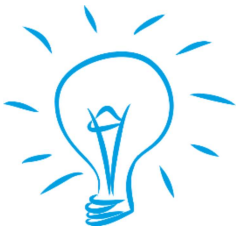
3. Zarządzanie danymi



W elektronicznej wymianie danych B2B (EDI) komunikaty zawierające kody produktów lub usług, identyfikatory jednostek transportowych, a także dokumenty prawne są wymieniane między partnerami w łańcuchu dostaw. Zazwyczaj mają one postać ciągów alfanumerycznych.

3.1 Informacja-Dane-Wiedza

W komputerowych systemach informatycznych reprezentacja informacji różni się w zależności od ich przeznaczenia i zastosowania. Może być alfanumeryczna lub binarna, biorąc pod uwagę fakt, czy reprezentuje tekst, rysunek, dźwięk lub program wykonywalny. Aby móc przechowywać, wyszukiwać, przetwarzać i przesyłać dane różnych typów (np. liczby, znaki, daty, waluty np.), muszą one być odpowiednio zakodowane. Alfanumeryczne formaty reprezentacji danych mają swoje korzenie w alfabecie ASCII (ask-key), ewoluowały w sensie krajowych zestawów znaków (np. standardy 8859-1, Latin 1 i 8859-2, Latin 2) i ostatecznie przekształciły się w międzynarodowe formaty UTF-8 i UTF-16. Umożliwiają one wspólną interpretację danych przez partnerów biznesowych należących do różnych grup etnicznych i środowisk geograficznych. Podczas gdy ciągi alfanumeryczne zależą od ich kodowania, dane numeryczne różnią się głównie rozmiarem i/lub precyzją.

Proces przekształcania danych z ich oryginalnej postaci analogowej do postaci cyfrowej jest popularnie nazywany digitalizacją. Odpowiednio zaprojektowane programy użytkowe i aplikacyjne akceptujące dane z różnych źródeł (np. skanery optyczne, czujniki elektryczne, EDI np.) umożliwiają organizacjom automatyzację pozyskiwania, przechowywania, przetwarzania i przesyłania danych w ramach i pomiędzy komputerowymi systemami informatycznymi.

Po zebraniu, dane różnych typów mogą być łączone i organizowane w tabele danych, bazy danych, hurtownie danych (porządek chronologiczny) lub bazy wiedzy (porządek koncepcyjny). Wyższe poziomy organizacji danych pozwalają na zautomatyzowaną klasyfikację, wnioskowanie i reprezentację zgromadzonej w ten sposób wiedzy do celów analitycznych.

3.2 Dane logistyczne

W logistyce EDI służy do przesyłania danych transakcyjnych między partnerami biznesowymi. Ponieważ mogą oni używać różnych języków i aplikacji, ich konwersja do wspólnego formatu (np. XML, JSON) jest konieczna, aby mogły być interpretowane przez systemy informatyczne różnych partnerów (W3schools, 2023). W celu szybkiej identyfikacji i manipulacji opracowano kody kreskowe i znaczniki RFID.

Aby umożliwić współpracę międzynarodową, konieczne było zdefiniowanie globalnie akceptowanych formatów danych dla celów logistycznych. Formaty danych logistycznych odpowiadają etykietom usług i produktów, identyfikatorom jednostek transportowych i kodom transakcji, zwykle mającym postać ciągów alfanumerycznych ze znacznikiem czasu. Dla uproszczenia manipulacji i szybkości przetwarzania, kody te zostały ustandaryzowane i zakodowane jako optycznie czytelne kody kreskowe lub elektromagnetycznie czytelne kody identyfikacji radiowej (RFID).

Kody kreskowe (EAN/UCC) to wielobranżowa i międzynarodowa forma numerowania przedmiotów (POS EAN-8 i EAN-13, zmienny EAN-128, pasek danych, opakowanie ITF-14, QR, matryca danych np.) Są one wykorzystywane do identyfikacji produktów, partii produktów lub przesyłek (kody 1D), a także usług (kody 2D). Spośród kodów kreskowych 2D kod QR jest najbardziej popularny i może być odczytywany również przez smartfony, co zwiększa jego użyteczność w różnych obszarach zastosowań.

Kody RFID są przede wszystkim używane w taki sam sposób jak kody kreskowe. Umożliwiają one jednoznaczną identyfikację przedmiotów lub usług. Zazwyczaj, oprócz opcjonalnego kodu kreskowego, etykiety RFID zawierają jeszcze więcej informacji na chipie wielkości główki od szpilki. Oprócz identyfikacji, etykiety RFID umożliwiają rejestrowanie informacji o śledzeniu, często wymaganych w zastosowaniach logistycznych. W przeciwieństwie do kodów kreskowych, RFID umożliwia ich skanowanie bez bezpośredniej linii wzroku, a także wielu etykiet jednocześnie.

Dzięki standardowi GS1 EPC Gen2 (ISO/IEC 18000-6:2013) ustanowiono standard technologiczny określający komunikację między tagami RFID a czytnikami. Podobnie jak kody kreskowe, standardy EPCglobal łączą technologię RFID ze znakowaniem EPC produktów,

logistycznych jednostek transportowych, lokalizacji, zapasów, przedmiotów zwrotnych, dokumentów np. W celu bezpośredniej, zautomatyzowanej identyfikacji i śledzenia jednostek logistycznych w łańcuchach dostaw.

