PODSTAWOWE POJĘCIA: BAZA DANYCH, SYSTEM ZARZĄDZANIA BAZĄ DANYCH, SCHEMAT LOGICZNY I FIZYCZNY BAZY DANYCH, MODEL BAZY DANYCH. RODZAJE MODELI BAZ DANYCH: RELACYJNY, SIECIOWY, HIERARCHICZNY, OBIEKTOWY – CHARAKTERYSTYKA MODELI I ICH WŁASNOŚCI ORAZ ZASTOSOWANIA. WŁASNOŚCI SYSTEMÓW ZARZĄDZANIA BAZAMI DANYCH.

System bazo-danowy – uniwersalny program udostępniający użytkownikowi informację.

Baza danych – miejsce gdzie przechowuje się informacje – tzw. dane; to inaczej kolekcja informacji umieszczona w określonym miejscu na nośniku; każda informacja w bazie danych musi mieć pewną wartość i strukturę bo inaczej byłaby bezużyteczna.

System Zarządzania Bazą Danych – program umożliwiający zapisanie i dostęp do danych poprzez odpowiednie metody wyszukiwania i zestawiania danych; często używa się pojęcia **System Baz Danych**, co oznacza bazę danych wraz z System Zarządzania Bazą Danych. System Zarządzania Bazą Danych sprawuje kontrolę nad *fizyczną strukturą danych*.
Fizyczna struktura danych wiąże się z miejscem przechowywania danych w komputerach, wyznacza sposoby dostępu do danych, decyduje o tym jak dane się aktualizuje, tworzy i usuwa.

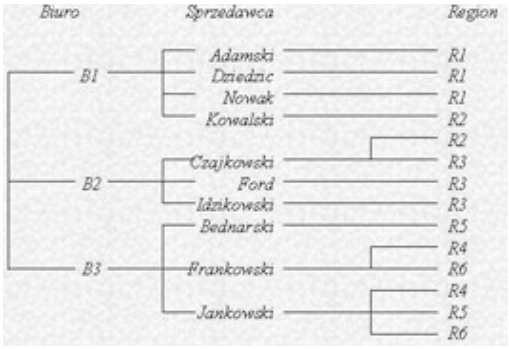
Perspektywa – punkt widzenia użytkownika.

Logiczny model danych - występuje pomiędzy *perspektywą użytkownika* a *fizycznym modelem danych* – to opis bazy danych, z którego można wyprowadzić *perspektywy* poszczególnych użytkowników.

Cechy Systemów Zarządzania Bazami Danych:
- zapewnia współbieżny dostęp do danych wielu użytkownikom;
- bezpieczeństwo i ochronę danych przed niepowołanym dostępem;
- umożliwia restart systemu po awarii sprzętu lub nośników, łączy czy związaną z konfliktami;
- musi odtworzyć bazę danych po awarii;
- optymalizacja dostępu – jak najszybciej należy dostarczyć informację użytkownikowi (czas i koszt);
- musi utrzymywać niezależność danych od aplikacji;
- oparcie fizycznej struktury danych na jednym logicznym modelu danych, który jest odzwierciedleniem perspektyw użytkowników;
- uwzględnia standardy języka zapytań;
- oddziela model fizyczny od modelu logicznego;
- automatycznie obsługuje spójność bazy danych – powiązania między informacjami muszą być spójne;
- redundancja danych – powtarzalność danych musi być ograniczona.

Współbieżny dostęp do danych polega na tym, że: 1. nie ma kłopotu z wieloma czytającymi użytkownikami, 2. jest problem z modyfikowaniem – gdy pojawi się użytkownik chcący zmodyfikować dane pojawia się problem kto ma pierwszy wykonać operację: czytający czy piszący.
Redundancja, awaryjność, bezpieczeństwo – system musi umieć odtworzyć sytuację sprzed awarii; dane muszą zachować swoją poprawność (stosuje się mechanizm dziennika);

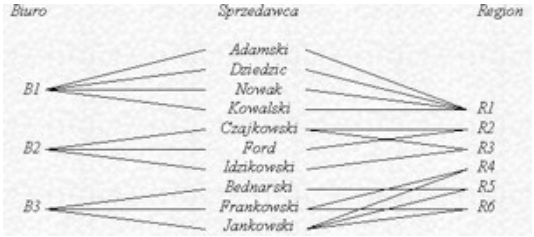
MODELE BAZ DANYCH
Model hierarchiczny danych przedstawia relacje pomiędzy elementami danych w postaci struktury drzewiastej o wielu gałęziach.



każda wartość obiektu danych jest logicznie powiązaną z jedną lub kilkoma wartościami innego obiektu danych . Model hierarchiczny pozwala zobrazować jedynie relacje jedno-jedno i jedno-wiele lub wiele-jedno. W celu uwzględnienia relacji wiele-wiele trzeba je zrestrukturyzować do postaci powtarzających się relacji jedno-wiele. w przypadku struktur drzewiastych, jedna gałąź może prowadzić do wielu podgałęzi, lecz wszystkie podgałęzie prowadzą zawsze do tylko jednej gałęzi wyższego poziomu. Aby dostosować się do tego typu struktury, konieczne jest powtarzanie pewnych obiektów w wielu podgłęziach w celu opisania relacji wiele-wiele poprzez kilka relacji jedno-wiele.

Sieciowy model danych

Na sieciowy model danych można spojrzeć jak na zmodyfikowaną wersję modelu hierarchicznego. Elementy danych są w nim zorganizowane w strukturę drzewiasta podobnie jak w przypadku modelu hierarchicznego. Jednak, inaczej niż w modelu hierarchicznym, model sieciowy pozwala na definiowanie relacji wiele-wiele w postaci struktury drzewiastej bez powtarzania poszczególnych wartości w ramach obiektu danych.



W obu tych modelach elementy danych są zorganizowane w sposób logiczny, co pozwala na powiązanie wszystkich elementów danych poprzez zdefiniowanie koniecznych połączeń. W przypadku wyszukiwania pojedynczej informacji w bazie danych, można przejść wzdłuż odpowiednich połączeń w celu odnalezienia poszukiwanej informacji. Aby było to możliwe, system musi jednak dysponować wbudowanym, dość szczegółowym opisem połączeniem w postaci wskaźników określających powiązania pomiędzy obiektami. Dodatkowo, ze względu na przyśpieszenie wyszukiwania, często konieczne jest ułożenie elementów danych w określonym porządku. Te działania porządkujące realizowane są przez takie operacje jak sortowanie czy indeksowanie.

Relacyjny model danych

FoxPro 2.0, FoxBase, Paradox, Rbase, oraz rodzina oprogramowania dBASE wykorzystują relacyjny model danych do strukturalizacji elementów danych. Może on być stosowany do strukturalizacji relacji typu jedno-jedno, jedno-wiele oraz wiele-wiele. W relacyjnym modelu danych wszystkie elementy danych oraz relacje pomiędzy obiektami danych są zorganizowane w tablice danych. Tablice danych wykorzystuje się do przechowywania elementów danych związanych z konkretnymi obiektami danych. Podobnie, wszystkie relacje pomiędzy obiektami są także zapisane jako elementy danych w tablicach danych. W efekcie, baza danych według modelu relacyjnego składa się ze zbioru tablic danych. Każdy wiersz w tablicy nazywany jest rekordem danych i zawiera pewną liczbę pól danych, z którego każde odpowiada kolumnie w tablicy danych. Zatem, elementy danych w relacyjnej bazie danych są zorganizowane w pliki bazy danych (tablice danych). Każdy plik bazy danych zawiera pewną liczbę rekordów (wierszy tablicy), a każdy rekord - pewną liczbę pól (kolumn tablicy).

Niezależnie od wielkości i złożoności relacyjnej bazy danych, musi ona cechować się pewnymi własnościami. Implementacja tych własności ma kluczowe znaczenie dla uniknięcia powtórzeń elementów danych i poprawnego ich powiązania. Te istotne własności mogą być sformułowane w następujący sposób:

-wszystkie elementy danych muszą być zorganizowane w tablice;

-elementy danych w pojedynczej tablicy muszą pozostawać w relacji jedno-jedno;

-w tablicy danych nie mogą występować powtórzenia tych samych wierszy;

-w tablicy danych nie mogą występować powtórzenia tych samych kolumn;

-w każdej komórce danych może być przechowywany tylko jeden element danych.

Tablica danych jest podstawową jednostką relacyjnej bazy danych. Każda tablica może zawierać dowolną liczbę wierszy (rekordów) i kolumn (pól). Liczba i kolejność kolumn, jak również wierszy, nie jest istotna. Powtarzające się wiersze nie tylko stanowią marnotrawstwo przestrzeni przechowywania danych, lecz przede wszystkim zwalniają proces przetwarzania danych poprzez nieuzasadnioną rozbudowę tablicy. Co ważniejsze, dublujące się wiersze powodują błędne wyniki (niepoprawna liczba wierszy) przy obliczeniach statystycznych w tabeli. Skrzyżowanie kolumny i wiersza w tabeli danych nazywane jest często komórką danych.

Obiektowy model danych – podstawowy element to **obiekt** z **atrybutami i metodami**. Poprzednie systemy zarządzania bazą danych pozwalały na przechowywanie w polach rekordów liczb, dat, Boolean, ciągów, pól MEMO (notatnikowych) Poprzez dodanie do metod do obiektu strukturę można było rozszerzyć o dźwięk, obiekty wektorowe itp. **Obiekty** należą do **klas**. Dopuszczalne jest istnienie obiektów złożonych wskazujących na inne obiekty lub klasy. W obiektowych B.D. wykorzystywane są abstrakcyjne typy danych – jest to kombinacja danych i operacji na nich.

Rodzaje **obiektowych baz danych**:

- cechy Systemów baz danych dodaje się do obiektowego języka programowania;
- cechy Systemów baz danych zostały dodane do języka programowania, który jest rozszerzeniem nieobiektowego języka programowania.
- Systemów baz danych, w których podejście obiektowe zostało dodane do relacyjnej bazy danych.

2. relacyjny model danych. Pojęcie języka definiowania i manipulacji danymi. Pojęcie relacji (DDL, DDL).

- DDL język definiowania danych (data definition language), umożliwia definiowanie struktury danych przechowywanych w bazie, czyli tworzenie schematu implementacyjnego.
- DML język manipulowania danymi (data manipulation language), umożliwia wypełnianie, modyfikowanie i usuwanie informacji z bazy danych.

3. Operacje na relacjach:

Operacje algebry relacji działają na jednej lub wielu relacjach, a ich wynikiem są inne relacje. Wyróżnia się następujące operacje relacyjne:

- Selekcja – umożliwiająca wybór krotek (wierszy) relacji spełniających określone warunki. Operacja ta nazywana jest również **podzbiorem poziomym**.
- Projekcja – umożliwia pobranie wartości wybranych atrybutów, wymienionych po słowie kluczowym SELECT z wszystkich krotek relacji. Operacja ta jest zwana także Podzbiorem pionowym.
- Produkt (iloczyn kartezjański) – jest operacją teorii zbiorów. Operacja ta umożliwia łączenie dwóch lub więcej relacji w taki sposób, że każda krotka pierwszej relacji, jest łączona z każdą krotką drugiej relacji, operacja ta, jest wykonywana na pierwszych dwóch, a następnie na otrzymanym wyniku i relacji następnej, aż do wyczerpania wszystkich argumentów.
- Połączenie – Operacja ta polega na łączeniu krotek dwóch lub więcej relacji z zastosowaniem określonego warunku łączenia. Wynikiem połączenia jest podzbiór produktu relacji.
- Operacje teorii mnogości, jak suma mnogościowa, produkt kartezjański, itp.

Przykłady:

Selekcja

$\Sigma_w(R)=\{K \in R: w(k)\}$ w jest warunkiem selekcji

selekcja jest komutatywna:

$\Sigma_v(\Sigma_w(R))= \Sigma_w(\Sigma_v(R))$

Nr_indeksu	Imię	Nazwisko	Wykształcenie
33445	Celina	Arbuz	WT
40435	Józef	Maliniak	SO

23890	Stefan	Piech	ST
22456	Wiktor	Fajfer	SO
26876	Krzysztof	Majer	ST

W języku SQL wykonanie selekcji umożliwia rozkaz SELECT z klauzulą WHERE. Przykładowo polecenie:
SELECT * FROM osoby ;
Spowoduje wybranie wszystkich krotek (wierszy) z relacji (tabeli) osoby.
W celu pobrania wierszy, dla których pole w kolumnie 'wykształcenie' jest równe „SO" należy wpisać:
SELECT * FROM osoby WHERE wykształcenie = 'SO'
Warunki selekcji mogą być złożone. Przykładowo, aby wybrać wszystkie osoby, które mają wykształcenie techniczne (średnie techniczne – ST lub wyższe techniczne WT) można odpowiednie warunki połączyć spójnikiem logicznym OR:
SELECT * FROM osoby WHERE wykształcenie = 'ST' OR wykształcenie = 'WT'.

Projekcja (rzut)

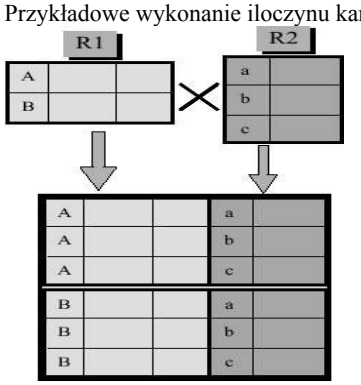
$\Pi_{S'}(R)=\{k(S'):k\in R\}$ gdzie S' jest podzbiorem schematu S
projekcja jest wzajemnie komutatywna z selekcją

$\Pi_{S'}(\Sigma_w(R))=\Sigma_w(\Pi_{S'}(R))$
o ile warunek selekcji ma sens projekcji, tj. dotyczy tylko atrybutów wybranych w projekcji.

Nr_indeksu	**Imię**	**Nazwisko**	**Wykształcenie**
33445	Celina	Arbuz	WT
40435	Józef	Maliniak	SO
23890	Stefan	Piech	ST
22456	Wiktor	Fajfer	SO
26876	Krzysztof	Majer	ST

Przykładową operację projekcji pokazaną na rysunku można wykonać za pomocą następującego rozkazu SELECT:
SELECT Nr_indeksu, Wykształcenie FROM osoby;
Operacje selekcji i projekcji mogą być łączone w jednym rozkazie SELECT. I tak chcąc otrzymać kolumny zawierające Nr_indeksu i Nazwisko osób mających średnie wykształcenie należy napisać:
SELECT Nr_indeksu, Nazwisko FROM osoby WHERE Wykształcenie = 'ST' OR Wykształcenie = 'SO'

Produkt
Przykładowe wykonanie iloczynu kartezjańskiego przedstawia rysunek:



Znajdowanie iloczynu dwóch relacji (tabel) jest również wykonane przez rozkaz SELECT. Przedstawioną na rysunku operację można wykonać za pomocą rozkazu:
SELECT * FROM R1, R2 ;
Operacja znajdowania iloczynu kartezjańskiego może być łączona zarówno z operacją selekcji, jak również projekcji lub oboma równocześnie.

Połączenie

$R'*R''=\{k'||k'': k'\in R' \wedge k''\in R'' \wedge w(k',k'')\}$ – w jest warunkiem złączenia
Krotki złączenia stanowią sklejenie (konkatenację) krotek relacji złącznych.

Autorzy Książki

1	Adam	Mickiewicz
2	Bjarne	Stroustrup

1	Pan Tadeusz	1
2	Język C++	2
3	Dziady	1

Adam	Mickiewicz	Pan Tadeusz
Adam	Mickiewicz	Dziady

Powyższy przykład uzyskać można następującym poleceniem SELECT:
SELECT imię, nazwisko, tytuł FROM autorzy, książki
WHERE autorzy.nazwisko=‘Mickiewicz’ and autorzy.nr=książki.autor;

Unia
Pozwala na zsumowanie zbiorów krotek dwóch lub więcej relacji (bez powtórzeń – zgodnie z teorią mnogości). Warunkiem poprawności tej operacji jest zgodność liczby i typów atrybutów (kolumn) sumowanych relacji. Przykład przedstawiony poniżej sumuje zbiory pracowników i właścicieli okrojone do imienia i nazwiska (za pomocą projekcji), w celu uzyskania informacji o wszystkich ludziach powiązanych z firmą:

```
SELECT imię, nazwisko FROM pracownicy
UNION
SELECT imię, nazwisko FROM właściciele ;
```

Przekrój
Przekrój pozwala znaleźć iloczyn dwóch lub więcej zbiorów krotek tzn. takich, które występują zarówno w jednej jak i w drugiej relacji. Podobnie jak w przypadku unii, warunkiem poprawności tej operacji jest zgodność liczby i typów atrybutów relacji bazowych.
Poniższy przykład znajduje wszystkie nazwiska(np. stosunek pracy, powiązania rodzinne), które występują w relacji pracownicy jak i w relacji właściciele.

```
SELECT nazwisko FROM pracownicy
INTERSECT
SELECT nazwisko FROM właściciele;
```

Różnica
Operacja obliczania różnicy dwóch relacji polega na znalezieniu wszystkich krotek, które występują w pierwszej relacji, ale nie występują w drugiej.
Przykład znajduje wszystkie osoby, które są współwłaścicielami spółki, ale nie są w niej zatrudnieni:

```
SELECT imię, nazwisko FROM właściciele
MINUS
SELECT imię, nazwisko FROM pracownicy;
```

Grupowanie
Klauzule GROUP BY i HAVING występujące w rozkazie SELECT pozwalają dzielić relację wynikową na grupy, wybierać niektóre z tych grup i na każdej z nich z osobna wykonywać pewne (dozwolone przez system) operacje. Operacje te działają tylko na wszystkich wierszach wchodzących w skład grupy. Na samym końcu zwracana jest tylko zbiorcza informacja o wybranych grupach (nie zwraca się wszystkich wierszy wchodzących w skład grupy).
Klauzula GROUP BY służy do dzielenia krotek relacji na mniejsze grupy. Sposób takiego podziału ilustruje przykład:

```
SELECT stanowisko, avg(placa_podstawowa) FROM pracownicy
GROUP BY stanowisko;
```

Istnieje możliwość odrzucenia pewnych krotek przed podziałem na grupy. Dokonuje się tego za pomocą klauzuli WHERE:

```
SELECT stanowisko, avg(placa_podstawowa) FROM pracownicy
WHERE stanowisko != ‘kierowca’ GROUP BY stanowisko;
```

Dzielenie na grupy może być zagnieżdżane, co umożliwia wydzielanie podgrup w uprzednio znalezionych podgrupach. W przykładzie poniżej wszyscy pracownicy są dzieleni na wydziały, w których pracują, a w ramach każdego wydziału grupowani według stanowiska:

```
SELECT wydział, stanowisko, avg(placa_podstawowa)
FROM pracownicy GROUP BY nr_wydziału, stanowisko;
```

Klauzula HAVING ogranicza wyświetlanie grup do tych, które spełniają określony warunek. Chcąc wyświetlić tylko te grup, w których płaca podstawowa przynajmniej jednego pracownika jest większa niż 3000 należy zastosować następujące zapytanie:

```
SELECT stanowisko, max(placa_podstawowa) FROM pracownicy
GROUP BY stanowisko HAVING max(placa_podstawowa) > 3000;
```

ZALEŻNOŚCI FUNKCYJNE I WIELOWARTOŚCIOWE. ZASADY NORMALIZACJI BAZ DANYCH

Relacja może służyć do modelowania „świata rzeczywistego" na kilka sposobów; na przykład każda krotka relacji może reprezentować pewną encję i jej atrybuty lub związek między encjami. Często dzięki znanym faktom o świecie rzeczywistym wiadomo, że nie każdy skończony zbiór krotek może być bieżącą wartością pewnej relacji, jeśli nawet krotki są odpowiedniej krotności i składowe wybrane spośród właściwych dziedzin. Można rozróżnić dwa rodzaje ograniczeń nałożonych na relacje.

1) Ograniczenia zależne od znaczenia elementów dziedzin. Te ograniczenia zależą od zrozumienia tego, że znaczenie mają składowe krotek (np. nikt nie ma 60 stóp wysokości o nikt w wieku 27 lat nie jest zatrudniony przez lat 37). Warto, aby system zarządzania bazą danych sprawdzał takie niewiarygodne wartości, powstające prawdopodobnie na skutek błędu podczas wprowadzania lub obliczania danych.
2) Ograniczenia zależne jedynie od równości lub nierówności wartości. Istnieją inne węzły, które nie zależą od wartości żadnej składowej krotki, lecz tylko od tego, czy określone składowe dwóch krotek są zgodne, czy nie. Omówimy tu najważniejsze z takich węzłów, zwane zależnościami funkcyjnymi.

Niech R(A1,A2,. .., An) będzie schematem relacji i niech X oraz Y będą podzbiorami {A1, A2,. .., An}. Mówimy, że X->Y, czytaj „X wyznacza funkcyjnie Y" lub „Y zależy funkcyjnie od X", jeśli dla dowolnej relacji r będącej bieżącą wartością R nie jest możliwe, aby r zawierała dwie krotki, mające składowe zgodne dla wszystkich atrybutów ze zbioru X, a jednocześnie co najmniej jedną niezgodną składową dla atrybutów ze zbioru Y.
Zależności funkcyjne powstają w różny sposób. Na przykład, jeśli R reprezentuje zbiór encji o atrybutach A1, A2,. .., An i X jest zbiorem atrybutów tworzących klucz do tego zbioru encji, można stwierdzić, że X->Y dla dowolnego podzbiory atrybutów Y. Wynika to z faktu, że krotki każdej możliwej relacji r reprezentują encje, a encje są identyfikowane za pomocą wartości atrybutów należących do klucza. Stąd dwie krotki, których atrybuty ze zbioru X są zgodne, musiałyby reprezentować tę samą encję, a zatem musiałyby być tą samą krotką. Podobnie, gdy R reprezentuje odwzorowanie jednoznaczne ze zbioru encji E_1 w zbiór encji E_2 i pośród A_1 są atrybuty tworzące X do E_1 oraz klucz Y do E_2, wówczas X->Y i faktycznie dowolny zbiór atrybutów R. Zależność Y->X nie będzie jednak zachodziła, chyba że odwzorowanie jest jednoznaczne.

Należy podkreślić, że zależności funkcyjne są stwierdzeniami odnoszącymi się do wszystkich możliwych relacji mogących tworzyć wartości schematu relacji R. Nie można na przykładzie konkretnej relacji r schematu R wywnioskować, jakie są zależności funkcyjne w R. Jeśli na przykład r jest zbiorem pustym, to okazuje się, że zachodzą wszelkie zależności, lecz w ogólności nie musi tak być, gdy zmienią się wartości relacji oznaczonej przez R. Dla określonej relacji R można jednak stwierdzić, że pewne zależności nie zachodzą.
Logiczne implikacje zależności
Przypuśćmy że R jest schematem relacji, a A,B i C są niektórymi z jego atrybutów. Przypuśćmy również że wiadomo, iż w R zachodzą zależności funkcyjne A->B B->C. Twierdzimy, że w R może także zachodzić A->C. Dla potrzeb dowodu przypuśćmy, że r jest relacją spełniającą A->B B->C, lecz że istnieją w niej dwie krotki t i u, takie że obie mają zgodne składowe dla A i niezgodne składowe dla B, to r narusza B->C, ponieważ nie są zgodne składowe dla C. Stąd r musi spełniać A->C. W ogólności, niech F będzie zbiorem zależności funkcyjnych dla schematu relacji R i X->Y zależnością funkcyjną. Mówimy że
X->Y wynika logicznie z F i zapisujemy to jako F |= X->Y, jeśli każda relacja r dla schematu relacji R, spełniająca zależności z F, spełnia również X->Y. Widzimy więc że jeśli F zawiera A->B B->C, to A->C wynika logicznie z F. To znaczy {A->B, B->C} |= A->C. Niech F^+, domknięcie F, będzie zbiorem zależności funkcyjnych wynikających logicznie z F,
tj. $F^+ = \{X->Y| F|= X->Y\}$.

SQL

Cechy SQL:

1. zawiera funkcje agregujące (suma,min...)

2. mechanizmy aktualizacji BD – aktualizacja pól kluczowych i zbiorów indeksowych (szybsze wyszukiwanie informacji)
3. mechanizmy zmiany wartości atrybutów, dopisywanie i usuwanie krotek
4. możliwość tworzenia własnych typów danych, tzw. **bloby**
5. selekcja może być wykonana za pomocą klauzul selekcji
6. łączenie tabel może być wykonane za pomocą selektora

1. Tworzenie tabeli

Nową tabelę tworzymy przy pomocy instrukcji

CREATE TABLE <nazwa tablicy>
(<nazwa kolumny><typ danych>
[,<nazwa kolumny><typ danych>...])

np.
CREATE *TABLE Test (aaa CHAR(10), bbb CHAR(12));*

2. Modyfikacja struktury istniejącej tabeli

ALTER TABLE <nazwa>
ADD <nazwa kolumny><typ danych>
[,<nazwa kolumny><typ danych>...],
DROP <nazwa kolumny>;

np.
ALTER TABLE Ilosc
ADD mistrz CHAR(15),
DROP nr_mistrz;

Do tablicy Ilość zostanie dodana kolumna o nazwie mistrz, a usunięta kolumna o nazwie nr_mistrz

3. Perspektywy

Perspektywy są to tablice wirtualne tzn. takie, które fizycznie w pamięci nie istnieją i służą tylko do innej organizacji już istniejących tablic. Perspektywa może zawierać kombinacje wierszy i kolumn wybranych z jednej lub kilku tablic. Może być wykorzystywana do wybrania i wyprowadzenia żądanych informacji analogicznie jak dla tablic. Przy pomocy perspektywy można również wprowadzać i aktualizować informację wtedy, gdy perspektywa powstała z kolumn pojedynczej tablicy. Perspektywę tworzy się przy pomocy instrukcji:

CREATE VIEW <nazwa perspektywy> [(<lista kolumn>)]
AS < podinstrukcja SELECT > [WITH CHECK
OPTION]

<lista kolumn> podaje nazwy kolumn, które mają wystąpić w tworzonej perspektywie. Jeżeli opcja ta nie wystąpi, perspektywa będzie zawierała nazwy kolumn wyszczególnione w instrukcji SELECT. Podanie nazw kolumn jest konieczne w przypadku, gdy pewna kolumna jest otrzymana w wyniku działania wyrażenia.

