mgr inż. Adam Krasuski
prof. dr hab. inż. Tadeusz Maciak
Szkoła Główna Służby Pożarniczej

ROZPROSZONE BAZY DANYCH – ARCHITEKTURA FUNKCJONALNA

***Abstract**: This paper focuses on taxonomy of Distributed Database Systems (DDBS). A technical aspects related to DDBS like concurrency control, distributed recovery and query processing are described. The paper also contains comparison between DDBS and other systems of distributed data sharing.*

Streszczenie: *Przedstawiony artykuł ma charakter przeglądowy a celem jego jest zwięzły i syntetyczny opis niektórych aspektów związanych z technologią rozproszonych baz danych. Zawarto w nim również wstępną ocenę przydatności tej technologii jako podstawy do budowy systemu wymiany informacji w **PSP**. Zaprezentowano podstawowy podział rozproszonych baz danych oraz różnice funkcjonalne występujące między poszczególnymi ich rodzajami. Szczególny nacisk położono na opis aspektów technicznych dotyczących spójności i bezpieczeństwa danych. Mają one kluczowe znaczenie przy konstruowaniu systemów przeznaczonych dla służb ratowniczych wymagających dużej niezawodności i bezpieczeństwa na wysokim poziomie.*

Wstęp

System wymiany informacji wewnątrz Państwowej Straży Pożarnej (**PSP**) obok wykorzystania powszechnych narzędzi takich jak poczta elektroniczna bazuje na programie o nazwie **EWID**. Program ten służy do gromadzenia informacji o zdarzeniach, w których interweniowała **PSP**. **EWID** został zbudowany na początku lat dziewięćdziesiątych i bez większych modyfikacji używany jest do dzisiaj. Zawartość bazy danych przechowywana jest w plikach typu **DBF** (*ang. database file*). Informacje generowane w jednostkach niższego szczebla (powiatach) zapisywane są lokalnie a także przekazywane wyżej z wykorzystaniem protokołu **FTP** (*ang. file transport protocol*). Poszczególne kopie nie są ze sobą synchronizowane a aktualizacja polega na dopisywaniu nowych wartości.

EWID przechowuje jedynie podstawowe informacje o zdarzeniach. Są to między innymi: czas, miejsce akcji, opis geograficzny i administracyjny a także rodzaj pożaru, jego wielkość i zaangażowane siły i środki. Dane wprowadzane są za pomocą pól wyboru w formularzu. Taki sposób powoduje brak dostępu do szerszych opisów i sprawozdań z prowadzonych działań. Jednostki niższego szczebla mają dostęp zawężony do informacji zgromadzonych na ich terenie. Ponadto ze względu na skrótowy opis zdarzeń mogą generować jedynie zestawienia statystyczne.

W niektórych jednostkach **PSP** wprowadzono komputerowe systemy wspomagania decyzji. Zazwyczaj są to bazy danych substancji niebezpiecznych. Tylko niektóre jednostki wprowadziły systemy przetwarzające informacje geograficzne z danego terenu. Brak jest natomiast narzędzi do dystrybuowania i pozyskiwania informacji z innych jednostek organizacyjnych **PSP**. Systemy wspomagania decyzji mają zazwyczaj opracowane scenariusze działań w związku z zagrożeniami występującymi na ich terenie, co obecnie, z powodu zagrożenia terrorystycznego jest niewystarczające. Analiza ataków terrorystycznych na przykładzie Osetii Północnej pokazuje, że nawet małe miasteczka typu Biesłan liczące 30 tysięcy mieszkańców narażone są na ryzyko ataku, efektem którego śmierć może ponieść kilkaset osób. Zdarzenia występujące w Polsce również potwierdzają, że służby ratownicze muszą mieć dostęp do informacji na poziomie globalnym. Zawalenie się Hali Targowej w Katowicach miało swój precedens w zawaleniu się budynku mieszkalnego w Gdańsku w 1995 roku. Podczas akcji śląscy ratownicy powinni mieć dostęp do analiz i opisów działań wypracowanych przez ratowników gdańskich.

Chcąc skutecznie zapobiegać zagrożeniom mogącym wystąpić na terenie danej gminy czy powiatu służby ratownicze powinny mieć dostęp do informacji globalnych z terenu całej Polski a nawet państw sąsiadujących. Koniecznym warunkiem realizacji tej strategi jest budowa systemu wymiany informacji mającego na celu integrację danych z terenu całej Polski. Tylko dostęp do informacji na poziomie globalnym jest w stanie zapewnić odpowiedni poziom bezpieczeństwa każdej najmniejszej gminie. Systemy wspomagania decyzji nie powinny zawierać jedynie scenariuszy do zwalczania zagrożeń istniejących na terenie danej gminy. Zagrożenia terrorem powodują, iż można się spodziewać niebezpieczeństw transgranicznych. Dlatego też budowa zintegrowanego systemu wymiany danych musi doczekać się swojej realizacji. Władze państwowe oraz dowództwo **PSP** jest świadome tej konieczności. Uruchomiony został projekt w ramach amerykańskiego offsetu myśliwca F-16 mający na celu budowę takiego zintegrowanego systemu danych. Zgodnie z planami, architektura jego będzie bazowała na modelu scentralizowanym. Choć model ten jest jednym z prostszych w realizacji, posiada liczne ograniczenia w organizacjach rozproszonych geograficznie takich jak **PSP**. Dlatego też koniecznym jest rozważenie innych rozwiązań bazujących na przykład na modelu rozproszonym.

Próbę oszacowania przydatności innych modeli niż scentralizowany do budowy systemu wymiany danych podjęto w Szkole Głównej Służby Pożarniczej. Prowadzone są tam badania własne mające na celu sprawdzenie możliwości wykorzystania technologii rozproszonych baz danych do budowy zintegrowanego systemu wymiany informacji. Badania

te poprzedzone zostały rozpoznaniem literaturowym z zakresu rozproszonych baz danych. Znajomość podstaw oraz motywów, które powodowały rozwój tego typu systemów jest konieczna aby można było rozważyć ich wykorzystanie w **PSP**.

Poniższy artykuł ma charakter przeglądowy a celem jego jest zwięzły i syntetyczny opis niektórych aspektów związanych z technologią rozproszonych baz danych. Zawarto w nim również wstępną ocenę przydatności tej technologii jako podstawy do budowy systemu wymiany informacji w **PSP**. Zaprezentowano podstawowy podział rozproszonych baz danych oraz różnice funkcjonalne występujące między poszczególnymi ich rodzajami. Szczególny nacisk położono na opis aspektów technicznych dotyczących spójności i bezpieczeństwa danych. Mają one kluczowe znaczenie przy konstruowaniu systemów przeznaczonych dla służb ratowniczych wymagających dużej niezawodności i bezpieczeństwa na wysokim poziomie.

1. Wprowadzenie

Pierwsi projektanci systemów baz danych kierowali się zasadą, iż do każdego zbioru danych należy zbudować oddzielną aplikację [1,2]. Z upływem czasu, rozwiązania takie, z uwagi na nadmiarowość danych oraz problemy z ich wymianą, zastępowano systemami centralnymi. Informacje firmy gromadzone były w jednym miejscu i udostępniane poszczególnym aplikacjom klienckim. Rozwiązanie takie było również najbardziej opłacalne ekonomicznie. Prawo Groscha [3] stanowiło, iż moc komputera jest proporcjonalna do kwadratu jego ceny. Powodowało to, iż zakup dużej ilości komputerów a co za tym idzie masowa produkcja były mało opłacalne. Dlatego też centralne przechowywanie danych, było polityką w pełni usprawiedliwioną zarówno ekonomicznie jak i technologicznie.

Presja użytkowników oraz ograniczenia techniczne wymusiły dalsze zmiany w systemach baz danych [4]. Zwolennicy zasady centralizacji dążyli do modelu, w którym wszystkie dane przedsiębiorstwa zgromadzone zostaną w jednym miejscu. Polityka taka w większych organizacjach typu banki lub ośrodki zdrowia, spowodowała liczne niedogodności. W trakcie swojej działalności przedsiębiorstwa te nagromadziły bardzo duże ilości danych. Przy tak dużych ich ilościach, powstał problem nawigacji w bazach scentralizowanych. Wyodrębniła się grupa użytkowników, którym administratorzy systemów nie byli w stanie zapewnić satysfakcjonującego poziomu dostępu do bazy. Optymalizując strukturę danych pod kątem wydajności globalnej, skomplikowano obsługę systemu użytkownikom lokalnym. Względy technologiczne uniemożliwiały natomiast tworzenie widoków lub perspektyw dla potrzeb lokalnej obsługi danych.

Mimo polityki centralnego przechowywania danych zaczęły powstawać tak zwane „wyspy informacji” [5]. Wynikało to między innymi z historycznej przeszłości firm. Duże przedsiębiorstwa bardzo często powstawały na skutek łączenia się i wchłaniania mniejszych. Każda z wchłoniętych firm miała zazwyczaj swój własny system do zarządzania informacjami. Przeniesienie danych do jednego centralnego systemu pociągało za sobą znaczne nakłady finansowe, co w wielu przypadkach było nieopłacalne.

