Podstawy Systemów Zarządzania Baz Danych

1. System Zarządzania Bazą Danych (SZBD)

System Zarządzania Bazą Danych (ang. *Database Management System – DBMS)* to zorganizowany zbiór narzędzi umożliwiających definiowanie i konstruowanie bazy danych, dostęp do danych przechowywanych w bazie danych oraz do wykonywania operacji modyfikujących te dane, czyli manipulowanie bazą danych. Jest to także uniwersalne oprogramowanie, które kontaktuje się z aplikacją użytkownika i z bazą danych.

Podstawowe funkcje SZBD to:

- Definiowanie bazy danych, zazwyczaj przy użyciu języka definicji danych (ang. *Data Definition Language* DDL). Definiowanie bazy danych za pomocą tego języka dotyczy określenia typów i struktur danych oraz ograniczeń dla przechowywanych informacji.
- Manipulowanie bazą danych obejmuje takie działania jak wykonywanie zapytań umożliwiających wybieranie z bazy danych określonych informacji, aktualizowanie bazy danych w taki sposób, aby przechowywane w niej informacje zawsze odzwierciedlały modelowany świat, a także wyprowadzanie informacji na zewnątrz bazy danych poprzez generowanie raportów. Umożliwienie użytkownikom wykonywania podstawowych operacji na danych odbywa się za pomocą języka manipulowania danymi (ang. *Data Manipulation Language* DML). Jest to podstawowa funkcja SZBD.

  Ponieważ baza danych jest „sercem" całego systemu informacyjnego, to funkcje wyszukiwania i przekształcania danych są realizowane za pomocą języka nazywany językiem zapytań (ang. *query language*).

  Najpopularniejszym aktualnie językiem zapytań jest SQL (ang. *Structered Query Language*), gdyż stanowi on obecnie standard języka relacyjnych baz danych.
- Udostępnia użytkownikom katalog (słownik danych), w którym zapamiętywane są opisy elementów danych. Katalog systemowy, zwany również słownikiem danych, jest miejscem przechowywania informacji opisujących dane w bazie, czyli inaczej: jest zbiorem „danych o danych" lub **metadanych**. Zakłada się, iż katalog systemowy jest dostępny zarówno dla użytkowników, jak i dla samego SZBD. Ilość pamiętanych informacji i sposoby ich wykorzystania mogą różnić się pomiędzy poszczególnymi komercyjnymi SZBD.

  W typowym katalogu systemowym są zapamiętywane:
  - nazwy, typy i rozmiary elementów danych,
  - nazwy związków,
  - więzy integralności,
  - nazwy użytkowników upoważnionych do dostępu do danych,
  - schematy bazy danych,
  - statystyki wykorzystania, takie jak np. częstości transakcji i liczniki odwołań do obiektów w bazie danych.

  Podstawowe korzyści wynikające z wykorzystywania katalogu systemowego to.
  - informacje o danych mogą być gromadzone i przechowywane centralnie, co pomaga utrzymać kontrolę nad takim zasobem, jak są dane,

- można opisać znaczenie poszczególnych danych, co umożliwia innym użytkownikom zrozumienie ich przeznaczenia,
- są określeni użytkownik lub użytkownicy, którzy są właścicielami lub mają prawo dostępu do danych,
- łatwiej jest wykrywać nadmiarowość i sprzeczności występujące w danych, ponieważ informacje są dostępnie centralnie,
- zmiany w strukturze danych mogą być rejestrowane,
- wprowadzone mogą być zabezpieczenia danych,
- zapewniona może być integralność danych,
- dostarczane mogą być odpowiednie informacje kontrolne.

❖ Obsługa transakcji – SZBD musi zawierać pewien mechanizm, który będzie nadzorował odpowiednie wykonywanie transakcji, tzn. iż zostaną wykonane wszystkie operacje wchodzące w skład transakcji, albo żadna z nich nie zostanie wykonana.

❖ Obsługa odtwarzania bazy danych – SZBD powinien zawierać pewien mechanizm służący do odtwarzania bazy danych w sytuacji, gdy ulegnie ona jakiemukolwiek uszkodzeniu.
W przypadku, gdy transakcja zostanie przerwana (czy to wskutek załamania systemu, czy uszkodzenia nośnika bądź innego błędu sprzętu czy oprogramowania, czy w końcu wskutek interwencji użytkownika, który wykrył w niej błąd), to we wszystkich tych przypadkach SZBD musi obsługiwać odtwarzanie niesprzecznego stanu bazy danych.

❖ Umożliwia kontrolę dostępu do bazy danych, gdyż zapewnia:
- system bezpieczeństwa, który zapobiega przed dostępem do bazy danych nieuprawnionych użytkowników,
- system integralności, który pilnuje spójności danych przechowywanych w bazie,
- system kontroli wielodostępu, który umożliwia jednoczesny dostęp do bazy wielu jej użytkownikom,
- system odtwarzania po awarii, który odtwarza spójny stan bazy danych sprzed wystąpienia awarii oprogramowania czy sprzętu.