Druga część instrukcji CREATE VIEW zawiera instrukcję SELECT definiującą wiersze i kolumny z wyróżnionej tablicy, które będą zawarte w perspektywie. Opcja WITH CHECK OPTION jest wykorzystywana wtedy, gdy chcemy, aby każdy wiersz umieszczony w perspektywie spełniał warunek wyszczególniony w opcji WHERE instrukcji SELECT.

Przykład

CREATE VIEW P1
AS SELECT material, ilosc
FROM Ilość WHERE material ='
'szyby';

Zostanie utworzona perspektywa o nazwie P1 zawierająca kolumny materiał, ilość z tablicy Ilość i wiersze, dla których materiał = 'szyby', Zawartość perspektywy P1 można wyprowadzić przy pomocy instrukcii:

SELECT *FROM P1;

Należy podkreślić, że przy każdej aktualizacji tablicy Ilość zostanie również zaktualizowana perspektywa P1,

Do usunięcia zbędnej perspektywy wykorzystuje się instrukcję **DROP VIEW.**

Przykład

DROP VIEW Mag;
Zostanie usunięta perspektywa Mag.

4. Instrukcja SELECT

Instrukcja ta służy do wyprowadzania informacji umieszczonych w tablicach

SELECT <klauzula>
FROM <klauzula>
[WHERE <klauzula>]
[GROUP BY <klauzula>]
[HAVING <klauzula>]
[ORDER BY <klauzula>]

np
SELECT place
FROM pracownik
WHERE place=3000

Wyświetli płace z tabeli pracownik, gdzie płace=300

4.1 GROUP BY i HAVING

Klauzula GROUP BY układa wiersze w grupy, w których określona kolumna ma tę samą wartość, a następnie w tablicy wynikowej redukuje tę grupę do pojedynczego wiersza. Dodatkowo klauzula HAVING podaje warunek, który musi spełniać każda grupa wyszczególniona w klauzuli GROUP BY, aby mogła się znaleźć w tabeli wynikowej

Np.
SELECT nr, ksiazka
FROM biblioteka
GROUP BY ksiazka
HAVING nr>3

Wyszukanie nr i ksiazka, z relacji biblioteka. Wcześniej pogrupowanie wg książek, gdzie spełniony mysi być waruneknr>3

5. DISTINCT

Eliminuje duplikaty w zakresie nazw atrybutów (wyprowadza tylko różne wiersze)

6. ORDER BY

Określa sposób sortowania wyników zapytania (asc – rosnąco, desc- malejąco). Sortowanie odbywa się albo względem atrybutów albo względem wyrażenia

SELECT [DISTINCT] atrybuty
FROM relacja
WHERE warunek
ORDER BY {atrybut, wyrażenie} {asc,desc}

7. Warunki

Warunki w formie zapytania SELECT zapisuje się za pomocą operatorów relacyjnych

a)BETWEEN AND

Sprawdzenie czy lewy operator wyrażenia zawarty w przedziale ograniczonych wartości spełnia wystąpienie. Wartości od do są podawane po słowach between and

WHERE zarobki BETWEEN 100 AND 900

b)IN

Operator umożliwia sprawdzenie czy lewy operand jest elementem listy będącej prawym operandem. Listę podaje się w nawiasie okrągłym

WHERE student IN („Kowalski”,”Nowak”)

c)LIKE

Sprawdzenie czy operand pasuje do podanego wzorca
WHERE student LIKE „Kow%”

d)IS NULL

umożliwia sprawdzenie, czy operand ma wartość nieokreśloną, pustą

8. Funkcje agregujące

a)COUNT() – zlicza liczbę krotek
b)SUM() – sumuje wartości w kolumnach numerycznych
c)MIN() - znajduje minimalną wartość tekstową, typu danych daty lub numeryczną
d)MAX() – znajduje wartość maksymalną
e)AVG() – oblicza średnią wartość w kolumnach numerycznych

```
SELECT AVG(place), SUM(pensja)
FROM pracownik
WHERE MAX(place)=800
```

9. Zapytania złożone

Zapytania w SQL mogą być zagnieżdżone – wynik jednego zapytania może być użyty jako warunek selekcji innego zapytania

Zapytanie zagnieżdżone wykonywane przed zapytanie zewn. nazywane jest zagnieżdżonym (sub-query)

```
SELECT a1,a2
FROM relacja
WHERE a1=
                    (SELECT a1
                     FROM relacja2
                     WHERE warunek);
```

Np.

```
SELECT etat, placa
FROM pracownik
WHERE placa=
                   (SELECT min(placa)
                    FROM kierownik)
```

10. Łączenie tabel

Łączenie dwóch tabel przy pomocy SELECT'a (wykorzystujemy najczęściej, gdy operujemy kodami a nie pełnymi informacjami). Operacja ta polega na łączeniu dwóch lub więcej relacji z zastosowaniem określonego warunku łączenia. Wynikiem połączenia jest podzbiór produktu relacji

```
SELECT imie, nazwisko, tytul
FROM autorzy, ksiazki
WHERE autorzy.nazwisko = 'Mickiewicz' and autorzy.nr = ksiazki.autor
```

Wyświetli pola imię, nazwisko, tytuł z połączonych dwóch tabel przy spełnionych warunkach.

11. Operacje mnogościowe

Suma (Union) krotek dwóch lub więcej relacji (krotki się nie powtarzają). Warunkiem poprawności jest by była wzajemna zgodność pól i ich typów w obu relacjach z których ta suma powstaje

```
SELECT imie, nazwisko
FROM studenci
UNION
SELECT imie, nazwisko
FROM nauczyciele
```

Iloczyn krotek – dotyczy dwóch lub więcej relacji, podobnie wymagana jest zgodność typów. Zamiast Union – Intersect

```
SELECT imie, nazwisko
FROM studenci
INTERSECT
SELECT imie, nazwisko
FROM nauczyciele
```

Różnica – polega na pobraniu krotek występujących w pierwszej relacji, a nie wyst. w drugiej relacji, podobnie wymagane warunki i podobna składnia. Występuje słowo minus

```
SELECT imie, nazwisko
FROM studenci
MINUS
SELECT imie, nazwisko
FROM nauczyciele
```

ANOMALIE

Problemy które powstają, gdy próbujemy do pojedynczej relacji włączyć zbyt wiele danych, nazywane są anomaliami.

Tytuł	Rok	Długość	Typ_filmu	Nazwa_studia	Nazwisko_gwiady
Gwiezdne Wojny	1977	124	Kolor	Fox	Carrie Fisher
Gwiezdne Wojny	1977	124	Kolor	Fox	Mark Hamil
Gwiezdne Wojny	1977	124	Kolor	Fox	Harrison Ford
Potężne Kaczory	1991	104	Kolor	Disney	Emilio Estevez
Świat Wayne'a	1992	95	Kolor	Paramount	Mike Meyers
Świat Wayne'a	1992	95	Kolor	Paramount	Dana Carvey

1. Redundancja – Dane niepotrzebnie powtarzają się w kilku krotkach. Jako przykład mamy tu długość filmu i rok produkcji.
2. Anomalie modyfikacji – Może się zdarzyć, że wartość danej zostanie zmodyfikowana w pewnej krotce, a w innej nie. Na przykład w pewnym momencie okaże się, że Gwiezdna Wojny trwają naprawdę 125 minut i beztrosko zmienimy wartość tego atrybutu w pierwszej krotce ale druga i trzecia pozostają bez zmian.
3. Anomalie usunięć – W przypadku gdy dla pewnego atrybutu zaczyna obowiązywać wartość pusta, to jako efekt uboczny może się zdarzyć utrata danych. Jeżeli np. ze zbioru gwiazd filmu usuniemy nazwisko Emilio Estevez, to w bazie nie będzie już żadnych danych o gwiazdach tego filmu. Z relacji zniknie wówczas ostatnia krotka zawierająca dane o filmie Potężne Kaczory, a wraz z nią informacje o tym, że film trwa 95 minut i że jest kolorowy

INDEKSOWANIE.

Podstawową techniką stosowaną w systemach DBMS z zakresu wyszukiwania danych i języków zapytań jest technika indeksowania. B. d. indeksuje się jednym lub wieloma kluczami. Dla każdego pojedynczego lub zagregowanego klucza określa się zbiory indeksowe pojedynczego lub zagregowanego indeksu. Niektóre zbiory b. d. przechowywane są w formie zindeksowanej dla b. d. mogą być tworzone zbiory indeksowe czasowo, w dowolnym momencie gdyż są one technikami użytecznymi. W niektórych systemach pamięć zajmowana przez indeksy jest większa niż pamięć zajmowana przez dane. Ważne jest by organizacja indeksu była najefektywniejsza. Projekt musi dążyć do tego by organizacja indeksów była jak najefektywniejsza tzn. powinien dążyć do zminimalizowania obszaru pamięci zajmowanego przez indeksy oraz czasu potrzebnego do ich poszukiwania. Indeks powinien być tak zaprojektowany by możliwe było dołączenie i usunięcie pozycji, oraz by utrzymywanie indeksów nie było czasochłonne.

Indeks jest tablicą na której wykonuje się operacje przeglądania. Jesteśmy zainteresowani czasem przeszukiwania indeksów. Pole użyte do szukania nazywa się **argumentem**, a pole które otrzymamy z tablicy nazywa się **funkcją.** Z jednym argumentem może być powiązana jedna lub wiele funkcji.

Funkcja indeksu b. d. może przyjąć jedną z następujących postaci:

- adres rekordu - w większości przypadków indeks podaje maszynowy adres rekordu; często zależnie od rekordu wymaga przeszukiwania kilku poziomów indeksów;
- względny adres rekordu - stało się to ważne z chwilą upowszechnienia techniki stronicowania
- symboliczny adres rekordu - jeżeli indeks podaje pozycję określonych rekordów to jest on w efekcie tablicą wskaźników, które mogą być adresami maszynowymi, względnymi lub symbolicznymi. W indeksach wtórnych i skorowidzach używa się niekiedy wskaźnika symbolicznego w celu oddzielenia struktury indeksu wtórnego od fizycznego rozmieszczenia danych. Indeks wtórny wiąże bezpośrednio z kluczami wtórnymi. Gdy utworzony zostanie zbiór indeksów dla takiego klucza może on wskazywać np.: na pierwsze takie wystąpienie w b. d. dla którego określony klucz ma taką a nie inna wartość. Zbiory indeksów wtórnych mogą być czasami b. duże i przepisywanie ich po każdej reorganizacji pliku byłoby czasochłonne. Dlatego też indeks może podawać klucz główny szukanego rekordu za pomocą dowolnej metody adresowania pliku będzie się znajdo-

wało żądany rekord. Używanie indeksów symbolicznych wydłuża czas potrzebny do znalezienia rekordu, lecz ułatwia utrzymanie indeksu.

- umiejscowienie porcji - niektóre indeksy nie wskazują pojedynczego rekordu lecz obszar, w którym mieści się większa liczba rekordów. Obszar ten zwany niekiedy porcją może być ścieżka dysku lub obszarem kontrolowanym lub obszarem dobranym jako odpowiedni z punktu widzenia zastosowanej techniki indeksowania. Po uzyskaniu dostępu do porcji trzeba porcję te przeszukać aby odnaleźć wymagany rekord. Liczba rekordów w porcji, na które wskazuje indeks, charakteryzuje rozdzielczość indeksu. Indeks wskazujący pojedyncze porcje określa się mianem **indeksu porcji**. Indeks wskazujący pojedyncze rekordy nazywa się **Indeksem rekordu.** Indeks porcji jest mniejszy i jego przeglądanie trwa krócej. Indeks porcji może być używany wtedy jeśli rekordy mogą być grupowane w tych samych porcjach oraz wtedy gdy zapis rekordów odbywa się sekwencyjnie a nie uporządkowaniem losowym. Wskaźniki zapisane w indeksie porcji mogą być adresami: maszynowymi, względnymi, symbolicznymi. Symbolicznym wskaźnikiem indeksu może być klucz główny pierwszego rekordu tej porcji.
- adres łańcucha - gdy używane jest w b. d. łączenie łańcuchowe
- wartości atrybutów - niektóre indeksy wtórne podają zmiany adresów wartości atrybutów. W wielu przypadkach pozwala to udzielić odpowiedzi na zapytania dotyczące kluczy wtórnych bez konieczności przeglądania rekordów danych. Wartość atrybutu dostarczana przez indeks wtórny po dodatkowych operacjach może być przekształcona w adres rekordu danych. Niekiedy indeks składa się z wierszy kluczy, z których jeden będący kluczem głównym pozwala poprzez dalsze operacje adresowania odnaleźć rekord danych. W niektórych systemach różnica między rekordami indeksu a rekordami danych staje się nieuchwytna. Rekordy indeksu zawierają dane, które nie muszą być powtarzane w rekordach danych.
- wyjście wielokrotne - każdy zapis w indeksie głównym podaje pojedynczy wskaźnik. Zapis może określać wiele wskaźników. Mogą one wskazywać rekordy porcje lub fragmenty łańcucha, będące adresami maszynowymi, względnymi, symbolicznymi lub wartościami atrybutów.

Jeżeli zapis w indeksie może przyjmować jedną zwartości skończonego zbioru, to przedstawiciele każdej wartości zapisu w postaci binarnej pozwala zaoszczędzić obszar pamięci. Ograniczeniem przechowywania zapisów w formie liczb binarnych jest to, że potrzeba dodatkowej tablicy w celu zapamiętania wartości w pierwotnej formie. Jeśli tablica wartości jest mała to może być przechowywana w pamięci operacyjnej lub w ostateczności może być dostępna poprzez operację stronicowania. W niektórych indeksach każdy argument może mieć wiele f-cji ze stronicowego zbioru funkcji. Konieczna jest wtedy minimalizacja liczby bitów koniecznych do zapamiętania tych f-cji. Są tu duże metody ich zapamiętania.

Indeks gęsty to taki indeks, który przyjmuje wszystkie możliwe wartości klucza. Indeksy wtórne są na ogół gęste, tzn. zawierają wszystkie możliwe wartości klucza głównego. Indeks główny nie musi być indeksem gęstym jeśli rekordy ułożone są w sekwencji swoich kluczy głównych; indeks może wskazać ścieżkę lub obszar pliku, który musi być przeszukany; w tym przypadku indeks nazywa się **indeksem rzadkim.** Za pomocą indeksu znaleźć można tylko konkretnie wybrane wartości. Indeks o takich cechach nazywa się **indeksem znaczonym.**

Wiele atrybutów indeksowych przyjmuje wartości dyskretne. Niekiedy **rozkład wartości atrybutów** jest ciągły. Wtedy rozkład wartości może zostać podzielony na odpowiednie odcinki, podobnie jak podziałka linijki. Operacje taką nazywa się **kwantyzację**. Redukuje ona liczbę wymaganych zapisów w indeksie. Argument indeksu może się odnosić do konkretnej wartości lub przedziału wartości. Taki indeks nazywa się **indeksem zakresu.**

Podczas projektowania plików wielokluczowych, przeznaczony do obsługi zapytań spontanicznych projektant musi postawić pytanie: dla których argumentów tworzyć indeks. Jednocześnie pożądane jest, by indeks nie był zbyt obszerny. Projektant może preferować indeksowanie atrybutów mających mały zbiór wartości, gdyż wówczas lista będzie składać się z małej liczby zapisów. Niestety atrybuty o małej liczbie wartości nie są zbyt selektywne. Np. w bibliotece wartość atrybutu „fantastyka" może się odnosić do dużej części książek, a tym samym indeks tej wartości jest w procesie indeksowania mało pomocy. Odwrotnie atrybuty dostarczając małą ilość pozycji na zapytanie będzie prawdopodobnie miał dużą liczbę indeksowanych wartości. Wyjściem z tego dylematu jest łącznie w indeksie pewnych atrybutów, pod kątem ich łącznego występowania w kryteriach przeszukiwania. Z powodu wielkości indeksów, w szczególności indeksów wtórnych ważne są techniki **kompresji** zapisów w indeksie. Wielkość niektórych indeksów ulegnie zmniejszeniu, jeśli przedstawi się je w postaci **binarnej**. Inne można

zmniejszyć przez usunięcie nieznaczących części kluczy. Przykładem takiego postępowania jest odcięcie końcowej części klucza w **indeksie wielopoziomowym**.

Do zlokalizowania rekordu konieczne jest by najbardziej znaczące znaki jego klucza różniły się od najwyższego klucza z następnej porcji.

Jeszcze większą kompresję można uzyskać za pomocą techniki zwanej **rozbiorem.**

Przy rozbiorze najwyższy poziom indeksu zawiera część klucza. Następny poziom indeksu zawiera jeden lub kilka następnych znaków klucza, jednak bez powtarzania tych, które już zostały zapisane na najwyższym poziomie. Trzy najwyższe poziomie indeksu składają się z zapisów jednoliterowych. Odpowiadają one trzem pierwszym literom. Czwarty poziom nie powtarza już tych trzech liter. Nie trzeba żadnych dodatkowych oznaczeń wskazujących początek i koniec zapisu. Jeśli porcje wskazywane przez czwarty poziom indeksu miałby zmienną pojemność, to ten właśnie poziomu indeksu mógłby mieć stałą długość. Wadą tego sposobu jest to, że niektóre bloki indeksu zawierać będzie więcej zapisów niż dziesięciozapisowe bloki w poprzedniej technice. W tym przypadku gdy kliki tworzą trzy poziomowe indeksy, to mogą być przeszukiwanie binarne. W tej technice może być opłacane dodanie dodatkowego poziomu indeksu, składającego się z zapisu jednoliterowego, pomiędzy poziomu 3 i 4. Zaleta metody może być to, że plik może być bardzo zmienny, a potrzeba będzie bardzo niewiele zmian w trzech górnych poziomach indeksu. Mechanizm dołączenia i usuwania elementów zb. indeksowego koncentrują się przede wszystkim od poziomu 4.

METODY DOSTĘPU DO BAZ DANYCH

Dostęp sekwencyjny
Możliwy dla plików sekwencyjnych, w których uporządkowane rekordy są uporządkowane według wartości jednego lub więcej pól. Jednym z najbardziej znanych algorytmów jest algorytm wyszukiwania binarnego, którego działanie polega na ciągłym zmniejszaniu obszaru wyszukiwania o połowę.

Dostęp haszowany
Możliwy dla plików haszowanych, które posiadają tylko jeden porządek haszowania

Dostęp przez indeks
Mechanizm dostępu który jest dodawany do bazy danych aby usprawnić jej działanie bez wpływu na strukture przechowywania danych. Na indeksie (plik o dwóch polach) wykonujemy przetwarzanie wykorzystujące algorytm taki jak wyszukiwanie binarne.
W praktyce większość indeksów jest implementowana za pomocą pewnych postaci B – drzew („drzewa wyważone” – hierarchiczna struktura danych).

ZASADY ZARZĄDZANIA WSPÓŁBIEŻNOŚCIĄ OPERACJI NA BAZIE DANYCH. POJĘCIE TRANSAKCJI W BAZIE DANYCH. METODY BLOKOWANIA. HARMONOGRAMOWANIE TRANSAKCJI. METODA OPTYMISTYCZNA W ZARZĄDZANIU DOSTĘPEM DO BAZY DANYCH. IMPASY I METODY ICH WYKRYWANIA

Transakcja – jest to jedno wykonanie programu. Programem może być proste zapytanie wyrażone w jednym z języków zapytań lub skomplikowany program w języku macierzystym z wbudowanymi odwołaniami do języka zapytań. Równocześnie może być w toku kilka niezależnych wykonań tego samego programu; każde z nich jest transakcją.

Impas – Sytuacja, w której każdy element zbiotu S dwóch lub większej ilczby transakcji czeka na blokadę jednostki właśnie zablokowanej przez pewną inną transakcję ze zbioru S, jest nazwana *impasem* (deadlock). Ponieważ każda transakcja ze zbioru S czeka, nie można więc odblokować jednostki potrzebnej do dalszego wykonywania pewnej innej transakcji ze zbioru S, a zatem wszystkie czekają bez końca.
Kilka sposobów rozwiązania tego problemu:

1. Wymaga się, aby każda transakcja zgłaszała od razu wszystkie swoje żądania blokowania. System albo przydziela je wszystkie, albo, gdy jedna lub więcej blokad jest nałożonych przez inną transakcję, nie przydziela żadnej i powoduje, że proces czeka.
2. Przypisuje się jednostkom pewne arbitalne uporządkowanie i wymaga się, aby wszystkie transakcje żądały blokad w tej kolejności.
3. Inny sposób radzenia sobie z impasem polega na tym, że ni robi się nic, aby mu zapobiegać. Bada się okresowo żądania blokowania i sprawdza, czy im-

pas występuje. Sprawdzania łatwo dokonać za pomocą algorytmu rysowania grafu oczekiwania. Jego wierzchołkami są transakcje, a krawędzie T1->T2 oznaczają, że transakcja T1 czeka na zablokowanie jednostki, którą blokuje T2; każdy cykl oznacza występowanie impasu, a jeśli nie ma cykli, to nie ma także żadnego impasu. Jeśli jednak impas został jednak wykryty, to trzeba zacząć ponownie wykonywać co najmniej jedną z powodujących go transakcji, a wyniki jej dotychczasowego działania muszą być usunięte z bazy danych.

ZARZĄDZANIE WSPÓŁBIEŻNOŚCIĄ

Pojęcie transakcji

- Transakcja jest to pewien ciąg czynności na bazie danych (np. pojedyncze wyszukiwanie informacji). Transakcja może być zapisana z poziomu programu lub w trybie konwersacyjnym

Współbieżność w rozproszonej bazie danych

Zapewnienie szeregowalności zachowania transakcji w środowisku rozproszonym jest znacznie utrudnione. Wynika to z faktu, że transakcje mogą być wykorzystywane w różnych węzłach i mogą mieć dostęp do danych w wielu węzłach. Jeśli do sterowania współbieżnością zastosować metody blokowania, to zazwyczaj przesyła się o wiele więcej komunikatów dotyczących samego blokowania niż przenoszonych danych. Ponieważ przesyłanie nawet krótkich komunikatów jest kosztowne, w środowisku rozproszonym zalecana jest lepsza gruba ziarnistość blokowania; oznacza to, że jednostki do blokowania powinny być dużymi obiektami, być może całymi relacjami, tak by liczba blokad nakładanych na transakcję nie była duża. Zakłada się, że w środowisku rozproszonym istnieje wiele węzłów lub miejsc, w których mogą być przechowywane dane i wykonywane transakcje. Może istnieć wiele kopii każdej jednostki danych zapamiętanych w różnych węzłach i jeśli tak, to częścią problemu sterowania współbieżnością jest zapewnienie identyczności wszystkich kopii. Zakłada się dalej, że operacjami na BD są transakcje READ i WRITE oraz, że transakcja może zablokować jednostkę. Transakcje są dwufazowe – to znaczy mają zapewnione wszystkie blokady jednostek, których chciały, dokonują na nich tych akcji których powinny dokonać, a następnie zdejmują blokady. Blokowanie należy interpretować tak, jak gdyby transakcje pojawiały się w chwili, w której zapewniono jej wszystkie blokady. Pogwałcenie szeregowalności może wystąpić wtedy gdy jedna transakcja blokuje zapis, a druga blokuje całkowicie tę samą jednostkę w tym samym czasie.

METODY BLOKOWANIA W ROZPROSZONEJ BAZIE DANYCH

- ***Całkowite blokowanie wszystkich – blokada zapisu jednej***

Zakładamy, że cała rozproszona baza danych znajduje się w określonej ilości węzłów i określona jednostka znajduje się w określonej ilości węzłów. Transakcja blokuje zapis jednostki A, gdy blokuje ona zapis dowolnej kopii tej jednostki. Transakcja całkowicie blokuje jednostki A, gdy całkowicie blokuje wszystkie jej kopie. Blokowanie zapisu można zapewnić dotąd, aż żadna transakcja nie blokuje całkowicie jednostki. Jednostka A nie może być jednocześnie blokowana całkowicie i blokadą zapisu. Wprowadzamy oznaczenia: n – liczba węzłów w bazie danych; dla każdej jednostki bazy danych jest n kopii jednostki. Rozważmy przepływ komunikatów w rozproszonej sieci. Gdy zakładamy blokadę na jednostkę A nie wiemy ile jest jej kopii. Aby wykonać WLOCK A musi być wysłana blokada zapisu do jednej kopii. Jeżeli ma być blokada całkowita, to trzeba wysłać komunikat do wszystkich węzłów zawierających jednostkę. Trzeba następnie czekać na odpowiedź z węzłów, że jest to możliwe. Potem przesyłamy nową wartość i komunikat o odblokowaniu. Komunikat przesyłający nową wartość będzie komunikatem długim. W rezultacie – przy blokowaniu całkowitym wysyłamy 3n komunikatów krótkich i n komunikatów długich. Jeżeli do komunikatu długiego dołączymy komunikat o odblokowaniu jednostki, to zaoszczędzimy 1n komunikatów krótkich (czyli będzie 2n komunikatów krótkich).Jeżeli żądanie blokady zapisu zostało odrzucone, to nie szukamy następnej kopii jednostki w bazie danych.

- ***Metoda blokowania większości***

Transakcja zablokowała zapis jednostki A jeżeli zablokowała zapis większości jej kopii. Transakcja zablokowała jednostkę A jeżeli zablokowała całkowicie większość jej kopii.

Szacowanie ilości komunikatów.

Aby założyć blokadę trzeba przesłać (n+1)/2 komunikatów o zablokowaniu. Wysyłamy n komunikatów z nową wartością i n+1 komunikatów o założeniu blokady i uzyskaniu odpowiedzi. Sam długi komunikat może zawierać informację o odblokowaniu. Przy odczycie mamy (n+1)/2 komunikatów o zablokowaniu i tyle samo musi być odpowiedzi o założeniu blokady. Jeżeli transakcja wykonuje się w węźle z kopią, to przy odczycie długiego komunikatu nie musi być przesłany komunikat krótki o potrzebie zablokowania tej jednostki. Mamy n- komunikatów krótkich i 0 komunikatów długich. Ale przesyłamy n komunikatów o odblokowaniu. Metoda większościowa jest efektywniejsza przy zapisie, natomiast jest ona bardziej kosztowna. Inna zaleta tej metody ujawnia się gdy transakcje często chcą blokować tę samą jednostkę. Gdy pojawiają się 2 transakcje w tym samym czasie i chcą zablokować co najmniej dwie kopie jednostki, to może się pojawić impas, bo dopiero po zablokowaniu wszystkich jednostek można działać. Uogólnieniem tej metody jest **k z n.** (k – ilość węzłów, które muszą być zablokowane spośród n jednostek). Zanim uzna się że jednostka A została całkowicie zblokowana, to musi być zablokowane k z n jednostek. Przy blokowaniu zapisu musi być zablokowane n-k+2 jednostek. Ta metoda obowiązuje dal k>n/2. Tak dobieramy k aby impas występował najrzadziej.

