyrażenie (np. [Total Sales]).
2. **Krok 2:** Patrzy na Kontekst Filtru z raportu (np. Rok = 2024, Kategoria = 'Rowery').
3. **Krok 3:** Patrzy na *nowe filtry*, które mu dałeś (np. Kategoria = 'Akcesoria').
4. **Krok 4:** **Modyfikuje kontekst!** Ignoruje stary filtr (Kategoria = 'Rowery') i zastępuje go nowym (Kategoria = 'Akcesoria'), ale zachowuje pozostałe (Rok = 2024).
5. **Krok 5:** Liczy wyrażenie w tym *nowym, zmodyfikowanym* Kontekście Filtru.

Magia: Przejście Kontekstu (Context Transition)

CALCULATE potrafi też robić magię. Kiedy użyjesz CALCULATE wewnątrz Kontekstu Wiersza (czyli w kolumnie obliczeniowej), ta funkcja zamienia bieżący Kontekst Wiersza w Kontekst Filtru.

To jest skomplikowane, ale to właśnie ta magia pozwala na zaawansowane obliczenia.

### Anty-Wzorce: Czego NIE Robić w Modelu Danych (Najczęstsze Błędy)

Oto krótka lista "anty-patternów", czyli najczęstszych błędów, które prowadzą do wolnych i niepoprawnych raportów. Unikaj ich jak ognia.

1. **Brak dedykowanej Tabeli Dat:** Tworzenie analiz czasowych (np. Rok-do-Roku) bez oddzielnej tabeli wymiaru DimDate jest proszeniem się o kłopoty. Filtry czasowe nie będą działać poprawnie. **Zawsze** twórz i używaj tabeli kalendarza.
2. **Brak Schematu Gwiazdy (np. płaska tabela):** Ładowanie wszystkiego do jednej, gigantycznej tabeli (jak w Excelu) jest skrajnie nieefektywne. Model Power BI jest zoptymalizowany pod kątem schematu gwiazdy (fakty + wymiary). Taka płaska tabela "puchnie" w pamięci i jest bardzo wolna.
3. **Nadużywanie Kolumn Obliczeniowych:** **To jest największy błąd.** Tworzenie kolumny obliczeniowej DAX dla czegoś, co można zrobić jako miarę, jest marnotrawstwem pamięci. Jeśli liczysz SUMA(A) / SUMA(B), zrób to jako miarę. Kolumny obliczeniowe tworzysz tylko do *kategoryzacji* (np. na osie wykresu).
4. **Nadużywanie Relacji Wiele-do-Wielu (N:N):** Czasem są konieczne, ale relacje N:N są skomplikowane, mogą być wieloznaczne i spowalniają model. Zawsze staraj się ich unikać, np. poprzez stworzenie tabeli pomostowej (bridge table), która rozbije relację N:N na dwie relacje 1:N.
5. **Nieuważne używanie iteratorów (funkcji X):** Funkcje jak SUMX są potężne, ale działają w Kontekście Wiersza (iterują wiersz po wierszu). Użycie SUMX na wielomiliardowej tabeli faktów, gdy można było użyć prostej miary SUM, jest bardzo kosztowne obliczeniowo.
6. **Ładowanie zbędnych danych:** Importowanie do modelu kolumn, których *nigdy* nie użyjesz w raporcie (np. ID\_transakcji, Notatki, DataModyfikacji) tylko "rozpycha" model, zużywa pamięć RAM i spowalnia odświeżanie. Ukryj je w Power Query, zanim trafią do modelu.

### Słownik Kluczowych Pojęć

Oto definicje najważniejszych terminów użytych w tym dokumencie, które pomogą Ci usystematyzować wiedzę.