Wraz ze wzrostem ilości przechowywanych informacji w systemach centralnego gromadzenia danych napotkano szereg problemów technicznych [4]. Centralne systemy generowały dużą ilość operacji wejścia/wyjścia na dyskach, co stanowiło zawsze wąskie gardło aplikacji i spowalniało jej pracę. Dodatkowo problemami były niezawodność oraz bezpieczeństwo sytemu. Celem zabezpieczenia się przed utratą danych stosowano kopie danych. Przy tak dużej ich ilości było to bardzo kosztowne. Koniecznym stawał się zakup zapasowego systemu o podobnych parametrach technicznych jak roboczy. Każda awaria zaś sytemu centralnego unieruchamiała w zasadzie cały system informatyczny instytucji.

Dane przechodziły ciągłą modyfikację na skutek regulacji prawnych, zmiany produktu czy też optymalizacji systemu. W przypadku wykorzystania centralnego systemu do przechowywania danych bardzo trudne i zarazem kosztowne okazywało się wprowadzanie każdej modyfikacji.

Ograniczenia te oraz presja użytkowników systemów centralnych wymusiły szukanie rozwiązań w dziedzinie systemów rozproszonych.

W połowie lat siedemdziesiątych prawo Groscha zostało podważone. Rozwój technologiczny umożliwił relatywnie tanią produkcję procesorów. Dodatkowo badania w dziedzinie komunikacji sieciowej umożliwiły efektywną wymianę danych między komputerami. Wszystko to doprowadziło do powstania nowego sposobu przechowywania danych – w systemach rozproszonych. W latach osiemdziesiątych systemy te uzyskały oficjalną nazwę Rozproszone bazy danych.

Obecnie określenie „Rozproszona baza danych” **DDB** (*ang. Distributed Database*) jest różnie rozumiana w środowiskach informatycznych. Producenci systemów baz danych postrzegają **DDB** poprzez problemy związane z łączeniem ich produktów. Dla twórców narzędzi dostępu oraz obsługi danych, **DDB** jest systemem, który posiada rozproszoną architekturę i wymaga zastosowania różnego rodzaju protokołów dostępu. Producenci sprzętu komputerowego widzą natomiast **DDB** jako system skomponowany z różnego rodzaju oprogramowania do obsługi baz danych pracujących na tej samej platformie sprzętowej. W zasadzie każda z tych definicji pasuje do luźno powiązanych baz danych. Za najbardziej

trafną uznaje się definicję ustanowioną przez C. J. Date [1]. Wprowadził on dwanaście zasad precyzujących idee rozproszonych baz danych. Zasady te mają na celu zdefiniowanie podstaw otwartej architektury umożliwiającej integrację systemów baz danych różnych producentów. Celem ich jest również utworzenie systemu baz danych, który swoją funkcjonalnością przewyższać będzie systemy scentralizowane.

2. Umiejscowienie DDB pomiędzy systemami współdzielenia danych

Definicja zaprezentowana przez Date'a jest bardzo rozbudowana i służy zazwyczaj do wytyczania kierunku rozwoju oprogramowania z dziedziny rozproszonych baz danych. W literaturze światowej natomiast najczęściej wykorzystywana jest krótsza definicja, zgodnie z którą rozproszoną bazą nazywamy [2,4-6]:

> *Logicznie połączone ze sobą zasoby współdzielonych danych, które fizycznie rozproszone są pomiędzy węzłami w sieci komputerowej.*

Analogicznie System zarządzania rozproszoną bazą danych **DDBMS** (*ang. Distributed Database Management System*) definiowany jest jako:

> *Oprogramowanie umożliwiające zarządzanie rozproszoną bazą danych, w sposób, w którym rozproszenie danych jest niewidoczne dla użytkowników.*

Definicja ta ma z kolei postać bardzo ogólną i w zasadzie wymogi jej spełniać może wiele aplikacji. Dokonując jej interpretacji można stwierdzić, iż **DDB** jest po prostu rozproszonym systemem plików **DSF** (*ang. Distributed File Systems*) [7]. W obu przypadkach zadaniem systemu jest zapewnienie użytkownikom dostępu do plików rozproszonych geograficznie. Występują jednak istotne czynniki odróżniające **DDB** od **DFS**. Objawia się to strukturą przechowywanych danych oraz funkcjonalnością tych systemów.

Najważniejsze czynniki rozróżniające **DDB** oraz **DFS** są następujące [8]:

1. **DFS** zapewnia użytkownikom prosty interfejs dostępu do plików rozmieszczonych na komputerach połączonych w sieć. Pliki te mają prostą strukturę (zazwyczaj są płaskie). Powiązania pomiędzy kilkoma plikami (o ile w ogóle istnieją) nie są zarządzane przez system, pozostawiając to użytkownikom. W przeciwieństwie do tego, **DDB** są zorganizowane zgodnie ze schematem, który definiuje strukturę danych oraz relacje miedzy nimi. Za zarządzanie strukturą oraz relacjami odpowiedzialny jest system.

2. Interfejs użytkownika w **DFS** umożliwia jedynie wykonywanie podstawowych operacji na plikach, typu: otwarcie, odczyt lub zapis oraz zamknięcie pliku. **DDB** natomiast posiadają pełną funkcjonalność centralnej bazy danych. Oznacza to, że umożliwiają wykonywanie operacji na danych z użyciem języków wysokiego poziomu (np. **SQL** *ang. Structured Query Language* lub **OQL** ang. *Object Query Language*), wsparcie dla transakcji (zbioru powiązanych zapytań) oraz implementację mechanizmów wielodostępu i odzyskiwania danych.
3. W rozproszonych bazach danych zapewniona jest przezroczystość dostępu do danych, co oznacza, iż użytkownik nie zdaje sobie sprawy z tego, iż dane są rozmieszone na innych komputerach niż jego. Natomiast w **DFS** użytkownik musi znać rozmieszczenie plików.

Rozproszony system plików może być realizowany w różnych architekturach sieciowych. Może to być serwer udostępniający swoje zasoby klientom (architektura klient/serwer) jak również może to być architektura **P2P** (*ang. peer-to-peer*) [9], w których nie ma wyraźnie sprecyzowanego serwera i klienta.

W sieci **P2P** każdy z węzłów może być serwerem, który udostępnia zasoby klientom, jak również klientem korzystającym z tych zasobów. Funkcjonalność tę ma część rozproszonych baz danych. Nie należy zatem kojarzyć sieci **P2P** z systemem współdzielenia plików. **DDB** mogą również być budowane w architekturze **P2P** [10,11].

Ważnym kryterium uznania danej bazy za rozproszoną jest geograficzne oddzielenie od siebie przechowywanych zasobów. Definicja nie precyzuje odległości między węzłami. Jest zatem możliwe utworzenie architektury rozproszonej bazy danych na jednym komputerze. Warunkiem jest aby komputer ten posiadał kilka procesorów i był w stanie przetwarzać zapytania równolegle. Systemy takie jednak stanowią odrębną dziedzinę techniki i znane są pod nazwą Równoległe bazy danych **PDB** (*ang. Parallel Database Systems*) [2,4,5].

W odróżnieniu od rozproszonych baz danych **PDB** mają na celu przede wszystkim zwiększenie wydajności, a nie fakt integrowania struktur danych rozproszonych geograficznie [2,12]. Ponadto węzły w rozproszonej bazie danych są osobno administrowalne i połączone siecią o dużo mniejszej przepustowości, natomiast węzły równoległej bazy danych są elementami tego samego komputera.

Obecnie coraz większą popularność w dużych przedsiębiorstwach zyskują hurtownie danych (*ang. Data warehouse*) [13]. Podobnie jak rozproszone bazy danych zasoby hurtowni

mogą być rozproszone geograficznie. Istnieją jednak dosyć istotne różnice pomiędzy tymi systemami. Do najważniejszych można zaliczyć [4]:

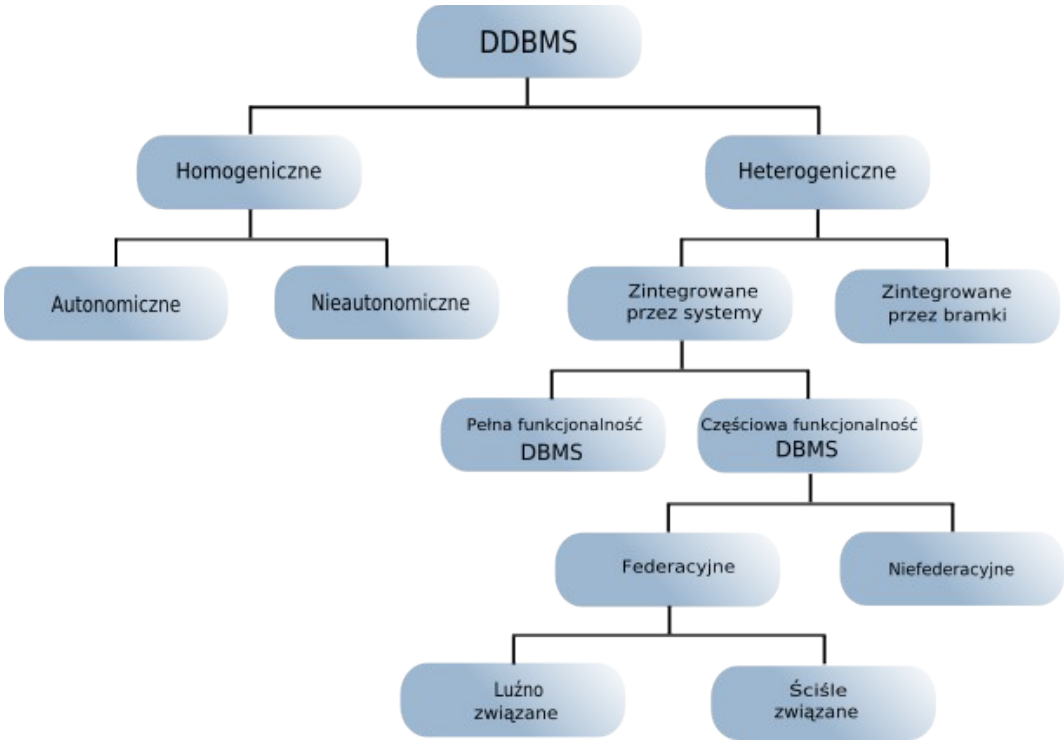
1. **DDB** przechowują dane bieżące, natomiast hurtownie danych przechowują również archiwalne;
2. Dane w hurtowni danych zazwyczaj są zagregowane;
3. Dane przechowywane w hurtowni mają naturę statyczną – raz załadowana hurtownia rzadko się zmienia. Natomiast dane w **DDB** ulegają ciągłej modyfikacji.
4. W **DDB** można wyznaczyć zbiór zapytań, które się powtarzają, natomiast w hurtowniach danych zapytania zazwyczaj są niestrukturalne i heurystyczne.

Przedstawione różnice pomiędzy **DDB** a innymi systemami rozproszonymi skłaniają do wyznaczenia praktycznej definicji rozproszonej bazy danych. Można zatem stwierdzić, iż **DDB** jest systemem posiadającym funkcjonalność centralnej bazy danych, jednak zasoby jej rozproszone są geograficznie. Integracja danych realizowana jest za pomocą sieci komputerowej.

Jak zaprezentowano powyżej **DDB** stanowią wydzielony fragment oprogramowania o rozproszonym przetwarzaniu danych. Przez kilka lat rozwoju tej dziedziny baz danych, powstało szereg rozwiązań realizujących założenia **DDB**. Różnią się one między sobą zarówno architekturą jak i możliwościami funkcjonalnymi. Zaistniała różnorodność wymusiła próby usystematyzowania tego oprogramowania.

3. Podział DDB ze względu na architekturę

Jeden z pierwszych podziałów rozproszonych baz danych został opracowany przez D. Bella oraz J. Grimson [4]. Zaproponowali oni następującą strukturę **DDBMS** (rysunek 1).

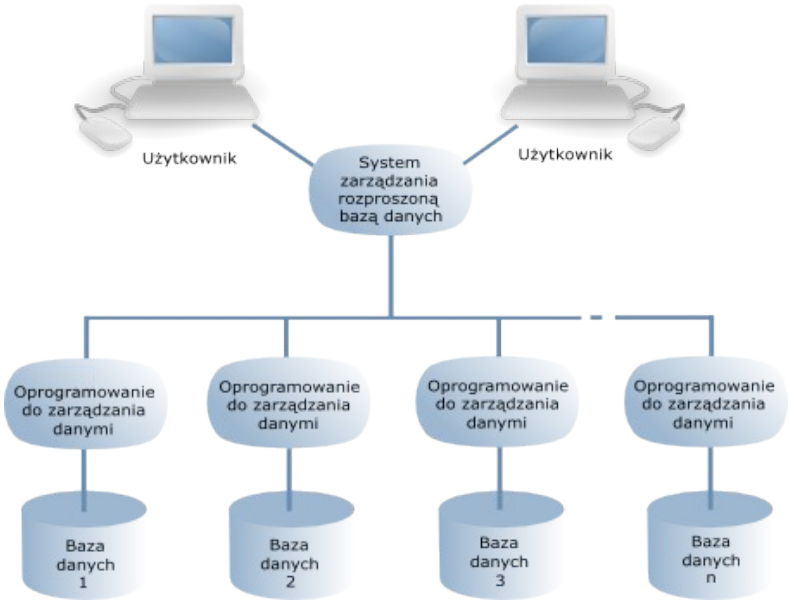


Rysunek 1. Podział rozproszonych baz danych, wg [4]

D. Bell oraz J. Grimson dokonali głównego podziału rozproszonych baz danych na homogeniczne oraz heterogeniczne. W homogenicznych wszystkie węzły wykorzystują ten sam rodzaj oraz wersję oprogramowania, natomiast w heterogenicznych mamy do czynienia z integracją różnorodnych struktur danych umiejscowionych w różnych architekturach sprzętowych lub systemowych. W poniższych rozdziałach zamieszczono szczegółowy opis rysunku 1.

3.1. Homogeniczne DDB

Homogeniczne bazy danych **HDDB** (*ang. Homogeneous Distributed Databases*) łączą różnorodne zasoby danych jednak wszystkie węzły wchodzące w skład systemu posiadają tą samą wersję oprogramowania [2]. Homogeniczne systemy zarządzania bazą danych przypominają centralne **DBMS** lecz w przeciwieństwie do nich dane są fizycznie rozproszone. Rysunek 2 przedstawia strukturę **HDDB**.



Rysunek 2. Architektura homogenicznej rozproszonej bazy danych, wg [4]

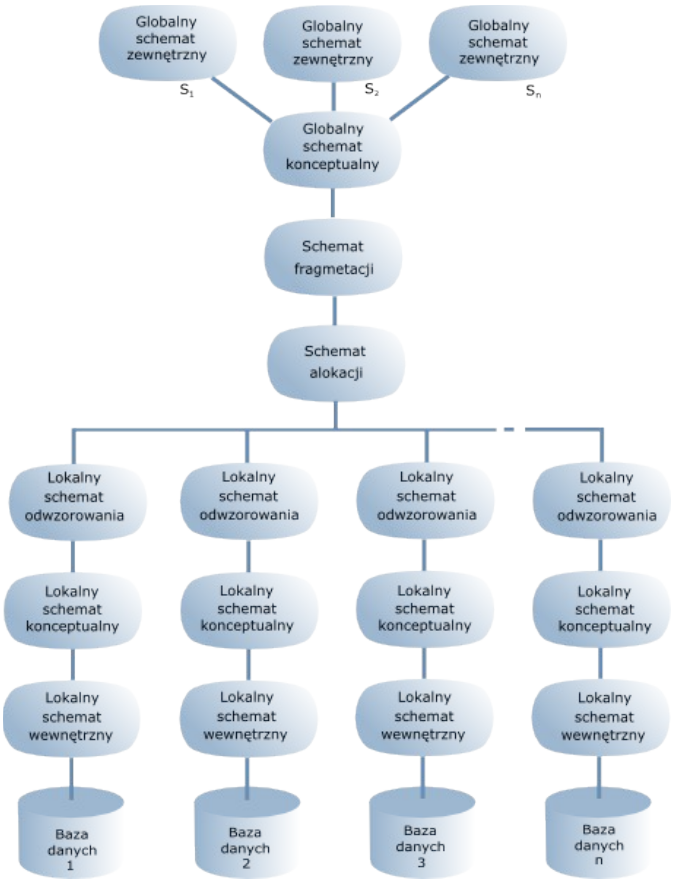
Na rysunku 2 zaznaczono, iż użytkownicy nie mają bezpośredniego dostępu do lokalnych baz danych. Każdy z nich do komunikacji z bazą wykorzystuje globalny interfejs.

Zgodnie z definicją, rozproszenie danych musi być ukryte przed użytkownikiem. W celu uproszenia struktury globalnej, tworzy się widoki bazy, dostosowane do potrzeb lokalnych użytkowników.

Do opisu danych wykorzystywany jest globalny schemat bazy, ukazujący zależności pomiędzy poszczególnymi relacjami. Jest on konstruowany poprzez połączenie wszystkich lokalnych schematów.

Jako standard opisu centralnych **DBMS** uznana została architektura **ANSI-SPARC** [14]. Zakłada ona trzy warstwy opisu danych przechowywanych w bazie: fizyczną, logiczną oraz zewnętrzną. Każdą z tych warstw opisują odpowiednie schematy: wewnętrzny, konceptualny oraz zewnętrzny.

Architekturę **ANSI-SPARC** zastosowano również do opisu rozproszonych baz danych. Jednak w ich przypadku dodatkowo rozszerzono ją celem uwzględnienia aspektu rozproszenia danych. Wyodrębniono dwie dodatkowe warstwy: alokacji oraz fragmentacji. Zmodyfikowany schemat rozproszonej bazy danych zgodny ze standardem **ANSI-SPARC** przedstawiono na rysunku nr 3.