Standardy EPCglobal stanowią również podstawę GDSN (ang. Global Data Synchronization Network). Umożliwia ona zautomatyzowane pozyskiwanie i wymianę danych specyfikacji produktów i ich opakowań, co pozwala przedsiębiorstwom na centralne zarządzanie tymi danymi, które mogą być zamiennie wykorzystywane przez nie i ich partnerów.

Tabela 3.1 podsumowuje różne technologie identyfikacji wraz z ich zastosowaniami. Przedstawia ona różne jedno- i dwuwymiarowe kody kreskowe, a także różne klasy kodów RFID wraz z ich możliwościami.

Tabela 3.1 Technologie znakowania

Technologia	Zastosowanie
Pasek jednowymiarowy	Artykuły detaliczne i komponenty produktów
Pasek dwuwymiarowy	Usługi (np. UPS, bilety lotnicze), produkty hurtowe wymagające śledzenia
RFID klasy 1 (pasywne, znaczniki R)	Produkty wymagające masowej identyfikacji, kontroli dostępu
RFID klasy 2 (pasywne, znaczniki RW)	Produkty wymagające śledzenia
RFID klasy 3 (półaktywne, znaczniki RW)	Kontrola dostępu z dodatkowymi informacjami o śledzeniu
RFID klasy 4 (aktywne, znaczniki RW)	Śledzenie i namierzanie w przestrzeni zamkniętej
RFID klasy 5 (aktywne znaczniki/interrogatory)	Śledzenie i namierzanie otwartej przestrzeni, usługi zbliżeniowe z włączonymi urządzeniami, usługi oparte na lokalizacji

Przyszłe trendy w znakowaniu, śledzeniu i namierzaniu podążają w dwóch głównych kierunkach: miniaturyzacji i różnorodności. Kody kreskowe (1D) umożliwią również znakowanie miniaturowych przedmiotów (np. kapsułek medycznych). Nowatorskie kody matrycowe (2D) nie tylko umożliwią korekcję błędów podczas skanowania, ale także szyfrowanie danych.

RFID nadal rozprzestrzenia się na inne obszary zastosowań, takie jak identyfikacja przez dostawców usług (np. karty kolejowe, rejestracja w pracy np.), płatności zbliżeniowe (np. bezprzewodowe przelewy pieniężne, płatności w automatach sprzedających) i inteligentne rozwiązania (np. inteligentne zarządzanie domem, zdalna obsługa inteligentnych urządzeń np.), a także e-waluty.

3.3 Organizacja danych

Oprócz posiadania określonego formatu, dane mogą być zorganizowane na różne sposoby, aby ułatwić zarządzanie nimi, ich przetwarzanie i prezentację. Chociaż dane wejściowe są w większości nieustrukturyzowane, ich przechowywanie, przesyłanie i przetwarzanie zwiększa ich organizację. W dalszej części przedstawiono typowe formy organizacji danych o rosnącej złożoności, od formatów częściowo ustrukturyzowanych (np. CSV) do ustrukturyzowanych (np. Arkusze kalkulacyjne, bazy danych itp.).

Arkusze kalkulacyjne

Pierwszą formą organizacji danych są dwuwymiarowe tablice pól, zwane również tabelami lub arkuszami kalkulacyjnymi. Zazwyczaj pierwszy wiersz arkusza kalkulacyjnego oznacza znaczenie wartości przechowywanych w kolumnach, po których następują wiersze danych.

Pole tabeli lub komórka to najmniejsza jednostka danych. Ma określony typ (ciąg znaków, liczba, data, waluta np.). Jej zawartość można adresować za pomocą oznaczeń wierszy i kolumn (np. A1, reprezentujące pierwszy wiersz kolumny A).

Każdy wiersz tabeli jest grupą powiązanych pól, reprezentujących rekord (np. transakcję, rekord studenta, dane produktu np.). Ponieważ wszystkie wiersze tabeli mają tę samą strukturę, możemy zdefiniować typ rekordu jako listę atrybutów (np. dane ucznia (imię, nazwisko, data urodzenia, miejsce urodzenia, identyfikator...) odpowiednich typów danych.

Bazy danych

Plik lub tabela bazy danych to zbiór rekordów tego samego typu. Baza danych (DB) składa się z wielu wzajemnie połączonych tabel. Stąd definicja bazy danych według ANSI:

- Dane z bazy danych są ze sobą połączone i posortowane,

- Dane z bazy danych mogą być jednocześnie wykorzystywane przez wielu użytkowników,
- Dane w bazie danych składają się z unikalnych rekordów,,
- Baza danych jest przechowywana w komputerze.