- ***Metoda kopii pierwotnej***

Zakładamy, że zarządzanie blokadami jest powierzone konkretnemu węzłowi. W szczególnym przypadku jeden węzeł sieci odpowiada za zarządzanie blokadami w całej sieci. Wszystkie blokady są zakładane na kopii pierwotnej jednostki.

- ***Metoda żetonu w formie pierwotnej***

Modyfikacją powyższej jest metoda, która dodatkowo w węźle pierwotnym wydaje żetony (odczytu i zapisu). Są to inaczej uprawnienia przyznawane określonym węzłom sieci. Dla dowolnej jednostki A może istnieć jeden żeton zapisu i wiele żetonów odczytu. Żetony te są przekazywane z węzła do węzła. Posiadanie przez węzeł żetonu zapisu A może zapewnić transakcji wykonywanie w tym węźle zablokowania całkowitego lub zapisu. Jeżeli węzeł ma żeton odczytu jednostki A, to może zapewnić blokowanie jej zapisu transakcji, która jest w tym węźle, ale nie zapewnia blokowania całkowitego. Ta metoda nazywana jest metodą żetonu kopii pierwotnej.

Jeżeli transakcja zarządza całkowitego zablokowania jednostki A, to należy spowodować by do tego węzła został przesłany żeton zapisu A. Gdy w węźle nie ma żetonu to przesyłamy komunikaty do pozostałych węzłów zawierających jednostkę z żądaniem wyrzeczenia się żetonu. Wszystkie żetony odczytu i zapisu muszą dojść do tego węzła.
Z każdego węzła zawierającego jednostkę A mogą być przesłane trzy komunikaty:
1) o zrzeczeniu się żetonu (zapisu i odczytu);
2) o stwierdzeniu braku żetonu w węźle;
3) informacja, że węzeł nie może zrzec się żetonu, bo trwa blokada jednostki A

Występują dodatkowe uwarunkowania: węzeł żądający żetonu musi wiedzieć do jakich węzłów trzeba wysłać komunikaty. Baza posiada rejestr posiadanych żetonów dla węzłów. Komunikaty wymagają potwierdzenia. Węzeł który pierwszy chce modyfikować jednostkę rezerwuje węzeł dla siebie. Inne transakcje są niejako odrzucane. To prowadzi do wielokrotnego żądania blokowania jednostki do zapisu. Żeton zapisu to 3n komunikatów krótkich

Żądanie odczytu jednostki A – ogólna zasada jest taka sama (też są żetony). Różnica jest taka, że jeżeli w węźle pojawi się transakcja żądająca odczytu A i jednostka posiada żeton odczytu, to nie ma komunikatów przesyłanych na zewnątrz. Jeżeli jednostka nie posiada żądanego żetonu, żąda żeton odczytu i sprawdza czy nie dokonuje się jakiś żeton. Odpowiedź z innych węzłów może być:

1) węzeł może przekazać żeton odczytu;
2) węzeł nie ma żadnego żetonu dla jednostki A;
3) węzeł nie może oddać takiego żetonu, bo zawiera transakcję całkowicie blokującą jednostkę A;

Gdy węzeł może dokonywać operacji odczytu to przesyłane jest 3n (4n) komunikatów. Jeżeli w węźle nie ma jednostki A, to potrzebny jest jeszcze długi komunikat odczytujący wartość A z innego węzła.

Metoda żetonu kopii pierwotnej wymaga większej ilości przesłanych komunikatów.
Główną zaletą jest to, że jeżeli w danym węźle wykonuje się większość transakcji dotyczących danej jednostki, to z reguły żeton zapisu znajduje się w tym węźle. Metoda węzła pierwotnego nie jest metodą bezpieczną (gdy zachodzi węzeł pierwotny).

- ***Metoda węzła centralnego***

Metoda ta polega na przypisaniu odpowiedzialności za blokowanie jednemu węzłowi. Węzłem pierwotnym jest tylko jeden węzeł-centralny. Nie musi zawierać kopii jednostki którą należy blokować. Blokada zapisu: do węzła centralnego przesyłane jest żądanie blokady zapisu. Jeżeli blokada nie może nie może być zapewniona to do źródła wysyła się odpowiedni komunikat. Żeby sprawdzić czy blokada jest założona węzeł centralny wysyła komunikaty do węzłów z kopiami. Jeżeli blokada może być założona to węzeł z kopią wysyła wartość do węzła żądającego blokady. Często jest potrzebny dodatkowy komunikat do węzła z kopią o konieczności przesłania żądanej wartości. Na podobnych zasadach odbywa się blokowanie całkowite dla modyfikowania jednostki. Węzeł żądający blokady wysyła na koniec żądanie zdjęcia blokady.
Niestety opisywana metoda jest wolniejsza od poprzedniej. Kolejną jej wadą jest to, że większość komunikatów jest przesyłana od / do węzła centralnego, a zatem węzeł centralny jest wąskim gardłem przy komunikacji. Dodatkowo metoda ta nie jest bezpieczna – szczególnie w przypadku awarii węzła centralnego cały mechanizm obsługi staje się bezużyteczny. Są jednak metody wykrywające awarie węzła centralnego i przejmujące jego funkcje.

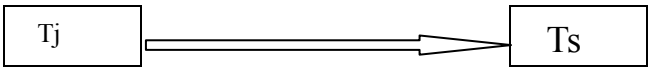
METODY BLOKOWANIA

Harmonogram – kolejność wykonywania transakcji z dokładnością do elementarnych operacji wykonywanych przez transakcję, tj. blokowania, czytania, odblokowania, zapisu, itp. Harmonogram jest **sekwencyjny**, jeżeli wszystkie operacje każdej transakcji występują kolejno po sobie. Harmonogram jest **szeregowalny** (dający się uszeregować), jeżeli wynik działania tego harmonogramu jest równoważny wynikowi otrzymanemu za pomocą pewnego harmonogramu sekwencyjnego. Jeżeli w momencie pojawienia się harmonogramu okaże się że jest on szeregowalny, to transakcja może być wykonywana w systemie bazodanowym. W module zarządzania blokadami sprawdza się czy harmonogram jest szeregowalny. Nie każdy harmonogram jest sprawdzany przy realizacji w implementacjach BD. Większość harmonogramów jest sprawdzana z punktu widzenia operacji zapisu, odczytu, zablokowania, odblokowania. Poprzez program sprawdzający szeregowalność harmonogramu można osiągnąć rozwiązanie problemu impasu. Dla wszystkich jednostek bazy danych wprowadza się jeden lub więcej protokołów przestrzeganych przez wszystkie transakcje. Na przykład protokół, w którym przyjęto określoną metodę blokowania jednostek w określonej kolejności. W normalnych harmonogramach występują jeszcze operacje zakładania i zdejmowania blokady, dopiero na podstawie tego rozpatrujemy problem szeregowalności.

- ***Algorytm szeregowania dla operacji Lock i UnLock realizowanych dla każdej operacji Read i Write.***

S – harmonogram składający się z transakcji T1...Tk

Metoda sprawdzania szeregowalności polegać będzie na utworzeniu grafu pierwszeństwa (graf zorientowany). Wierzchołkami będą transakcje. W grafie należy określić krawędzie. Wyznacza się je tak:
Przyjmujemy, że mamy dwie transakcje: Tj – ma zadanie Lock albo UnLock. Jeżeli operacja w harmonogramie jest operacją odblokowania, czyli operacją na transakcji Tj, Tj:UNLOCK[Am], to szukamy w harmonogramie takiej transakcji Ts, która blokuje tę samą jednostkę bazy danych Am. Ts:LOCK[Am]. W grafie dokonujemy wtedy połączenia Tj i Ts:



Jeżeli graf pierwszeństwa zawiera cykl, to taki harmonogram nie jest szeregowalny. Gdy w grafie nie ma cyklu, to harmonogram jest szeregowalny i można do grafu zastosować metodę sortowania topologicznego, która pozwoli na sprowadzenie tego szeregowalnego harmonogramu do harmonogramu sekwencyjnego. Polega to na znalezieniu w grafie pierwszeństwa takiego wierzchołka Ti, do którego nie dochodzą żadne inne krawędzie. Taki wierzchołek wpisujemy jako pierwsza transakcja w realizacji, usuwamy ten wierzchołek z grafu i znowu szukamy w podgrafie. Powtarzamy to, aż nie usuniemy wszystkich wierzchołków z grafu pierw-

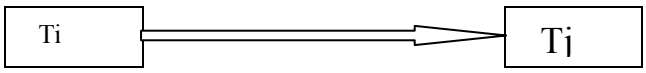
szeństwa. Kolejność usuwania wierzchołków definiuje harmonogram sekwencyjny, który dowodzi szeregowalności harmonogramu.

- ***Metoda zakładania blokad różnego typu posiadających wspólny UnLock***

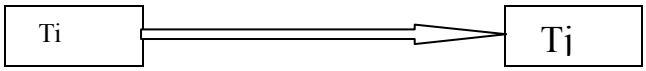
Poprzedni model jest dość ograniczony, bo przy każdej operacji trzeba jednostkę zablokować i odblokować. Lepszym sposobem blokowania jest zakładanie blokad różnego typu, posiadających wspólny UnLock. W tym nowym modelu istnieją dwie blokady: zapisu (współdzielona - Rlock) i całkowita (wyłączna - WLock).
Tabela zgodności blokad: „T" dla T1[RlockA] i T2 [RlockA], dla reszty „F"
Jeżeli mamy wiele transakcji T1...Tk zakładających Rlock, to ona tak długo obowiązuje, aż transakcje odczytają wszystkie dane. Gdy pojawi się WLock (było już Rlock) to taka transakcja ustawia się w kolejce i czeka aż Rlock nie zostanie zwolniona na danej jednostce bazy danych. Obowiązuje jedno odblokowanie UnLock.

Algorytm konstrukcji grafu:

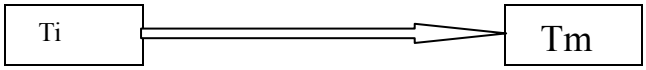
1) W harmonogramie jest transakcja Ti, która blokuje zapis jednostki A (RlockA), a Tj jest następną w harmonogramie transakcją, która chce całkowicie zablokować A (WlockA). Tj występuje po Ti:



2) Przypuśćmy, że Ti blokuje całkowicie A (WlockA), a Tj jest następną transakcją, która chce całkowicie zablokować a (WlockA); prowadzimy krawędź z Ti do Tj



Niech Tm będzie transakcją RlockA, gdy Ti zdjęła wcześniej blokadę na A (UnLockA), a Tj całkowicie blokuje A, to poprowadzimy krawędź z Ti do Tm



- ***Model poszerzony o dwufazowość***

Kolejny model został poszerzony o tzw. **dwufazowość**, czyli wg protokółu dwufazowego oznaczamy go jako 2PL lub **Two Phase Locking**. Polega on na tym, że każda transakcja przebiega w dwóch fazach: zablokowania i odblokowania, więc nie ma oddzielnego odblokowania. Moment założenia wszystkich blokad nazywa się punktem akceptacji. Zapis realizuje się dopiero w fazie odblokowania, a jednostka zostaje odczytana w fazie blokady. Można dodatkowo założyć, że jednostka zablokowana i zwolniona, może mieć nową blokadę dopiero po jakimś czasie. Inaczej wygląda harmonogram szeregowalności, graf będzie multigrafem, gdzie niektóre ścieżki mogą być traktowane wariantowo. Algorytm jest podobny do poprzednich. Ta metoda jest najczęściej implementowana w SZBD. Jest połączona z wypełnieniem i realizacją dziennika transakcji.

- ***Metoda hierarhicznego blokowania***

W niektórych bazach danych można zakładać blokady na całe relacje, krotki, wybrane elementy. Gdy w systemie jest możliwe powyższe blokowanie, to nazywamy ją metodą hierarchicznego blokowania. Jeżeli weźmiemy zbyt małe jednostki do blokowania, to możemy otrzymać niespójność bazy danych. Ważne jest jaką jednostką dysponujemy. Całą bazę możemy rozpatrywać jako hierarchię drzewiastą, korzeń to baza danych, a kolejne wierzchołki to relacje. Dla relacji można wyznaczać kolejne jednostki, które możemy blokować.
Metodę hierarchiczną opracował Grey. Oparta jest ona na większej ilości blokad:

1) współdzielna i wyłączna (S i X),
2) intencjonalna blokada współdzielna (IS),
3) intencjonalna blokada wyłączna (IX),
4) blokada mieszana (SIX), która jest połączeniem S i IX.

Blokada intencjonalna współdzielona (IS) wierzchołka W w drzewie hierarchii blokad oznacza, że wystąpiła intencja odczytu co najmniej jednego następnika wierzchołka W. Blokada IS pozwala zakładać blokady IS lub S na wszystkie następniki wierzchołka W. Blokada intencjonalna wyłączna (IX) wierzchołka W oznacza, że wystąpiła intencja zapisu co najmniej jednego jego następnika. Blokada IX pozwala na zakładanie blokad IX, X, IS, S oraz SIX na następnikach

wierzchołka W. Blokada mieszana (SIX) wierzchołka W pozwala na dostęp współdzielony do danych należących do wierzchołka W i do jego następników oraz pozwala na zakładanie na tych następnikach blokad wyłącznych intencjonalnych.

		T2				
		IS	IX	S	SIX	X
T1	IS	T	T	T	T	
	IX	T	T			
	S	T		T		
	SIX	T				
	X					

Algorytm hierarchicznego blokowania transakcji obejmuje 4 kroki:

1) Pierwszym wierzchołkiem dla którego transakcja T żąda dostępu o danych jest korzeń hierarchii ziarnistej lub korzeń poddrzewa.
2) Transakcja T może założyć blokadę IS lub S wierzchołka nie będącego korzeniem ⇔ gdy T blokuje poprzednik wierzchołka blokadą IS lub IX
3) Transakcja T może założyć blokadę IX lub SIX lub X na wierzchołku W, który nie jest korzeniem ⇔ gdy blokuje go poprzednik blokadą IS lub SIX.
4) Wszystkie blokady założone przez transakcję T muszą być odblokowane po zakończeniu tej transakcji lub w trakcie jej wykonywania w kolejności odwrotnej do kolejności ich zakładania.

W tej stosuje się jeszcze modyfikację – sposób zapisu drzewa zmieniany jest na B-drzewo. Do harmonogramu blokowania hierarchicznego stosuje się specjalne protokoły ostrzeżeń, które przestrzegają ten algorytm przed właściwą realizacją blokad.

- ***Bezpieczeństwo przy awarii na tle metod blokowania transakcji:***

System musi mieć dodatkowe mechanizmy zabezpieczenia przed awariami. Dotyczy to tylko scentralizowanych baz danych. Inne metody dotyczą systemów rozproszonych. Mówimy o awarii powodujących błędną interpretację danych i są odwracalne (do naprawy przez SZBD). Aby zabezpieczyć system baz danych stosuje się metody:

- Okresowe sporządzanie kopii BD bez wiedzy użytkowników,
- Stworzenie kopii jest realizowanie jako odrębna transakcja na danych. Może być realizowane planowo lub tworzy się sama po upływie jakiegoś czasu.
- Mechanizmy dzienników: Dziennik Bazy Danych obejmuje identyfikator transakcji powodującej zmianę w BD, starą i nową modyfikowanej jednostki danych. Mogą być też zapisywane dodatkowe informacje, np. czy wypełniono transakcję do końca.

METODY OPTYMISTYCZNE

Metody optymistyczne współbieżnej realizacji operacji na BD polegają na synchronizacji transakcji. Do metod tych zaliczamy metodę opartą o etykiety – znaczniki czasowe oraz metody tzw. walidacji. Metoda optymistyczna nie traci czasu na blokady oraz na ich usuwanie. Czas poświęcony na powtórzenie transakcji, które muszą zostać odrzucone może być znacznie krótszy niż czas zużyty na oczekiwanie na blokady lub usuwanie impasu. Metoda optymistyczna jest efektywna pod warunkiem, że małe jest prawdopodobieństwo konfliktu dwóch transakcji. Szansa wzajemnego oddziaływania dwóch transakcji wykonywanych w tym samym czasie jest mała pod warunkiem, że każda z nich ma dostęp tylko do drobnego fragmentu BD. Dla omawianych metod niejako zakłada się wprowadzenie uporządkowania sekwencyjnego i wykrywania jego naruszeń. Efekt taki można uzyskać jeśli posiada się system tworzenia tzw. znaczników czasowych (*timestemps*).

Znaczniki czasowe są nadawane pojawiającym się transakcjom do obsługi. Znaczniki czasowe są liczbami generowanymi w każdym takcie zegara. Takty wewnętrznego zegara komputera występują z tak dużą częstotliwością, że dowolne dwa zdarzenia (np. zgłoszenie transakcji) nie mogą zajść w tym samym takcie. Wynika stąd, że każdej transakcji nadaje się inny znacznik czasowy, który jest bieżącym czasem zegara. Znacznik czasowy nadajemy w chwili inicjalizacji transakcji lub w momencie jej odczytu lub zapisu na dysk. Żadne dwie transakcje nie mają tego samego znacznika czasowego. Za kryterium poprawności sposobu ich przyporządkowania przyjmujemy, że transakcje powinny zachowywać się tak, jak gdyby kolejność ich znaczników była uporządkowaniem sekwencyjnym transakcji. Znaczniki liczbowe będą dużymi liczbami. Dla większości zastosowań wystarczy 16 – bitowy znacznik czasowy dla każdej transakcji. Rozważymy teraz w jaki sposób za pomocą znaczników czasowych zmusza się nie odrzucone transakcje do tego, by zachowywały się tak, jak gdyby były wykonywane sekwencyjnie. Dla każdej jednostki BD zapamiętuje się dwa czasy:

- czas odczytu – największy znacznik czasowy należący do dowolnej transakcji odczytującej tę jednostkę.
- czas zapisu – największy znacznik należący do dowolnej transakcji zapisującej tę jednostkę.

Robiąc to można uzyskać wrażenie, że transakcji wykonywana jest w jednej chwili wskazywanej przez jej znacznik czasowy. Znaczniki czasowe przypisane transakcji i zapamiętywanie czasów odczytu i zapisu służą do sprawdzenia, czy nie zdarza się nic fizycznie niemożliwego.

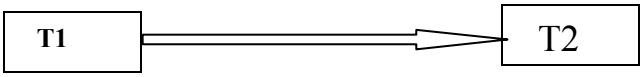
- nie jest możliwe by transakcja mogła odczytać wartość jednostki, jeśli tę jednostkę zapisano już po wykonaniu tej transakcji. Inaczej mówiąc transakcja ze znacznikiem czasowym t1 nie może odczytać jednostki o czasie zapisu t2 takim, że t2 > t1.Jeśli transakcja chce tego dokonać, to należy ja odrzucić i ponownie rozpocząć. (1)
- nie jest możliwe by transakcja mogła zapisywać jednostkę, jeśli stała wartość tej jednostki ma być później odczytana. Oznacza to, że transakcja ze znacznikiem t1 nie może zapisać jednostki o czasie odczytu t2, jeśli t2 > t1. Taką transakcję trzeba odrzucić. (2)

Nie budzi wątpliwości, ze dwie transakcje mogą odczytywać te same jednostki bez jakiegokolwiek konfliktu, tzn. transakcja ze znacznikiem t1 może czytać tą samą jednostkę o czasie odczytu t2 jeśli t2 > t1. Mniej oczywistym wydaje się fakt, że nie trzeba odrzucać transakcji ze znacznikiem t1, jeśli próbuje ona zapisać jednostkę o czasie zapisu t2, nawet jeśli t2 > t1.Fakt ten uzasadniamy w następujący sposób: jednostka jest zapisywana przez transakcję t1 a następnie przez t2 w uporządkowaniu sekwencyjnym opartym na znacznikach czasowych. Widać jednak z warunków (1) i (2), że między chwilami t1 i t2 jednostka nie jest odczytywana przez żadną transakcję gdyż w przeciwnym razie jej czas odczytu przekroczyłby czas t1, w którym zapisuje ją pierwsza transakcja. Spowodowałoby to odrzucenie transakcji na mocy punktu (2). Wynika stad, że przestrzegając pewnych reguł i wykonując transakcje ze znacznikami czasowymi zamiast blokowania można zachować uporządkowanie sekwencyjne.

Metody usuwania impasu

Sytuacja, w której każdy element zbioru S dwóch lub większej ilości transakcji czeka na blokadę jednostki właśnie zablokowanej przez pewną transakcję ze zbioru S nazywa się impasem (*deadlock*). Istnieje kilka sposobów na pokonanie tego problemu:

- Każda transakcja zgłasza od razu swoje żądania blokowania zaś system albo przydzielała je wszystkie albo w przypadku gdy jedna lub więcej blokad cząstkowych jest już nałożona przez inne transakcje, nie przydziela takiej blokady i powoduje, że cała taka transakcja oczekuje na blokadę.
- Przypisuje się jednostkom pewne arbitralne uporządkowanie i wymaga się by wszystkie żądały blokad w tej kolejności.
- Nie robi się nic by zapobiegać impasom. Bada się za to okresowo żądania blokowania i sprawdza, czy impas występuje. Stosuje się tu zazwyczaj metodę opartą na rysowaniu grafu oczekiwań. Wierzchołkami takiego grafu są transakcje, a jego krawędzie oznaczają, że transakcja T1 czeka na zablokowanie jednostki która jest zablokowana przez T2



Jeżeli taki graf oczekiwań zostanie utworzony i w grafie tym będą występowały cykle to mamy wówczas dowód na istnienie impasu. Jeżeli taki impas został wykryty, to trzeba zacząć ponownie wykonywać jedną z powodujących ten impas transakcję, a wyniki dotychczasowego działania tej transakcji muszą być usunięte z BD. Taki proces ponownego startu przy impasie może być i często jest skomplikowany jeśli nie opracowano specjalnego sposobu zapisu zmian w BD.

Strategia dwufazowego wypełniania.

Przy tworzeniu zabezpieczeń antyawaryjnych dla rozproszonych BD istotną rolę odgrywa m.in. proces badania czy dana transakcja została wypełniona. Stąd wniosek, że akcja wypełnienia transakcji powinna być zapisana w dzienniku BD. Jeśli zachodzi potrzeba usunięcia skutków awarii, to badając dziennik można dowiedzieć się które transakcje zostały wypełnione. Do określenia czy dana transakcja została wypełniona, czy nie, przyjmuje się często tzw. strategię dwufazowego wypełniania, którą definiujemy następująco:

- Transakcja nie może zapisywać danych do BD dopóty, dopóki nie zostanie wypełniona.
- Transakcja nie może być wypełniona dopóty, dopóki nie zapisze w dzienniku wszystkich dokonywanych przez siebie zmian elementarnych.

Z powyższych uwarunkowań wynika, ze pierwsza faza wypełnienia, to zapis danych do dziennika, a faza druga to zapis danych do BD. Jeżeli dodatkowo transakcje przestrzegają protokołu dwufazowego i odblokowanie odbywa się po wypełnieniu (zapis do dziennika i zapis do BD) to żadna transakcja nie może odczytywać z BD wartości zapisanej przez transakcję nie wypełnioną. Gdy w systemie wystąpi awaria, to możliwe jest badanie dziennika i powtórzenie wszystkich wypełnionych transakcji zapisanych w dzienniku, które nie mogły zapisać zmian do BD.

Porównanie metod: Metoda całkowitego blokowania vs. Metoda Większościowa

Zapis.

Przy zapisie nieco lepsza jest metoda większościowa, bo wymaga średnio mniej komunikatów na zapis.

Odczyt.

Jeśli typowe transakcje wymagają równej liczby blokad całkowitych i blokad zapisu, to żadna z tych metod nie ma przewagi, nawet przy policzeniu całkowitej liczby komunikatów lub przy przypisaniu większej wagi komunikatom krótkim niż długim. Faktycznie dla n = 1 obie metody są identyczne. Z drugiej strony, jeśli większość blokad zakłada się dla odczytu, to znacznie przydatniejsza jest metoda całkowitego blokowania wszystkiego a jeśli przeważają blokady całkowite, to lepsza jest metoda większościowa.

Ponadto metoda większościowa spisuje się lepiej w następujących sytuacjach:

- Transakcje często współzawodniczą o blokadę tej samej jednostki.
- Sprawiamy, że transakcje, chcąc zablokować jednostkę, żądają blokad we wszystkich węzłach lub w ich większości.

Przy zastosowaniu metody całkowitego blokowania wszystkiego, każda z dwóch realizujących transakcji rozpoczętych w mniej więcej tym samym czasie potrafi prawdopodobnie zapewnić sobie blokady co najmniej jednej kopii jednostki o którą współzawodniczą. Ta sytuacja powoduje impas: są to sytuacje wykrywalne ale czasochłonne. Pobierają one zarówno czas rzeczywisty, ponieważ każda transakcja czeka dopóty, dopóki impas nie zostanie usunięty, jak i czas systemowy, gdyż często jest niezbędne wykonywanie procedury wykrywania impasu. Przy zastosowaniu metody większościowej jedna transakcja zawsze uzyska zablokowanie drugiej, podczas gdy druga może czekać lub być odrzucona.

ROZPROSZONE B. D.

System rozproszonej bazy danych jest t taki system b. d., w którym dane są rozłożone do różnych konfiguracji sprzętowych i programowych, na ogół rozmieszczonych w różnych geograficznie miejscach organizacji, do której została.
Zasadniczym celem rozproszonej b. d. jest, aby użytkownik widział b. d. jako scentralizowaną. Powinna ona być przeźroczysta (rodzaje przeźroczystości: geograficzna, fragmentaryzacji, replikacji.
Rozłożenie b. d. realizuje się poprzez:

- Fragmentaryzację – utrzymywanie podzbiorów wszystkich danych (fragmentów danych). Fragmentaryzacja może być realizowana poziomo (podzbiór wierszy), lub pionowo (podzbiór kolumn).,
- Replikację – utrzymywanie kopii całości lub części danych przechowywanych w b. d. Jednym z głównych celów replikacji jest zmniejszenie kosztu dostępu do bazy danych.