Rysunek 3. Wzorcowa architektura dla DDBMS wg [2]

Przedstawiony na rysunku 3 schemat alokacji przechowuje informacje dotyczące lokalizacji danego fragmentu relacji [2,4]. Relacje w **DDB** mogą być rozmieszczone na cztery podstawowe sposoby:

- centralnie w jednym miejscu,
- podzielone na fragmenty i rozproszone,
- replikowane całkowicie w każdym węźle,
- replikowane selektywnie.

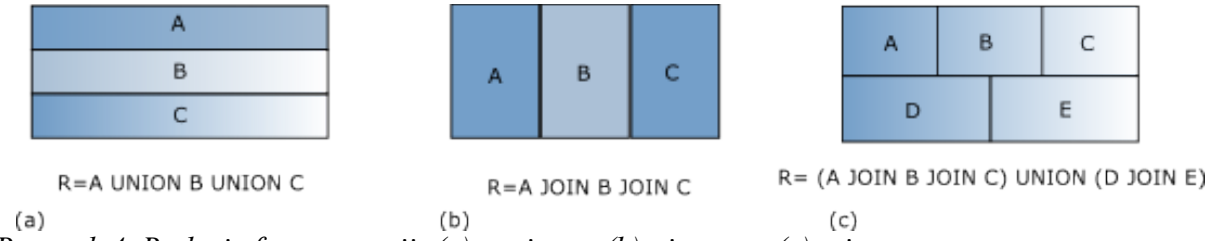
Centralizacja polega na utworzeniu w wybranym węźle jednej bazy danych i zarządzającego nią **DBMS** oraz połączenie tego węzła z wieloma użytkownikami rozproszonymi w sieci (metoda ta nazywana jest również przetwarzaniem rozproszonym).

Metoda podziału na fragmenty polega na podzieleniu bazy na rozłączne fragmenty i umieszczenie ich w różnych węzłach (każdy fragment w innym węźle). Rozmieszczenie danych w węzłach, które najczęściej z nich korzystają, pozwala osiągnąć wysoki poziom lokalności odwołań.

Całkowita replikacja, polega na przechowywaniu kompletnej kopii całej bazy w każdym węźle. Rozwiązanie takie maksymalizuje lokalność odwołań, wiarygodność, dostępność oraz wydajność. Z drugiej strony jednak są tutaj największe koszty przechowywania, komunikacyjne oraz modyfikacji danych.

Selektywna replikacja jest połączeniem metod fragmentacji, replikacji i centralizacji. Część danych jest dzielona na fragmenty w celu uzyskania wysokiego poziomu lokalności odwołań, część natomiast wykorzystywanych przez wiele węzłów oraz rzadko modyfikowanych jest replikowana. Pozostałe dane, dla których nie zastosowano replikacji i fragmentacji są scentralizowane [6].

Widoczny na rysunku 4 schemat fragmentacji przechowuje informacje dotyczące sposobu podzielania globalnej relacji (tabeli) na tabele lokalnych baz danych [2,4]. Wyróżniamy dwie główne metody fragmentacji: poziomą i pionową. Fragmenty poziome są zbiorami krotek, natomiast pionowe zbiorami atrybutów. Stosowane są również dwa inne rodzaje fragmentacji: mieszana oraz pochodna (rysunek 4).



Rysunek 4. Rodzaje fragmentacji: (a) pozioma, (b) pionowa, (c) mieszana

Fragmentacja pochodna polega na złączeniu relacji powstałych z wykonanych wcześniej fragmentacji.

3.1.1. Autonomiczność systemów homogenicznych

Homogeniczne rozproszone bazy danych dzielą się dodatkowo na autonomiczne i nieautonomiczne. Autonomiczność bazy jest określana jako możliwość modyfikacji (ingerencji) zawartości bazy danych z poziomu lokalnego [6,8]. Należy przy tym wspomnieć, iż autonomiczność nie odnosi się wyłącznie do homogenicznych baz danych, jest ona również atrybutem opisującym heterogeniczne bazy.

Większość systemów homogenicznych jest nieautonomiczna. Modyfikacja każdej z baz lokalnych dokonywana jest poprzez globalny interfejs. Wynika to z konstruowania **HDDB**. Tworzenie ich zazwyczaj odbywa się przez planowanie globalnej struktury a następnie struktury relacji każdego z węzłów. Dzięki temu dużo łatwiej wprowadzić można

globalny interfejs zarządzania bazą. Jest on znacznym ułatwieniem administracyjnym. Przeciwieństwem do tego typu projektowania są systemy heterogeniczne, gdzie tworzenie bazy zazwyczaj odbywa się poprzez łączenie już istniejących systemów i globalna administracja każdym z węzłów staje się trudna w realizacji [5].

3.2. Heterogeniczne DDB

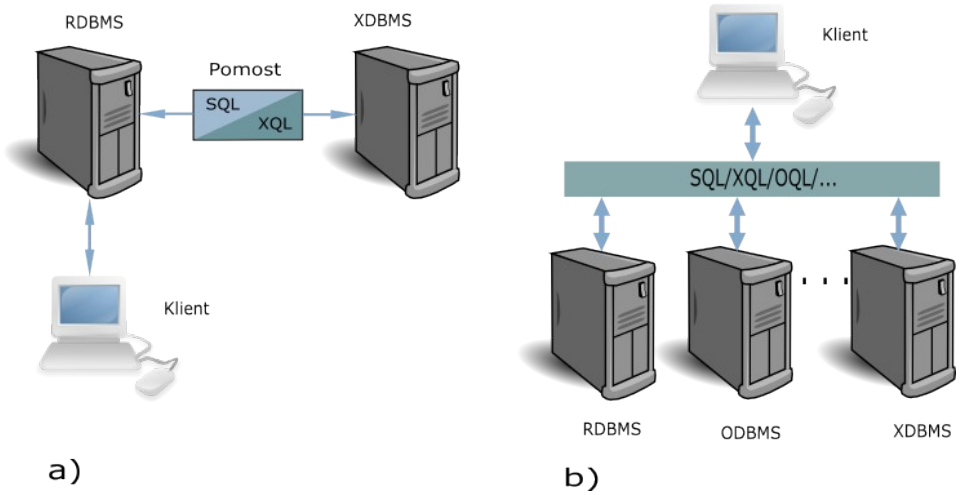
W systemach heterogenicznych węzły mogą wykorzystywać różne oprogramowania, które nie muszą opierać się na tym samym modelu danych [2,4]. W efekcie w skład heterogenicznych baz danych mogą wchodzić relacyjne, sieciowe, obiektowe oraz inne modele danych.

Heterogeniczne bazy danych dzielą się na podklasy:

- zintegrowane poprzez pomosty;
- zintegrowane poprzez systemy;

W bazach zintegrowanych przez pomosty pomiędzy poszczególnymi węzłami umieszczane jest oprogramowanie, które ma za zadanie dopasowanie protokołów komunikacyjnych oraz konwersję zapytań na właściwe dla danej bazy [15].

W integracji przez systemy zadania dopasowania protokołów komunikacyjnych oraz tłumaczenia zapytań kierowanych do baz przejmuje jedna aplikacja. Połączone są z nią wszystkie niejednorodne bazy danych. Aplikacja łącząca posiada specjalne **API** (*ang. Application User Interface*) umożliwiające programom klienckim korzystanie z baz przyłączonych do systemu [4]. Rysunek 5 przedstawia koncepcję integracji przez pomosty oraz przez systemy.



Rysunek 5 Heterogeniczne bazy danych, koncepcja. a) zintegrowane przez pomosty, b) zintegrowane przez systemy

Aplikacje zintegrowane przez systemy dzielą się dodatkowo na zapewniające pełną funkcjonalność centralnego **DBMS** oraz takie, które jej nie zapewniają ponieważ nastawione są na bardziej pragmatyczne zagadnienia, typu konwersja zapytań lub uzyskanie maksymalnej wydajności. Systemy, które realizują częściową funkcjonalność **DBMS** nazywane są systemami wielobazowymi **MDBS** (*ang. mutli-database systems*) [2, 4, 12].

3.2.1.Systemy wielobazowe

Systemy wielobazowe łączą różnorodne bazy danych, posiadające lokalną autonomię. Integracja baz danych odbywa się za pomocą warstw pośredniczących (*ang. middleware*). Warstwa ta może mieć charakter opakowujący bądź mediacyjny [6, 18]. Dostęp do lokalnych baz danych może być bezpośredni lub przy wykorzystaniu globalnego interfejsu. Funkcjonalność ta determinuje podział **MDBS** na federacyjne (sfederowane) oraz niefederacyjne (niesfederowane) [4,12,16]. W federacyjnych bazach danych użytkownicy mogą również korzystać z obu wymienionych interfejsów, natomiast w niesfederowanych mogą wykorzystywać jedynie interfejs globalny. W obu przypadkach musi być zapewniona przezroczystość lokalizacji, co oznacza, że użytkownicy nie zdają sobie sprawy z tego, że pracują w systemie rozproszonym.