❖ Usługi wspierające niezależność danych – SZBD powinien zawierać elementy wspierające niezależność programów od rzeczywistej struktury bazy danych. Fizyczna niezależność danych jest zwykle łatwiejsza do osiągnięcia, gdyż dostępnych jest na ogół kilka różnych typów zmian fizycznej charakterystyki bazy danych, które nie wpływają na sposoby widzenia danych przechowywanych w bazie. Logiczna niezależność danych jest już trudniejsza do osiągnięcia – o ile dodanie nowej encji, atrybutu, czy związku może być zazwyczaj bez problemu zaakceptowane, o tyle ich usunięcie zazwyczaj już nie. W niektórych systemach nie jest dozwolona żadna zmiana składnika już występującego w jego strukturze logicznej

❖ Obsługa transmisji danych – SZBD musi być w stanie współpracować z oprogramowaniem transmisji danych. Większość użytkowników korzysta z bazy danych z poziomu stacji roboczych, które mogą mieć podłączony SZBD, ale mogą znajdować się w bardzo dużej odległości od serwera i komunikować się z nim poprzez sieć. W obu przypadkach SZBD otrzymuje żądania w postaci przesyłanych komunikatów i odpowiada w podobny sposób. Wszystkie komunikaty są obsługiwane przez program zarządzający komunikacją (DCM od ang. *Data Communication Manager*). Wprawdzie DCM nie jest cześcią SZBD, ale ten drugi z wymienionych musi z nim

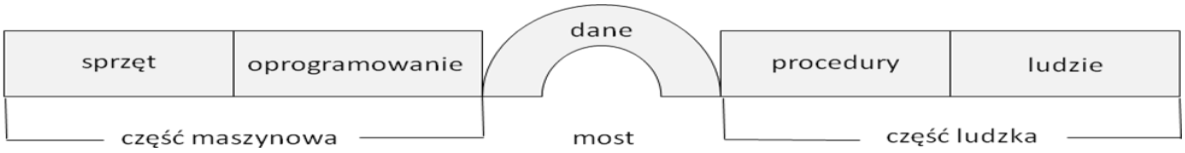
współpracować. Topologię, w której użytkownicy mają zdalny dostęp ze swoich komputerów do (na ogół) jednej centralnej bazy danych będziemy określać mianem *przetwarzania rozproszonego*.

- Programy narzędziowe – SZBD powinien zawierać zestaw programów narzędziowych. Programy narzędziowe pomagają administratorowi efektywnie zarządzać bazą danych. Niektóre z tych programów są tworzone przez samego administratora, inne muszą być dostarczone przez producenta SZBD. Przykładami programów narzędziowych dostarczanych przez producentów SZBD są:
  - system bezpieczeństwa, który zapobiega przed dostępem do bazy danych nieuprawnionych użytkowników,
  - system integralności, który pilnuje spójności danych przechowywanych w bazie,
  - system kontroli wielodostępu, który umożliwia jednoczesny dostęp do bazy wielu jej użytkownikom,
  - system odtwarzania po awarii, który odtwarza spójny stan bazy danych sprzed wystąpienia awarii oprogramowania czy sprzętu.
- Usługi wspierające niezależność danych – SZBD powinien zawierać elementy wspierające niezależność programów od rzeczywistej struktury bazy danych. Fizyczna niezależność danych jest zwykle łatwiejsza do osiągnięcia, gdyż dostępnych jest na ogół kilka różnych typów zmian fizycznej charakterystyki bazy danych, które nie wpływają na sposoby widzenia danych przechowywanych w bazie. Logiczna niezależność danych jest już trudniejsza do osiągnięcia – o ile dodanie nowej encji, atrybutu, czy związku może być zazwyczaj bez problemu zaakceptowane, o tyle ich usunięcie zazwyczaj już nie. W niektórych systemach nie jest dozwolona żadna zmiana składnika już występującego w jego strukturze logicznej

2. Elementy środowiska SZBD

Wyróżnia się pięć podstawowych elementów środowiska SZBD:

1. sprzęt,
2. oprogramowanie,
3. dane,
4. procedury,
5. ludzie.
6. Graficznie można to zobrazować w następujący sposób:



3. Elementy środowiska SZBD - sprzęt

Zarówno sam SZBD, jak i obsługiwane aplikacje, wymagają sprzętu komputerowego, na którym będą uruchamiane. Sprzętem może być:

- pojedynczy komputer,
- duży, wielodostępny komputer,
- sieć komputerów.

Dobór sprzętu zależy od:

- samych potrzeb informacyjnych instytucji, w której dany SZBD będzie wykorzystywany,

- od rodzaju wykorzystywanego SZBD – pewne SZBD mogą być uruchamiane na konkretnym sprzęcie, pod konkretnym system operacyjnym, zaś inne działają na różnorodnych komputerach pod różnymi systemami operacyjnymi.

SZBD wymaga do pracy niewiele pamięci operacyjnej i dyskowej, co nie oznacza, iż taka konfiguracja jest zawsze zadowalająca.

4. Elementy środowiska SZBD - oprogramowanie

Ten element środowiska SZBD obejmuje:

- program SZBD,
- programy aplikacji – pisane zazwyczaj w języku trzeciej generacji (3GL), np. C, C++, Java, Visual Basic, Ada, Pascal, PL/SQL lub w języku czwartej generacji, np. tzw. osadzony SQL,
- system operacyjny,
- oprogramowanie sieciowe dla systemów baz danych użytkowanych w sieci.

Sam SZBD może posiadać swoje własne narzędzia czwartej generacji, których główną zaletą jest to, iż przy stosunkowo małym skomplikowaniu, pozwalają szybko tworzyć aplikacje dzięki udostępnieniu:

- nieproceduralnych języków zapytań,
- generatorów zapytań,
- generatorów formularzy,
- generatorów raportów,
- generatorów grafiki,
- generatorów aplikacji.

5. Języki czwartej generacji

Języki czwartej generacji, w przeciwieństwie do języków trzeciej generacji, są nieproceduralne – co oznacza, iż użytkownik zapisuje **co** ma być zrobione, a nie **jak** ma to być zrobione.

Języki czwartej generacji obejmują:

- języki prezentacji danych, takie jak języki zapytań czy generatory raportów,
- języki specjalizowane, takie jak arkusze kalkulacyjne czy języki baz danych,
- generatory programów wykonujących definiowanie, wstawianie, modyfikowanie i wyszukiwanie danych w bazie,
- języki bardzo wysokiego poziomu służące do generowania kodu aplikacji.
- Przykładami języków czwartej generacji są język SQL (ang. *Structured Query Language*) i język QBE (ang. *Query-By-Example*).