O rozproszeniu można też mówić w stosunku do funkcji realizowanych przez b. d. Na przykład w systemie typu klient-serwer mamy do czynienia z rozproszeniem funkcji, ale na ogół nie mamy rozproszenia danych. System klient-serwer, gdzie dane nie są rozproszone, nie jest rozproszony. Taki system to scentralizowana b. d..

Typy systemów rozproszonych baz danych:

1) jednorodna b. d. – składa się z wielu serwerów obsługujących wielu klientów, połączenia i odległości pomiędzy elementami systemu realizowane są przez odległe łącza. Zawsze opieramy się na tym samym rodzaju systemu zarządzania b. d.. Takie systemy są realizowane na tego samego rodzaju sprzęcie,
2) niejednorodna b. d. – są to takie bazy, które zrealizowane są na różnych platformach sprzętowych i programowych, np. w 1 węźle VAGS i Oracle, w 2 VAGS i Inges, a w 3 IBM i BD2. Aktualnie są realizowane przez łącza typu Gateway. Problemy z zarządzaniem są dość duże. Połączenia są realizowane na bieżąco.
3) Federacyjny system b. d.- przypominają federację w sensie politycznym, składają się z pewnej liczby autonomicznych baz danych, czasami się je scala do wykonywania jakichś funkcji. Są to funkcje trudne do przewidzenia. Podejście do tych systemów jest takie, by produkty tych systemów były jak najbardziej otwarte.

Przykłady systemów rozproszonych:

1) Systemy bankowe – w oddziałach są systemy obsługująca dany oddział, do wszystkiego ma jednak dostęp centrala. Dane są fragmentami rozmieszczone w oddziałach. Często występuje replikacja.
2) Systemy ewidencji ludności – realizowane w odrębnych miejscach, gdzie istnieją wojewódzkie systemy ewidencji ludności. Większość zadań realizowana jest w stolicach województw, czasami jest jednak realizowane zadanie dla całego kraju, np. wiąże się ono z nadaniem numeru pesel.

Mimo faktu, że rozproszona b. d. wymaga wielu dodatkowych mechanizmów takich jak np. zabezpieczanie, istnieje wiele ich **zalet**:

- Gdy organizacja ma charakter rozproszony, to rozproszony system b. d. jest dla nich bardziej odpowiedni niż system scentralizowany,
- Gdy organizacja jest rozproszona geograficznie to mamy większą kontrolę nad danymi w miejscu przechowywania tych danych,
- Replikacja danych zwiększa niezawodność systemu, nawet gdy jest realizowana w innym oddziale,
- Działanie systemu b. d. może być istotnie poprawione, jeżeli się dokona prawidłowego rozproszenia danych.

SYSTEMY B. D. TYPU KLIENT-SERWER

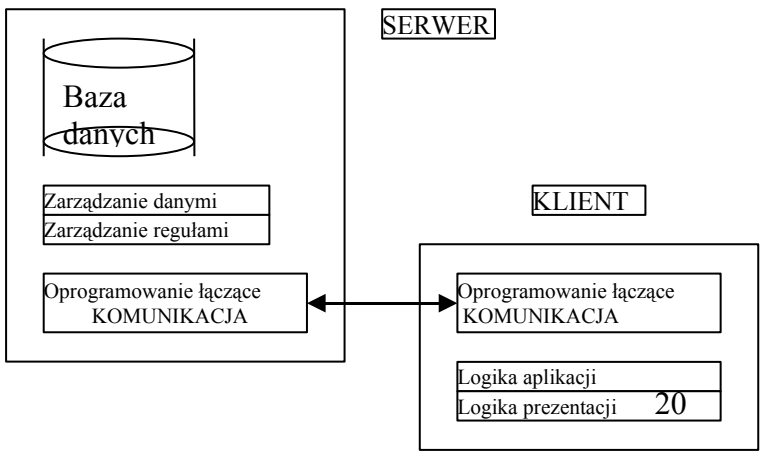
Odnoszą się najczęściej do lokalnych sieci komputerowych. Często mylnie są traktowane jako systemy rozproszone. Przynajmniej jeden serwer jest przeznaczony do spełniania funkcji b. d. dla pozostałych komputerów, które działają jako klienci.
Baza danych przechowywana jest na serwerze. Tu znajdują się także funkcje obsługi b. d.. Na komputerach klientów znajdują się interfejsy użytkowników oraz narzędzia do tworzenia aplikacji bazodanowej.
Serwery b. d. spełniają dwie funkcje:

A) serwer plików – zapytania są wyrażane przez klienta w jego aplikacji, są one interpretowane. Określa się dla nich na jakich plikach będą realizowane operacje do tego zapytania. Pliki te są udostępniane przez serwer plików.
B) Serwer SQL – lepsze z punktu widzenia użytkowników. Zapytania zgodne z SQL wydane z komputera klienta jest wysłane bezpośrednio do serwera SQL. Serwer je wykonuje i w odpowiedzi wysyła do aplikacji tylko wynik operacji zapytania. System oparty na serwerze SQL pozwala wytwarzać systemy otwarte.

W architekturze klient-serwer w oprogramowaniu współdziałają ze sobą dwa procesy: podrzędny i nadrzędny. Proces klienta zaczyna współdziałanie poprzez wysłanie zapotrzebowania (w SQL lub do pliku). Proces serwera na nie reaguje. Serwery SQL lub plików mogą być zaopatrzone w funkcje serwerów pocztowych lub druku.

Każda konwencjonalna aplikacja bazodanowa składa się z 4 części:

1) zarządzanie danymi – funkcje – zarządzanie transakcją, współbieżnością, bezpieczeństwem, przechowywaniem danych.
2) Zarządznie regułami – polega na realizowaniu funkcji zapewniających spójność b. d. (wewnętrzną i zewnętrzną).
3) Logika aplikacji – realizuje funkcje przekształcające dane i zgłaszające zapotrzebowanie na usługi serwera lub zapotrzebowanie na inne oprogramowanie znajdujące się w komputerze.
4) Zarządzanie i logika prezentacji – realizuje funkcje związane z przyjmowaniem i zapotrzebowaniem użytkownika oraz prezentuje użytkownikowi wyszukane dane. Przekształca dane wejściowe do postaci wymaganej przez serwer i odwrotnie – komunikaty serwera do postaci wymaganej przez użytkownika.



W części serwera i klienta musi istnieć oprogramowanie łączące. Jest ono przeźroczystym łącznikiem aplikacji klienta z danymi serwera. Oprogramowanie łączące może być standardowym pakietem lub też produktem danej firmy tworzącej dany SZBD np. oprogramowanie łączące ODBC – standardowe, zwane otwartym łączem z bazami danych. Standard opracowany przez Microsoft na zasadzie interfejsu. ODBC sprowadza się do definiowalnej biblioteki wywołań dostępu do b. d. Umożliwia programiście otworzenie połączenia aplikacji ze środowiskiem b. d. Sprowadzamy dane z b. d. do aplikacji klienta. Program sterownika ODBC odwołuje się do sterowników poszczególnych systemów zarządzania b. d. (DBMS), a te do różnych systemów ZBD.
W przypadku niestandardowym, np. Oracle, modół łączący nosi nazwę SQL*NET (ODBC jako otwarty może być mniej korzystny przy konkretnym systemie, np. Oracle, jednak w przypadku baz niejednorodnych lepszy będzie ODBC).

Przy **projektowaniu** systemu **Klient-serwer** musimy wziąć pod uwagę:

- Na komputerze klienta umieszczać jak najwięcej logiki prezentacji i aplikacji, daje to odciążenie serwera i przyśpieszenie współpracy z użytkownikiem,
- Na komputerze klienta należy umieścić sprawdzanie niektórych reguł poprawności danych wprowadzanych przez użytkownika (odciąża to serwer),
- System musi być tak zaprojektowany, by minimalizować ruch w sieci przez minimalizację liczby i rozmiaru zgłoszeń do serwera,
- Umieszczać sprawdzanie podstawowych reguł poprawności danych, gdy te reguły dotyczące całej organizacji bazodanowej na serwerze, jeżeli są związane z fragmentem, to umieszczać je w logice prezentacji.

Każdy **SZBD** (nie tylko rozproszony) **składa** się z trzech części:

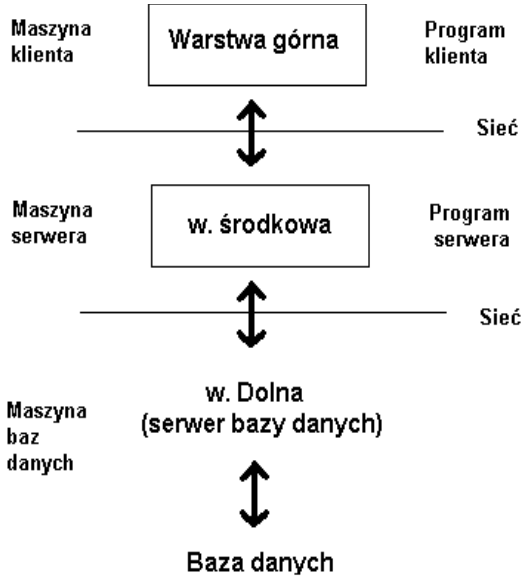
1) Jądro - samo jądro z b. d. musi utrzymywać określony model danych. W modelu relacyjnym dane widziane są przez jądro jako tabele.
2) Interfejs DBMS – jest dostosowany do narzędzi i do jądra, jest połączeniem między nimi – przekształca polecenie narzędziowe na polecenie jądra. Najczęściej mają możliwość wytwarzania poprzez kompilację aplikacji w języku jądra.
3) Zestaw narzędzi: języki 4 generacji, języki wytwarzania aplikacji bazodanowej, graficzny interfejs użytkownika, interfejsy języka naturalnego, interfejsy programowe do programowania aplikacji.

Funkcje podstawowe jądra DBMS:

a) organizacja plików – dotyczy sposobu przechowywania danych na urządzeniach zewnętrznych. Sam system przyjmuje jakiś określony model danych. Jądro przekształca ten model w fizyczną reprezentację danych na urządzeniu zewnętrznym. Dla modelu relacyjnego są to najczęściej pliki sekwencyjne o nieuporządkowanej strukturze. Nowe dane dopisywane są na końcu pliku. Bywają także pliki uporządkowane funkcją maskującą. Inne z kolei wykorzystują klastry.
b) Mechanizmy dostępu – sposób zapisu wymaga odpowiednich mechanizmów dostępu do plików. Dla relacyjnych b. d. wyróżnia się techniki oparte na prostym indeksowaniu lub na B-drzewach.
c) Zarządzanie transakcjami – kontrola współbieżności i spójności b. d.
Zarządzanie współbieżnością – ogranicza się do systemu scentralizowanego, opartego na modelu relacyjnym. Podstawowym zadaniem SZBD jest udostępnianie bazy wielu użytkownikom – jest to jedno z najbardziej skomplikowanych zadań jądra. Musimy zapobiegać straconej modyfikacji i innym problemom.
d) zarządzanie słownikami – słownik (katalog systemowy) zawiera pewne informacje ogólne o systemie bazodanowym. Zakładany jest najczęściej w momencie zakładania przez administratora b. d. lub później. Zawiera on grupy użytkowników, ich uprawnienia i hasła dostępu. Mogą być tam opisy relacji (nazwy relacji, typy kolumn), definicje i deklaracje kluczy, definicje perspektyw (podtabel), definicje uprawnień, informacje na temat indeksów i informacji o pliku. Jądro może ten słownik organizować na różny sposób (jeden dla całości przyklejany jako nagłówek do każdego pliku z danymi).
e) Zarządzanie zapytaniami,
f) Sporządzanie kopii i odtwarzanie bazy danych po awarii.

Aplikacja baz danych składa się z trzech warstw:

- Dolna – dostęp do baz danych,
- Środkowa – reguły w zakresie baz danych,
- Górna – instrukcje użytkownika.
-



Warstwa dolna dostępu do baz danych – zajmuje się kontrolą dostępu do bazy danych, składa się z programu, który realizuje bezpośredni dostęp do bazy danych (sterowniki i programy sieciowe). Warstwa ta odbiera zapytania (instrukcje) SQL z warstwy środkowej i je wykonuje. Po otrzymaniu wyszukaniu informacji z bazy danych warstwa ta ustala ewentualny komunikat o błędzie (wykonuje na własnych serwerach SQL) i przekazuje go do warstwy środkowej.

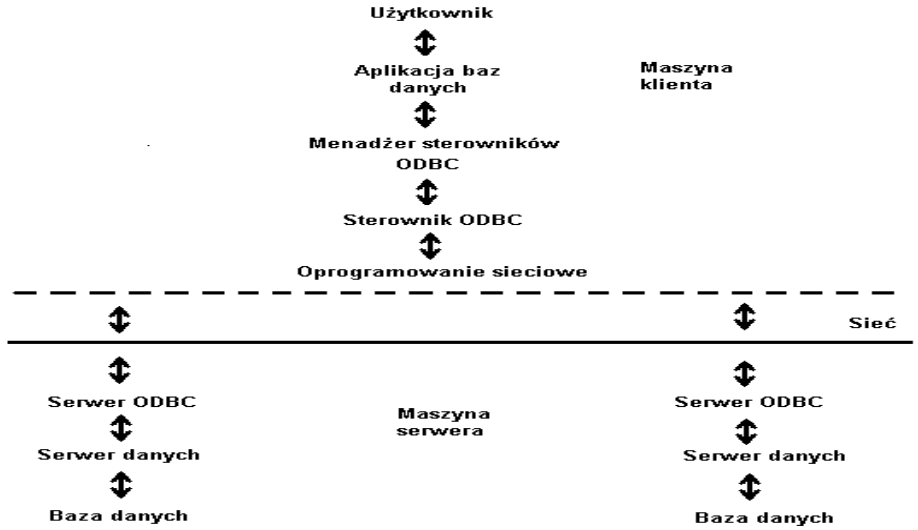
Warstwa środkowa – (reguł dziedziny danych) tworzy kod odpowiadający za implementacje reguł zgodności formatu danych i reguł spójności danych. Reguły spójności przetwarzane są na kod, wymuszają na bazie danych modyfikacje. Warstwa środkowa pobiera z warstwy górnej zadanie, instrukcję do wykonania i porównuje te instrukcje z zaprogramowanymi regułami, jeśli jest zgodność z ustalonymi normami to przesyła do warstwy górnej celem wykonania zapytań SQL i jeżeli jest zgodna to przesyła do warstwy dolnej celem wykonania.

Warstwa górna – wyświetla dane z bazy i instrukcje dla użytkownika. Odbiera cel zadania użytkownika, wyświetla komunikaty o błędzie. Warstwę górną można inaczej nazywać kodem prezentacji – sposób udziału interfejsu na ekrany, przepływ i ułożenie pól, sposób wyświetlania komunikatów, rodzaje symboli graficznych, używanie kolorów, ikon, grafiki.

Użytkownik + Warstwa Górna → Środkowa → Dolna + Baza danych

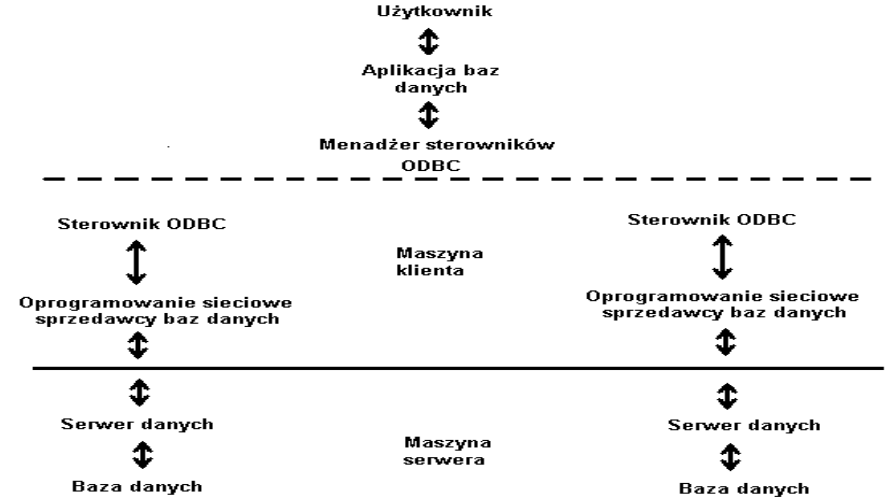
Warstwa Dolna i Górna (bez kontekstowe) niezależnie od tego co jest w bazie przechowywane, warstwa środkowa jako jedyna zna zawartość bezy danych (logiczne części rozdzielania). Warstwa Górna i Środkowa (scalone) Dolna składa się z 1, 2 lub 3 programów (SQL i Builder'ów)

Jeśli warstwy są oddzielnymi programami można podzielić je na:
- Warstwa Dolna – serwer
- Warstwa Dolna – klient (może być połączona z w. środkową)



W dostępie do bazy danych używa się sterowników (standardowych):
- ODBC
- JDBC firmy SUN
- DBE (data base engine)

Różne sterowniki ODBC stosowane są dla różnych baz danych (dwie architektury zorientowane na klienta i na serwer)
- (ODBC) Na Klienta **– najwcześniej stosowany**



Aplikacja może pochodzić od różnych produktów ale działających razem z użyciem sterowników ODBC
- **(ODBC) Na Serwer –** pojawiła się niedawno Trzy główne różnice rozróżniające architektury:
- ma serwer, maszyna klienta nie potrzebuje do komunikacji z serwerem danych specjalnego oprogramowania sieciowego (wystarczy podstawowe opr. protokołu TCP/IP)
- maszyna klienta potrzebuje uogólnionego sterownika ODBC dedukowanych w bazie danych

- maszyna serwera wymaga serwera ODBC (ster. ODBC) dla każdej obsługiwanej bazy danych (inaczej skonstruowana)

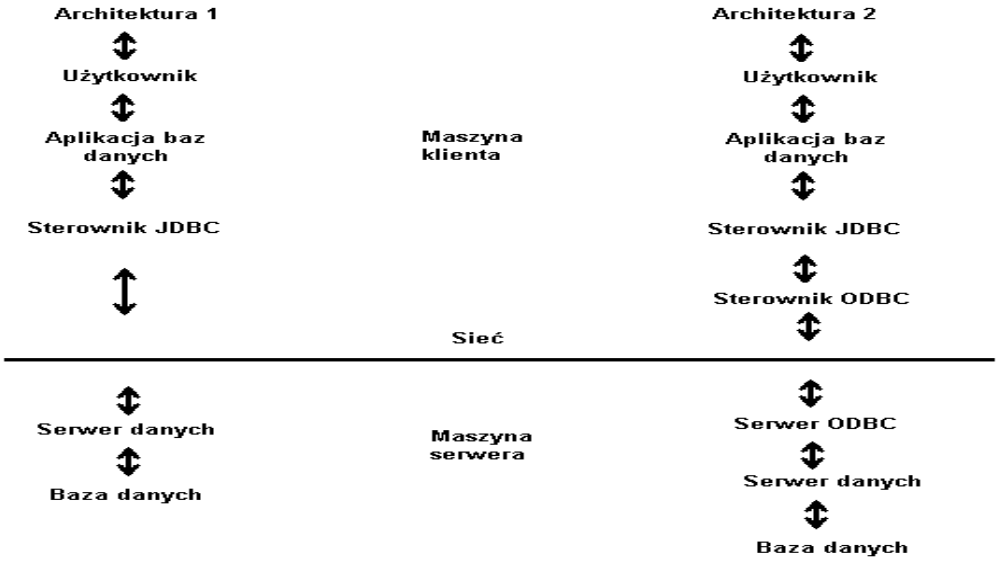
Chodzi głównie o koszty zakupu danej architektury, na klienta łatwiejsza w obsłudze i implementacji.

Sterownik JDBC – interfejsem jest zbiór klas zdefiniowanych w języku Java (jawny odpowiednik ODBC) interfejs programowania aplikacji działa wywołując (przekazując wywołanie) ster. ODBC. Wykorzystuje się dwie architektury:

- Pierwsza polega na komunikacji bezpośredniej sterownika JDBC z serwerem danych i tłumaczy instrukcje JDBC na wywołania CLI rozpoznawane przez daną baze.
  CLI – zbiór funkcji zaimplementowanych w języku programowania (np. C++) funkcje te przekazują polecenia SQL bezpośrednio do obsługi bazy danych przez serwer bazy danych (Java na C, a C na SQL a ten na bazę danych)

- Sterownik JDBC wymienia dane ze sterownika ODBC na serwerze i tłumaczy instrukcje JDBC na wywołania ODBC, a dopiero zadanie te realizuje ODBC poprzez kontakt z odpowiednim serwerem danych (bardziej ogólne i uniwersalne)

Architektura JDBC – serwer danych i architektura JDBC - ODBC



OCHRONA BAZ DANYCH SCENTRALIZOWANYCH I ROZPROSZONYCH PRZED AWARIĄ

Zabezpieczenie baz danych przed awarią

Jest to podstawowa funkcja każdego SZBD. Musi on mieć mechanizm dziennika bazy danych, gdzie odnotowuje się informacje na temat transakcji. Każda transakcja jest zapisywana w postaci identyfikatora transakcji oraz starej i nowej wartości elementarnej jednostki. Są różne rozwiązania, najczęściej tylko zapisuje się wartości zmieniane, ale są też systemy zapisujące wszystkie wartości. W pierwszym przypadku jest więc to tylko zabezpieczenie przed awarią, w drugim może także odnosić się do parametrów eksploatacyjnych.
Różne systemy przyjmują różne jednostki elementarne. Niektóre dzienniki zapisują całe stare i nowe wartości, inne natomiast zapisują tylko starą i nową zawartość pola. Gdy się coś dopisuje nowego, to zapisuje się starą wartość i to co dodane.

Mówi się że transakcja się wypełniła, gdy wykonała się od początku do końca.

W SZBD mówi się że transakcja się wypełniła gdy wszystkie zmiany są zapisane i w dzienniku i w bazie danych oraz we wszystkich obszarach obszarach roboczych.
Przyjmuje się dwufazową strategię wypełnienia:
1) transakcja nie może zapisać do bazy danych dopóty, dopóki się nie wypełniła w dzienniku baz danych;
2) transakcja nie może być wypełniona dopóty, dopóki nie zapisze w dzienniku wszystkich dokonywanych przez siebie zmian na jednostce elementarnej.

Najczęściej wykorzystuje się mechanizmy blokowania. Między zablokowaniem i odblokowaniem realizowane są również inne mechanizmy.
Zabezpieczenie przed awarią polega na tym, że gdy wystąpi awarią, to na podstawie dziennika odtworzone zostają operacje, które się wypełniły (od pewnego momentu). Na podstawie dziennika anuluje się transakcje niewypełnione i powtarza się wykonanie tych transakcji, które się wypełniły. Ważna jest wielkość dziennika, metody jego przeszukiwania, itp...
Blokowanie nie jest jedyna metodą realizowania współbieżności. Inna jest tzw. **metoda optymistyczna (metoda znaczników czasowych, metoda walidacji).** Elementy bazy danych nie są tu blokowane (oszczędza to czas realizacji). Ta metoda nie traci czasu na blokowanie, lecz realizuje transakcję lub umieszcza ją w kolejce Ta metoda wymaga pewnego czasu na realizację transakcji. Jest efektywna gdy nie ma konfliktów między transakcjami. Zasada tej metody polega na nadawaniu każdej transakcji znacznika czasowego. Są one liczbami generowanymi w każdym takcie zegara. Istnieje szeregowalność pojawiających się transakcji. Każda transakcja ma inny znacznik czasowy, niezależnie od tego co chce robić. Znaczniki powinny być jak najmniejszymi liczbami. Należy często kopiować bazę, aby znaczniki nie były zbyt duże. Najczęściej są one 16-bitowe.

Dla każdej jednostki bazy danych zapamiętywane są:
- czas odczytu – najmniejszy znacznik czasowy należący do transakcji odczytującej tą jednostkę;

- czas zapisu – największy znacznik czasowy, który zapisuje jednostkę.

Transakcje w systemie pojawiają się szeregowo. Znacznik jest porównywany z czasem zapisu i odczytu, aby sprawdzić, czy transakcję można realizować czy też umieścić ją w kolejce.
Podczas realizacji transakcji nie jest możliwe aby transakcja mogła odczytywać jednostki jeżeli zostały one zapisane już po wykonaniu tej transakcji. Czyli jeżeli transakcja ma znacznik czasowy t_1 i chce czytać, to nie może tego zrobić, jeżeli da tej jednostki czas zapisu t_2 jest większy od t_1 ($t_2 > t_1$). Transakcja jest odrzucana.
Jeżeli dwie transakcje mają różne znaczniki czasowe i chcą zapisywać, to nie są odrzucane ale sprawdza się czy nie pojawił się jakiś odczyt pomiędzy nimi.
Załóżmy że transakcja zgłasza się znacznikiem czasowym t, gdy będzie chciała wykonać operację na jednostce X to dla tej jednostki będą podane dwa czasy: t_r i t_W.
Zasady:
1) operację wykonujemy jeżeli X = READ i $t \geq t_W$ lub jeżeli X =WRITE i znaczniki czasowe $t \geq t_r$ i $t \geq t_W$ Jeżeli $t>t_r$, to czas odczytu jednostki X przybiera wartość t. Jeżeli $t>t_W$ to czas zapisu jednostki X przybiera wartość t;
2) Jeżeli X=WRITE i zachodzi zależność $t_r \leq t \leq t_W$ to transakcja nie jest realizowana (nie robimy ale i nie odrzucamy);
3) Jeżeli X=READ i $t<t_W$ lub X=WRITE i $t<t_r$ to transakcję odrzucamy

OCHRONA BAZ DANYCH PRZED NIEWŁAŚCIWYM DOSTĘPEM I ZASADY ZACHOWANIA INTEGRALNOŚCI. INTEGRALNOŚĆ REFERENCYJNA. WIĘZY WARTOŚCI ATRYBUTÓW I WIĘZY GLOBALNE

W każdym systemie DBMS są środki, które zapobiegają zarówno zapamiętywaniu w b.d. niepoprawnych danych jak i odczytywaniu danych przez osoby nieupoważnione, niepowołane.