Z powyższej definicji można wyciągnąć pewne wnioski na temat architektury klient – serwer, w której serwer przechowuje bazę danych, do której dostęp mają jego klienci. Oczywiście, aby uzyskać dostęp do bazy danych, należy ustanowić sieć komunikacyjną między serwerem a jego klientami. Serwer DB jest zwykle określany jako jego „back-end", podczas gdy klienci reprezentują „front-end". System zarządzania bazą danych (DBMS) na serwerze umożliwia swoim klientom dostęp do danych przechowywanych w bazie danych za pośrednictwem interfejsu aplikacji (API) i funkcji DBM. Funkcje DBM to mechanizmy umożliwiające wprowadzanie, pobieranie, przetwarzanie i prezentację danych DB. Aby wywołać te funkcje, zdefiniowano standardowe języki zapytań (SQL).

Model relacyjnej bazy danych

Istnieją różne formy organizacji DB, przy czym najbardziej powszechny jest model relacyjny (RDB). Podstawową ideą tego modelu jest fakt, że użytkownik nie może z góry znać wszystkich możliwych zastosowań danych przechowywanych w bazie danych. Ponieważ zwykle nie ma ustalonych ścieżek przeszukiwania plików bazy danych, opracowano różne języki zapytań do pobierania danych i manipulowania nimi. Model RDB opiera się na koncepcji encji i relacji:

- Encja to osoba/rzecz/koncepcja, którą można jednoznacznie zidentyfikować i która posiada atrybuty,
- Relacja reprezentuje sposób powiązania dwóch lub więcej jednostek danych.

Tabele RDB, reprezentujące encje lub relacje, są połączone za pomocą kluczy. Zestaw atrybutów, które jednoznacznie identyfikują encję, nazywany jest kluczem podstawowym. Gdy klucz podstawowy pojawia się jako pole w innej tabeli w celu wypełnienia relacji z oryginalną tabelą, nazywany jest kluczem drugorzędnym lub obcym. Podczas gdy tabela może zawierać tylko jeden klucz główny, aby jednoznacznie identyfikować swoje rekordy, może zawierać wiele kluczy drugorzędnych.

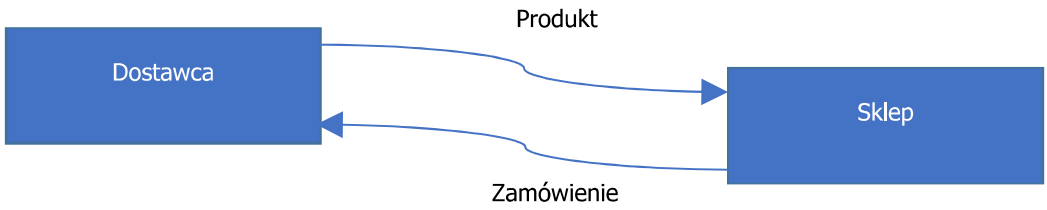
Ogólnie rzecz biorąc, istnieją dwa podejścia do konstruowania RDB: analityczne i syntetyczne.

Analityczne podejście do budowy RDB obejmuje następujące cztery kroki:

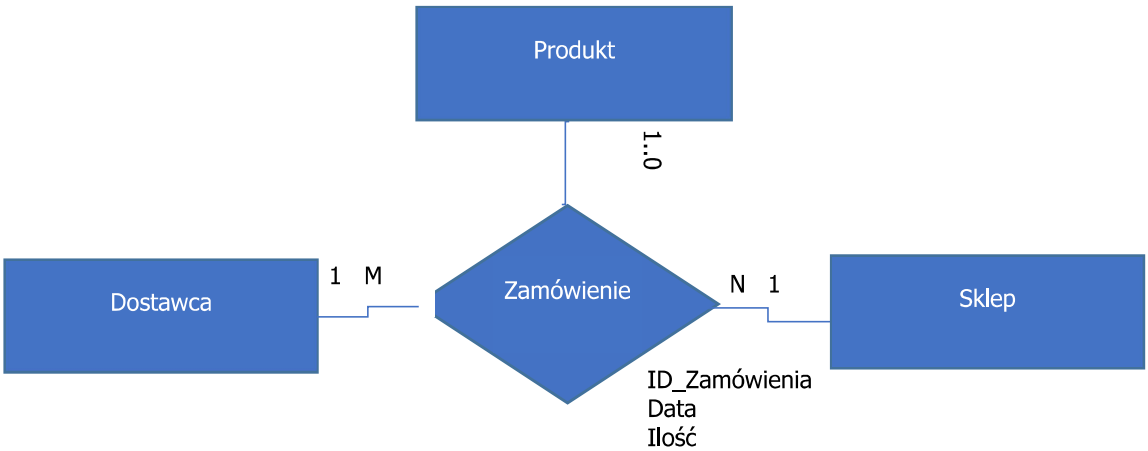
1. Analiza świata rzeczywistego – model globalny.

2. Określenie encji i relacji – model konceptualny (np. diagram E-R).
3. Określenie modelu logicznego – schematu relacyjnego.
4. Budowa bazy danych (DBMS) – model fizyczny.

Aby zilustrować to podejście, rozważmy przykład sieci sklepów ogólnospożywczych i ich dostawców (rysunek 3.1). Każdy sklep ma wielu dostawców. Każdy dostawca może zaopatrywać różne sklepy. Cykl uzupełniania zapasów rozpoczyna się od zamówienia ze sklepu. W zamian dostawca dostarcza towary do sklepu. Zamówienia to transakcje, w których łączone są dane dotyczące sklepu, dostawcy i dostarczanych towarów (rysunek 3.2).