6. Elementy środowiska SZBD - dane

Z punktu widzenia użytkownika najważniejszym elementem środowiska SZBD są same dane. Jak wynika z graficznej ilustracji środowiska SZBD dane są jakby mostem łączącym część sprzętową z częścią ludzką.

Sama baza danych zawiera:

- dane operacyjne,

- metadane, czyli, jakbyśmy mogli powiedzieć, „dane o danych” – to dane o obiektach przechowywanych w bazie, które umożliwiają łatwiejszy do nich dostęp i łatwiejsze wykonywanie na nich różnorakich operacji. Metadane opisują:
  - rekordy,
  - pola, z których zbudowane są rekordy,
  - inne obiekty będące w sferze zainteresowań użytkowników lub wymagane przez SZBD,
- katalog systemowy, zwany inaczej słownikiem danych lub katalogiem danych – SZBD odwołuje się do niego, zanim zacznie wykonywać samą operację właściwego wyszukiwania pożądanych danych w bazie. Katalog systemowy zawiera przykładowo:
  - nazwy elementów danych przechowywanych w bazie,
  - typy i rozmiary elementów danych,
  - więzy przypisane do elementów danych,
  - nazwy użytkowników uprawnionych do korzystania z SZBD,
  - nazwy elementów danych przechowywanych w bazie,
  - nazwy elementów danych, do których dany użytkownik ma dostęp wraz z typami dozwolonego dostępu: wprowadzanie, aktualizacja, usuwanie i/lub czytanie.

7. Elementy środowiska SZBD - procedury

Procedury – to instrukcje i polecenia, które dotyczą procesu projektowania bazy danych, jak i późniejszego jej użytkowania. Użytkownicy bazy danych oraz pracownicy zarządzający bazą potrzebują udokumentowanych procedur pozwalających korzystać z systemu, wśród których mogą znajdować się przykładowo polecenia:

- uruchomienia i zamknięcia SZBD,
- logowania się do SZBD,
- wykorzystania konkretnej funkcji SZBD czy aplikacji,
- zmiany struktur i ilości tabel,
- poprawy efektywności bazy danych,
- tworzenia kopii bazy danych,
- obsługi awarii sprzętu lub oprogramowania – mogą one obejmować procedury rozpoznające, które elementy uległy awarii, a także sposoby odtworzenia bazy danych po naprawieniu awarii.

8. Elementy środowiska SZBD - ludzie

Jak wynika choćby z graficznej ilustracji środowiska SZBD, końcowym elementem tego układu są ludzie.
Możemy wymienić cztery rodzaje osób występujących w środowisku SZBD:

- administratorzy danych i baz danych:
  - administrator danych jest odpowiedzialny za:
    - planowanie bazy danych
    - projektowanie bazy,
    - kontrolowanie ustalonych standardów,
    - przestrzeganie strategii i procedur postępowania,

- administrator bazy danych jest odpowiedzialny za:
  - fizyczną realizację bazy danych, w tym za fizyczny projekt i implementację, kontrolę bezpieczeństwa i spójności, konserwację systemu operacyjnego oraz zapewnienie sprawnego działania aplikacji użytkowników.

❖ projektanci bazy danych:
- projektant logicznej bazy danych – zajmuje się:
  - analizą danych,
  - związków między danymi
  - ograniczeniami, które muszą spełniać dane przechowywane w bazie,
- projektant fizycznej bazy danych – zajmuje się:
  - odwzorowywaniem logicznego projektu bazy danych w postaci tabel i więzów integralności,
  - wyborem odpowiednich struktur i metod dostępu do danych,
  - zaprojektowaniem procedur gwarantujących bezpieczeństwo danych,

❖ twórcy (programiści) aplikacji – zajmują się tworzeniem programów aplikacji, które pozwolą użytkownikom realizować potrzebne funkcje dotyczące pracy w bazie,

❖ użytkownicy – są „klientami" bazy danych:
- użytkownicy naiwni (laicy) :
  - na ogół nie są świadomi istnienia SZBD,
  - kontaktują się z bazą przy pomocy specjalnie napisanych programów aplikacji, które pozwalają realizować konieczne operacje w jak najprostszy sposób,
- użytkownicy profesjonalni :
  - znają strukturę bazy i funkcje realizowane przez SZBD,
  - mogą używać języka wysokiego poziomu (np. SQL), by wykonywać potrzebne operacje (np. na danych),
  - niektórzy z nich mogą pisać programy dla własnego użytku.