Integracja wielu niejednorodnych systemów baz danych w jedną rozproszoną bazę nastręcza często dużo problemów ze konstruowaniem jej schematu. Budowa globalnego schematu konceptualnego wymaga czasem zaangażowania tak dużych zasobów, iż staje się to przedsięwzięciem nieopłacalnym. W trakcie konstruowania schematu należy rozwiązywać problemy zarówno semantyki jak i składni zapytań poszczególnych baz. Dlatego też często

rezygnuje się z budowy globalnego schematu konceptualnego ograniczając się do schematów fragmentarycznych.

Ze względu na wyżej wymienione własności wśród federacyjnych baz danych powstały dwie podklasy: ściśle związane (ze zdefiniowanymi schematami konceptualnymi) oraz luźno związane (bez schematów).

Bazy ściśle związane muszą mieć zdefiniowane schematy globalne. Schemat ten jest konstruowany poprzez połączenie konceptualnych schematów lokalnych baz danych. Wybór zasobów lokalnych, które mają być przyłączone do systemu globalnego, spoczywa na administratorach lokalnych baz danych. Funkcjonalność ta komplikuje budowę schematów globalnych.

Federacyjne ściśle związane bazy danych ze względu na złożoność schematu globalnego, muszą posiadać możliwość definiowania widoków (tworzenia wirtualnych tabel, perspektyw). Ponadto czasami konieczne jest definiowanie również schematów pomocniczych. Opisują one zasady przekształcania (mapowania) schematów lokalnych na globalne. Mapowanie może polegać między innymi na konwersji jednostek, wtedy gdy opis danych na jednej z baz jest w kilometrach natomiast na innej w milach.

Luźno powiązane **MDBS** nie posiadają globalnego schematu konceptualnego. Ze względu na najbardziej nieograniczoną możliwość w integrowaniu różnorodnych struktur danych nazywane są również Interoperacyjnymi (*ang. Interoperable Database Systems*) [17, 18]. W luźno powiązanych bazach danych użytkownik jest odpowiedzialny za tworzenie lokalnych widoków.

Przedstawiony podział rozproszonych baz danych przebiega głównie między systemami homogenicznymi oraz heterogenicznymi. Systemy homogeniczne są prostsze w konstrukcji i zarządzaniu, mimo to większym zainteresowaniem cieszą się systemy heterogeniczne.

Jak wspomniano we wstępie system wymiany informacji pomiędzy poszczególnymi komórkami organizacyjnymi **PSP** bazuje na systemie EWID. Dzięki temu dane przechowywane we wszystkich jednostkach **PSP** mają jednakową strukturę oraz semantykę. Taki stan rzeczy zwiększa szanse na wprowadzenie w **PSP** homogenicznych rozproszonych baz danych. Zastosowanie modelu homogenicznego niesie ze sobą wiele zalet i powinno być wprowadzane tam gdzie jest to możliwe. **HDDB** dzięki występującemu schematowi globalnemu oraz identycznemu oprogramowaniu są łatwiejsze w administrowaniu. Wprowadzenie heterogenicznych rozproszonych baz danych zazwyczaj jest podyktowane koniecznością integracji systemów już istniejących.

Trudność w implementacji homogenicznych baz danych polega na konstruowaniu przejrzystego a zarazem obejmującego swym zasięgiem całą strukturę schematu globalnego. W przypadku **PSP** jednak konstrukcja schematu globalnego nie powinna nastręczać trudności dzięki temu, że struktura **PSP** jest hierarchiczna i relatywnie prosta. Zatem homogeniczne bazy danych wydają się najlepszym rozwiązaniem dla **PSP**.

W **PSP** powinno dążyć się do tego aby lokalne bazy danych były autonomiczne. Zmniejszy to nakłady związane z administrowaniem systemem poprzez przeniesienie ciężaru zarządzania bazą na lokalnych administratorów.

W związku ze specyfiką organizacji można założyć iż ponad 80 % wszystkich zapytań kierowanych do bazy będzie miało zasięg lokalny. Schemat alokacji zatem powinien bazować na całkowitej fragmentacji danych. Spowoduje to iż ruch w systemie będzie minimalizowany ze względu na lokalność odwołań. Jednakże ze względu na czasowe odwołania globalne i wysoki ich priorytet (w sytuacjach akcji ratowniczej) należy dodatkowo zabezpieczyć istotne fragmenty danych replikacją. Replikacja ta może być wykonywana w komendach wojewódzkich a kopie danych powinny służyć jedynie do odczytu w sytuacjach awarii lub gdy żądanie ma wysoki priorytet.

Powstanie tak różnorodnych struktur rozproszonych baz danych wynikało z potrzeb użytkowników jak również sposobów realizacji pewnych problemów technicznych. W związku z rozproszeniem danych problemy związane z **DDB** są dużo bardziej skomplikowane niż w systemach scentralizowanych. Konieczne zatem staje się dokładniejsze omówienie tej problematyki.

4. Aspekty techniczne rozproszonych baz danych

Rozproszone bazy danych powinny dążyć do osiągnięcia takiej samej funkcjonalności jak systemy scentralizowane. Najważniejszymi funkcjami jakie muszą zapewnić użytkownikom to [6,8,12]:

- wsparcie dla zapytań oraz ich optymalizacja,
- dostępność transakcji,
- kontrola wielodostępu,
- integralność pomiędzy poszczególnymi relacjami,
- bezpieczeństwo,
- niezawodność.

Zagadnienia te w systemach rozproszonych są techniczne o wiele trudniejsze do rozwiązania, ze względu na szereg dodatkowych problemów wynikających z rozproszenia

danych. Wymienić tutaj można zależność od sieci transmisyjnej, niejednorodność stosowanego oprogramowania, różnorodność modeli danych i wiele innych. W latach dziewięćdziesiątych problemy te w większości zostały rozwiązane, niemniej prowadzone są nadal badania mające na celu ulepszenie istniejących algorytmów [2, 5].

4.1.Przetwarzanie rozproszonych zapytań

Przetwarzanie zapytania jest procesem, w którym zapytanie otrzymane od aplikacji użytkowej przekształcane jest na niskiego poziomu operację na danych. Optymalizacja zapytania jest natomiast procesem wyboru najlepszej strategii wykonania tego zapytania [8].

W **DDBMS** proces przetwarzania zapytania realizowany jest w czterech etapach [6, 12]:

- dekompozycja zapytania,
- lokalizacja danych, które mają być przetwarzane,
- optymalizacja globalna,
- optymalizacja lokalna w poszczególnych bazach danych.

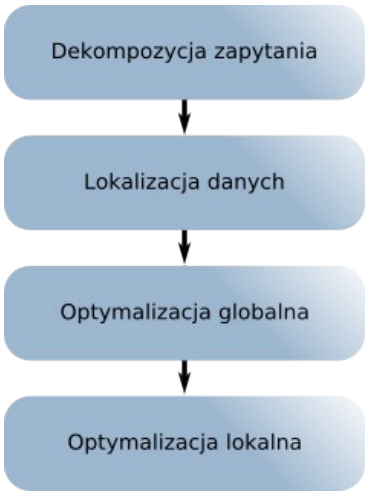
Dekompozycja zapytania, to jego przetworzenie na algebrę relacji. W procesie tym zapytanie poddawane jest semantycznej analizie, upraszczane a w przypadku wykrycia błędu składni odrzucane. Zapytania przetworzone przez moduł dekompozycji odnoszą się do relacji globalnych. Optymalizacja ich przybiega w taki sposób jakby wszystkie zasoby umieszczone były w systemie centralnym. W procesie tym nie są brane pod uwagę informacje dotyczące lokalizacji danych oraz ich podziału na fragmenty. Dlatego po dekompozycji poddawane są dalszej obróbce. Głównym celem kolejnego modułu (lokalizacji danych), jest zlokalizowanie danych przy użyciu schematu alokacji i fragmentacji. Z ich pomocą, zapytanie globalne przetwarzane jest na szereg zapytań odnoszących się do poszczególnych fragmentów (węzłów).

Każda globalna relacja może być odtworzona z lokalnych fragmentów na kilka sposobów. Moduł lokalizacji danych ma za zadanie odrzucić te metody, które są błędne lub też absorbują zbyt dużo zasobów systemowych.

Po podziale zapytania globalnego na szereg operacji na fragmentach, wynik kierowany jest do kolejnego modułu – optymalizacji globalnej. Jego zadaniem jest wyznaczenie optymalnego zbioru zapytań na fragmentach danych, po rozważeniu kosztów komunikacji z poszczególnymi bazami.

Ostatni etap to optymalizacja poszczególnych zapytań w lokalnych bazach danych. Jest to proces bardzo podobny do stosowanego w systemach scentralizowanych. W procesie

tym brane są pod uwagę aspekty wykorzystania indeksów relacji, czy też ich skanowania sekwencyjnego. Na rysunku 6 przedstawiono sekwencję realizacji zapytania globalnego.