Rysunek 3.1 Model globalny



Rysunek 3.2 Model koncepcyjny

DOSTAWCA (ID_Dostawcy#, Nazwisko_dostawcy, Kontakt_z_dostawcą)

SKLEP (ID_Sklepu#, Nazwa_sklepu, Adres_sklepu)

ZAMÓWIENIE (ID_Dostawcy #, ID_Sklepu #, ID_Zamówienia#, Data, Ilość, EPC)

PRODUKT (EPC#, Nazwa_produktu, Cena_produktu)

Rysunek 3.3 Model logiczny

Model logiczny reprezentuje tabele RDB poprzez typy ich rekordów. W modelu logicznym (rysunek 3.3.) niektóre atrybuty zawierają symbol hashtagu (#). Oznacza to, że reprezentowane przez nie pole jest lub należy do (złożonego) klucza głównego. Niektóre pola są podkreślone. Są one nazywane kluczami drugorzędnymi lub obcymi, ponieważ odwołują się do kluczy głównych powiązanych tabel.

Syntetyczne podejście do RDB obejmuje następujące trzy kroki:

1. Analiza danych - lista wszystkich istotnych atrybutów.
2. Określenie modelu logicznego poprzez normalizację - schemat relacyjny.
3. Model fizyczny (DBMS).

Normalizacja lub synteza kanoniczna (Kent, 1983) zapewnia, że poprzez inżynierię odwrotną z odpowiednich atrybutów tworzona jest baza danych spełniająca warunki RBD (por. tabela 3.2). Podczas gdy początkowa postać normalna (0PN) atrybutów reprezentuje tabelę nieuporządkowanych atrybutów, kolejne postacie normalne, zgodnie z definicją (Codd, 1970), reprezentują wyższe poziomy organizacji danych. Można twierdzić, że osiągając trzecią postać normalną, osiągnięto schemat spełniający wymagania dla modelu logicznego RDB.

Tabela jest w Pierwszej postaci normalnej (1PN), jeśli reprezentuje relację. Zapewnia to, że wszystkie powtarzające się grupy danych są przechowywane oddzielnie, a zatem nie powtarzają się.

Tabela w Drugiej postaci normalnej (2PN) jest w 1PN. Dodatkowo, żaden klucz-atrybut nie może być częściowo funkcjonalnie zależny od klucza głównego. W ten sposób wszystkie klucze, które jednoznacznie identyfikują określone atrybuty, są przechowywane oddzielnie. Głównie ma to na celu zapewnienie, że tylko atrybuty zależne od wszystkich (części) klucza głównego są przechowywane w jednej tabeli.

Tabela w Trzeciej postaci normalnej (3PN) jest w 2PN. Dodatkowo, żadne atrybuty niekluczowe nie są przejściowo zależne od klucza głównego. Oznacza to, że wszystkie atrybuty niekluczowe, które mogą reprezentować klucz dla pewnych innych atrybutów niekluczowych, są przechowywane w oddzielnej tabeli, przy czym tylko ich klucz jest przechowywany w oryginalnej tabeli jako klucz obcy.

Postępując zgodnie z krokami normalizacji od 1PN do 3PN, uzyskuje się model logiczny, który odpowiada przepisom relacyjnej bazy danych (RBD). Wyższe formy normalizacji służą głównie do optymalizacji modelu RBD.

Tabela w Boyce-Codd PN (BCPN) jest w 3PN; dodatkowo każdy wyznacznik jest kluczem. Powoduje to usunięcie wszystkich relacji nieobjętych istniejącą kolejnością kluczy z oryginalnej tabeli, tworząc w ten sposób dodatkowe tabele dla każdego klucza kandydującego. Tabela w 4PN jest w 3NF i BCPN. Dodatkowo, każdy atrybut wielowartościowy, który jest częściowo zależny od klucza, znajduje się we własnej tabeli. 4NF ma na celu usunięcie wszystkich możliwych pozostałych atrybutów wielowartościowych z oryginalnej tabeli. Tabela w 5PN jest w 4PN; dodatkowo każda operacja JOIN jest przewidziana przez klucze. Tabela w 6PN jest w 5PN; dodatkowo, wszystkie nietrywialne zależności JOIN są brane pod uwagę.

Można zauważyć, że przy wyższych formach organizacji liczba tabel rośnie z każdym krokiem. Dlatego rozsądne jest obserwowanie fragmentacji danych, aby zapobiec tworzeniu niepotrzebnych, rzadko używanych tabel.