9. Architektura wielodostępnego SZBD

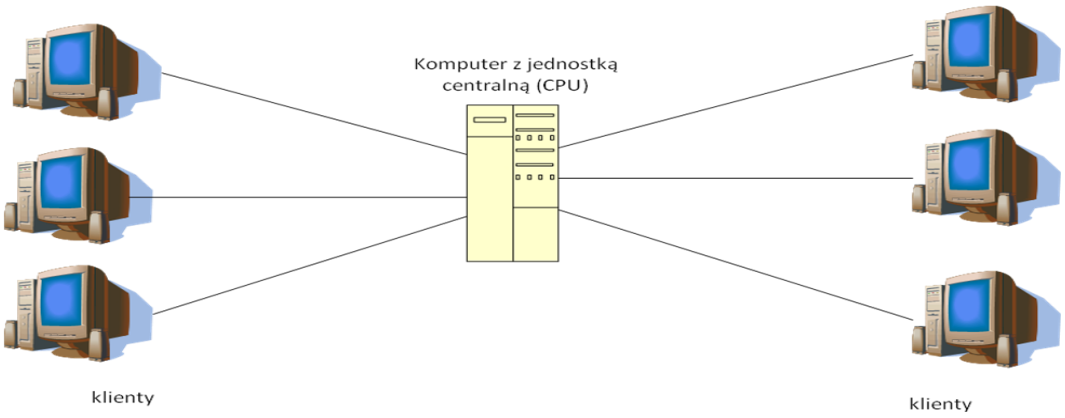
Architektury systemów zarządzania bazami danych ewoluowały zgodnie z ogólnymi trendami panującymi w świecie systemów komputerowych. Wcześniejsze architektury opierały się na centralnych komputerach, które obsługiwały przetwarzanie dla wszystkich funkcji systemu., włącznie z udostępnianiem aplikacji użytkowników i programów pełniących rolę interfejsów użytkowników, a także całą funkcjonalność systemu zarządzania bazą danych. Stosowanie takich rozwiązań wynikała z faktu, że większość użytkowników uzyskiwała dostęp do tych systemów za pośrednictwem komputerowych terminali, które nie oferowały wystarczającej mocy obliczeniowej, i które w związku z tym pełniły wyłącznie rolę sprzętu wyświetlającego otrzymywane dane. Wszystkie operacje związane z przetwarzaniem danych były wówczas zdalnie wykonywane w centralnym systemie komputerowym, a terminale odpowiadały jedynie za wyświetlanie otrzymywanych z tego komputera informacji i instrukcji sterujących – terminale były połączone z centralnym komputerem za pośrednictwem rozmaitych sieci komunikacyjnych.
Wraz z postępującym spadkiem cen sprzętu komputerowego, większość użytkowników zastąpiła swoje terminale komputerami osobistymi oraz stacjami roboczymi. Jednak systemy baz danych początkowo wykorzystywały te komputery w taki sam sposób, w jaki wcześniej używały terminali wyświetlających, zatem nadal istniały

scentralizowane systemy zarządzania baz danych, w których cała funkcjonalność, wykonanie programów aplikacji oraz przetwarzanie interfejsu użytkownika odbywało się na jednym komputerze.

Wśród architektur powszechnie wykorzystywanych do implementacji wielodostępnych systemów zarządzania bazą danych wyróżniamy:

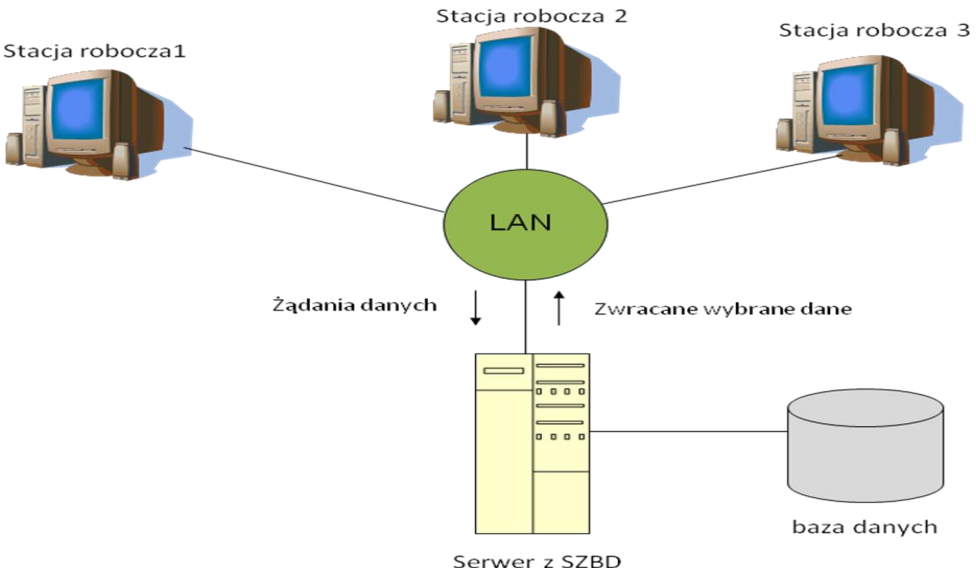
- przetwarzanie zdalne,
- serwer plików,
- architekturę klient-serwer.

10. Topologia przetwarzania zdalnego



W systemach o takiej architekturze głównym elementem był komputer z jedną jednostką centralną (CPU od ang. *Central Processing Unit*), do którego podłączano terminale. Całe przetwarzanie odbywało się w ramach jednego fizycznego komputera. Terminale użytkowników były „nieme", tzn. niezdolne do samodzielnego funkcjonowania. Przesyłały komunikaty do aplikacji poprzez pewien podsystem systemu operacyjnego odpowiedzialny za sterowanie komunikacją, a aplikacje korzystały kolei z usług SZBD. W ten sam sposób komunikaty powrotne były przekazywane do terminali użytkowników. Niestety, w takiej architekturze centralny komputer musiał podołać olbrzymiemu obciążeniu, ponieważ do jego obowiązków należało nie tylko wykonywanie aplikacji i obsługa SZBD, ale także realizacja ważnych zadań na rzecz terminali (np. formatowanie danych, które mają być pokazane na ekranie).

11. Architektura serwera plików



W środowisku z serwerem plików przetwarzanie jest rozproszone w sieci komputerowej, która jest na ogół siecią lokalną (LAN od ang. *Local Area Network*). Zadaniem serwera plików jest przechowywanie plików używanych przez aplikacje i SZBD. Same aplikacje i SZBD są jednak wykonywane na każdej ze stacji roboczych i w razie potrzeby odwołują się do plików na serwerze.

Architektura serwera plików ma zatem trzy główne wady:

1. Występuje w niej duże natężenie ruchu w sieci.
2. Dla każdej stacji roboczej potrzebna jest pełna kopia SZBD.
3. Współbieżność, odzyskiwanie danych po awarii oraz kontrola integralności danych są dużo trudniejsze do uzyskania, ponieważ wiele SZBD korzysta jednocześnie z tych samych plików.