Rysunek 6: Przetwarzanie zapytania w systemie rozproszonym

4.2.Kontrola wielodostępu w systemach rozproszonych

Jeżeli wielu użytkowników zamierza jednocześnie dokonać modyfikacji danych w bazie, wówczas operacje te powinny być synchronizowane przez system zarządzania bazą danych. Synchronizacja taka jest realizowana przez mechanizm kontroli wielodostępu (*ang. concurrency control*) [6,19,20]. Głównym jego zadaniem jest izolacja oraz szeregowanie transakcji użytkowników. Protokół implementujący te mechanizmy kontroluje aby transakcje dotyczące tego samego fragmentu danych były uruchamiane dopiero wtedy kiedy zakończone są poprzednie. Ponadto celem zapewnienia wydajności systemu, dąży do tego aby transakcje odnoszące się do różnych elementów danych wykonywane były równolegle.

Istnieje około dwudziestu algorytmów realizujących kontrolę wielodostępu. W systemach rozproszonych najbardziej popularne to [19, 21-23]:

a) dwufazowe blokowanie;
a) znaczniki czasowe;
b) szeregowanie za pomocą wielowymiarowych znaczników czasowych;
c) optymistyczny mechanizm bez blokad.

ad. a) Dwufazowe blokowanie **2PL** (*ang. two-phase locking*). Protokół **2PL** bazuje przydzieleniu każdemu elementowi danych (rekord, atrybut) odrębnej blokady. Jeżeli jakaś transakcja chce dokonać operacji na tych danych musi otrzymać od systemu dostęp do blokady. Do danych przypisane są dwa rodzaje blokad: zapisu oraz odczytu. Zanim system przydzieli transakcji blokadę sprawdza czy jest ona wolna. Gdy warunek ten jest spełniony,

transakcja niezależnie do rodzaju operacji (odczyt/zapis) otrzymuję blokadę odczytu. Może ją teraz zmienić na zapisu, ponieważ jest pewna, że nikt w tym czasie nie dokonuje operacji odczytu (blokada odczytu jest zajęta). Po wykonaniu operacji zwalniane są obie blokady.

Dana transakcja może otrzymać blokadę tylko wtedy kiedy poprzednio nie zwolniła innej blokady. Ma to na celu rozpoznawanie czy transakcja jest w fazie rosnącej czy wygasającej i zabezpieczanie przed możliwymi zakleszczeniami systemu [24]. Dwufazowe blokowanie jest najczęściej stosowaną metodą kontroli wielodostępu. W systemach rozproszonych w zależności od sposobu zarządzania dzieli się dodatkowo na:

1. Scentralizowane **2PL**, w których jeden węzeł w sieci jest odpowiedzialny za zarządzanie tabelą blokad dla całej rozproszonej bazy.
2. 2PL z kopią główną, jest najczęściej wykorzystywany w systemach z replikacją danych, gdzie dane mogą posiadać kilka kopii w różnych węzłach. Jedna z nich jest uznawana jako kopia podstawowa i tylko ona powinna być blokowana w przypadku dostępu do danych. Każdy z węzłów zna lokalizację kopii głównych i prośba o przydzielenie blokady jest kierowana do węzła na którym umieszczona jest kopia główna. Węzeł ten jest odpowiedzialny za kontrolę blokad. Jeżeli baza danych nie jest replikowana wówczas metoda 2PL z kopią główna sprowadza się do rozproszonego **2PL** [12].
3. Rozproszone **2PL**, istnieje wielu zarządców blokad, każdy z nich jest odpowiedzialny za dane przechowywane w jego węźle.

ad. b) Znaczniki czasowe **TS** (*ang. time stamping*). Największym problemem z systemami blokad jest możliwość występowania zakleszczeń. Mają one miejsce wtedy gdy kilka transakcji czeka na zwolnienie wzajemnie blokowanych danych. Problem ten może być rozwiązany przez użycie znaczników czasowych. Każda transakcja ma przyporządkowany znacznik czasu. Ponadto każda operacja wewnątrz transakcji (jeżeli na daną transakcję składa się kilka zapytań) ma również przypisany taki znacznik. W przypadku wystąpienia konfliktu, priorytet ma transakcja o niższym znaczniku czasowym. W systemach centralnych transakcji przydzielane są znaczniki czasowe na podstawie zegara systemowego. W przypadku **DDBMS** należy stosować bardziej wyrafinowane algorytmy, ponieważ zintegrowane bazy danych mogą być niezsynchronizowane pracując w różnych strefach geograficznych. Powszechnie stosowanym rozwiązaniem w rozproszonych **DBMS** jest wykorzystanie w roli znacznika połączenie wartości lokalnego znacznika czasu z unikalnym identyfikatorem węzła [4,21,22]. Dzięki temu transakcja identyfikowana jest z danym węzłem i w przypadku ustalania priorytetu węzły porównują między sobą udostępnione znaczniki czasu.

ad. c) Szeregowanie za pomocą wielowymiarowych znaczników czasowych. W metodzie tej podobnie jak w poprzedniej każdej transakcji przyporządkowywany jest znacznik czasu [19,22]. Jest on odpowiedzialny za ustalanie hierarchii transakcji. Kiedy transakcja dokonuje odczytu jakiegoś fragmentu danych ustawia im swój znacznik. Dane przechowują zatem czas ostatniego odczytu. Kiedy transakcja dokonuje modyfikacji fragmentu danych tworzy jego kopię z ustawionym znacznikiem modyfikacji. Jeżeli w trakcie modyfikacji fragment danych uzyska znacznik odczytu wyższy niż modyfikacji operacja jest wycofywana i powtarzana.

ad. d) Optymistyczny mechanizm bez blokad [22]. Poprzednie mechanizmy kontroli wielodostępu wymagały nadzorowania zapisu/odczytu danych przy użyciu blokad lub znaczników czasowych. Pociągało to za sobą wykorzystywanie znacznych zasobów systemowych, szczególnie wtedy gdy angażowało to operacje na dysku. W pewnych sytuacjach jednak kontrola wielodostępu nie jest wymagana np. w przypadku gdy transakcja dokonuje jedynie odczytu danych.

W modelu optymistycznym transakcja dzielona jest na trzy fazy: odczytu, wartościowania oraz zapisu. W trakcie pierwszej fazy transakcja przeprowadza modyfikację danych w lokalnym buforze pamięci, w trakcie fazy wartościowania system sprawdza spójność zmodyfikowanych danych, natomiast w trzeciej fazie zmiany lokalne otrzymują status globalny. W przypadku odczytu danych transakcje przechodzą wyłącznie dwie pierwsze fazy. Metoda wykorzystuje również znaczniki czasu jako podstawę przeprowadzania wartościowania transakcji.

W przypadku zintegrowanego systemu wymiany danych działającego wewnątrz **PSP** należy się spodziewać iż większość transakcji będzie miała na celu odczyt danych. Dane mogą być modyfikowane w niewielkim stopniu i zazwyczaj sprowadzone to będzie do dodawania nowych wartości. Wynika to przede wszystkich ze specyfiki systemu. Sposobem wykorzystania systemu skłania on się bardziej w kierunku hurtowi danych. Jednakże dane w nim przechowywane nie będą zagregowane oraz typ zapytań ma charakter transakcyjny a nie heurystyczny. W związku z tym w implementacji systemu w **PSP** można by się było kierować w stronę optymistycznego mechanizmu bez blokad. Mechanizm ten ma swoje dodatkowe zalety w postaci małego obciążenia systemu w porównaniu do konkurencji. Jest to istotne w systemach gdzie liczy się prędkość przetworzenia żądań. Jednakże nie może być on zaimplementowany w czystej postaci. Musi zostać poddany modyfikacji w celu zwiększenia jego niezawodności.

4.3.Synchronizacja transakcji w systemie rozproszonym

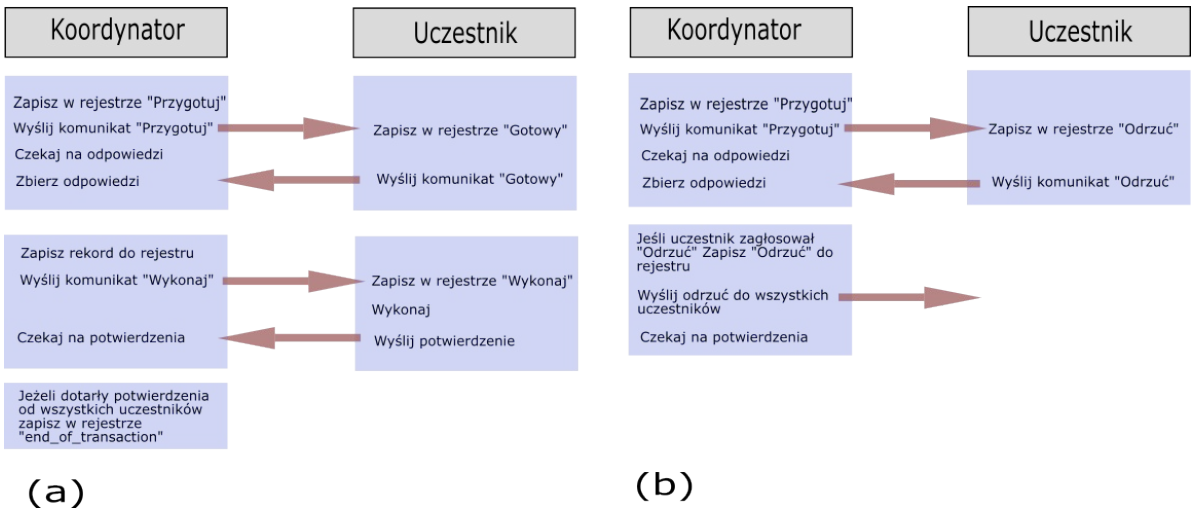
Niezawodność działania rozproszonej bazy danych charakteryzują dwa czynniki: atomowość (niepodzielność) oraz trwałość rozproszonych transakcji. Atomowość wymaga tego, aby wszystkie operacje transakcji zostały wykonane lub wszystkie odrzucone. Zasada trwałości jest natomiast spełniona wtedy gdy po wykonaniu transakcji zmiany mają charakter trwały.