Tabela 3.2 Przykład normalizacji Relacyjnej Bazy Danych

<table>
<tr><td>Początkowa Postać Normalna

ZAMÓWIENIE

• ID_Dostawcy*
• Nazwa_Dostawcy
• Kontakt_Dostawcy
• ID_Sklepu*
• Nazwa_sklepu
• Adres_sklepu
• ID_Zamówienia*
• Data
• EPC
• Ilość
• Nazwa_produktu
• Cena_produktu</td><td>Pierwsza Postać Normalna

ZAMÓWIENIE

• ID_Dostawcy#
• Nazwa_Dostawcy
• Kontakt_Dostawcy
• ID_Sklepu #
• Nazwa_sklepu
• Adres_sklepu
• ID_Zamówienia #
• Data
• EPC
• Ilość
• Nazwa_produktu
• Cena_produktu</td></tr>
<tr><td colspan="2">* Klucze kandydujące powtarzającej się grupy atrybutów, jednoznacznie identyfikujące zamówienie</td></tr>
<tr><td>Druga Postać Normalna

DOSTAWCY

• ID_Dostawcy #
• Nazwa_Dostawcy
• Kontakt_Dostawcy

SKLEP

• ID_Sklepu #
• Nazwa_sklepu
• Adres_sklepu</td><td>

ZAMÓWIENIE

• ID_Zamówienia #
• ID_Dostawcy #
• ID_Sklepu #
• EPC
• Nazwa_produktu
• Cena_produktu
• Data
• Ilość</td></tr>
</table>

Trzecia Postać Normalna ZAMÓWIENIE • ID_Dostawcy # • ID_Sklepu # • ID_Zamówienia# • Data • Ilość • ID_Produktu	 PRODUKT • EPC# • Nazwa_produktu • Cena_produktu

Języki zapytań

Są one głównie dwojakiego rodzaju: Structured Query Language (SQL) i Query by Example (QBE). Podczas gdy SQL jest uważany za język programowania, QBE jest używany głównie z DBMS bezpośrednio do zarządzania DB i hurtowniami danych.

Standard SQL (ISO/IEC 9075, 1986-2016) to język programowania czwartej generacji służący do manipulowania bazami danych. Umożliwia wyszukiwanie, dodawanie, modyfikowanie, a także usuwanie rekordów danych. Pomimo standaryzacji istnieją niewielkie różnice w jego implementacji w różnych systemach zarządzania bazami danych (DBMS).

W dalszej części język jest krótko przedstawiony z najczęściej używanymi opcjami. Zgodnie z konwencją słowa kluczowe SQL są pisane wielkimi literami, a każde zdanie kończy się średnikiem. Zdania są prezentowane z linkami do materiałów bibliografii oferujących dalsze informacje.

Każda manipulacja bazą danych zaczyna się od jej utworzenia. Komenda:

CREATE DATABASE [*UTWÓRZ_BAZĘ_DANYCH*] *nazwa_bazy_danych*;

tworzy nową pustą bazę danych o określonej nazwie.

Jak wspomniano powyżej, dane w bazach danych są zorganizowane w tabelach rekordów danych określonego typu, w których wszystkie wiersze mają wspólną strukturę. Aby utworzyć tabelę, używane jest następujące polecenie:

CREATE TABLE [*UTWÓRZ_TABELĘ*] *nazwa_tabeli* (
kolumna1 typ_danych1,
kolumna 2 typ_danych 2,
kolumna 3 typ_danych 3,
….);

Każda nazwana kolumna reprezentuje atrybut o określonym typie danych. Na przykład w:

CREATE TABLE *Sklep* (*ID_Sklepu* int *[liczba całkowita]* NOT NULL PRIMARY KEY *[KLUCZ PODSTAWOWY NIEPUSTY]* (,…);

CREATE TABLE *Zamówienie* (*ID_Zamówienia* int *(liczba całkowita)* NOT NULL PRIMARY KEY *[KLUCZ PODSTAWOWY NIEPUSTY]*, …, *Id_Produktu* int *[liczba całkowita]* FOREIGN KEY REFERENCES *Produkt* (*EPC*) *[KLUCZ OBCY ODWOŁANIA Produkt(EPC)]*);

tworzone są dwie tabele. Pierwsza z nich zawiera dane sklepów, podczas gdy druga zawiera dane ich zamówień, odwołując się do pierwszej tabeli poprzez numer produktu jako klucz obcy.

Podczas gdy dane w tabeli są już posortowane według klucza głównego, można je dodatkowo posortować według innych atrybutów, pod warunkiem, że są one indeksowane. Można je indeksować, tworząc indeks na podanym atrybucie (atrybutach) za pomocą następującego polecenia:

CREATE INDEX *nazwa_indeksu* ON *nazwa_tabeli* (*nazwa_kolumny*);

Każda manipulacja danymi na indeksowanej tabeli trwa nieco dłużej, ponieważ dla jej spójności należy sprawdzić nie tylko dane dostarczane przez klucze i odpowiednio je uporządkować, ale także inne atrybuty z określonego indeksu.

Najczęstszą operacją na bazie danych jest zapytanie o dane za pomocą instrukcji SELECT:

SELECT *kolumna1, kolumna2*, …
FROM *nazwa_tabeli*;

To zapytanie o dane zwraca dane w kolumnie1, kolumnie2 itd. z tabeli. Zdania zapytania są zwykle tworzone poprzez podanie dodatkowych opcji, odfiltrowanie danych, spełnienie określonego warunku (warunków):

WHERE definiuje warunek, który określa kryteria wyboru rekordów;

GROUP BY łączy rekordy, mające wspólną właściwość umożliwiającą funkcje agregacji;

HAVING określa funkcje agregujące dla grup zdefiniowanych przez instrukcję GROUP BY;

ORDER BY określa atrybuty, według których uporządkowane są zwracane rekordy.