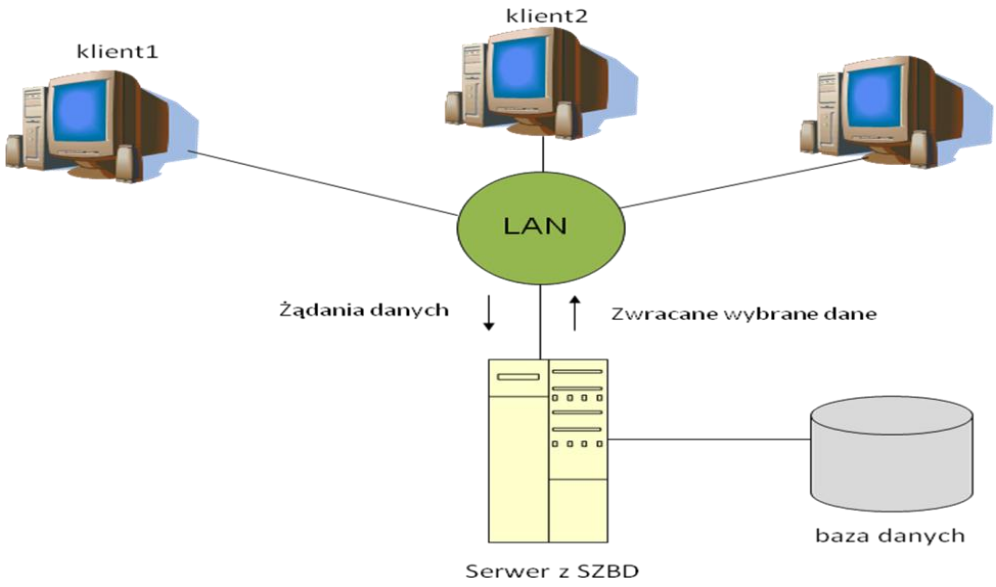
12. (Dwuwarstwowa) architektura klient-server

Aby przezwyciężyć niedogodności wcześniejszych rozwiązań (tzw. scentralizowane SZBD) została stworzona architektura klient – serwer. Nazwa klient-serwer odnosi się do sposobu interakcji komponentów oprogramowania w ramach tej architektury.

Już z samej nazwy wynika, że musi istnieć proces klienta, który potrzebuje pewnych zasobów oraz serwer, który je udostępnia. Nie jest przy tym wymagane, aby klient i serwer znajdowały się na tej samej maszynie. Normalną praktyką jest umieszczanie serwera w jednym miejscu sieci lokalnej, a klientów – w pozostałych.

W porównaniu do systemów scentralizowanych, do warstwy klienta w pierwszej kolejności przeniesiono zatem takie elementy jak interfejs użytkownika oraz aplikacje. Ponieważ SQL z czasem zyskał pozycję standardowego języka relacyjnych systemów zarządzania bazami danych, takie posunięcie stworzyło logiczny punkt podziału pomiędzy klientem a serwerem. Oznacza to, że obsługa zapytań i funkcjonalność transakcji pozostały po stronie serwera. W związku z tym, w rozwiązaniach zbudowanych zgodnie z tą architekturą serwer jest często nazywany **serwerem zapytań** lub **serwerem transakcji.** W relacyjnych systemach zarządzania bazami danych taki serwer jest równie często nazywany **serwerem SQL**, ponieważ większość rozwiązań tego typu opiera się właśnie na języku i standardzie języka SQL.

W takich architekturach typu klient-serwer programy interfejsu użytkownika i proste aplikacji mogą działać po stronie klienta. Kiedy okaże się, że niezbędny jest dostęp do funkcjonalności systemu zarządzania bazami danych, odpowiedni program nawiąże połączenie z takim systemem (działającym po stronie serwera). Kiedy odpowiednie połączenie zostanie utworzone, program klienta będzie mógł się komunikować z systemem zarządzania bazą danych. Standard nazywamy **otwartym łączem baz danych** (ang. *Open Database Connectivity – ODBC)* oferuje **interfejs programowy aplikacji** (ang. *Application Programming Interface*), który umożliwia programom działającym po stronie klienta wywołanie systemu zarządzania bazami danych – jedynym wymaganiem jest zainstalowanie na komputerach klienta i serwera odpowiedniego oprogramowania. Większość producentów systemów zarządzania bazami danych oferuje w swoich pakietach specjalne sterowniki ODBC. Oznacza to, że pojedynczy program klienta może łączyć się z wieloma relacyjnymi systemami zarządzania bazami danych i wysyłać – za pośrednictwem interfejsu API standardu ODBC – zapytania i żądania transakcji, które będą następnie przetwarzane po stronie serwerów baz danych. Uzyskane w ten sposób wyniki zapytań zostaną odesłane do programu klienta, który może je dodatkowo przetworzyć na lub od razu zaprezentować użytkownikowi. Opisana powyżej architektura nazywana jest **architekturą dwuwarstwową**, ponieważ składniki oprogramowania są dzielone pomiędzy dwa systemy: klienta i serwera. Zaletą tego typu architektur jest ich prostota i bezproblemowa zgodność z istniejącymi systemami informatycznymi. Jednak rozwój Internetu (w szczególności popularność stron WWW) odmienił nieco role klientów i serwera, doprowadzając ostatecznie do powstania architektury trójwarstwowej.