Istnieją dwa źródła niepoprawnych danych:
- pomyłki (np. przy wprowadzaniu danych)
- złe użycie b.d.,

Mówiąc o ochronie b.d. warto wyróżnić dwa zagadnienia:
1) Utrzymywanie integralności,
2) Ochrona (czyli kontrola danych) dostępu

Ad.1 Zagadnienie to łączy się z zapobieganiem nieumyślnym błędom
Ad.2 Zależy nam głównie na tym, aby pewni użytkownicy mieli dostęp tylko do określonego podzbioru b.d. i tylko ten ewentualny podzbiór mogli aktualizować.

Bardzo ciekawym problemem jest ochrona statystycznych baz danych. W takich bazach problem polega nie na tym by ograniczyć użytkownikowi możliwość dostępu do jakiejkolwiek szczegółowej części b.d., lecz na tym aby mu przeszkodzić w dedukowaniu z informacji statystycznych, takich jak średnie zarobki itp., danych szczegółowych, np. dochodu poszczególnych osób.

Integralność
Zakłada się by systemy DBMS nakładały na bazy danych dwa rodzaje więzów, więzy strukturalne oraz zależności funkcjonalne. Nie wszystkie jednak zależności funkcjonalne mogą być podtrzymywane przez systemu DBMS. Zezwalają one np. na deklarowanie, że zbiór pól lub atrybutów tworzy klucz do typu rekordu lub relacji. Inny rodzaj więzi dotyczy wartości bieżących zapamiętywanych w b.d. Więzi te noszą miano więzów integralności. Podstawową ideą dotyczącą więzów integralnościowych jest to, że więzy te mogą być wyrażone w języku manipulacji danymi.

Przykład
Więzy integralności określa się tak, że nikomu nie wolno mieć ujemnego salda w zakresie np. udziałów członkowskich. Można je wyrazić w postaci wymogu, że:

$$\Pi_{NAZWISKO}(CZŁONKOWIE)=\Pi_{NAZWISKO}(\vartheta_{SALDO\geq 0}(CZŁONKOWIE)$$

Drugą częścią deklaracji więzów integralnościowych jest opis, kiedy warunki wcześniej określone mają być sprawdzane.

Przykład
Więzy integralności dotyczą w tym przypadku wartości bieżących zapamiętywanych w b.d. Niech pola EGZAMIN, PRACA, LAB oznaczają procentowy udział czasu przeznaczonego na egzamin, pracę domową i laboratorium, to dla zachowania integralności będziemy oczekiwać, że w każdym rekordzie suma wartości będzie wynosić 100%. Opis tego faktu stanowi warunki integralności.

Ogólnie deklaracja więzów integralnościowych wymaga podania tych więzów w postaci pewnego opisu, kiedy i jak te warunki integralności mają być sprawdzone. W DBMS-ach zezwala się aby takie „więzy" działały jako „przerwania" wysokiego poziomu, np. w językach manipulacji danymi zezwala się umieszczać warunki postaci

ON <lista rozkazów> CALL <procedura>

w deklaracjach typów rekordów lub w deklaracjach typów kolekcji. Taka <lista rozkazów> może zawierać rozkazy typu FIND, INSERT, REMOVE zaś <procedura> jest dowolnym programem napisanym w języku manipulacji danymi, np. dla typu kolekcji S zadeklarowano

ON INSERT CALL P1

to procedura P1 powinna sprawdzać, czy pewne pola bieżącego rekordu zadania, którym jest wstawiany rekord podrzędny, nie występują już w zbiorze wystąpień. W ten sposób możemy zapewnić integralność b.d. polegającą na tym, że wraz z kluczem dla rekordu nadrzędnego wyznaczają funkcyjnie resztę pól w typie rekordu podrzędnego. Warunek ON w zapisie więzów integralności jest uaktywniony zawsze wtedy, kiedy jest wykonany rozkaz z tej listy i bieżący rekord zadania jest odpowiedniego typu.

Czasami w systemach DBMS nakłada się więzy integralności na aktualizację, tak by istniały związki między starymi i nowymi-uaktualnionymi wartościami pewnych atrybutów. Tu oprócz więzów dla samej krotki włączamy również linię reprezentującą starą krotkę. Np. przy pobraniu towaru z magazynu nie można zmniejszyć ilości towaru pozostającego na stanie magazynu, więcej niż wynosi ilość towaru w magazynie przed jej pobraniem.

Najczęściej więzy integralności nie są sprawdzane po wykonaniu pojedynczych rozkazów, lecz tylko wtedy kiedy zostaną wykonane całe ciągi rozkazów. To pozostawia pewną swobodę co do kolejności specyfikowania rozkazów dopóty, dopóki są wprowadzane grupowo. Np. w systemie obsługi magazynu nakładamy więzy w taki sposób, że na towar, którego nikt nie dostarcza nie można złożyć zamówienia. Jeśli jednak jako „zawartość ekranu" wprowadzimy kilka zamówień oraz informacje o dostawcach i po tym sprawdzimy więzy integralności, to więzów tych nie naruszymy. Jeśli jednak system wprowadzi zamówienie i sprawdzi więzy integralności przed wprowadzeniem informacji o dostawcach to naruszenie integralności nastąpi.

OCHRONA

Zagadnienie ochrony danych przed nieuprawnionym dostępem ma wiele różnych aspektów. Należy:

- chronić b.d. przed niepożądaną modyfikacją lub zniszczeniem danych
- chronić przed nieuprawnionym odczytem danych

Ochrona danych nie jest tylko domeną systemów DBMS. Z podobnymi problemami spotykamy się też na poziomie systemu operacyjnego. Dlatego tylko o niektórych powszechnych metodach ochrony tylko wspomnimy. Zajmiemy się pewnymi szczególnymi problemami związanymi z DBMS-ami. Jednym z problemów ochrony jest metoda identyfikacji użytkownika. Ogólnie mówiąc, różnym użytkownikom nadaje się różne uprawnienia do korzystania z różnych b.d. lub różnych części b.d. np. relacji, atrybutów, określonych atrybutów w określonych relacjach. Te uprawnienia mogą dotyczyć: odczytu, ustawiania, usuwania, aktualizowania. Najbardziej rozpowszechnioną metodą identyfikacji użytkowników jest wprowadzenie haseł znanych tylko systemowi i użytkownikowi. System chroni te hasła co najmniej tak dobrze jak dane, chociaż nie ma żadnej gwarancji ich pełnego bezpieczeństwa.

Innym ważnym problemem jest ochrona fizyczna. Chcąc całkowicie niezawodnie bronić b.d. trzeba brać pod uwagę różne możliwości fizycznego jej naruszenia – od wymuszenia ujawnienia haseł, aż do kradzieży czy uszkodzenia nośników pamięci. Przed skutkami naruszenia haseł lub zawartości b.d mogą uchronić również różne sposoby szyfrowania danych. Te sposoby wchodzą w zakres tzw. Kryptologii, kryptografii i będą tam dokładniej omawiane chociaż trzeba pamiętać, że szyfrowanie wymaga również tzw. Deszyfracji co powoduje wydłużenie czasu dostępu do danych. W zakresie ochrony b.d. znajduje się również problem utrzymywania i przekazywania uprawnień. Systemu DBMS muszą utrzymywać dla każdego użytkownika listę uprawnień do korzystania z każdego chronionego fragmentu b.d. Jednym z takich uprawnień może być uprawnienie do nadawania uprawnień innym.

Rozważmy teraz dwa mechanizmy ochrony b.d. zaprojektowane przede wszystkim specjalnie dla systemów DBMS, są nimi:

- perspektywy
- użycie języka zapytań do definiowania uprawnień

Istnieją perspektywy dwóch rodzajów:

- read-only – perspektywa niezmienna
- perspektywa zezwalająca na odczytywanie oraz zapisywanie obiektów będących ich częścią. Aktualizacje perspektyw są dopuszczalne i są odzwierciedlane w schemacie przejściowym b.d.

Niezmienna perspektywa najczęściej używana jest wtedy, gdy właściciel b.d. (administrator) chce ogólnie udostępnić prawo doczytywania danych zachowując dla siebie lub ograniczonego kręgu prawo do aktualizowania tych danych. Perspektywy niezmienne nie mogą być aktualizowane.

Poważnym problemem w perspektywach modyfikowalnych jest to, że aktualizowanie perspektyw powoduje również skutki uboczne w części b.d. nie należącej do perspektywy. W systemach b.d. nie jest jasne, co oznacza usunięcie niektórych składowych np.: krotek, jeśli w relacji występują inne atrybuty nie należące do perspektywy ; użytkownik widzący tylko perspektywę nie powinien mieć możliwości ich usunięcia. Z tego powodu system musi odrzucać aktualizację wielu perspektyw. Podobnie jest w systemach hierarchicznych b.d. gdzie można mieć perspektywę na konkretny rekord lecz bez jego typu „potomków". Usuwając ten rekord musimy mieć możliwość usunięcia „potomków" dla zachowania choćby integralności b.d. Taka akcja może być nielegalna. Podobna sytuacja może mieć miejsce w modelu sieciowym gdy chce się usunąć rekord nadrzędny nie wiedząc nic o jego rekordach podrzędnych, ponieważ nie należą do tej perspektywy.

Drugim ważnym mechanizmem ochrony b.d. jest wykorzystanie **języka manipulacji danymi do definiowania uprawnień dostępu** każdego użytkownika do b.d. Tego języka można używać do definiowania uprawnień na podobnych zasadach jak używanie go do definiowania więzów integralności. Tzn. można spowodować by do każdego zapytania zadanego przez użytkownika i dotyczącego wyznaczenia relacji automatycznie była dołączana **selekcja** i **rzut.** W systemach DBMS można stosować również tzw. blokady jawności dla chronionego obiektu w postaci procedur wyrażonych w języku manipulacji danymi.

Perspektywę można uważać za tabelą utworzoną z innej tabeli (relację z innej relacji). Perspektywa jako taka nie istnieje fizycznie w b.d. Jest to tablica wirtualna, której zawartość zmienia się, kiedy tylko określające ją tabele ulegają zmianie. Perspektywa użytkownika była uważana za trzeci poziom bazy danych po poziomie fizycznym i logicznym. W wielu przypadkach jest ona po prostu zbiorem kolumn pochodzących z tabel utworzonych na poziomie logicznym. Za pomocą perspektywy użytkownik może widzieć bazę danych w specyficzny dla siebie sposób. Perspektywa to instrukcja języka manipulacji danymi (języka zapytań); aby np.: utworzyć w ORACLE perspektywę należy napisać:

CREATE VIEW <nazwa perspektywy> AS <zapytanie>

Gdzie:

<nazwa perspektywy> - to nazwa nowej perspektywy
<zapytanie> - instrukcja typu SELECT, która ma być wykonywana przy każdym odwołaniu się do perspektywy.

Każda zmiana dokonana w określających perspektywę tabelach będzie także uwzględniona w perspektywie. Perspektywa może mieć jednak własny system ochrony. Za pomocą doboru tabel i perspektyw programista b.d. może zapewnić każdemu użytkownikowi odpowiedni dla niego obraz bazy danych.
Do usuwania perspektyw mamy w ORACLE instrukcję

DROP VIEW <nazwa perspektywy

Ponieważ perspektywy są używane tak jak tabele, naturalne byłoby dodawać, usuwać i modyfikować wiersze z perspektyw. Stanowi to ważny problem. Dlatego w systemach DBMS są wprowadzane dodatkowe ograniczenia na możliwość wstawiania danych przez perspektywy, a w niektórych z nich w ogóle jest to zabronione w celu uproszczenia procedur sprawdzających. Modyfikacja wiersza przez perspektywę jest równoważna usunięciu go, a następnie dodaniu nowego wiersza, co prowadzi do podobnych problemów, jak te związane ze wstawieniami (wstawienie danych do perspektywy niw wstawia danych w kolumnach nie należących do perspektywy a tylko do tablicy nad którą utworzono perspektywę).
Dodatkowo obok perspektyw wykorzystuje się komendy ochrony danych. NP. w ORACLE gdy użytkownicy mają założone poprawne konta jako podstawowy sposób wprowadzania ochrony służy instrukcja GRANT. Instrukcja ta umożliwia lub zabrania innym użytkownikom dostępu do tabel lub wierszy w obrębie tabel. Chodzi tu o usuwanie, wstawianie, wybór i modyfikację. Aby np.: użytkownik, który jest rozpoznawany w systemie jako „Piotr", miał prawo wstawiać, wybierać lub modyfikować, ale nie usuwać całe wiersze z tabeli „klient" należy wywołać komendę GRANT w następującej postaci:

GRANT INSERT,SELECT,UPDATE ON klient TO Piotr

Gdy taka instrukcja zostanie wykonana to zostaną określone odpowiednie prawa dostępu. Jeżeli użytkownik typu „Piotr" wstawi jakiś wiersz, a chce go teraz usunąć może to zrobić za pomocą instrukcji

ROLLBACK

Wycofanie transakcji. Wówczas wiersz tan nie będzie przechowywany w bazie danych.
Do zapisu uprawnień dostępu stosuje się najczęściej tzw, tablice uprawnień. W tych tablicach można zapisywać uprawnienia do relacji nadawane pewnej osobie lub wszystkie uprawnienia związane z relacją. Pierwszy sposób może być zrealizowany przy pomocy specjalnej tablicy uprawnień zapisywanej w postaci zbioru dyskowego odpowiedni

HURTOWNIE DANYCH – PODSTAWOWE POJĘCIA I METODY

Problem magazynu (hurtowni) danych w firmie występuje kilka baz, różnie zarządzanych, tworzymy aplikację zarządzającą typami baz danych. Są to dane heterogeniczne. Sposób zarządzania jest taktyczny, strategiczny. Dostęp do takich danych określa się na 6 sposobów

1. Konwersja danych ze starego środowiska do nowego. Nowa aplikacja działa na przekonwertowanych danych. Większość to umożliwia
2. Pomosty między systemami bazodanowymi – gateway'e (3 rodzaje baz – 3 pomosty)
3. System otwartej łączności ODBC (Open Data BaseConnectivity - Microsoft) – udostępnia bibliotekę funkcji. Aplikacja odwoływuje się do bazy przez interfejs ODBC. On uruchamia managera sterowników, który wywołuje konkretny sterownik, który dopiero działa na bazie. Schemat:
   - Aplikacja wywołuje zapytanie na dane SQL
   - Menedżer sterowników w imieniu aplik. ładuje odpowiedni sterow. do DBMS, pod którym są dane
   - Sterownik BD odpowiednio realizuje modyfikując je
   - przesył danych przez sterow. menedżer i aplik. - użytkownik. Integracja danych na poz. aplikacji
4. Mechanizm SQL Links – podobne do ODBC – lecz produkt Borlanda, obsługuje tylko SQL. Głównym elementem jest dynamiczna biblioteka wykorzystywana do komunikacji z DBMS
5. Specyfikacja OLE DB wraz ze zbiorem dostępu do danych. Mechanizm ten łączy i wstawia obiekty. Utworzony w celu przenoszenia informacji między aplikacjami Windows. Obsługuje różne środowiska i formaty. Ma mechanizm wyszukiwania czy identyfikowania danych. Interfejsy wyszukiwania możemy instalować w różnych źródłach. OLE DB wychodzi poza tradycyjny sposób widzenia BD i rozszerza jego możliwości. Umożliwia dzielenie relacji na fragmenty, udostępnia zdarzenia potrzebne do komunikacji między tymi elementami. Łączy dane z procesem wyszukiwania w jeden bezpośredni obiekt, który może być później używany. Funkcje zgodne z OLE mają dostęp do sterowników ODBC. Mechanizm OLE może być używany jako bezpośr. lub pośr.w środowisku Internetu i do aplikacji, które niekonieczne korzystają z SQL'a
6. Zfederowane systemy bazodanowe – system – nakładka na istniejące systemy bazodanowe – składowe tej federacji. Oprogramowanie zarządza komponentami. W sposób lokalny komunikują się z innymi komponentami.za pośrednictwem tej nakładki. Zaletą jest przechowywanie jednego (globalnego) schematu danych utworzonego z schematów lokalnych. Nakładka zarządza współbieżnością lokalnych BD (między nimi, one wewnętrznie robią to same)

BD – kolekcja danych, zorganizowana kolekcja danych. Baza danych występuje w następujących modelach:
- model rekordowy, itp.
- model plikowy

Zazwyczaj w bazie danych obowiązuje jeden model danych. Istnieją także heterogeniczne bazy ze zróżnicowanymi modelami danych.

SBD – działa w otoczeniu systemu operacyjnego oraz systemu sieciowego. Przy pomocy dyrektyw rozporządza danymi z poziomu:
- query (zapytanie) – np. z możliwością napisania programu aplikacyjnego
- aplikacji na systemie zarządzania bazą danych. Sposoby realizacji:

a) biblioteka funkcji
b) serwer bazy danych – dostęp do fizycznej bazy danych

musi być wyposażony w środki do wyszukiwania, aktualizowania, dołączania i usuwania informacji.
System bazy danych = system zarządzania bazą danych + baza danych

DBMS – system zarządzania bazą danych – system ten umożliwia przechowywanie dużej ilości danych, przy czym użytkownik nie wie, gdzie i jak informacje te są przechowywane. Posługuje się danymi (informacjami) w ich zewnętrznej, użytkowej postaci bez konieczności programowania. Czas reakcji systemu z punktu użytkownika musi być jak najkrótszy (w tym wyszukiwanie informacji). DBMS musi rozwiązywać następujące problemy:
- możliwość ochrony danych przed niewłaściwym dostępem
- współbieżna realizacja operacji bazy danych – współdziałanie wielu użytkowników
- redundancja danych (minimalizacja redundancji)
- ochrona przed awarią
- restart systemu po awarii
- sprawdza integralność bazy danych,
- chroni przed niespójnym stanem,
- zapewnienie spójności po awariach sprzętowych i programowych

Schemat baz danych – jest to forma użytkowa bądź zapisu danych. Może występować w dwóch rodzajach: schemat logiczny (struktura logiczna) i schemat fizyczny (struktura fizyczna).

Schemat logiczny baz danych – forma danych w użytkowej, zewnętrznej postaci. Jest niezależny od schematu fizycznego.

Schemat fizyczny baz danych – sposób umieszczenia danych na fizycznych urządzeniach służących do pamiętania. Jest niezależny od schematu logicznego.

Model danych – sposób reprezentacji danych (rekordów), a także ich wzajemne powiązanie.

Model logiczny danych – opis kolekcji danych przechowywanych w BD niezależnie od fizycznej realizacji, np.: rekord, tabela, tablica rekordów, obiekt z definicją metod obsługi obiektu, lista, drzewo itp.. Model konceptualny (koncepcyjny) jest to inaczej obraz danych postrzeganych przez użytkownika (sposób wykorzystania, interpretacji danych).

Model fizyczny danych – sposób fizycznej realizacji modelu logicznego

Podstawowe typy danych – wyróżniamy następujące typy danych:
- liczba (INT)
- znaki (CHAR)
- logiczny (BOOL)
- data (DATE)
- czas (TIME)
- memo

NORMALIZACJA

Pierwsza postać normalna (1NF)

Jest to najsłabsza postać jeśli chodzi o wymogi. W pierwszej postaci normalnej wymaga się, by dziedzinami atrybutów relacji – tabeli były tzw. Wartości elementarne – nierozkładalne.
Przykład:

nazwisko studenta	języki obce, które zna student

W kolumnie drugiej tej relacji umieszczono szereg informacji, które są elementarnymi jako pojedyncze ale dostęp do tych pojedynczych informacji wymaga operacji rozbioru drugiego pola. Jeśli podczas pracy z wyżej podaną tabelą będziemy np.: wyszukiwać wszystkich studentów, którzy znają język np.: hiszpański, to taka tabela nie jest w pierwszej postaci normalnej. Trzeba ją wówczas zaprojektować inaczej.

Bycie wartością elementarną jest raczej umowne. Dalsze postacie normalne wymagają znajomości pojęcia zależności funkcyjnej (często zwanej również funkcjonalną). W omawianych powyżej przykładach można zauważyć zależność miedzy sobą:
adres czytelnika zależy np.: od nazwiska czytelnika (ale nazwisko nie zależy od adresu)
Inaczej mówiąc:

numer czytelnika jednoznacznie określa adres czytelnika,
kod PESEL jednoznacznie określa nazwisko i imię osoby,
nr i seria dowodu osobistego określa osobę (ale nie odwrotnie) itp.

Sformalizujmy pojęcie **zależności funkcyjnej**, a mianowicie:

Niech X i Y będą podzbiorami atrybutów relacji R. Powiemy, że Y zależne funkcyjnie od X co symbolicznie zapiszemy:
X → Y (z X wynika Y)
jeśli nie jest możliwe by relacja R zawierała dwie krotki, które by miały różne wartości atrybutów ze zbioru X i jednocześnie miały różne wartości atrybutów ze zbioru Y.

Przykład:

numer czytelnika → nazwisko, adres; nie może być w/w relacji dwóch takich krotek, by numer czytelnika powtarzał się w bazie BIBLIO,

PESEL → nazwisko osoby; nie powinno być dwu osób z takim samym kodem PESEL,

nazwisko czytelnika (not →) adres czytelnika; not → oznacza tu, że nie zachodzi zależność funkcyjna; w tym przypadku nie zawsze tak jest

Druga postać normalna (2NF)

Aby sprawdzić, że relacja jest w drugiej postaci normalnej trzeba:

- Wyznaczyć zależności funkcyjne w relacji,
- klucz główny tabeli
- jeśli każdy atrybut nie należący do klucza głównego zależy funkcyjne tylko od klucza głównego, to relacja jest w drugiej postaci normalnej (co zapisujemy 2NF)

Rozważymy relację KLIENCI_BANKU, która zawiera następujące atrybuty:
*NR_KLIENTA,
IMIĘ
NAZWISKO,
TEL_DOM – numer telefonu domowego klienta,
*DATA_OPŁATY – data wpłaty
WIELKOŚĆ_WPŁATY – kwota opłaty

Kluczem głównym tej relacji jest NR_KLIENTA + DATA_OPŁATY ponieważ dany klient może dokonać wielu wpłat, w wielu dniach, a w danym dniu kilku klientów może dokonywać wpłat. Atrybuty TEL_DOM zależy funkcyjnie tylko o NR_KLIENT a nie zależy funkcyjnie od DATA_OPŁATY. Relacja powyższa jest zatem w 2NF.

Zasadą sprawdzania czy relacja jest w 2NF jest sprawdzenie czy wszystkie atrybuty relacji zależą od całego klucza głównego, czy tylko od jego części. Jeśli relacja nie jest w 2NF, to należy dokonać takiego podziału relacji na podrelacje, by atrybuty, które zależą od części klucza, znalazły się w odrębnej tabeli-relacji, razem z tą częścią klucza, od której zależą.

Dla rozważanego przykładu relację KLIENCI_BANKU należy rozłożyć na dwie relacje, a mianowicie:

KLIENCI (*NR_KLIENTA, IMIĘ, NAZWISKO, TEL_DOM)

oraz

OPŁATY (*NR_KLIENTA, *DATA_OPŁATY, WIELKOŚĆ_WPŁATY)

Trzecia postać normalna (3NF)

Zwykle jest tak, że jeśli sprowadza się relację do 2 NF, to jest ona również w 3NF. Tak jest w przypadku relacji omawianych w poprzednim przykładzie. Nie jest to regułą. Są często przypadki, że baza danych 2NF zawiera jeszcze anomalie i dlatego koniecznym jest dalsze normalizowanie. W celu zaprezentowania normalizacji 3NF rozważmy relację klientów banku, opisaną następująco:
KLIENT(NR_KLIENTA, NAZWISKO, IMIĘ, STANOWISKO, ŚREDNI_ZAROBEK) gdzie kluczem jest atrybut NR_KLIENTA. Przykładowe wypełnienie tej bazy jest następujące:

1	KOWALSKI	JAN	księgowy	3000
2	NOWAK	ADAM	spawacz	1800
3	ADAMSKI	JANUSZ	księgowy	3000

W tej relacji mogą występować anomalie aktualizacji i anomalie istnienia różnych średnich zarobków tych samych zawodów (np.: księgowy). Przeciętne zarobki zależą od NR_KLIENTA ponieważ każdy klient ma określone stanowisko, które określa średni zarobek.
NR_KLIENTA→STANOWISKO→ŚREDNI_ZAROBEK.

Taka zależność nazywa się **zależnością funkcyjną przechodnią.**

Jeśli relacja jest typu 2NF i zawiera zależności funkcyjne przechodnie, to nie jest w trzeciej postaci normalnej (3NF). Jeśli dla atrybutów A →B →C zachodzi zależność funkcyjna przechodnia, to relację ABC trzeba rozbić na dwie relację wg zasady:

- w pierwszej relacji pozostają atrybuty AB,
- w drugiej relacji pozostają atrybuty BC.

W rozważanym przykładzie, w celu uzyskania 3NF rozbijamy relację KLIENCI na dwie relacje, a mianowicie:
KLIENCI (NR_KLIENTA, NAZWISKO, IMIĘ, STANOWISKO)

oraz

STANOWISKA(STANOWISKO, ŚREDNI_ZAROBEK)

Relacja jest w 3NF jeśli jest w 2NF i nie zawiera przechodnich zależności funkcyjnych.

Czwarta postać normalna (4NF)

Stosuje się do relacji, które są w 3NF i nie zawierają wielowartościowych zależności funkcyjnych, które zdefiniowano poniżej

Niech relacja R zawiera atrybuty X, Y, Z. Mówimy, że X wyznacza wieloznacznie Y co zapisujemy:
X→→Y
Jeżeli w relacji R istnieją krotki (x,y,z) i (x,y1,z1) a z tego wynika, że istnieją również krotki (x,y1,z) oraz (x,y,z1).
Z symetryczności definicji wynika również, że w takiej relacji R zachodzi również zależność funkcyjna wielowartościowa postaci: X→→Z.
Jeżeli X wyznacza wieloznacznie Y, to mówimy, że Y zależy wielowartościowo funkcyjnie od X.
Jeżeli w relacji R={A,B,C} zachodzi A→→B i A→→C, to należy tę relację rozłożyć na dwie relacje AB i AC.