Transakcje mogą być przerwane na skutek różnego rodzaju awarii, wynikających czy to problemów komunikacyjnych, wad sprzętu, oprogramowania czy też zakleszczeń transakcji. W celu poprawnego przeprowadzania transakcji wymagane jest uprzednie sprawdzenie możliwości jej przeprowadzenia a w przypadku nastąpienia awarii możliwość wycofania wprowadzonych zmian.

Aby spełnić wymogi atomowości oraz trwałości transakcji wymagana jest implementacja dwóch protokołów: atomowego wypełniania transakcji oraz rozproszonego odtwarzania. W **DDBMS** najczęściej wykorzystywane algorytmy realizujące atomowe wypełnianie to dwufazowe lub trzyfazowe wypełnianie [2,6].

4.3.1.Wypełnianie dwufazowe

Zgodnie z nazwą wypełnianie dwufazowej **2PC** (*ang. two-phase commit*) składa się z dwóch faz: głosowania oraz decyzyjnej. W pierwszej fazie koordynator transakcji odpytuje wszystkie węzły biorące udział w transakcji o gotowość jej przeprowadzenia. Jeżeli jeden z węzłów głosuje za odrzuceniem transakcji lub nie udzieli odpowiedzi w określonym limicie czasu wówczas koordynator poleca wszystkim uczestnikom odrzucenie transakcji. Jeżeli natomiast wszyscy głosują za wypełnieniem transakcji wówczas koordynator nakazuje im jej wykonanie. W protokole tym zakłada się, że każdy węzeł posiada swój własny lokalny rejestr, na podstawie którego może wiarygodnie wypełnić lub wycofać transakcję. Wypełnienie dwufazowe wymaga, aby procesy oczekiwały na wiadomości z innych węzłów. Stosuje się jednak limity czasu oczekiwania, aby zapobiec niepotrzebnemu blokowaniu procesów. Rysunek 6 przedstawia działanie protokołu **2PC**.



Rysunek 7. Schemat działania protokołu 2PC: (a) protokół 2PC dla uczestnika głosującego za wypełnieniem; (b) protokół 2PC dla uczestnika głosującego za odrzuceniem

Protokół **2PC** posiada kilka wad, między innymi możliwość zablokowania węzłów w trakcie transakcji. Przykładowo zablokowany zostanie proces, który przekroczy limit czasu po głosowaniu za wypełnieniem, lecz przed otrzymaniem globalnego wyniku od koordynatora, który w tym czasie mógł ulec awarii. Prawdopodobieństwo wystąpienia tego typu awarii jest niewielkie, wobec czego **2PC** jest stosowany dosyć często. Znany jest jednak alternatywny protokół nieblokujący, nazywany wypełnianiem trzyfazowym [2].

4.3.2.Wypełnianie trzyfazowe

Wypełnianie trzyfazowe **3PC** (*ang. three-phase commit*) jest protokołem nieblokującym, z wyjątkiem sytuacji w której awarii ulegną wszystkie węzły. Podstawowe rozwiązanie, na którym oparto **3PC**, polega na usunięciu okresu „niepewności", w którym uczestnicy głosujący za wypełnieniem transakcji oczekują, aż koordynator prześle im informacje o globalnym odrzuceniu lub wypełnieniu transakcji. W **3PC** wprowadzono trzecią fazę nazywaną wstępnym wypełnieniem (*ang. pre-commit*), występującą między głosowaniem a globalną decyzją. Po otrzymaniu wszystkich głosów od uczestników zgłaszających gotowość do przeprowadzenia transakcji, koordynator wysyła komunikat „wypełnianie wstępne". Po otrzymaniu komunikatu globalnego wypełniania wstępnego uczestnik wie, że wszyscy głosowali za wypełnieniem. W związku tym może wypełnić transakcję, chyba, że ulegnie uszkodzeniu. Każdy uczestnik wysyła potwierdzenie odebrania komunikatu **PRE-COMMIT**, a koordynator po otrzymaniu wszystkich potwierdzeń dokonuje

globalnego wypełniania. Głosy odrzucania transakcji są dokładnie tak samo obsługiwane jak w **2PC**.

Ze względu na większą niezawodność w systemie wymiany informacji w **PSP** powinien być wykorzystany protokół **3PC**.

4.4.Rozproszony protokół odtwarzania

Transakcja jest zbiorem zapytań, powiązanych ze sobą w ten sposób, iż w przypadku niepowodzenia jednego z nich odrzucane są wszystkie. Mechanizm jej działania zakłada więc wycofanie zmian wprowadzonych przez zapytania jeżeli w trakcie transakcji wystąpiła awaria lub inny powód jej nie zatwierdzenia. Wycofywanie transakcji oraz przywracanie bazy danych do stanu sprzed transakcji jest dziedziną, za które odpowiedzialny jest rozproszony protokół odtwarzania bazy danych (*ang. Distributed recovery protocols*) [2,6,12,25].

Protokół odtwarzania bazy danych traktuje rozproszony system, jako zbiór procesów komunikujących się ze sobą poprzez sieć. Odporność na uszkodzenia realizowana jest poprzez okresowy zapis stanu bazy danych na trwałych nośnikach. Jeżeli wystąpiła awaria systemu, możliwy jest powrót do stanu, którym został zapisany na dysku, zachowując część wykonanych operacji. Momenty w których dokonywany jest zapis nazywane są punktami kontrolnymi (*ang. checkpoint*). Jeżeli system nie wymaga od użytkowników żadnej ingerencji i działa automatycznie zgodnie z ustaloną polityką nazywany jest przeźroczystym [25].

W systemach rozproszonych działanie protokołu komplikuje się, ponieważ procesy komunikują się poprzez sieć. Jeżeli na jednym z węzłów nastąpi awaria, węzeł przywraca bazę do stanu poprzedzającego awarię, przykładowo punku *m*. Dodatkowo węzeł musi wysyłać wiadomości do pozostałych uczestników aby one również przywróciły bazy do stanu sprzed *m*. Jeżeli wyznaczane punktów kontrolnych w poszczególnych węzłach nie jest zsynchronizowane, w łatwy sposób może dojść do „efektu domino”. W trakcie tego procesu bazy wzajemnie wysyłają do siebie informację o wycofywaniu zmian. Może to doprowadzić że przywrócony zostanie stan początkowy, to jest moment uruchomienia bazy, a wszystkie modyfikacje utracone.

W celu uniknięcia efektu domino stosowane jest skoordynowane wyznaczanie punktów kontrolnych w każdym węźle [6].

Inną metodą zapewniającą odporność na uszkodzenia jest wykorzystanie dzienników (*ang. log*). Jest to połączenie wykonywania punktów kontrolnych z zapisywaniem wyjątków. Metoda bazuje na tym, iż wszystkie niestandardowe operacje wykonywane przez procesy są wychwytywane i zapisywane w dzienniku. Stąd też informacje potrzebne

powtórzenia tej operacji są zachowane i w razie potrzeby mogą być odtworzone. Poprzez zapisywanie ich w kolejności wystąpienia możliwy jest powrót do stanu bazy z dowolnego momentu w przeszłości.

4.5.Protokół replikacji

Zwiększenie wydajności systemu jest możliwe poprzez replikację danych. Replikacja zakłada, że część danych posiada swoje kopie w miejscach gdzie zapotrzebowanie na dostęp do nich jest duże. Replikacja jest opłacalna wtedy, gdy dane są rzadko modyfikowane i często wykorzystywane przez lokalnych użytkowników.

Zadaniem rozproszonego systemu zarządzania bazą danych jest utrzymanie spójności pomiędzy wszystkimi kopiami. Najważniejszym kryterium utrzymania spójności danych jest równoważność wszystkich kopii **OCE** (*ang. One copy equivalence*), który mówi iż wszystkie kopie logicznej części danych powinny być identyczne w momencie gdy transakcja je modyfikująca zakończy działanie [12].

Jeżeli zachowana jest przezroczystość replikacji, transakcje przeprowadzały będą operacje na logicznych danych x. Protokół replikacji będzie zaś odpowiedzialny za to aby zmapować operację z logicznej części danych na ich fizyczne kopie x (x_1, x_2, ..., x_n). Typowy protokół kontroli replikacji który zapewnia funkcjonalność **OCE** znany jest jako **ROWA** (*ang. Read Once/Write All*). **ROWA** zamienia odczyt logicznych danych x na odczyt z wybranej fizycznej ich kopii x_i. Właściwa kopia wyznaczana jest na podstawie kryterium wydajności. Każda modyfikacja logicznych danych x wymaga modyfikacji wszystkich ich kopii fizycznych.