Przyjęło się, iż w bazach danych to klient obsługuje interfejs użytkownika i sterowanie w aplikacjach, działając jako zaawansowana stacja robocza przeznaczona do wykonywania programów. Klient przyjmuje żądania użytkowników, sprawdza ich poprawność składniową i tworzy żądanie do bazy danych w języku SQL lub w innym języku bazy danych właściwym dla struktury sterowania aplikacji. Następnie przesyła je jako komunikat do serwera, czeka na odpowiedź i formatuje tę odpowiedź zgodnie z wymaganiami użytkownika. Serwer przyjmuje i przetwarza żądania do bazy danych, a następnie przesyła wyniki z powrotem do klienta. Przetwarzanie żądania polecenia polega na sprawdzeniu uprawnień, zapewnieniu integralności, konserwowaniu katalogu systemowego oraz na realizacji zapytań i aktualizacji. Dodatkowo konieczne jest zapewnienie obsługi współbieżności oraz odzyskiwania danych po awarii.

13. Podział funkcji w architekturze klient-server

Zestawienie funkcji w architekturze klient-serwer:

Klient	Serwer
Obsługuje interfejs użytkownika	Akceptuje i przetwarza żądania do bazy danych od klientów
Akceptuje wprowadzane danych i sprawdza ich poprawność składniową	Sprawdza uprawnienia
Obsługuje sterowanie w aplikacji	Zapewnia nienaruszalność więzów integralności
Tworzy żądanie do bazy danych i przesyła je do serwera	Realizuje przetwarzanie zapytań/aktualizacji i przekazuje odpowiedzi klientom
Przekazuje odpowiedź z powrotem do użytkownika	Konserwuje katalog systemowy
	Obsługuje współbieżny dostęp
	Realizuje odzyskiwanie danych po awarii

Architektura klient-serwer ma następujące zalety:

- umożliwia szerszy dostęp do istniejących danych
- zwiększa wydajność systemu – jeśli klienci i serwery rozmieszczone są na różnych komputerach, to różne jednostki centralne mogą równolegle wykonywać aplikacje. Łatwiejsza też powinna być właściwa konfiguracja komputera, na którym działa serwer, ponieważ jedynym jego zadaniem jest przetwarzanie danych,
- pozwala na redukcję kosztów sprzętu, gdyż tylko serwer wymaga dostatecznie dużej pamięci i mocy procesora do przechowywania danych i zarządzania bazą danych

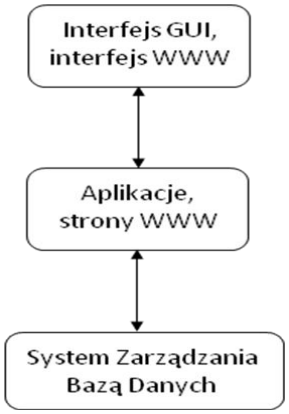
- Redukuje koszty komunikacji – aplikacje wykonują cześć operacji w komputerach klientów i dzięki temu przez sieć przesyłają tylko żądania dostępu do bazy danych. W efekcie zmniejsza się liczba przesyłanych w sieci danych.
- Rozszerza zakres niesprzeczności danych – serwer może obsługiwać sprawdzanie integralności w taki sposób, by więzy były definiowane i weryfikowane tylko w jednym miejscu.
- Odwzorowuje się w naturalny sposób w architekturę systemów otwartych.

Niektórzy producenci baz danych używają pojęcia architektury klient-serwer do podkreślenia zdolności swoich produktów do tworzenia rozproszonych baz danych, tzn. kolekcji wielu logicznie powiązanych baz danych rozmieszczonych w róznych miejscach sieci komputerowej. O ile jednak architektura klient-serwer może być użyta do realizacji rozproszonych baz danych, o tyle sama w sobie nie musi stanowić rozproszonego SZBD.

14. Trójwarstwowa architektura klient-server

Trójwarstwowa architektura klient – serwer:

- dodaje pomiędzy klientem a serwerem jeszcze jedną warstwę pośredniczącą, nazywaną **warstwą aplikacji** lub **serwerem WWW** – w zależności od jej rzeczywistego zastosowania,
- serwer aplikacji lub serwer WWW pełni rolę pośrednika przechowującego reguły biznesowe (procedury lub ograniczenia), które są wykorzystywane podczas udostępniana danych przechowywanych na serwerze danych,
- dodatkowa warstwa może także zwiększyć bezpieczeństwo bazy danych poprzez sprawdzenie danych uwierzytelniających klienta jeszcze przed przekazaniem otrzymanych żądań do serwera bazy danych,
- komputery klienta zawierają graficzne interfejsy użytkownika (GUI) i kilka dodatkowych reguł biznesowych charakterystycznych dla wykonywanych aplikacji,
- serwer pośredniczący przyjmuje i przetwarza żądania nadesłane przez klienta, po czy przesyła je dalej do serwera bazy danych (w postaci odpowiednich poleceń bazy danych). Następnie ten sam serwer pośredniczący przekazuje (częściowo) przetworzone dane z serwera bazy danych do klientów, gdzie dane te mogą być dalej przetwarzane i filtrowane jeszcze przed ich ostatecznym zaprezentowaniem użytkownikom w formacie wykorzystywanego GUI.



Logiczna trójwarstwowa architektura klient-server

Ponadto architektura trójwarstwowa systemów zarządzania bazami danych umożliwia stosowanie trzech poziomów schematów bazy danych (tzw. trójpoziomową architekturę systemów baz danych):

- schemat wewnętrzny, który opisuje fizyczną strukturę przechowywania bazy danych
- schemat koncepcyjny, który jest wysokopoziomowym opisem całej bazy danych,
- schemat zewnętrzny, który opisuje perspektywy różnych grup użytkowników.