* **Tabela Faktów (Fact Table):** Centralna tabela w schemacie gwiazdy, która zawiera **miary** (dane liczbowe, np. KwotaSprzedazy) oraz **klucze obce** łączące ją z tabelami wymiarów. Przechowuje informacje o zdarzeniach biznesowych (np. każda sprzedaż, każda rozmowa).
* **Tabela Wymiarów (Dimension Table):** Tabela, która opisuje dane i nadaje im kontekst. Odpowiada na pytania "Kto, Co, Gdzie, Kiedy?". Zawiera **atrybuty** (np. NazwaKlienta, KategoriaProduktu) i jest połączona z tabelą faktów za pomocą **klucza głównego**.
* **Fakt (Fact):** Pojedyncza, mierzalna wartość liczbowa opisująca zdarzenie biznesowe. To samo co **Miara** (Measure) w tabeli faktów. Przykład: IlośćSztuk = 2.
* **Miara (Measure):** W kontekście DAX, jest to formuła (np. SUM(Sales[SalesAmount])), która dynamicznie oblicza wartość w zależności od **Kontekstu Filtru** (czyli filtrów na raporcie). Miary nie przechowują danych, tylko je obliczają w locie.
* **Kolumna Obliczeniowa (Calculated Column):** Nowa kolumna dodana do tabeli za pomocą formuły DAX. Jej wartość jest obliczana raz (podczas odświeżania danych) dla **każdego wiersza** i przechowywana w modelu. Działa w **Kontekście Wiersza**.
* **Klucz Główny (Primary Key, PK):** Unikalny identyfikator dla każdego wiersza w tabeli. W **tabeli wymiarów** (np. DimProduct), ProductID jest kluczem głównym, który jednoznacznie identyfikuje każdy produkt. Wartości w tej kolumnie nie mogą się powtarzać.
* **Klucz Obcy (Foreign Key, FK):** Kolumna w jednej tabeli (np. w **tabeli faktów**), która odwołuje się do **klucza głównego** w innej tabeli (np. w **tabeli wymiarów**). Służy do tworzenia relacji. ProductID w tabeli FactSales jest kluczem obcym wskazującym na ProductID w tabeli DimProduct.
* **Schemat Gwiazdy (Star Schema):** Najpopularniejszy model projektowania hurtowni danych, składający się z centralnej tabeli faktów otoczonej przez tabele wymiarów, z którymi jest połączona. Wygląda jak gwiazda.
* **Model Danych (Data Model):** W Power BI, jest to struktura składająca się z tabel (załadowanych np. z Power Query), **relacji** między nimi oraz **obliczeń** (kolumn obliczeniowych i miar) napisanych w języku DAX.
* **ETL (Extract, Transform, Load):** Proces inżynierii danych służący do budowania hurtowni. Polega na **Ekstrakcji** danych ze źródeł, **Transformacji** (czyszczeniu, modelowaniu, łączeniu) oraz **Ładowaniu** ich do docelowej hurtowni.
* **OLTP (Online Transaction Processing):** System (baza danych) zoptymalizowany do szybkiego **zapisywania** transakcji (np. baza sklepu internetowego).
* **OLAP (Online Analytical Processing):** System (hurtownia danych) zoptymalizowany do szybkiego **odczytywania** i analizowania danych (np. model Power BI).
* **SCD (Slowly Changing Dimensions):** Techniki (np. Typ 1, Typ 2) zarządzania zmianami danych historycznych w tabelach wymiarów (np. zmianą adresu klienta).
* **Kontekst Wiersza (Row Context):** Mechanizm DAX, który pozwala formule "widzieć" wartości tylko z **bieżącego wiersza** tabeli. Występuje głównie w kolumnach obliczeniowych i funkcjach iteracyjnych (np. SUMX).
* **Kontekst Filtru (Filter Context):** Mechanizm DAX, który **filtruje** dane *przed* wykonaniem obliczenia przez miarę. Składa się ze wszystkich filtrów aktywnych na raporcie (z fragmentatorów, osi wykresów itp.).
* **DAX (Data Analysis Expressions):** Język formuł używany w Power BI (i innych narzędziach Microsoft) do tworzenia kolumn obliczeniowych, miar i tabel obliczeniowych w celu analizy danych.
* **CALCULATE:** Najważniejsza funkcja w języku DAX, która potrafi **modyfikować Kontekst Filtru**. Służy do dodawania, usuwania lub zmieniania filtrów, w których obliczane jest wyrażenie.