Na przykład:

```
SELECT "Sklep"."Nazwa_Sklepu", "Produkt"."Nazwa_produktu", "Zamówienie"."Ilość" FROM "Zamówienie", "Produkt", "Dostawca", "Sklep" WHERE "Zamówienie"."ID_Produktu" = "Produkt"."EPC" AND "Zamówienie"."ID_Dostawcy" = "Dostawca"."ID_Dostawcy" AND "Zamówienie"."ID_Sklepu" = "Sklep"."ID_Sklepu" ORDER BY "Sklep"."Nazwa_Sklepu" ASC
```

zwraca listę sklepów z zamówionymi produktami i ilościami, uporządkowaną rosnąco według nazwy sklepu.

Najważniejszą operacją w procesie selekcji jest operacja JOIN. Często zastępuje ona warunek WHERE jako JOIN ON, po którym następuje warunek. Porównuje wartości kolumn i na podstawie porównania określa, czy powinny one zostać uwzględnione w wyniku. W LEFT JOIN rekord jest zwracany, jeśli kryteria są spełnione w lewej tabeli i odwrotnie w operacji RIGHT JOIN. Jak wspomniano powyżej, warunek musi być spełniony w obu tabelach, aby był zgodny z operacją INNER JOIN lub FULL JOIN. Ponieważ ta ostatnia jest najczęściej używana, można użyć JOIN jako synonimu. Odnosząc się do warunków piątej i szóstej postaci normalnej, jest to ta sama operacja JOIN, która musi zostać spełniona, aby spełnić warunki odpowiedniej postaci normalnej.

Do wprowadzania nowych danych do tabeli używana jest operacja INSERT INTO:

```
INSERT INTO [WPROWADŹ DO] nazwa_tabeli (kolumna1, [kolumna2, ... ])
VALUES [WARTOŚCI] (wartość1, [wartość2, ...]);
```

Aby operacja zakończyła się powodzeniem, wartości muszą spełniać wszystkie warunki atrybutów określonych przez nazwy kolumn. Nie trzeba określać nazw kolumn, jeśli wszystkie wartości są wymienione. W przypadku, gdy w tabeli przewidziane są pewne wartości DEFAULT, nie trzeba ich określać, chyba że są inne.

Po wprowadzeniu danych można je zmodyfikować za pomocą instrukcji UPDATE:

```
UPDATE nazwa_tabeli
SET kolumna1=wartość1, kolumna2=wartość2,...
WHERE pewna_kolumna=jakaś_wartość;
```

W instrukcji podawane są nowe wartości dla pól w wymienionych kolumnach. Kryteria wyboru wiersza są oznaczone specyfikatorem WHERE, który określa wszystkie wartości kolumn, do których odnosi się instrukcja UPDATE. Aby zapobiec niepożądanym zmianom, należy zachować szczególną ostrożność przy formułowaniu kryteriów wyboru.

Rekord lub wiele rekordów można usunąć z tabeli za pomocą operacji DELETE:

DELETE FROM *[USUŃ Z] nazwa_tabeli*
WHERE *pewna_kolumna=jakaś_wartość*;

Podobnie jak w przypadku instrukcji UPDATE, specyfikator WHERE jest używany do określenia wszystkich wierszy, które powinny zostać usunięte.

Oczywiście zarządzanie bazą danych na tym się nie kończy. Każdy element bazy danych można również usunąć, zmienić i/lub zastąpić nowym. W przypadku, gdy indeks, tabela lub baza danych ma zostać usunięta, można zastosować następujące instrukcje:

DROP INDEX *nazwa_indeksu* ON *nazwa_tabeli*;

DROP TABLE *nazwa_tabeli*;

DROP DATABASE *nazwa_bazy_danech*;

Jeśli chcemy tylko usunąć dane z tabeli, można użyć instrukcji TRUNCATE:

TRUNCATE TABLE *nazwa_tabeli*;

W przypadku, gdy chcemy dodać lub usunąć atrybut (kolumnę) do/z tabeli, możemy to zrobić za pomocą instrukcji ALTER:

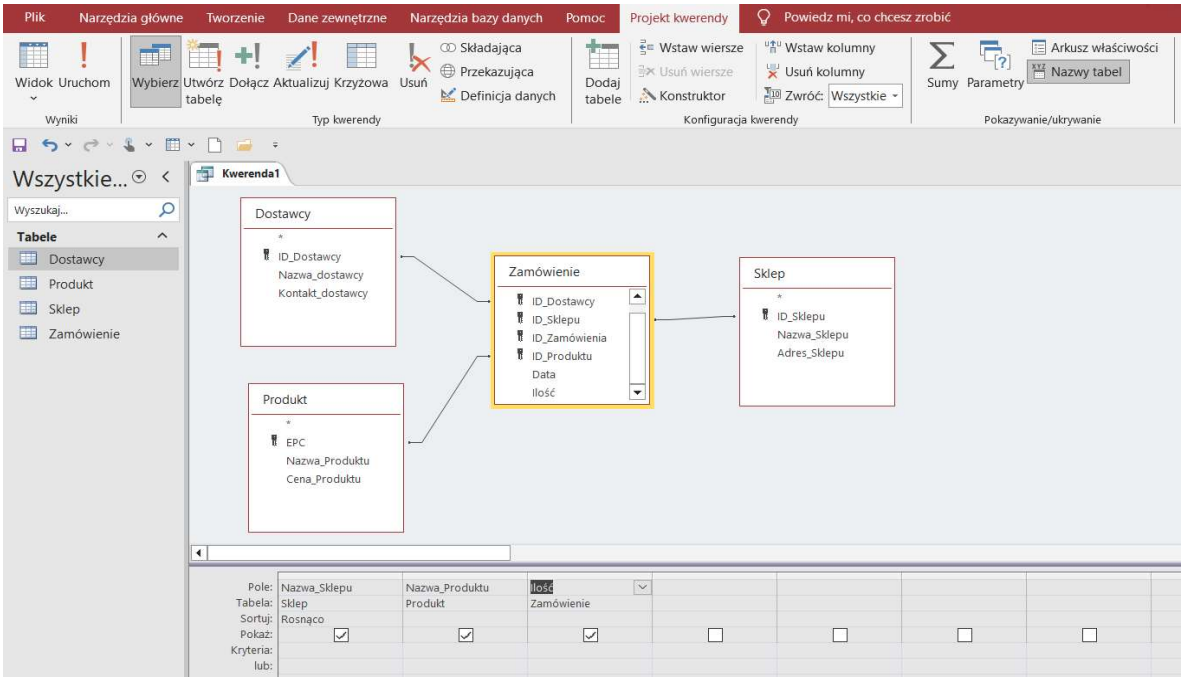
ALTER TABLE *nazwa_tabeli* ADD *nazwa_kolumny typ_danych*;

ALTER TABLE *nazwa_tabeli* DROP COLUMN *nazwa_kolumny*;

Na tym kończy się ten krótki przegląd języka SQL i jego najczęstszych scenariuszy użycia. SQL jest powszechnie używany w architekturach klient-serwer z systemem DBMS hostowanym przez serwer. Aby uzyskać do niego dostęp, instrukcje SQL są wydawane przez program aplikacji klienckiej lub interfejs sieciowy DBMS serwera.

Z drugiej strony, QBE jest również często używany z relacyjnymi systemami DBMS z graficznym interfejsem użytkownika (GUI), takimi jak MS Access lub LibreOffice Base. Dzięki QBE baza danych i jej tabele są tworzone znacznie bardziej interaktywnie, a ich struktura jest łatwiejsza w utrzymaniu. Jak sama nazwa wskazuje, oferuje również prostszą formę wprowadzania i wyszukiwania danych. Aby przeprowadzić wyszukiwanie, należy zebrać wszystkie tabele, które są używane w wyszukiwaniu, a następnie ustalić warunki jako wzorce w polach kolumn, aby odfiltrować odpowiednie dane (rysunek 3.4). Sformułowanie zapytania jest uzupełniane przez istniejące relacje między tabelami. Jak zwykle, wynikiem takiego zapytania jest kolejna

tabela z wynikowymi danymi, które mogą być później dalej przetwarzane. W ten sposób można również tworzyć zapytania kaskadowe lub wielofazowe.



Rysunek 3.4 Zapytanie według przykładu odpowiadające powyższemu zapytaniu SQL

Filtry i maski danych

Aby zapobiec błędnym lub niekompletnym danym, które mogłyby zakłócić ich przetwarzanie i interpretację, należy zastosować dodatkowe środki ostrożności:

1. Filtrowanie pustych wierszy i kolumn.
2. Stosowanie silnego typowania danych w celu zapobiegania błędom obliczeniowym.
3. Definiowanie masek wejściowych w celu zapobiegania wprowadzaniu nieprawidłowych danych.
4. Zniekształcanie danych w celu zapobiegania błędnym wynikom.

Puste wiersze i kolumny są częstym źródłem błędów, które wynikają głównie ze słabych interfejsów w aplikacjach do gromadzenia danych. Anulowane lub niekompletne transakcje zwykle skutkują pustymi wierszami lub brakującymi danymi w dziennikach transakcji. Tylko częściowo można sobie z nimi poradzić w aplikacjach arkuszy kalkulacyjnych, w których puste wiersze i brakujące dane można wykryć, sprawdzając całkowitą liczbę w stosunku do liczby niezerowych danych w wierszach/kolumnach. Można je odfiltrować, usuwając puste

wiersze/kolumny, jednak nie zawsze jest to pożądane działanie, ponieważ możemy również stracić cenne dane. Najlepszym sposobem, aby temu zapobiec, jest użycie systemu DBMS, który z jednej strony pozwalałby na wprowadzanie tylko pełnych danych transakcyjnych, z drugiej zaś zapobiegałby pustym wierszom/kolumnom, ponieważ w bazach danych ich nie ma.