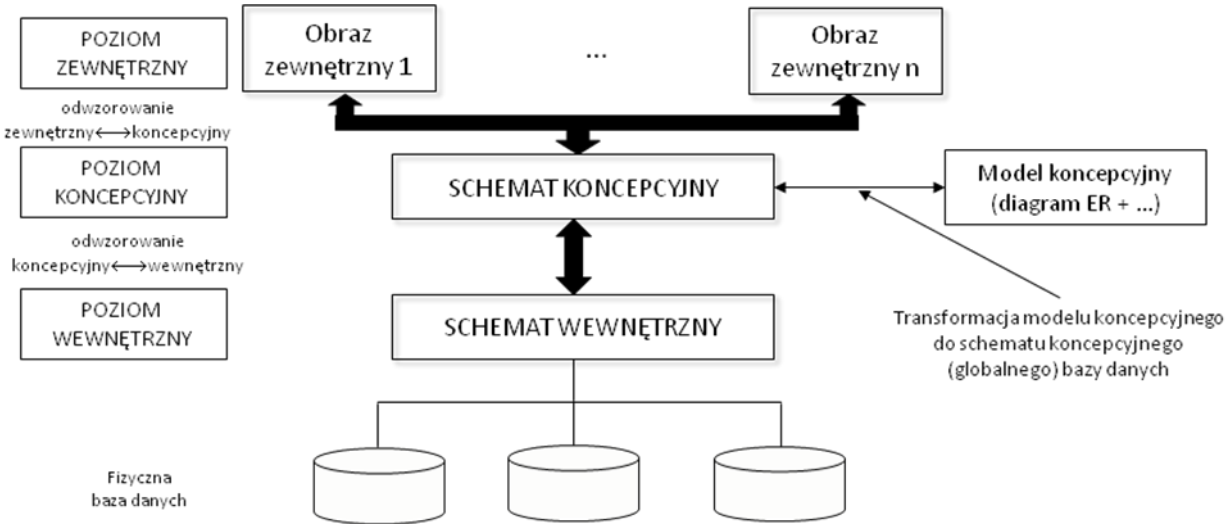
15. Trójpoziomowa architektura systemów baz danych

Jedną z popularnych architektur systemów baz danych jest architektura trójpoziomowa, która została wprowadzona z myślą o ułatwieniu uzyskania (i wizualnego przedstawienia) następujących własności baz danych:

- oddzielenie programów od danych,
- obsługa wielu perspektyw użytkowników
- wykorzystanie katalogu do przechowywania opisu (schematu) bazy danych.

Trójpoziomowa architektura systemów baz danych umożliwia stosowanie trzech poziomów schematów bazy danych:

- **poziom wewnętrzny** (zawiera **schemat wewnętrzny**), który opisuje fizyczną strukturę przechowywania bazy danych. Wewnętrzny schemat wykorzystuje fizyczny model danych i opisuje wszystkie szczegóły związane z przechowywaniem informacji zawartej w bazie danych oraz z jej ścieżkami dostępu.
- **poziom koncepcyjny** (zawiera **schemat koncepcyjny**), który jest wysokopoziomowym opisem całej bazy danych. Schemat koncepcyjny ukrywa szczegóły fizycznych struktur przechowywania danych i koncentruje się na opisywaniu encji, typów danych, związków, operacji użytkownika oraz ograniczeń. Do opisywania schematów koncepcyjnych podczas implementowania systemu bazy danych wykorzystuje się zwykle reprezentacyjny model danych.
- **poziom zewnętrzny** (zwany także **poziomem perspektyw**) zawiera wiele schematów zewnętrznych lub schematów użytkowników. Każdy taki schemat zewnętrzny opisuje tylko tę część bazy danych, która znajduje się w obszarze zainteresowań konkretnej grupy użytkowników, i ukrywa przed tą grupą użytkowników pozostałe części bazy danych.) który opisuje perspektywy różnych grup użytkowników. Jest on implementowany w oparciu o reprezentacyjny model danych, często bazujący na zewnętrznym projekcie schematu w wysokopoziomowym modelu danych.



Trójpoziomowa architektura SZBD

- Większość systemów zarządzania bazami danych nie oddziela tych trzech poziomów całkowicie od siebie całkowicie – obsługuje architekturę tego typu tylko do pewnego stopnia.
- Architektura trójwarstwowa Systemów Zarządzania Bazami Danych ściśle oddziela i w pełni umożliwia stosowanie tych trzech poziomów schematów baz danych.