Przykładowe relacje, które nie są w 4NF:

a) relacja zawiera informacje o wykładowcach, grupach studenckich, w których oni realizują zajęcia dydaktyczne oraz dni tygodnia, w których te zajęcia są realizowane:

KOWALSKI	A	PONIEDZIAŁEK
KOWALSKI	B	PONIEDZIAŁEK
KOWALSKI	C	PONIEDZIAŁEK
KOWALSKI	A	WTOREK
KOWALSKI	B	ŚRODA
KOWALSKI	C	CZWARTEK

aby zaradzić takiej zależności funkcyjnej należy rozłożyć relację na dwie, a mianowicie

KOWALSKI	A
KOWALSKI	B
KOWALSKI	C

KOWALSKI	PONIEDZIAŁEK
KOWALSKI	WTOREK
KOWALSKI	ŚRODA
KOWALSKI	CZWARTEK

b) relacja zawiera informacje o znajomości języków obcych i języków programowania wśród studentów III roku kierunku INFORMATYKA:

123/97	angielski	C
123/97	rosyjski	C
123/97	angielski	PASCAL
123/97	rosyjski	PASCAL
123/97	angielski	PROLOG
123/97	rosyjski	PROLOG
124/97	hiszpański	C

relację tę należy rozbić na dwie następujące relacje:

123/97	angielski
123/97	rosyjski
124/97	hiszpański

oraz

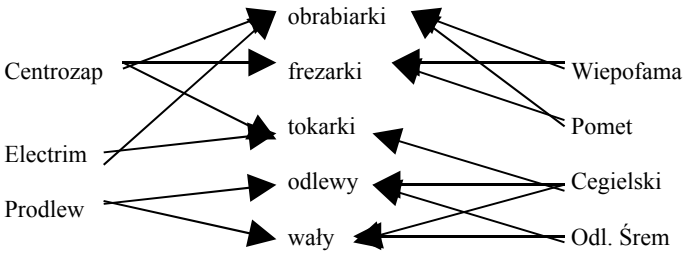
123/97	C
123/97	PASCAL
123/97	PROLOG
124/97	C

Piąta postać normalna (5NF)

W celu przedstawienia problemów związanych z piątą postacią normalną, rozważmy relację DOSTAWY_ZAGRANICZNE, zaprezentowane poniżej i spełniającą warunki czwartej postaci normalnej:

Centrozap	Wiepofama	obrabiarki
Centrozap	Wiepofama	frezarki
Centrozap	Pomet	obrabiarki
Centrozap	Pomet	frezarki
Centrozap	Cegielski	tokarki
Electrim	Wiepofama	obrabiarki
Electrim	Pomet	obrabiarki
Electrim	Cegielski	tokarki
Prodlew	Odl. Śrem	odlewy
Prodlew	Odl. Śrem	wały
Prodlew	Cegielski	odlewy
Prodlew	Cegielski	wały

W tej relacji przedstawiono informacje o producentach pewnych wyrobów i sprzedaży tych wyrobów przez centrale handlu zagranicznego. Związki występujące między centralami, a sprzedawanymi przez nie wyrobami, przedstawia następujący rysunek:



Przykładowo, centrala CENTROZAP sprzedaje OBRABIARKI, FREZARKI i TOKARKI, natomiast producent WIEPOFAMA produkuje OBRABIARKI i FREZARKI. Przedstawiona relacja jest w czwartej postaci normalnej, gdyż występuje w niej wyłącznie trywialna wielowartościowa zależność funkcyjna PRODUCENTa i WYROBu od CENTRALI_HZ. Ten stan rzeczy wynika z faktu występowania w relacji DOSTAWY_ZAGRANICZNE połączeniowej zależności funkcjonalnej, nie wynikającej z zależności atrybutów od klucza postaci:

#DOSTAWY_ZAGRANICZNE[(CENTRALA_HZ, PRODUCENT), (PRODUCENT, WYRÓB), (CENTRALA_HZ, WYRÓB)]

Dublowania się danych można uniknąć dekomponując tę relację na mniejsze, będące w piątej postaci normalnej.

Dana relacja jest w piątej postaci normalnej wtedy i tylko wtedy, gdy jest w czwartej postaci normalnej i w przypadku wystąpienia w niej połączeniowej zależności funkcjonalnej R{R1,...,RM} zależność ta wynika z zależności atrybutów klucza.

W celu doprowadzenia relacji DOSTAWY_ZAGRANICZNE do piątej postaci normalnej należy zdekomponować ją na trzy relacje: EKSPORTERZY, PRODUKCJA i SPRZEDAŻ w sposób następujący:

a) relacja EKSPORTERZY:

Centrozap	Wiepofama
Centrozap	Pomet
Centrozap	Cegielski
Electrim	Wiepofama
Electrim	Pomet
Electrim	Cegielski
Prodlew	Odl. Śrem
Prodlew	Odl. Śrem
Prodlew	Cegielski

b) relacja PRODUKCJA:

Wiepofama	obrabiarki
Wiepofama	frezarki
Pomet	obrabiarki
Pomet	frezarki
Odl. Śrem	odlewy
Odl. Śrem	wały
Cegielski	tokarki
Cegielski	odlewy
Cegielski	wały

c) relacja SPRZEDAŻ:

Centrozap	obrabiarki
Centrozap	frezarki
Centrozap	tokarki
Electrim	obrabiarki
Electrim	tokarki
Prodlew	odlewy
Prodlew	wały

Reasumując należy zauważyć, że normalizacja relacji ma na celu doprowadzenie jej do stanu, w którym elementarna informacja jest zapamiętana w bazie danych jednokrotnie. Dzięki temu unika się anomalii związanych z dublowaniem się danych, zmniejsza się zajętość pamięci i czas wykonania operacji.

Kiedy normalizuje się bazy danych?

W wielu przypadkach po normalizacji baz danych przechodzi się do rozważania możliwości wykonania operacji odwrotnej, która polega na złączaniu tabel rozłożonych przy pomocy normalizacji. Dlaczego?

Normalizacja polega na rozdziale tabel, to „wąskie" tabele są w działaniu na bazie danych często łączone ze sobą. Operacja złączenia jest najwolniejszą operacją w komputerze. Pominięcie tych złączeń jest równoważne przyspieszeniu dostępu do danych oczywiście kosztem pojawienia się anomalii. Jeśli na relacji aktualizacje odbywać się będą w bazie danych rzadko, to można dopuścić istnienie takich anomalii w tabelach.

Operacje

normalizacja $\leftrightarrow$ *optymalny dostęp*

współzawodniczą ze sobą w przyjęciu odpowiedniego rozwiązania.

Wniosek: Nie istnieje żadna sztywna reguła, na podstawie której można przekształcać normalizację nad czasem dostępu. Zawsze trzeba ocenić wady i zalety; wybrać rozsądne rozwiązania. Żeby to zrobić trzeba wykazać się gruntowną znajomością problemu i dysponować doświadczeniem projektowym. Ostateczne decyzje o zmianie w relacjach (ich rozdziale lub złączeniu) można podjąć w trakcie użytkowania (robi to administrator bazy danych). Takiej reorganizacji bazy danych nie powinien odczuć bezpośredni użytkownik bazy danych.

RELACJE

Relacja to tabela, w której wiersze nazywa się rekordami lub krotkami a każda kolumna tej tabeli jest przeznaczona do zapamiętania odpowiednich wartości - **atrybutów**. Atrybuty te mają swoją nazwę i typ.

Wiele relacji (zbiór relacji) to **relacyjna baza danych**.

Atrybut lub zestaw atrybutów relacji, które jednoznacznie identyfikują każdą krotkę relacji nazywa się kluczem relacji. Z punktu widzenia możliwości wyszukiwania rekordów z relacji interesujące są takie klucze, które zawierają minimalną ilość atrybutów. Takie klucze to **klucze kandydujące**. Jeśli w relacji jest wiele kluczy to jeden z nich ustala się jako **klucz główny**, pozostałe to **klucze wtórne**.

Indeksowanie bd. polega na utworzeniu zbioru wskazań do rekordów uporządkowanego ze względu na wartości danej grupy atrybutów.

KLASYFIKACJA JĘZYKÓW ZAPYTAŃ: PRZYKŁADOWE JĘZYKI ZAPYTAŃ, CHARAKTERYSTYKA, JĘZYKI DEKLARATYWNE I PROCEDURALNE

Języki zapytań są to mechanizmy umożliwiające wyszukiwanie
Do najważniejszych cech języka zapytań należą

- siła ekspresji – określa zakres tego co w danym języku można zrobić
- pełność – oznacza, że język zapytań musi mieć tę cechę, że w języku zapytań można zapisać taką kwerendę, która pozwoli odszukiwać w bazie danych dowolne informacją jednostkową lub zestaw informacji jednostkowych w bazie danych o ile te dane się tam znajdują
  - rozszerzalność operacyjna języka – jest to wyposażenie języka zapytań w takie mechanizmy które pozwolą
  - zapisać różnorodne operacje arytmetyczne
  - użycie funkcji agregujących
  - możliwość korzystania z wyrażeń warunkowych
  - możliwość nadawania nowopowstałym relacjom arbitralnych nazw
  - prezentacja (wydruk lub wyświetlanie) wskazanych relacji
- elastyczność języka, dotyczy to przede wszystkim czasu oczekiwania na odpowiedź po zadaniu pytania. Dotyczy to również rozmiaru pamięci roboczej potrzebnej do obsługi kwerend

Cennym rozszerzeniem języka zapytań jest jego **zanurzenie w językach programowania**, co pozwala profesjonalnemu użytkownikowi bazy danych dołączać do zdań języka zapytań zdań zapisanych w języku programowania
Jednym z ważniejszych kryteriów języków zapytań jest proceduralność. Przykładem języka proceduralnego jest język oparty na algebrze zapytań. W **proceduralnych językach** zapytań użytkownik buduje kwerendę w postaci procedury, którą można przedstawić jako ciąg kroków takich jak, selekcja, rzut, złączenie itd. Kwerenda jest tu zarówno opisem tego o co pytamy jak i metodą prowadzącą do uzyskania odpowiedzi.
Języki nieproceduralne. W tych językach kwerenda zapisana jest żądaniem użytkownika. Każde jego żądanie składa się z dwóch części

- pierwsza część opisuje żądany schemat relacji wynikowej
- druga jest opisem warunków jakie muszą spełniać krotki relacji

Klasyfikacja języków zapytań (dotyczy ona abstrakcyjnych języków zapytań, które w swojej czystej postaci nie są implementowane). Klasyfikacja jest dwustopniowa.
Na pierwszym poziomie języki zapytań dzieli się na
- języki algebraiczne (ISBL)
- języki oparte na rachunku predykatów

Języki oparte na rachunku predykatów, to języki nie proceduralne, w których kwerenda określa kształt (schemat) relacji wynikowej oraz warunek (predykat), który musi być spełniony przez krotki składające się na relacje wynikową.

Na drugim poziomie języków algebraicznych nie dzieli się. Dalszemu podziałowi ulegają języki oparte na rachunków predykatów
- języki oparte na relacyjnym rachunku krotek (QUEL)
- języki oparte na relacyjnym rachunku dziedzin (QBE)

Opis poszczególnych języków

Język ISBL jest językiem algebry relacji, do których zalicza się iloczyn i różnicę mnogościową oraz selekcję projekcję (rzutowanie) i łączenie.

Język QUEL jest językiem opartym o relacyjny rachunek krotek. Przedstawicielem systemu DBMS, który działa z takim językiem zapytań jest system INGRES. System ten działa w środowisku UNIX. Inspiracją był rachunek predykatów zaproponowany w abstrakcyjnym języku ALPHA. W języku tym można zadawać pytania dotyczące dwóch lub więcej relacji. Można usuwać, dodawć, sortować krotki. Można używać funkcji agregujących typu sumownie, wyznaczanie wartości największej i najmniejszej. Można QUEL zanużyć w programach napisanych w języku C.

Język QBE oparty na relacyjnym języku dziedzin. Twórca IBM. Język zyskał popularność ze względu na swój dialogowy i wizualny charakter.Zakresem zmiennej jest wartość atrybutu a nie zbiór krotek. W tym języku można określać warunki wyszukiwania dla określonych atrybutów np. X<J+3. Istotnym rozszerzeniem tego języka było wprowadzenie funkcji agregujących SUM, AVG, MIN, MAX, CNT (ilość). Język jest wyposarzony w mechanizmy typowe dla DML.
- zmienianie zawartości tabel (pól, krotek)
- definiowanie nowych tabel
- dodawanie pól lub atrybutów lub ich usuwanie

Język SQL jest językiem łączącym w sobie cechy języków algebraicznych i języka rachunku krotek. Został opracowany przez IBM w eksperymentalnym systemie SYSTEM R. Język ten stał się standardem w systemach b.d. Język ten zawiera w sobie część typu DML oraz DDL.
Podstawowa formuła zapytania w języku SQL
SELECT $R_{i1}A_1...,R_{ik}...A_k$
FROM $R_1...R_j$
WHERE Φ;
W formie tej lista $R_i,A_1...R_{ik}...A_k$ określa atrybuty relacji wynikowej Φ zaś określa warunki wyszukiwania. Formuła ta może być osiągnięta z użyciem złożenia operacji relacyjnych projekcji z selekcją. SQL pozwala tworzyć bardziej złożone konstrukcje niż QUEL. SQL wyposażony jest w operatory agregujące. Ma dodatkowe mechanizmy do aktualizacji b.d, zmiany wartości atrybutów, dopisywania krotek do relacji lub usuwania ich.

PODSTAWOWE OPERACJE NA BAZACH DANYCH: PROJEKCJA, SELEKCJA, RESTRYKCJA, ŁĄCZENIE, ILOCZYN KARTEZJAŃSKI, RÓWNOZŁĄCZENIE – CHARAKTERYSTYKA, PRZYKŁADY

a) Selekcja

- selekcja = wybór krotek (wierszy):

$$\Sigma_w(R) = \{K \in R : w(k)\}$$

  gdzie w jest warunkiem selekcji
- selekcja jest komutatywna:

$$\Sigma_v(\Sigma_w(R)) = \Sigma_w(\Sigma_v(R))$$

- przykład
  z przedstawionej relacji wybrano ludzi których nazwisko = Fajfer

Nr indeksu	Imię	Nazwisko
33445	Celina	Arbuz
22456	Wiktor	Fajfer
26876	Krzysztof	Majer

  w wyniku czego powstanie

22456	Wiktor	Fajfer

b) Projekcja (rzut)

- projekcja = wybór atrybutów (kolumn):

$$\pi_{S''}(R) = \{k(S') : k \in R\}$$

gdzie S' jest podzbiorem schematu S
- projekcja jest wzajemnie komutatywna z selekcją

$$\pi_{S''}(\Sigma_w(R)) = \Sigma_w(\pi_{S''}(R))$$

o ile warunek selekcji ma sens projekcji, tj. dotyczy tylko atrybutów wybranych w projekcji
- przykład
zostaną wybrane wartości atrybutów „Nr indeksu" i „Nazwisko" z krotek, dla których „Nr indeksu">25000

Nr indeksu	Imię	Nazwisko
33445	Celina	Arbuz
22456	Wiktor	Fajfer
26876	Krzysztof	Majer

w wyniku czego powstanie

Nr indeksu	Nazwisko
33445	Arbuz
26876	Majer

c) Złączenie

- operacja na dwóch relacjach – podzbiór iloczynu kartezjańskiego dwóch relacji:

$$R'*R'' = \{k' || k'' : k' \in R' \wedge k'' \in R'' \wedge w(k', k'')\}$$

gdzie w jest warunkiem złączenia
- krotki złączenia stanowią sklejenie (konkatenację) krotek relacji złączanych
- przykład

DOSTĘP DO BAZ DANYCH POPRZEZ WWW

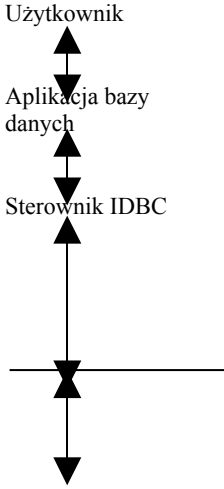
ODBC – ogólnie
- ODBC – sterownik dostępu do baz danych. Znajduje się w stacji użytkownika i wymaga połączenia z serwerem przez oprogramowanie sieciowe. Pracują dla różnych baz danych (Informix, Oracle).

- IDBC - Interfejsem programownia jest zbiór klas zdefiniowanych w języku Java. Interfejs ten przekazuje wywołanie sterownikowi ODBC. Używa się dwóch architektur współpracy z bazą danych.
  - sterownik IDBC komunikuje się bezpośrednio z serwerem bazy danych i tłumaczy instrukcje IDBC na wywołanie CLI rozpoznawane przez daną bazę;
  - Sterownik iDBC wymienia dane ze sterownikiem ODBC na serwerze i tłumaczy instrukcje IDBC na wywołanie ODBC, który je wykonuje przez kontakt z odpowiednim serwerem baz danych.

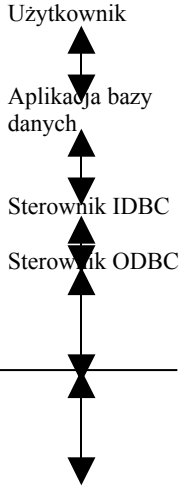
CLI – zbiór funkcji zaimplementowanych na język programowania. Funkcje te przekazują polecenie SQL bezpośrednio do obsługi baz danych przez serwer baz danych.

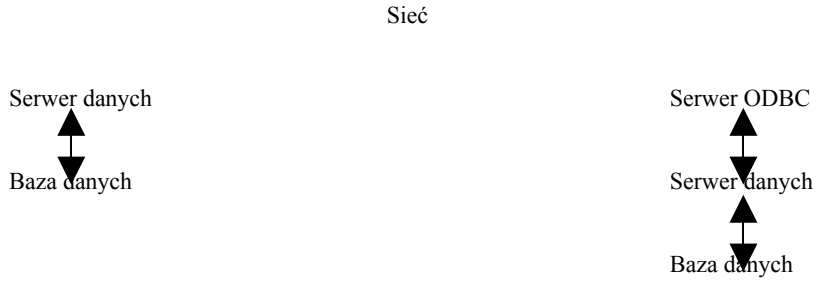
Dostęp do baz danych przez sieć (nowoczesne rozwiązanie)
Common Deathway Interface – dostęp przez WWW, umieszczenie aplikacji w minimalnej liczbie kopii.

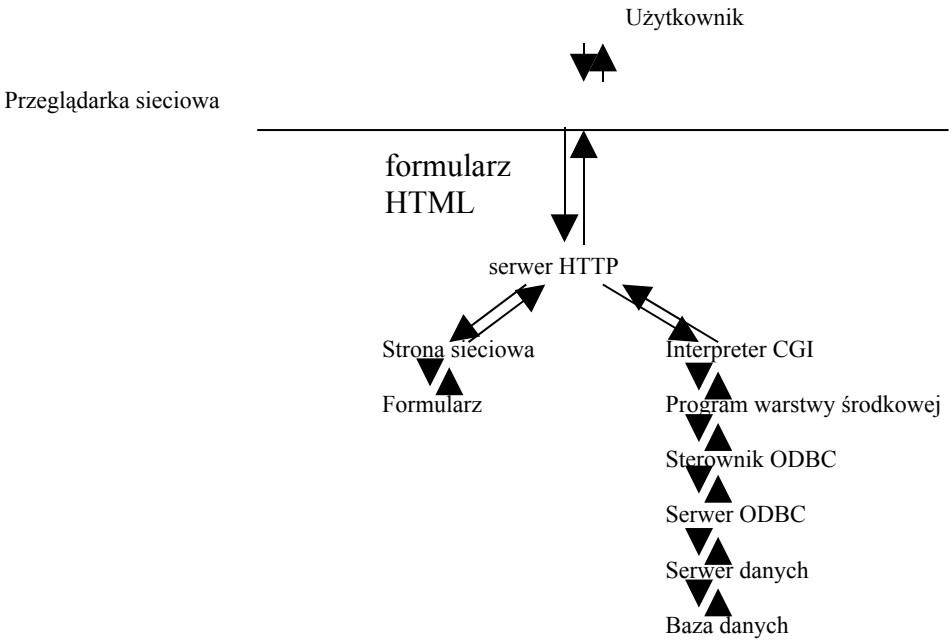
Arch1.



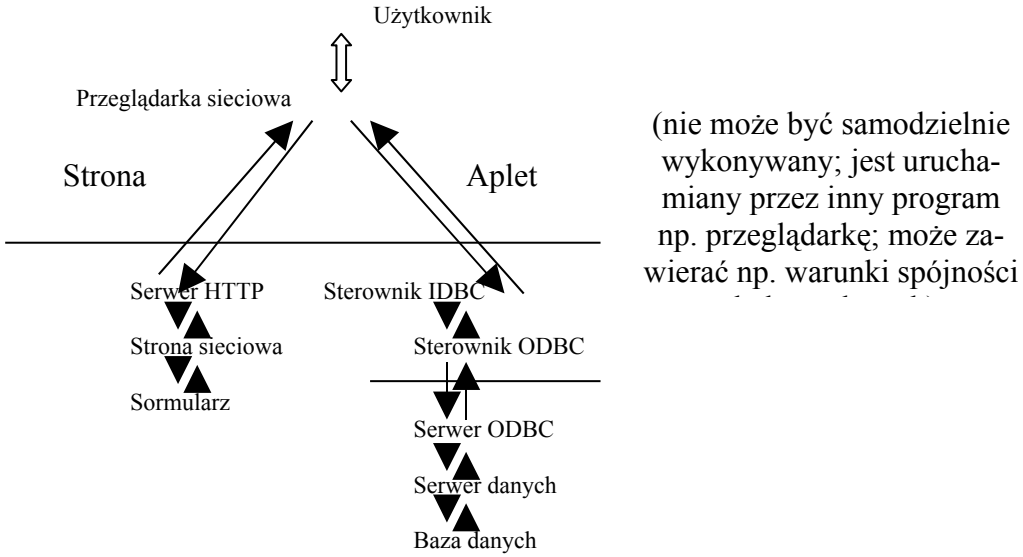
Arch2.







- **Dostęp przez aplet JAVY**



OBIEKTOWE BAZY DANYCH (OBD) – WŁASNOŚCI CHARAKTERYSTYKA (OBD) ; WIELOWERSYJNE BAZY DANYCH, METODY ZARZĄDZANIA OBIEKTOWYMI BAZAMI DANYCH.

Około 90% obecnie występujących baz danych to bazy relacyjne (Oracle, Progress, DBase, Informix, FoxPro, Access, Gupta). Przed nimi były baza hierarchiczne i sieciowe. Zaletami relacyjnych są:

- prostota struktur danych
- nieproceduralny język zapytań (co ma być wyszukane, nie jak) – deklaratywny. Na standardzie SQL'a.

- niezależność fizyczna i logiczna danych (logika – aplikacja, fizyka – nośnik)
- optymalizatory zapytań, ochrona danych

Prostota może być jednak także wadą, czasami proste związki nie wystarczają, wymagane są złożone struktury (zwłaszcza ich związki) (np. obrazy graficzne, zasłanianie i odsłanianie powierzchni). Przy danych tekstowych chcielibyśmy operować różnymi metodami.(np. analizy leksykograficzne).
Występują 2 trendy rozwoju:

- wzbogacenie relacyjnego o inne, bardziej złożone związki lub możliwości (np. Informix), gdzie użytkownik może definiować swoje relacje np. graficzne
- obiektowe bazy danych

Relacyjny model danych ma 2 nieadekwatności: statyczną i dynamiczną

Statyczna – relacje na ogół odzwierciedlają rzeczywistość. W tabeli chcemy zapisać najważniejsze cechy, atrybuty modelu. Tabele później możemy modyfikować. Może powstać tabela nie odzwierciedlająca rzeczywistości. (Przy użyciu prostych semantyczne związków możemy utworzyć bezsensownie strukturę)

Dynamiczna – dane powinny być kojarzone z operacjami na nich wykonywanymi, powinny stanowić całość. Te bazy nie łączą tabel, relacji z metodami, funkcjami DBMS (np. wyszukiwanie danych). Wyeliminować można to przez powstanie baz obiektowych. Obiekty (atrybuty i metody) są proste i złożone. Istnieją hierarchia kompozycji obiektów i hierarchia dziedziczenia metod.
Obiektowa baza danych – występują obiekty z metodami i 2 hierarchie: kompozycji i dziedziczenia metod. Obiekty należą do klas. Obiektowe BD zapewniają:

- ochronę danych,
- współbieżność
- oddzielenie struktury fizycznej i logicznej
- spójność bazy danych (nie może powstać stan dwuznacznych danych po operacji)

Istnieją 3 rodzaje obiektowych baz danych:

1) Cechy baz danych dodano do obiektowego języka programowania (Gem Stone). Dane w obiektach należą do klas. (Między nimi przesyłane są komunikaty: nazwa odbiornika, nazwa metody i parametry metody)
2) Do cech systemu baz danych mających język programowania nieobiektowego dodano obiektowe podejście (Ontos na C++). Cechy – prosty zapis struktury połączonych z metodami.
3) Podejście obiektowe dodane do relacyjnych BD. Niekoniecznie muszą to być relacje, mogą być obiekty (IRIS). Bardzo rozbudowane, język SQL także rozbudowany do OSQL (obiektowego). Są obiekty i są funkcje. Popularnym przykładem jest także O2 – obiektowy. Działamy na zdegenerowanych obiektach do postaci relacyjnej. POET – baza z dość ciężkim interfejsem wyszukiwania danych (trzeba pisać program w C++ by móc na niej pracować, nie ma select'a)

Obiekty mogą być powiązane różnorako, Przez przynależność do klas i.. . Przy modyfikacji atrybutu obiektu musimy wziąć pod uwagę, czy nie jest komponentem bardziej złożonego obiektu. Metody blokowania muszą dotyczyć hierarchii dziedziczenia i kompozycji. Struktury – drzewa kompozycji muszą być zapamiętane. Modyfikacja i algorytm bardzo skomplikowane. Współbieżność – algorytm wydłużony – operowanie może być tradycyjne. Występuje w dużym stopniu problem spójności danych – także rozbudowany algorytm. Ponadto występują kłopoty z indeksowaniem.