Protokół **ROWA** jest prosty i niezawodny, jednak wymaga aby podczas transakcji modyfikacji danych wszystkie kopie logicznej części danych były dostępne. Awaria jakiegokolwiek z węzłów zmniejsza spójność bazy danych.

Istnieją również mniej rygorystyczne co do dostępności węzłów protokoły **ROWA**. Wymagają one aby tylko większość fizycznych danych została zmodyfikowana podczas transakcji.

5. Podsumowanie

W artykule dokonano przeglądu rodzajów rozproszonych baz danych. Dokonano umiejscowienia **DDB** w systemach rozproszonego przetwarzania. Artykuł miał na celu zapoznanie z technologią rozproszonych baz danych a także rozważenie potencjalnych możliwości wprowadzenia ich w PSP i korzyści z tego płynących. Podsumowując informacje

zawarte w artykule można sporządzić porównanie systemów rozproszonych i scentralizowanych. Na korzyść systemów rozproszonych przemawiają między innymi argumenty [2, 4, 6]:

- niskie koszty (moc obliczeniowa do ceny jej uzyskania);
- przyspieszenie obliczeń (dzielenie obciążenia);
- niezawodność (awaria jednego urządzenia nie powinna uniemożliwiać działania systemu, lecz co najwyżej pogorszyć jego wydajność);
- dostępność (lokalnego dostępu poprzez geograficzne rozproszenie);
- możliwość stopniowej rozbudowy;
- skalowalność (zdolność systemu do adaptowania się do wzrastających wymagań).

Niewątpliwą wadą systemów rozproszonych jest ich złożoność i zwiększone trudności z zapewnieniem bezpieczeństwa danych. Złożoność przedkładać się może na występowanie większych ilości awarii. W systemach tego typu jest to sytuacja trudna do przyjęcia dlatego też możliwość wprowadzenia **DDB** w **PSP** musi poprzedzona odpowiednią analizą i zwiększeniem ich niezawodności. Duży nacisk musi być również położony na bezpieczeństwo danych. Informacje przechowywane w tych systemach mają charakter poufny i muszą być należycie zabezpieczone. O ile w systemach scentralizowanych wystarczy odpowiednia ochrona budynku bądź pomieszczenia w którym znajduję się komputer, o tyle w systemach rozproszonych niezbędna jest odpowiednia globalna polityka bezpieczeństwa.

Mimo wymienionych trudności, systemy rozproszone posiadają zalety potwierdzające ich potencjalnie dużą użyteczność w **PSP**. Teoretycznie większe prawdopodobieństwo wystąpienia awarii niwelowane jest faktem, iż awarie te nie wyłączają systemu z użycia a jedynie zmniejszają jego funkcjonalność. W przeciwieństwie do systemów scentralizowanych gdzie awaria jednego z elementów systemu wyłącza go z użytku. Dzięki takim właściwościom systemy rozproszone rozbudowane o dodatkowe mechanizmy bezpieczeństwa znacznie lepiej nadają się dla służb bezpieczeństwa od scentralizowanych. Trudniej jest bowiem odpowiednim aktem terroru unieszkodliwić cały system.

Problematyczna staje się natomiast polityka bezpieczeństwa w systemach rozproszonych. Konieczne jest stosowanie szyfrowania połączeń jak również reglamentowanie dostępu do bazy osobom nieuprawnionym. Istniejące obecnie mechanizmy mogą nie być w stanie zapewnić bezpieczeństwa systemu na najwyższym poziomie dlatego konieczne są dodatkowe prace w tym kierunku.

Znajomość dziedziny rozproszonych baz danych pozwoli podjąć bardziej wyważoną ocenę odnośnie wyboru architektury bazy danych stanowiącej podstawę budowy zintegrowanego systemu wymiany dokumentacji dla **PSP**.

6. Literatura

1. Date C. J.: Wprowadzenie do systemów baz danych. WNT Warszawa 2000
2. Connolly T. Begg C.: Systemy Baz Danych – projektowanie, wdrażanie i zarządzanie w praktyce. RM Warszawa 2004
3. Mendelson H.: Economies of scale in computing: Grosch's law revisited. Communications of the ACM, vol. 30, nr 12, December 1987. str. 1066 - 1072
4. Bell D. Grimson J.: Distributed Database Systems. Addison-Wesley Reading MA 1994
5. Burleson D. K.: Managing Distributed Databases: Building Bridges between Database Islands. John Wiley & Sons New York 1994
6. Ozsu T. M., Valduriez P.: Principles of Distributed Database Systems. Prentice-Hall Wyd. drugie Englewood Cliffs, NJ 1999
7. Silberschatz A., Galvin P. B.: Podstawy systemów operacyjnych. WNT Warszawa 2000
8. Ozsu T. M., Yao B.: Distributed Databases. *Encyclopedia on Distributed Computing Systems, J. Urban and P. Dasgupta (eds.), Kluwer Publishers 1999. http://db.uwaterloo.ca/~ddbms/publications/ozsu/Distdb/distdb.pdf*
9. Clarke R.: Peer-to-Peer (P2P) – An Overview. Xamax Consultancy Pty Ltd, November 2004 http://www.anu.edu.au/people/Roger.Clarke/EC/P2POview.html
10. Fanconi E., Kuper G., Lopatenko A.: A robust logical and computational characterization of peer-to-peer database systems. International Workshop On Databases, Information Systems ona Peer-to-Perr Computing 2003 http://www.inf.unibz.it/~franconi/papers/p2p-03.pdf
11. Fanconi E., Kuper G., Lopatenko A.: A distributed algorithm for robust data sharing and updates in P2P database networks http://www.inf.unibz.it/~franconi/papers/p2pdb-04.pdf
12. M.T. Özsu , "Distributed Database Systems", In *Encyclopedia of Information Systems,* Hossein Bidgoli (ed.), Academic Press, 2003, str. 673-682 http://db.uwaterloo.ca/~ddbms/publications/ozsu/EIC/eic.pdf
13. Inmon W. H.: Building a Data Warehouse (3rd Edition). John Wiley and Sons, Inc. 2002
14. Tsichritzis, D., Klug, A. (eds.): The ANSI/X3/SPARC Framework. Information Systems 3, 1978, str. 173-191

15. Wrembel R.: Integracja danych z wykorzystaniem oprogramowania gateway. Materiały pokonferencyjne Hurtownie danych i Business Intelligence. Warszawa 2004, str. 139-154.
16. Abbott K. R., McCarthy D. R.: Administration and Autonomy In A Replication-Transparent Distributed DBMS. 14th VLDB Conference Los Angeles 1988, str. 195-205. http://www.vldb.org/conf/1988/P195.PDF
17. Dogac A., Dengi C., Kiliac E.: METU Interoperable Database System. ACM SIGMOD Record, vol. 24, nr 3, September 1995 str. 56-61 http://citeseer.ist.psu.edu/79542.html
18. Stonebraker M., Brown P., Hebrach M.: Interoperabilty, Distributed Application and Distibuted Databases: The Virtual Table Interface. IEEE Data Engineerinng Buttetin, nr 3, vol. 21, September 1998, str. 25-34.
19. Pitkanen T.: Information searching and distributed databases. ACM Computer Survey, Vol. 13, No. 2, June 1997, str. 149—183. http://saato014.hut.fi/Hyotyniemi/publications/97_report106/PITKANEN/node1.html
20. Harris T. J., Perrizo W., Ding Q.: Mutliversion post ordering: a new concurrency control method. North Dakota State University http://cs.hbg.psu.edu/~ding/publications/CAINE-1059A.pdf
21. Lamport L.: Time, Clocks and Ordering of Events in Distributed System. Communications of ACM, Nr 7, vol. 21, July 1978, str. 558-565. http://research.microsoft.com/users/lamport/pubs/time-clocks.pdf
22. Munson J., Dewan P.: A concurrency control framework for collaborative systems. Computer Supported Cooperative Work. Proceedings of the 1996 ACM conference on Computer supported cooperative work, str. 278-287 http://www.cs.unc.edu/~dewan/242/s99/notes/trans/node6.html
23. Bernstein P.A., Goodman N.: Concurrency Control in Distributed Database Systems, ACM Computing Surveys, vol. 13, no. 2, str. 185-221, June 1981.
24. Tanenbaum A. S., Steen M.: Distributed Systems: Principles and Paradigms. Prentice Hall 2002
25. Elnozahy M., Alvisi L., Wang A.: A Survey of Rollback-Recovery Protocols in Message-Passing Systems. ACM Computing Surveys 2002, vol. 34, nr 3, str. 375-408. http://citeseer.ist.psu.edu/200905.html
26. System C2 - zintegrowany, ogólnopolski system powiadamiania ratunkowego i zarządzania kryzysowego. Projekt offsetowy 2-104A-G http://www.ziz.com.pl/cpr/PrezentacjaC2%20_Lublin.pdf