Niewłaściwe wpisywanie danych jest kolejnym częstym źródłem błędów. Jeśli dane alfanumeryczne, takie jak data lub kwota waluty, zostaną wprowadzone zamiast danych numerycznych, spowoduje to błędy podczas przetwarzania tych danych. Zarówno w arkuszach kalkulacyjnych, jak i w bazach danych poszczególnym komórkom danych, reprezentującym wartości atrybutów w rekordach danych, można przypisać typy danych, które zaalarmują nas, gdy wprowadzimy dane w niewłaściwym formacie. W ten sposób można zapobiec sytuacji, w której błędne dane przeszkadzają w ich przetwarzaniu.

Podczas konstruowania baz danych można zastosować ograniczenia do pól danych reprezentujących atrybuty encji lub relacji. Oprócz przypisania im odpowiedniego typu, można zdefiniować maski wejściowe pozwalające na wprowadzanie danych, takich jak daty, waluty, kody EAN itp. tylko w określonym formacie. Zwykle rozwiązuje to wiele nieporozumień, które w przeciwnym razie mogłyby wystąpić podczas przetwarzania danych.

Innym częstym źródłem błędów są nieobiektywne dane, reprezentujące dane, które są o rzędy wielkości większe lub mniejsze niż oczekiwano. Ponownie, mogą one zakłócać nasze przetwarzanie, dając błędne wyniki. Są one trudniejsze do wykrycia i można je odfiltrować jedynie poprzez analizę danych. W aplikacjach arkusza kalkulacyjnego dobrą powszechną praktyką byłoby określenie minimalnych, maksymalnych i średnich wartości danych w odpowiednich kolumnach w celu wykrycia możliwych odchyleń. Jeśli zostaną wykryte, można je zaznaczyć i zająć się nimi ręcznie, jeśli jest ich kilka, lub odfiltrować i zmodyfikować za pomocą zapytania w tabeli bazy danych, jeśli jest ich wiele. Tak czy inaczej, należy je dokładnie ocenić, aby nie pogorszyć sytuacji, w którym to przypadku lepiej byłoby usunąć te dane.

Hurtownie danych i bazy wiedzy

Dane w hurtowniach danych są pobierane z RDB i katalogowane w porządku chronologicznym. Zwykle na danych przeprowadzane są analizy biznesowe, a wyniki są przechowywane w celu późniejszego wykorzystania przez dział analityczny. Po zapisaniu w hurtowni dane te zazwyczaj nie są zmieniane, aby zachować ich spójność.

Oprócz porządku chronologicznego, podczas tworzenia baz wiedzy brane są pod uwagę zamówienia kontekstowe. W tym przypadku jednostki są reprezentowane jako pochodne jednostki najwyższego poziomu lub jej jednostki zależnej. Ich relacje są ustalane bardziej swobodnie, ponieważ mają być aktualizowane i ulepszane w miarę ich używania. Są one ustanawiane w formie reguł opartych na właściwościach encji. W związku z tym forma, w jakiej są przechowywane, jest nieco inna. Często są one przechowywane w formie ontologii zawierających głębszą wiedzę na temat zebranych danych. Podobnie jak w przypadku przechowywania wyników zapytań w hurtowniach danych, zapytania w bazach wiedzy są również przechowywane do późniejszego wykorzystania w celu renderowania bieżących wyników, ponieważ jednostki, relacje i instancje danych ulegają zmianie.

W przeciwieństwie do baz danych i hurtowni, które są specyficzne dla aplikacji, bazy wiedzy mogą być niezależne od aplikacji i często są używane w różnych dziedzinach przez różne aplikacje. Przykład został przedstawiony w (Gumzej i in., 2023).

3.4 Wnioski

W tym rozdziale omówiono różne aspekty zarządzania danymi w logistyce. Oprócz reprezentacji danych i standardów ich przechowywania, przedstawiono mechanizmy organizacji i wyszukiwania danych. Na koniec omówiono niektóre typowe błędy w zautomatyzowanym przetwarzaniu danych, aby zachować czujność świadomego czytelnika. Oprócz wymienionych przykładów, więcej można znaleźć w powiązanych materiałach edukacyjnych.

BIBLIOGRAFIA DO ROZDZIAŁU 3.

- Codd E.F. (1970). A relational model of data for large shared data banks. Communications. ACM 13, 6, pp. 377–387.
- Gumzej, R., Kramberger, T., Dujak, D. (2023). A knowledge base for strategic logistics planning, Proceedings of the 23rd International Scientific Conference Business Logistics in Modern Management: October 5-6, 2023, Osijek, Croatia, Dujak, Davor (ed.) Osijek: Josip Juraj Strossmayer University of Osijek, Faculty of Economics and Business, pp. 317-330. [available at: https://blmm-conference.com/past-issues/, Dostęp November 3rd, 2023]

- GS1 (2023). GS1 Standards. [available at: https://www.gs1.org/standards, Dostęp October 27th, 2023]
- Kent, W. (1983). A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM, vol. 26, pp. 120-125.
- W3schools (2023). XML Tutorial [Dostępne: https://www.w3schools.com/xml/, access November 3rd, 2023]
- W3schools (2023). JSON - Introduction Dostępne: https://www.w3schools.com/js/js_json_intro.asp, access November 3rd, 2023]
- W3schools (2023). Database Normalization [Dostępne: https://www.w3schools.in/DBMS/database-normalization/, access December 7th, 2023]