WIELOWERSYJNE BD

np. proces projektowania samochodu przez kilka zespołów, w pewnym momencie projekt musi być widoczny dla innych zespołów. Muszą występować warianty historyczne, w algorytmie podejmowania decyzji ważną rolę grają stare wartości. Są dwa typy takich baz:

- z wersjami obiektów – historyczne
- z wersjami obiektów – wariantowe

W systemie baz danych istnieje wiele wersji obiektów, nie muszą być ze sobą spójne. Mogą istnieć elementy nie tworzące kompletnej wersji (wariantowe), wielowersyjne zaś:

- z wersjami obiektów
- z wersjami baz danych

Z wersjami baz danych występuje problem redundantności, z reguły występują systemy z wersjami obiektów. Wśród obiektów wyróżnia się obiekt generyczny – ostatni, aktualny. Do hierarchii kompozycji, dziedziczenia dochodzi hierarchia wariantów (wersji). Może to być postać rozbudowanego grafu skierowanego (drzewo wywodu wersji - dla każdego obiektu). W drzewie wywodu wersji wszystkie muszą być blokowane w wypadku blokady. Powinno się też eliminować powtarzanie informacji w wersjach.
Każdy obiekt musi mieć identyfikator. W przypadku wersji – także jej identyfikator. Nie wszystkie obiekty muszą mieć wersje.
Wersje z bazami danych. Blokujemy obiekty w całej wersji – jest to o wiele prostsze jeśli chodzi o redundancję. Przy blokowaniu wykorzystujemy drzewo hierarchii wersji. Transakcje są realizowane w jednej wersji, a także w innych wersjach: obiektowe – wewnątrz jednej wersji BD a bazowe – na wielu wersjach.

DEDUKCYJNE BAZY DANYCH

Model ten powstał z klasycznego modelu relacyjnych baz danych, do którego dodano mechanizmy wnioskowania takie jak w języku PROLOG – np. COBOL, LDL, EX (zastosowania w systemach, systemach wspomagania decyzji). Zawierają skończony zbiór klauzul i język logiki 1-go rzędu. Musi reprezentować stałe i warunki. Do klauzul typu fakty dodane są reguły logiki – wynika, and, or... Fakty to inaczej predykaty bazowe (tabele z danymi), np. stan magazynu – tabela z ilością i ceną. Na bazie predykatów bazowych tworzy się predykaty wirtualne.

Możliwe są predykaty logiki 1-go rzędu, które umożliwiają tworzenie nowych predykatów. Na podstawie predykatów bazowych możemy opracować predykat wirtualny, taki jak np. tani towar. Jest on wynikiem implikacji towaru (podst. pred) i podanej ceny. Możliwe jest użycie pętli typu while, while do i funkcji agregujących (sum, max...), użycie uwarunkowań postwarunkowych przed lub po, typu istnieje – nie istnieje. Można tworzyć zdania z faktów predykatów wirtualnych (odbywa to się w pamięci)

PROBLEM MAGAZYNU (HURTOWNI) DANYCH

W firmie występuje kilka baz, różnie zarządzanych, tworzymy aplikację zarządzającą typami baz danych. Są to dane heterogeniczne. Sposób zarządzania jest taktyczny, strategiczny. Dostęp do takich danych określa się na 6 sposobów

1. Konwersja danych ze starego środowiska do nowego. Nowa aplikacja działa na przekonwertowanych danych. Większość to umożliwia
2. Pomosty między systemami bazodanowymi – gateway'e (3 rodzaje baz – 3 pomosty)
3. System otwartej łączności ODBC (Open Data BaseConnectivity - Microsoft) – udostępnia bibliotekę funkcji. Aplikacja odwoływuje się do bazy przez interfejs ODBC. On uruchamia managera sterowników, który wywołuje konkretny sterownik, który dopiero działa na bazie. Schemat:

- Aplikacja wywołuje zapytanie na dane SQL
- Menedżer sterowników w imieniu aplik. ładuje odpowiedni sterow. do DBMS, pod którym są dane
- Sterownik BD odpowiednio realizuje modyfikując je
- przesył danych przez sterow. menedżer i aplik. - użytkownik. Integracja danych na poz. aplikacji

4. Mechanizm SQL Links – podobne do ODBC – lecz produkt Borlanda, obsługuje tylko SQL. Głównym elementem jest dynamiczna biblioteka wykorzystywana do komunikacji z DBMS
5. Specyfikacja OLE DB wraz ze zbiorem dostępu do danych. Mechanizm ten łączy i wstawia obiekty. Utworzony w celu przenoszenia informacji między aplikacjami Windows. Obsługuje różne środowiska i formaty. Ma mechanizm wyszukiwania czy identyfikowania danych. Interfejsy wyszukiwania możemy instalować w różnych źródłach. OLE DB wychodzi poza tradycyjny sposób widzenia BD i rozszerza jego możliwości. Umożliwia dzielenie relacji na fragmenty, udostępnia zdarzenia potrzebne do komunikacji między tymi elementami. Łączy dane z procesem wyszukiwania w jeden bezpośredni obiekt, który może być później używany. Funkcje zgodne z OLE mają dostęp do sterowników ODBC. Mechanizm OLE może być używany jako bezpośr. lub pośr.w środowisku Internetu i do aplikacji, które niekonieczne korzystają z SQL'a
6. Zfederowane systemy bazodanowe – system – nakładka na istniejące systemy bazodanowe – składowe tej federacji. Oprogramowanie zarządza komponentami. W sposób lokalny komunikują się z innymi komponentami.za pośrednictwem tej nakładki. Zaletą jest przechowywanie jednego (globalnego) schematu danych utworzonego z schematów lokalnych. Nakładka zarządza współbieżnością lokalnych BD (między nimi, one wewnętrznie robią to same)

TEMPORALNE BAZY DANYCH

Zagadnienie czasu, wykorzystują uwarunkowania czasowe. Występuje problem kwantu czasu. SQL nie ma mechanizmów czasowych, trzeba rozbudować język o te mechanizmy „TSQL". Klauzula where zmodyfikowana o zależności dotyczące czasu (atrybut czasowy w określonym przedziale) Można np. zapytać o wartość czasu pomiędzy dwoma miesiącami. Klauzule first, second, third – można określić kto był pierwszym odbiorcą towaru. Klauzula moving window, która pozwala przeglądać BD w określonym przedziale czasowym. Są odpowiedzią na wyzwanie, że statyczne dla potrzeb podejmowania decyzji powinny określać zmieniającą się rzeczywistość.

Aplikacja złożona
przestrzenne uwarunkowanie zarządzania informacją (ewidencja gazowni, okablowanie w telekomunikacji). Łączenie różnych środowisk wizualizujących informację przestrzennie. Posiada atrybuty wizualizacji przestrzennej. Do złożonych zaliczmy również aplikacje oparte na hipermediach – zbiór węzłów, różnych rodzajów mediów połączonych w całość.(np. MicroGaleria)

OCHRONA BAZ DANYCH

W każdym systemie DBMS są środki, które zapobiegają zarówno zapamiętywaniu w b.d. niepoprawnych danych jak i odczytywaniu danych przez osoby nieupoważnione, niepowołane.

Istnieją dwa źródła niepoprawnych danych:

- pomyłki (np. przy wprowadzaniu danych)
- złe użycie b.d.,

Mówiąc o ochronie b.d. warto wyróżnić dwa zagadnienia:

3) Utrzymywanie integralności,
4) Ochrona (czyli kontrola danych) dostępu

Ad.1 Zagadnienie to łączy się z zapobieganiem nieumyślnym błędom
Ad.2 Zależy nam głównie na tym, aby pewni użytkownicy mieli dostęp tylko do określonego podzbioru b.d. i tylko ten ewentualny podzbiór mogli aktualizować.

Bardzo ciekawym problemem jest ochrona statystycznych baz danych. W takich bazach problem polega nie na tym by ograniczyć użytkownikowi możliwość dostępu do jakiejkolwiek szczegółowej części b.d., lecz na tym aby mu przeszkodzić w dedukowaniu z informacji statystycznych, takich jak średnie zarobki itp., danych szczegółowych, np. dochodu poszczególnych osób.

Integralność

Zakłada się by systemy DBMS nakładały na b.d. dwa rodzaje więzów, więzy strukturalne oraz zależności funkcjonalne. Nie wszystkie jednak zależności funkcjonalne mogą być podtrzymywane przez systemu DBMS. Zezwalają one np. na deklarowanie, że zbiór pól lub atrybutów tworzy klucz do typu rekordu lub relacji. Inny rodzaj więzi dotyczy wartości bieżących zapamiętywanych w b.d. Więzi te noszą miano

więzów integralności. Podstawową ideą dotyczącą więzów integralnościowych jest to, że więzy te mogą być wyrażone w języku manipulacji danymi.

Przykład

Więzy integralności określa się tak, że nikomu nie wolno mieć ujemnego salda w zakresie np. udziałów członkowskich. Można je wyrazić w postaci wymogu, że:

$$\Pi_{NAZWISKO}(CZŁONKOWIE)=\Pi_{NAZWISKO}(\vartheta_{SALDO\geq 0}(CZŁONKOWIE)$$

Drugą częścią deklaracji więzów integralnościowych jest opis, kiedy warunki wcześniej określone mają być sprawdzane.

Przykład

Więzy integralności dotyczą w tym przypadku wartości bieżących zapamiętywanych w b.d. Niech pola EGZAMIN, PRACA, LAB oznaczają procentowy udział czasu przeznaczonego na egzamin, pracę domową i laboratorium, to dla zachowania integralności będziemy oczekiwać, że w każdym rekordzie suma wartości będzie wynosić 100%. Opis tego faktu stanowi warunki integralności.

Ogólnie deklaracja więzów integralnościowych wymaga podania tych więzów w postaci pewnego opisu, kiedy i jak te warunki integralności mają być sprawdzone. W DBMS-ach zezwala się aby takie „więzy" działały jako „przerwania" wysokiego poziomu, np. w językach manipulacji danymi zezwala się umieszczać warunki postaci

ON <lista rozkazów> CALL <procedura>

w deklaracjach typów rekordów lub w deklaracjach typów kolekcji. Taka <lista rozkazów> może zawierać rozkazy typu FIND, INSERT, REMOVE zaś <procedura> jest dowolnym programem napisanym w języku manipulacji danymi, np. dla typu kolekcji S zadeklarowano

ON INSERT CALL P1

to procedura P1 powinna sprawdzać, czy pewne pola bieżącego rekordu zadania, którym jest wstawiany rekord podrzędny, nie występują już w zbiorze wystąpień. W ten sposób możemy zapewnić integralność b.d. polegającą na tym, że wraz z kluczem dla rekordu nadrzędnego wyznaczają funkcyjnie resztę pól w typie rekordu podrzędnego. Warunek ON w zapisie więzów integralności jest uaktywniony zawsze wtedy, kiedy jest wykonany rozkaz z tej listy i bieżący rekord zadania jest odpowiedniego typu.

Czasami w systemach DBMS nakłada się więzy integralności na aktualizację, tak by istniały związki między starymi i nowymi-uaktualnionymi wartościami pewnych atrybutów. Tu oprócz więzów dla samej krotki włączamy również linię reprezentującą starą krotkę. Np. przy pobraniu towaru z magazynu nie można zmniejszyć ilości towaru pozostającego na stanie magazynu, więcej niż wynosi ilość towaru w magazynie przed jej pobraniem.

Najczęściej więzy integralności nie są sprawdzane po wykonaniu pojedynczych rozkazów, lecz tylko wtedy kiedy zostaną wykonane całe ciągi rozkazów. To pozostawia pewną swobodę co do kolejności specyfikowania rozkazów dopóty, dopóki są wprowadzane grupowo. Np. w systemie obsługi magazynu nakładamy więzy w taki sposób, że na towar, którego nikt nie dostarcza nie można złożyć zamówienia. Jeśli jednak jako „zawartość ekranu" wprowadzimy kilka zamówień oraz informacje o dostawcach i po tym sprawdzimy więzy integralności, to więzów tych nie naruszymy. Jeśli jednak system wprowadzi zamówienie i sprawdzi więzy integralności przed wprowadzeniem informacji o dostawcach to naruszenie integralności nastąpi.

OCHRONA

Zagadnienie ochrony danych przed nieuprawnionym dostępem ma wiele różnych aspektów. Należy:

- chronić b.d. przed niepożądaną modyfikacją lub zniszczeniem danych
- chronić przed nieuprawnionym odczytem danych

Ochrona danych nie jest tylko domeną systemów DBMS. Z podobnymi problemami spotykamy się też na poziomie systemu operacyjnego. Dlatego tylko o niektórych powszechnych metodach ochrony tylko wspomnimy. Zajmiemy się pewnymi szczególnymi problemami związanymi z DBMS-ami. Jednym z problemów ochrony jest metoda identyfikacji użytkownika. Ogólnie mówiąc, różnym użytkownikom nadaje się różne uprawnienia do korzystania z różnych b.d. lub różnych części b.d. np. relacji, atrybutów, określonych atrybutów w określonych relacjach. Te uprawnienia mogą dotyczyć: odczytu, ustawiania, usuwania, aktualizowania. Najbardziej rozpowszechnioną metodą identyfikacji użytkowników jest wprowadzenie haseł znanych tylko systemowi i użytkownikowi. System chroni te hasła co najmniej tak dobrze jak dane, chociaż nie ma żadnej gwarancji ich pełnego bezpieczeństwa.

Innym ważnym problemem jest ochrona fizyczna. Chcąc całkowicie niezawodnie bronić b.d. trzeba brać pod uwagę różne możliwości fizycznego jej naruszenia – od wymuszenia ujawnienia haseł, aż do kradzieży czy uszkodzenia nośników pamięci. Przed skutkami naruszenia haseł lub zawartości b.d mogą uchronić również różne sposoby szyfrowania danych. Te sposoby wchodzą w zakres tzw. Kryptologii, kryptografii i będą tam dokładniej omawiane chociaż trzeba pamiętać, że szyfrowanie wymaga również tzw. Deszyfracji co powoduje wydłużenie czasu dostępu do danych. W zakresie ochrony b.d. znajduje się również problem utrzymywania i przekazywania uprawnień. Systemu DBMS muszą utrzymywać dla każdego użytkownika listę uprawnień do korzystania z każdego chronionego fragmentu b.d. Jednym z takich uprawnień może być uprawnienie do nadawania uprawnień innym.

Rozważmy teraz dwa mechanizmy ochrony b.d. zaprojektowane przede wszystkim specjalnie dla systemów DBMS, są nimi:

- perspektywy
- użycie języka zapytań do definiowania uprawnień

Istnieją perspektywy dwóch rodzajów:

- read-only – perspektywa niezmienna
- perspektywa zezwalająca na odczytywanie oraz zapisywanie obiektów będących ich częścią. Aktualizacje perspektyw są dopuszczalne i są odzwierciedlane w schemacie przejściowym b.d.

Niezmienna perspektywa najczęściej używana jest wtedy, gdy właściciel b.d. (administrator) chce ogólnie udostępnić prawo doczytywania danych zachowując dla siebie lub ograniczonego kręgu prawo do aktualizowania tych danych. Perspektywy niezmienne nie mogą być aktualizowane.

Poważnym problemem w perspektywach modyfikowalnych jest to, że aktualizowanie perspektyw powoduje również skutki uboczne w części b.d. nie należącej do perspektywy. W systemach b.d. nie jest jasne, co oznacza usunięcie niektórych składowych np.: krotek, jeśli w relacji występują inne atrybuty nie należące do perspektywy ; użytkownik widzący tylko perspektywę nie powinien mieć możliwości ich usunięcia. Z tego powodu system musi odrzucać aktualizację wielu perspektyw. Podobnie jest w systemach hierarchicznych b.d. gdzie można mieć perspektywę na konkretny rekord lecz bez jego typu „potomków". Usuwając ten rekord musimy mieć możliwość usunięcia „potomków" dla zachowania choćby integralności b.d. Taka akcja może być nielegalna. Podobna sytuacja może mieć miejsce w modelu sieciowym gdy chce się usunąć rekord nadrzędny nie wiedząc nic o jego rekordach podrzędnych, ponieważ nie należą do tej perspektywy.

Drugim ważnym mechanizmem ochrony b.d. jest wykorzystanie **języka manipulacji danymi do definiowania uprawnień dostępu** każdego użytkownika do b.d. Tego języka można używać do definiowania uprawnień na podobnych zasadach jak używanie go do definiowania więzów integralności. Tzn. można spowodować by do każdego zapytania zadanego przez użytkownika i dotyczącego wyznaczenia relacji automatycznie była dołączana **selekcja** i **rzut.** W systemach DBMS można stosować również tzw. blokady jawności dla chronionego obiektu w postaci procedur wyrażonych w języku manipulacji danymi.

Perspektywę można uważać za tabelą utworzoną z innej tabeli (relację z innej relacji). Perspektywa jako taka nie istnieje fizycznie w b.d. Jest to tablica wirtualna, której zawartość zmienia się, kiedy tylko określające ją tabele ulegają zmianie. Perspektywa użytkownika była uważana za trzeci poziom bazy danych po poziomie fizycznym i logicznym. W wielu przypadkach jest ona po prostu zbiorem kolumn pochodzących z tabel utworzonych na poziomie logicznym. Za pomocą perspektywy użytkownik może widzieć bazę danych w specyficzny dla siebie sposób. Perspektywa to instrukcja języka manipulacji danymi (języka zapytań); aby np.: utworzyć w ORACLE perspektywę należy napisać:

CREATE VIEW <nazwa perspektywy> AS <zapytanie>

Gdzie:

<nazwa perspektywy> - to nazwa nowej perspektywy
<zapytanie> - instrukcja typu SELECT, która ma być wykonywana przy każdym odwołaniu się do perspektywy.

Każda zmiana dokonana w określających perspektywę tabelach będzie także uwzględniona w perspektywie. Perspektywa może mieć jednak własny system ochrony. Za pomocą doboru tabel i perspektyw programista b.d. może zapewnić każdemu użytkownikowi odpowiedni dla niego obraz bazy danych.
Do usuwania perspektyw mamy w ORACLE instrukcję

DROP VIEW <nazwa perspektywy

Ponieważ perspektywy są używane tak jak tabele, naturalne byłoby dodawać, usuwać i modyfikować wiersze z perspektyw. Stanowi to ważny problem. Dlatego w systemach DBMS są wprowadzane dodatkowe ograniczenia na możliwość wstawiania danych przez perspektywy, a w niektórych z nich w ogóle jest to zabronione w celu uproszczenia procedur sprawdzających. Modyfikacja wiersza przez perspektywę jest równoważna usunięciu go, a następnie dodaniu nowego wiersza, co prowadzi do podobnych problemów, jak te związane ze wstawieniami (wstawienie danych do perspektywy niw wstawia danych w kolumnach nie należących do perspektywy a tylko do tablicy nad którą utworzono perspektywę).

Dodatkowo obok perspektyw wykorzystuje się komendy ochrony danych. NP. w ORACLE gdy użytkownicy mają założone poprawne konta jako podstawowy sposób wprowadzania ochrony służy instrukcja GRANT. Instrukcja ta umożliwia lub zabrania innym użytkownikom dostępu do tabel lub wierszy w obrębie tabel. Chodzi tu o usuwanie, wstawianie, wybór i modyfikację. Aby np.: użytkownik, który jest rozpoznawany w systemie jako „Piotr", miał prawo wstawiać, wybierać lub modyfikować, ale nie usuwać całe wiersze z tabeli „klient" należy wywołać komendę GRANT w następującej postaci:

GRANT INSERT,SELECT,UPDATE ON klient TO Piotr

Gdy taka instrukcja zostanie wykonana to zostaną określone odpowiednie prawa dostępu. Jeżeli użytkownik typu „Piotr" wstawi jakiś wiersz, a chce go teraz usunąć może to zrobić za pomocą instrukcji

ROLLBACK

Wycofanie transakcji. Wówczas wiersz tan nie będzie przechowywany w bazie danych.
Do zapisu uprawnień dostępu stosuje się najczęściej tzw, tablice uprawnień. W tych tablicach można zapisywać uprawnienia do relacji nadawane pewnej osobie lub wszystkie uprawnienia związane z relacją. Pierwszy sposób może być zrealizowany przy pomocy specjalnej tablicy uprawnień zapisywanej w postaci zbioru dyskowego odpowiedni szyfrowanego.

PODSUMOWANIE RELACYJNYCH BAZ DANYCH

1) Relacyjna baza danych jest widziana przez użytkownika jako zbiór tabel (relacji). Dla każdej jest dostępny zbiór operatorów umożliwiających wydzielenie danych. Powiązania danych są realizowane jedynie za pomocą danych (nie ma jawnych wskaźników między danymi).
2) Język do obsługi baz danych jest językiem deklaratywnym.
3) Użytkownik nie podaje ścieżki dostępu do danych i nie musi znać fizycznej reprezentacji danych.
4) Jeśli użytkownik chce obsługiwać tablicę, musi ją otworzyć,
5) W relacyjnej bazie danych istnieje pełna niezależność danych.
6) Wyróżniamy języki DDL (Data Definition.. Language) i DML (Data Manipulation Language).
7) Systemy zarządzania bazami danych można rozróżniać między sobą ze względu na:
   a) model danych – w systemach zarządzania b. d.: relacyjne, hierarchiczne, sieciowe, obiektowe, obiektowo-relacyjne. Obowiązuje w nich określony model danych,
   b) rozmieszczenie b. d. obsługiwanych przez dany system:
      - z centralną b. d. – obsługiwana przez serwer na zasadzie klient-serwer. Jest to baza jednoserwerowa (jednorodna)
      - rozproszona b. d. – b. d. umieszczone są w węzłach rozproszonej sieci serwerów. W węzłach mogą być zaimplementowane jednorodne lub różnorodne serwery (systemy) – heterogeniczne bazy danych, a system przyjmuje miano magazynu danych.
   c) niektóre systemy bazodanowe pamiętają historię zdarzeń (np. nie tylko dane, ale i ich wersje) – są to bazy wielowersyjne (historyczne albo wariantowe). Mogą pamiętać wersje obiektów lub wersje całej bazy.
   d) nowe b. d. nie tylko wyszukują informacje. Są także bazy dedukcyjne – system oparty na relacyjnej bazie danych zaopatrzony w mechanizmy dedukcyjne wnioskujące na podstawie zawartych danych. Dedukcyjne bazy danych oparte są na mechanizmach dedukcji lub na logice.
   e) zapis czasu. Jeżeli system bazodanowy uwzględnia czas to nazywamy go temporalnym. Dziś reprezentacja czasu w bazie danych jest bardzo ważnym problemem. Bazy temporalne mogą wpływać na algorytmy podejmowania decyzji.
   f) systemy bazodanowe są na ogół powiązane z językami IV generacji lub systemami szybkiego tworzenia aplikacji (RAPID) – generator menu, aplikacji, formularzy, raportów. Z poziomu każdego z tych generatorów jest możliwy dostęp do zapytań języka SQL.
   g) niektóre systemy baz danych udostępniają narzędzia pozwalające na tworzenie bazy od początku do końca (od generatora poprzez SQL, aż do kodu źródłowego).
   h) niezawodność, odtwarzanie systemu po awarii. Najczęściej ten problem realizuje się przy pomocy dziennika – zapis relacji wykonywanych na bazie. Awaria może być spowodowana awarią sprzętu lub awarią wynikającą z aplikacji. System powinien posiadać mechanizm pozwalający na odtworzenie b. d. sprzed awarii (za pomocą dziennika b. d. – dziennika transakcji). W tym dzienniku zapisuje się identyfikatory transakcji dokonujących aktualizacji, identyfikator modyfikowanej krotki, wartość krotki przed i po modyfikacji. Dokonuje się także przepisywania co jakiś czas danych na inny zapasowy nośnik.

ZABEZPIECZENIE BAZ DANYCH PRZED AWARIĄ

Jest to podstawowa funkcja każdego SZBD. Musi on mieć mechanizm dziennika bazy danych, gdzie odnotowuje się informacje na temat transakcji. Każda transakcja jest zapisywana w postaci identyfikatora transakcji oraz starej i nowej wartości elementarnej jednostki. Są różne rozwiązania, najczęściej tylko zapisuje się wartości zmieniane, ale są też systemy zapisujące wszystkie wartości. W pierwszym przypadku jest więc to tylko zabezpieczenie przed awarią, w drugim może także odnosić się do parametrów eksploatacyjnych.
Różne systemy przyjmują różne jednostki elementarne. Niektóre dzienniki zapisują całe stare i nowe wartości, inne natomiast zapisują tylko starą i nową zawartość pola. Gdy się coś dopisuje nowego, to zapisuje się starą wartość i to co dodane.

Mówi się że transakcja się wypełniła, gdy wykonała się od początku do końca.
W SZBD mówi się że transakcja się wypełniła gdy wszystkie zmiany są zapisane i w dzienniku i w bazie danych oraz we wszystkich obszarach obszarach roboczych.
Przyjmuje się dwufazową strategię wypełnienia:
3) transakcja nie może zapisać do bazy danych dopóty, dopóki się nie wypełniła w dzienniku baz danych;
4) transakcja nie może być wypełniona dopóty, dopóki nie zapisze w dzienniku wszystkich dokonywanych przez siebie zmian na jednostce elementarnej.

Najczęściej wykorzystuje się mechanizmy blokowania. Między zablokowaniem i odblokowaniem realizowane są również inne mechanizmy. Zabezpieczenie przed awarią polega na tym, że gdy wystąpi awaria, to na podstawie dziennika odtworzone zostają operacje, które się wypełniły (od pewnego momentu). Na podstawie dziennika anuluje się transakcje niewypełnione i powtarza się wykonanie tych transakcji, które się wypełniły. Ważna jest wielkość dziennika, metody jego przeszukiwania, itp...