16. Zalety SZBD

- kontrola redundancji danych – tradycyjne systemy oparte na przetwarzaniu plików bardzo rozrzutnie wykorzystują pamięć, ponieważ mogą tę samą informacje przechowywać w kilku plikach. W przeciwieństwie do tego, systemy baz danych są tak skonstruowane, aby eliminować powtarzanie się informacji poprzez scalanie plików i tym samym unikanie przechowywania wielokrotnych kopii tych samych danych.
  Tym niemniej, bazy danych nie eliminują całkowicie redundancji, lecz kontrolują jej rozmiar i ograniczają do sytuacji, gdy istotnie jest niezbędna (choćby w przypadku odwzorowania związku pomiędzy danymi istnieje konieczność powtarzania informacji – np. należy w takim przypadku powtórzyć klucz). Niekiedy należy także powielić pewne dane, aby usprawnić działanie systemu.
- kontrola spójności danych – poprzez eliminowanie redundancji zmniejsza się ryzyko wystąpienia sprzeczności wśród przechowywanych danych. Jeśli przechowywane dane występują tylko raz, to każdą ich modyfikację wystarczy przeprowadzić tylko raz i zmodyfikowane dane są od razu dostępne dla użytkowników. W przypadku, gdy dane mają wiele kopii i system wie o tym fakcie, to trzeba zapewnić ich zgodność przy każdej ich modyfikacji. Niestety, niektóre współczesne SZBD nie odznaczają się własnością automatycznej kontroli tego rodzaju spójności.
- wspólny dostęp do danych – baza danych jest wykorzystywana przez wielu uprawnionych użytkowników. W ten sposób także nowe aplikacje oparte na wspólnej bazie mogą dodawać jedynie tylko te informacje, które jeszcze nie są przechowywane. Ponadto nowe aplikacje mogą wykorzystywać mechanizmy istniejące w SZBD (np. definiowania bazy, itd.), zamiast realizować je we własnym zakresie
- poprawa integralności danych – SZBD wymusza przestrzeganie więzów integralności, czyli tzw. reguł spójności, które w bazie danych nie mogą zostać naruszone.
- większy zasób informacji w oparciu o te same dane – okazuje się, iż w oparciu o te same dane można uzyskać więcej informacji.
- poprawa bezpieczeństwa – pojęcie bezpieczeństwa bazy danych wiąże się z ochroną przechowywanych danych przed nieuprawnionymi użytkownikami. Administrator bazy danych definiuje ogólne mechanizmy bezpieczeństwa, które później są realizowane przez SZBD. Ponieważ w systemach bazodanowych mamy do czynienia z integracją danych, to taki sposób pracy powodowałby, iż, bez odpowiednich środków bezpieczeństwa, dane w przechowywane bazach danych byłyby bardziej narażone niż w systemach opartych na przetwarzaniu plików.
- przestrzeganie standardów – okazuje się, iż integralność danych umożliwia administratorowi bazy danych zdefiniować pewne normy i standardy, a następnie wymusić na użytkownikach ich przestrzeganie.
- wzrost wydajności – SZBD zawiera wiele wbudowanych funkcji, które dla aplikacji opartych na przetwarzaniu plików programista musiałby samodzielnie dopiero zaprogramować. Niektóre SZBD zawierają także środowisko czwartej generacji składające się z narzędzi ułatwiających tworzenie aplikacji dla baz danych. W związku z tym zwiększa się efektywność pracy programistów i zmniejsza się także czas tworzenia oprogramowania, co z kolei pozwala zmniejszyć koszty.
- kontrola wielodostępu– zwiększenie możliwości wielodostępu wynika z faktu, iż SZBD nie dopuszcza do utraty danych lub ich integralności, co miało miejsce w przypadku systemów opartych na przetwarzaniu plików, np. w trakcie odwoływania się przez kilku użytkowników jednocześnie do tego samego pliku danych.

- ❖ rozszerzona obsługa składowania i odtwarzania bazy danych po awarii – w przypadku systemów opartych na przetwarzaniu plików, to na użytkowniku spoczywa odpowiedzialność za zabezpieczenie danych przed skutkami awarii komputera lub błędami programów aplikacji. Natomiast nowoczesne SZBD zawierają mechanizmy minimalizujące utratę danych spowodowaną awarią w trakcie pracy.
- ❖ poprawa zakresu i czasu dostępu do danych – integracja danych przyczynia się do możliwości uzyskania szerszego dostępu do danych, a to z kolei zapewnia znacznie większą funkcjonalność systemów bazodanowych. Wiele SZBD zawiera języki zapytań i narzędzia tworzenia raportów, co pozwala użytkownikom na zadawanie zapytań i uzyskiwania natychmiastowej na nie odpowiedzi, bez konieczności dodatkowych nakładów pracy.

17. Wady SZBD

- ➢ złożoność – SZBD są bardzo skomplikowanymi programami, co wynika choćby z faktu, iż posiadają wiele różnorodnych funkcji. Korzystają z nich nie tylko zwykli użytkownicy, ale także projektanci i programiści baz danych oraz ich administratorzy.
- ➢ duży rozmiar – SZBD są programami, które zajmują bardzo dużo pamięci dyskowej, co także jest konsekwencją faktu, iż posiadają wiele różnorodnych, a przy tym złożonych, funkcji.
- ➢ koszt – na rynku oprogramowania istnieje wiele SZBD, które ceny są zróżnicowane w zależności od ich środowiska i funkcjonalności oraz liczby obsługiwanych stanowisk. Ponadto z naprawdę dużymi i kosztownymi SZBD, obsługującymi bardzo dużą rzeszę użytkowników, wiążą się spore koszty konserwacji, które muszą być przeprowadzane przynajmniej raz w roku.
- ➢ dodatkowe koszty sprzętu – ponieważ SZBD i same bazy danych potrzebują „dużo" pamięci, to dodatkowy koszt może być spowodowany koniecznością zakupu dodatkowych dysków pamięci. Ponadto niektóre SZBD mogą wymagać zakupu „lepszego" komputera.
- ➢ sprawność – ponieważ SZBD obsługują bardzo różnorakie aplikacje, natomiast systemy oparte na przetwarzaniu plików były zazwyczaj pisane w konkretnym celu, dlatego niektóre z aplikacji obsługiwanych przez SZBD mogą działać nieco wolniej niż poprzednio (w systemach specjalnie dla nich przeznaczonych).
- ➢ koszt przeniesienia – koszty przeniesienia aplikacji pod nowy system SZBD mogą być w niektórych przypadkach nieporównywalnie większe w stosunku do kosztów samego SZBD i dodatkowego sprzętu. Do tych kosztów można zaliczyć także konieczność przeszkolenia pracowników pod kątem użytkowania nowego systemu oraz konieczność zatrudnienia nowych pracowników, których zadaniem byłoby samo przeniesienie i uruchomienie nowego systemu.
- ➢ większy zasięg awarii – zasoby danych, w przeciwieństwie do systemów przetwarzania plików, są po kontrolą SZBD centralnie zmagazynowane. Ponadto ponieważ wszyscy użytkownicy i same aplikacje są uzależnieni od dostępu do SZBD, to awaria dowolnego systemu w SZBD może doprowadzić do przerwania pracy i wstrzymania wszystkich operacji.