Blokowanie nie jest jedyna metodą realizowania współbieżności. Inna jest tzw. **metoda optymistyczna (metoda znaczników czasowych, metoda walidacji).** Elementy bazy danych nie są tu blokowane (oszczędza to czas realizacji). Ta metoda nie traci czasu na blokowanie, lecz realizuje transakcję lub umieszcza ją w kolejce Ta metoda wymaga pewnego czasu na realizację transakcji. Jest efektywna gdy nie ma konfliktów między transakcjami. Zasada tej metody polega na nadawaniu każdej transakcji znacznika czasowego. Są one liczbami generowanymi w każdym takcie zegara. Istnieje szeregowalność pojawiających się transakcji. Każda transakcja ma inny znacznik czasowy, niezależnie od tego co chce robić. Znaczniki powinny być jak najmniejszymi liczbami. Należy często kopiować bazę, aby znaczniki nie były zbyt duże. Najczęściej są one 16-bitowe.

Dla każdej jednostki bazy danych zapamiętywane są:
- czas odczytu – najmniejszy znacznik czasowy należący do transakcji odczytującej tą jednostkę;
- czas zapisu – największy znacznik czasowy, który zapisuje jednostkę.

Transakcje w systemie pojawiają się szeregowo. Znacznik jest porównywany z czasem zapisu i odczytu, aby sprawdzić, czy transakcję można realizować czy też umieścić ją w kolejce.
Podczas realizacji transakcji nie jest możliwe aby transakcja mogła odczytywać jednostki jeżeli zostały one zapisane już po wykonaniu tej transakcji. Czyli jeżeli transakcja ma znacznik czasowy t_1 i chce czytać, to nie może tego zrobić, jeżeli da tej jednostki czas zapisu t_2 jest większy od t_1 ($t_2 > t_1$). Transakcja jest odrzucana.
Jeżeli dwie transakcje mają różne znaczniki czasowe i chcą zapisywać, to nie są odrzucane ale sprawdza się czy nie pojawił się jakiś odczyt pomiędzy nimi.

Załóżmy że transakcja zgłasza się znacznikiem czasowym t, gdy będzie chciała wykonać operację na jednostce X to dla tej jednostki będą podane dwa czasy: t_r i t_W.
Zasady:

4) operację wykonujemy jeżeli X = READ i $t \geq t_W$ lub jeżeli X =WRITE i znaczniki czasowe $t \geq t_r$ i $t \geq t_W$ Jeżeli $t>t_r$, to czas odczytu jednostki X przybiera wartość t. Jeżeli $t>t_W$ to czas zapisu jednostki X przybiera wartość t;
5) Jeżeli X=WRITE i zachodzi zależność $t_r \leq t \leq t_W$ to transakcja nie jest realizowana (nie robimy ale i nie odrzucamy);
6) Jeżeli X=READ i $t<t_W$ lub X=WRITE i $t<t_r$ to transakcję odrzucamy

OPTYMALIZACJA ZAPYTAŃ
Zadaniem SZBD jest jak najszybsze udostępnienie informacji użytkownikowi (gdy użytkownik zadaje zapytanie, to baza danych powinna jak najszybciej znaleźć odpowiedź). Zamiast mówić o optymalizacji czasu mówi się o optymalizacji całego zapytania. Będzie to suboptymalizacja (czyli nie najlepsza).
Optymalizacja zapytania polega na przeanalizowaniu kontekstu w celu przeorganizowania zapytania – zamienia kolejność wykonywania operacji na bazie, tak aby odpowiedzieć użytkownikowi na zapytanie i nie pogubić wyników.
Np.: złączanie tabel:

- jedna z kolumn A,B
- druga z kolumn C,D

Przyjmujemy, że w obu tabelach jest określona ilość rekordów: n_{AB}, n_{CD}.
Aby te relacje połączyć sprowadzamy je przez obszar roboczy pamięci operacyjnej zawierający m bloków dal relacji AB i jeden blok dal relacji CD.
n_{AB} m
n_{CD} 1
b_{AB} – ilość rekordów AB w jednym bloku z m.
Obliczamy całkowitą ilość dostępów do bloków, niezbędną dla odczytów AB – i

$$i1 = INT\left(\frac{n_{AB}}{m * b_{AB}} + 0{,}5\right)$$ podobnie: $$i2 = INT\left(\frac{n_{CD}}{b_{CD}} + 0{,}5\right)$$ INT – część całkowita

Rozważmy pierwszy sposób łączenia relacji: ściągamy pierwszą partię rekordów AB oraz i2*dostęp do CD. Potem ściągamy drugą partię i (i2-1)*b_{CD} itd., aż do wykonania (i2-1)*b_{CD} (dostęp do CD) i ściągamy i2 porcji AB.
Oznaczamy: ild_I – ilość dostępów przy pierwszym sposobie: $ild_I = i1(i2-1) + 1$
Drugi sposób: najpierw ściągamy paczkę rekordów CD a potem do tego pojedynczy AB, pierwsza partia rekordów CD i i1 rekordów z AB, druga partia rekordów CD oraz i1-1 dostępu do AB itd., na końcu i2 partię rekordów CD i dla niej wykonujemy i1-1 dostępów do AB.
$ild_{II} = i2(i1-1) + 1$

Wprowadzamy oznaczenia:
δ_{warunek} – selekcja , $\prod_{\text{nazwa podrelacji}}$ – rzut
Mamy relacje AB i CD i dokonujemy następującego zapytania:

$$\prod_A (\delta_{B=C \cap D="99"} (AB \times CD))$$

odpowiedź nie zmieni się, gdy zapytanie to zostanie przekształcone do innej formy:

$$\prod_A (\delta_{B=C} (AB \times \delta_{D="99"} (CD))$$

wykonanie zapytania w tej formie będzie szybsze jeżeli ilość rekordów wśród CD o wartości 99 będzie niewielka.

Optymalizator zapytań polega na zmianie początkowego zapytania w wersję oszczędniejszą czasowo. Algorytm optymalizacji oparty jest na drzewie wyrażeń.

Relacja równozłączeniowa – mamy dwie relacje R i S i równozłączenie dotyczy dwóch wybranych atrybutów z tych relacji (C z R i D z S).
Równozłączenie zapiszemy przy pomocy symboli:

$$R \underset{C=D}{|X|} S$$

Operacja selekcji może być zastąpiona równaniem: $$\Pi = (AB \underset{B=C}{|X|} \delta_{D="99"} (CD))$$

Wnioski z działań algebraicznych na bazach danych:

1) Selekcje wykonuje się tak wcześnie jak jest to tylko możliwe. Selekcja wpływa na zmniejszenie czasu wykonywania o rząd wielkości, bo ma ona tendencję do zawężania pośrednich wyników wielokrotnych;
2) Łączyć należy niektóre selekcje z poprzedzającym je iloczynem kartezjańskim w celu utworzenia złączenia lub rozłączenia, które może być efektywniejsze od samego iloczynu kartezjańskiego;
3) Należy łączyć jednoargumentowe w pewne ciągi – w jedną operację wielowarunkową.

Przy wykonywaniu przekształceń należy posłużyć się następującymi prawami:

1) prawo przemienności dla złączeń i iloczynów

$$E_1 \underset{F}{|x|} E_2 \equiv E_2 \underset{F}{|x|} E_1$$

2) prawo łączności dla złączeń i iloczynów

$$(E_1 \underset{F_1}{|x|} E_2) \underset{F_2}{|x|} E_3 \equiv E_1 \underset{F_1}{|x|} (E_2 \underset{F_2}{|x|} E_3)$$

3) prawo kaskada rzutów
Zakładamy że atrybuty $\{ A_1..A_n \} \subset \{ B_1..B_m\}$ wykonujemy najpierw
$\prod_{A1...An} (\prod_{B\,B2...Bm} (R)) \equiv \prod_{A1...An}$ ®
4) kaskada selekcji
$\delta_{F1} (\delta_{F2} R)) \equiv \delta_{F2} (\delta_{F1} (R)) \equiv \delta_{F1 \cap F2} (R)$
5) przemienność selekcji i rzutów
$\prod_{A1...A\,k} (\delta_F (E)) \equiv \delta_F (\prod_{A1..A\,k} (E))$
6) przemienność selekcji i iloczynu kartezjańskiego
$\delta_F (E_1 \times E_2) \equiv \delta_F(E_1) \times E_2$
Z tego prawa wypływają również inne związki:
- jeżeli $F = F_1 \cap F_2$ to $\delta_F (E_1 \times E_2) \equiv \delta_{F1}(E_1) \times \delta_{F2}(E_2)$
- jeżeli F_1 dotyczy atrybutów relacji E_1, F_2 dotyczy zarówno E_1 jak i E_2, to można zapisać:
$\delta_F (E_1 \times E_2) \equiv \delta_{F2}(\delta_{F1}(E1) \times E_2)$

7) przemienność selekcji z sumą
$\delta_F (E_1 \cup E_2) \equiv \delta_F(E_1) \cup \delta_F(E_2)$
8) przemienność selekcji z różnicą
$\delta_F (E_1 \setminus E_2) \equiv \delta_F(E_1) \setminus \delta_F(E_2)$
9) przemienność rzutów z iloczynem kartezjańskim
Przyjmujemy E_1E_2 – relacje $A_1...A_n$ – atrybuty relacji E_1
$B_1..B_m$ – atrybuty dokładnie tylko E_1
$C_1...C_k$ – atrybuty tylko relacji E_2
$\prod_{A1...A\,n} (E_1 \times E_2) \equiv \prod_{B1...B\,m} (E_1) \times \prod_{C1...C\,k} (E_2)$
10) przemienność rzutów z sumą
$\prod_{A1...A\,n} (E_1 \cup E_2) \equiv \prod_{A1...An} (E_1) \cup \prod_{A1...An} (E_2)$

W oparciu o powyższe prawa wprowadzany jest algorytm optymalizacji zapytań (nie ma gwarancji, że uzyskamy optymalne wyrażenie).
ALGORYTM:
1) Stosuje się regułę kaskady selekcji, aby każdą selekcję warunków typu END rozdzielić na kaskadę selekcji: $\delta_{F1\cap F2\cap...\cap Fk} (E) \equiv \delta_{F1} (. .. \delta_{Fk} (E))$
2) Dla każdej selekcji stosujemy reguły przemienności selekcji z innymi operatorami relacyjnymi tak by wykonać selekcję najwcześniej
3) Dla każdego rzutu stosujemy regułę kaskady rzutu i rozdzielamy rzut na kaskadę rzutów. Następnie stosujemy regułę przemienności rzutu z iloczynem kartezjańskim i sumą oraz uogólnioną regułę przemienności selekcji i rzutów. Wszystko to stosujemy po to, by rzuty wykonywały się jak najwcześniej. Przy stosowaniu uogólnionej reguły przemienności selekcji i rzutów może się okazać, że zginą niektóre rzuty. Jednak zazwyczaj ta reguła rozbija rzut na dwa rzuty
4) Kaskadę selekcji i rzutów połączyć w jedną selekcję, rzut lub selekcję, po której nastąpi rzut. Stosujemy reguły przemienności selekcji i rzutów, aby najpierw była selekcja a potem rzut.

Kroki 1)-4) pozwolą na określenie drzewa. Wierzchołkami będą operatory relacji lub selekcja, rzut. Kolejne gałęzie będą rozwijane, jeżeli dany wynik selekcji będzie poprzedzany innym operatorem.

Innym sposobem optymalizacji jest realizacja problemów zapytań prostych dotyczących pojedynczych relacji. W przypadku zapytań koniunkcyjnych stosujemy metodę opartą na stosowaniu specjalnych tablic, które przekształcamy aż do otrzymania zoptymalizowanych tablic. Po kolei rugujemy poszczególne wiersze tablicy (matrycy).
Optymalizacja zapytań w oparciu o specjalną matrycę dla specjalnej formy zapytań.
Rozważamy tylko zapytania koniunkcyjne.
Zapytanie koniunkcyjne jest to zapytanie o pewne atrybuty $A_1...A_2$ relacji R pod warunkiem że istnieją symbole nierozróżnialne $b_1...b_m$ i zachodzi warunek koniunkcji postaci $P_1 \cap P_2. .. \cap P_k$
gdzie P_i (dla i=1..k) jest jedną z dwóch postaci:
1) $R (c_1...c_l)$ - to oznacza że $c_1...c_l$ należą do relacji R, a c_j (j =1..l) jest atrybutem $A_1...A_n$ lub jest symbolem nierozróżnialnym $b_1...b_m$ lub jest stałą.
2) c-d – oznacza, że c i d należą do $A_1...A_n$ lub są symbolami nierozróżnialnymi lub są stałymi, a – jest jednym z operatorów <,>,≤, ≥.

Symbole nierozróżnialne mogą to być atrybuty relacji nie występujące jawnie w zapytaniu (są niejako symbolami pomocniczymi). Ogólnie zapytanie koniunkcyjne dotyczy pojedynczych relacji.
Dla zapytań koniunkcyjnych konstruuje się tablicę, a następnie przekształca się ją poprzez rugowanie wierszy. Obowiązuje zasada równoważności tablic.
Matryca jest tablicą dwuwymiarową. W pierwszym wierszu zapisujemy nazwy atrybutów. Wiersz następny nazywany jest reduktorem i składa się z symboli pustych lub odróżnialnych, występujących w zapytaniu koniunkcyjnym. Symbole występujące w dalszej części tabeli to symbole nierozróżnialne.
Metoda matryc polega na tworzeniu matryc równoważnych.
Twierdzenie.
Mówimy że tablica T_1 jest zawarta w T_2 ($T_1 \subseteq T_2$) ⇔, gdy istnieje takie odwzorowanie h symbol matrycy T_2 w symbole matrycy T_1, że:
1) h zastosowane do reduktu tablicy T_2 jest reduktem tablicy T_1;

2) h zastosowane do każdego wiersza T_2 daje wiersz tablicy T_1 z tym samym znacznikiem;
3) więzy w tablicy T_2 wynikają z przesłanki, że matrycy T_1 więzy te są nałożone na symbole, na które h przekształca symbole z tablicy T_2.
Matryce T_1 i T_2 są równoważne ($T_1 \equiv T_2$) $\Leftrightarrow$ gdy $T_1 \subseteq T_2$ i $T_2 \subseteq T_1$.
Skonstruowanie tablicy najmniejszej z możliwych i równoważnej tablicy wyjściowej jest ostatnim krokiem optymalizacji.

Minimalizacja matryc.
Niech T_0 – matryca początkowa, T_m – matryca uzyskana z T_0 poprzez przekształcenia (ma najmniejszą możliwą ilość wierszy).
Jeżeli T_0 jest dowolną matrycą, to istnieje matryca do niej równoważna z mniejszą liczbą wierszy uzyskaną drogą eliminacji z tablicy T_0 0 lub więcej liczby wierszy. Istnieje wzajemna równoważność tych tablic z T_0, jeżeli te matryce uzyskano przy pomocy odwzorowania równoważności, Eliminując kolejne wiersze dochodzimy do matrycy o najmniejszej liczbie wierszy tak, że tablice są jeszcze równoważne.
W tej drodze mamy prostsze struktury do przeglądania i sprawdzania. Tablica, która nie może dalej być redukowana jest odpowiednikiem zapytania,.
Oba podejścia (tablice i drzewa) wykorzystuje się w programach komercyjnych.

Współbieżność i czas w bazach rozproszonych.
Baza danych jest podzielona na wiele węzłów. Żeby zablokować jednostkę bazo- danową zawsze zakładamy że znajduje się ona w innym węźle. Między węzłem pracy a węzłem gdzie jest jednostka do blokowania trzeba przesłać komunikaty. Wydaje się, że lepsza jest grubsza ziarnistość blokowania (blokowanie większych obiektów). Zakładamy, że oprócz węzłów i relacji istnieje kilka kopii bazy danych. Kopie muszą być identyczne. Przyjmujemy dwufazowy protokół realizacji transakcji. Realizujemy zapis i odczyt.

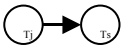
Metody blokowania.
1) metoda całkowitego blokowania wszystkich jednostek;
2) blokowanie zapisu jednej jednostki;
Ta metoda zakłada że blokada zakładana jest na wszystkie kopie. Blokowanie kopii jest częścią blokowania jednostki. Transakcja blokuje zapis jednostki A, gdy zablokuje zapis dowolnej jej kopii. Transakcja całkowicie blokuje jednostkę A, gdy blokuje wszystkie jej kopie.
Jest to pierwsza z metod blokowania. Są również inne- np. metoda blokowania większości, metoda żetonu, metoda węzła centralnego. Każda ma inny koszt. Wyróżniany komunikaty krótkie (przesłanie informacji, że chcemy zablokować) i długie (odczyt, zapis).

MODELE BLOKOWANIA B. D.
- Szeregowalność harmonogramu dla modelu o dwóch operacjach – zakładanie i zdejmowanie blokady.

Transakcja – pewien ciąg czynności na bazie danych (np. pojedyncze wyszukiwanie informacji). Może być zapisana z poziomu programu lub w trybie konwersacyjnym.

Harmonogram – kolejność wykonywania transakcji z dokładnością do elementarnych operacji wykonywanych przez transakcję, tj. blokowania, czytania, odblokowania, zapisu, itp.
Harmonogram jest sekwencyjny, jeżeli wszystkie operacje każdej transakcji występują kolejno po sobie.
Harmonogram jest szeregowalny (dający się uszeregować), jeżeli wynik działania tego harmonogramu jest równoważny wynikowi otrzymanemu za pomocą pewnego harmonogramu sekwencyjnego. Jeżeli w momencie pojawienia się harmonogramu okaże się że jest on szeregowalny, to transakcja może być wykonywana w systemie bazodanowym.
W module zarządzania blokadami sprawdza się czy harmonogram jest szeregowalny. Nie każdy harmonogram jest sprawdzany przy realizacji w implementacjach b. d. Większość harmonogramów jest sprawdzana z punktu widzenia operacji zapisu, odczytu, zablokowania, odblokowania.
Poprzez program sprawdzający szeregowalność harmonogramu można osiągnąć rozwiązanie problemu impasu. Dla wszystkich jednostek bazy danych wprowadza się jeden lub więcej protokołów przestrzeganych przez wszystkie transakcje. Na przykład protokół, w którym przyjęto określoną metodę blokowania jednostek w określonej kolejności. W normalnych harmonogramach występują jeszcze operacje zakładania i zdejmowania blokady, dopiero na podstawie tego rozpatrujemy problem szeregowalności.

1) **Algorytm jak szeregować gdy występują operacje Lock i UnLock i są one realizowane dla każdej operacji Read i Write.**
S – harmonogram składający się z transakcji T1...Tk
Metoda sprawdzania szeregowalności polegać będzie na utworzeniu grafu pierwszeństwa (graf zorientowany). Wierzchołkami będą transakcje. W grafie należy określić krawędzie. Wyznacza się je tak:
Przyjmujemy, że mamy dwie transakcje: Tj – ma zadanie Lock albo UnLock. Jeżeli operacja w harmonogramie jest operacją odblokowania, czyli operacją na transakcji Tj, Tj:UNLOCK[Am], to szukamy w harmonogramie takiej transakcji Ts, która blokuje tę samą jednostkę bazy danych Am. Ts:LOCK[Am]. W grafie dokonujemy wtedy połączenia Tj i Ts:



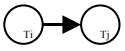
Jeżeli graf pierwszeństwa zawiera cykl, to taki harmonogram nie jest szeregowalny. Gdy w grafie nie ma cyklu, to harmonogram jest szeregowalny i można do grafu zastosować metodę sortowania topologicznego, która pozwoli na sprowadzenie tego szeregowalnego harmonogramu do harmonogramu sekwencyjnego. Polega to na znalezieniu w grafie pierwszeństwa takiego wierzchołka Ti, do którego nie dochodzą żadne inne krawędzie. Taki wierzchołek wpisujemy jako pierwsza transakcja w realizacji, usuwamy ten wierzchołek z grafu i znowu szukamy w podgrafie. Powtarzamy to, aż nie usuniemy wszystkich wierzchołków z grafu pierwszeństwa. Kolejność usuwania wierzchołków definiuje harmonogram sekwencyjny, który dowodzi szeregowalności harmonogramu.

2) Poprzedni model jest dość ograniczony, bo przy każdej operacji trzeba jednostkę zablokować i odblokować. Lepszym sposobem blokowania jest zakładanie blokad różnego typu, posiadających wspólny UnLock. W tym nowym modelu istnieją dwie blokady: zapisu (współdzielona - Rlock) i całkowita (wyłączna - WLock).
Tabela zgodności blokad: „T” dla T1[RlockA] i T2 [RlockA], dla reszty „F”

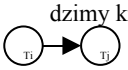
Jeżeli mamy wiele transakcji T1...Tk zakładających Rlock, to ona tak długo obowiązuje, aż transakcje odczytają wszystkie dane. Gdy pojawi się WLock (było już Rlock) to taka transakcja ustawia się w kolejce i czeka aż Rlock nie zostanie zwolniona na danej jednostce bazy danych. Obowiązuje jedno odblokowanie UnLock.

Algorytm konstrukcji grafu:

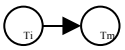
3) W harmonogramie jest transakcja Ti, która blokuje zapis jednostki A (RlockA), a Tj jest następną w harmonogramie transakcją, która chce całkowicie zablokować A (WlockA). Tj występuje po Ti:



4) Przypuśćmy, że Ti blokuje całkowicie A (WlockA), a Tj jest następną transakcją, która chce całkowicie zablokować a (WlockA); prowadzimy krawędź z Ti do Tj



Niech Tm będzie transakcją RlockA, gdy Ti zdjęła wcześniej blokadę na A (UnLockA), a Tj całkowicie blokuje A, to poprowadzimy krawędź z Ti do Tm



3) Ten model został poszerzony o tzw. dwufazowość, czyli wg protokółu dwufazowego oznaczamy go jako 2PL lub TwoPhaseLocking. Polega on na tym, że każda transakcja przebiega w dwóch fazach: zablokowania i odblokowania, więc nie ma oddzielnego odblokowania.
Moment założenia wszystkich blokad nazywa się punktem akceptacji. Zapis realizuje się dopiero w fazie odblokowania, a jednostka zostaje odczytana w fazie blokady. Można dodatkowo założyć, że jednostka zablokowana i zwolniona, może mieć nową blokadę dopiero po jakimś czasie.
Inaczej wygląda harmonogram szeregowalności, graf będzie multigrafem, gdzie niektóre ścieżki mogą być traktowane wariantowo. Algorytm jest podobny do poprzednich. Ta metoda jest najczęściej implementowana w SZBD. Jest połączona z wypełnieniem i realizacją dziennika transakcji.
4) W niektórych bazach danych można zakładać blokady na całe relacje, krotki, wybrane elementy.Gdy w systemie jest możliwe powyższe blokowanie, to nazywamy ją metodą hierarchicznego blokowania.
Jeżeli weźmiemy zbyt małe jednostki do blokowania, to możemy otrzymać niespójność bazy danych. Ważne jest jaką jednostką dysponujemy. Całą bazę możemy rozpatrywać jako hierarchię drzewiastą, korzeń to baza danych, a kolejne wierzchołki to relacje.

Dla relacji można wyznaczać kolejne jednostki, które możemy blokować.
Metodę hierarchiczną opracował Grey. Oparta jest ona na większej ilości blokad:

5) współdzielna i wyłączna (S i X),
6) intencjonalna blokada współdzielna (IS),
7) intencjonalna blokada wyłączna (IX),
8) blokada mieszana (SIX), która jest połączeniem S i IX.

Blokada intencjonalna współdzielona (IS) wierzchołka W w drzewie hierarchii blokad oznacza, że wystąpiła intencja odczytu co najmniej jednego następnika wierzchołka W. Blokada IS pozwala zakładać blokady IS lub S na wszystkie następniki wierzchołka W.
Blokada intencjonalna wyłączna (IX) wierzchołka W oznacza, że wystąpiła intencja zapisu co najmniej jednego jego następnika. Blokada IX pozwala na zakładanie blokad IX, X, IS, S oraz SIX na następnikach wierzchołka W.
Blokada mieszana (SIX) wierzchołka W pozwala ma dostęp współdzielony do danych należących do wierzchołka W i do jego następników oraz pozwala na zakładanie na tych następnikach blokad wyłącznych intencjonalnych.

		T2				
		IS	IX	S	SIX	X
T1	IS	T	T	T	T	
	IX	T	T			
	S	T		T		
	SIX	T				
	X					

Algorytm hierarchicznego blokowania transakcji obejmuje 4 kroki:

5) Pierwszym wierzchołkiem dla którego transakcja T żąda dostępu o danych jest korzeń hierarchi ziarnistej lub korzeń poddrzewa.
6) Transakcja T może założyć blokadę IS lub S wierzchołka nie będącego korzeniem ⇔ gdy T blokuje poprzednik wierzchołka blokadą IS lub IX
7) Transakcja T może założyć blokadę IX lub SIX lub X na wierzchołku W, który nie jest korzeniem ⇔ gdy blokuje go poprzednik blokadą IS lub SIX.
8) Wszystkie blokady założone przez transakcję T muszą być odblokowane po zakończeniu tej transakcji lub w trakcie jej wykonywania w kolejności odwrotnej do kolejności ich zakładania.

W tej stosuje się jeszcze modyfikację – sposób zapisu drzewa zmieniany jest na B-drzewo.
Do harmonogramu blokowania hierarchicznego stosuje się specjalne protokoły ostrzeżeń, które przestrzegają ten algorytm przed właściwą realizacją blokad.

Bezpieczeństwo przy awarii na tle metod blokowania transakcji:

System musi mieć dodatkowe mechanizmy zabezpieczenia przed awariami. Dotyczy to tylko scentralizowanych baz danych. Inne metody dotyczą systemów rozproszonych. Mówimy o awarii powodujących błędną interpretację danych i są odwracalne (do naprawy przez SZBD).
Aby zabezpieczyć system baz danych stosuje się metody:

- Okresowe sporządzanie kopii b. d. bez wiedzy użytkowników,
- Stworzenie kopii jest realizowanie jako odrębna transakcja na danych. Może być realizowane planowo lub tworzy się sama po upływie jakiegoś czasu.
- Mechanizmy dzienników: Dziennik Bazy Danych obejmuje identyfikator transakcji powodującej zmianę w b. d., starą i nową modyfikowanej jednostki danych. Mogą być też zapisywane dodatkowe informacje, np. czy wypełniono transakcję do końca.

GŁÓWNE ELEMENTY SZBD

- procesor języka manipulacji danymi - tłumaczenie zapytań;
- moduł współbieżnej obsługi transakcji;
- optymalizacja zadań - skracanie czasu dostępu do danych
- procesor zarządzający danymi - zawiera tablicę ochrony dostępu;
- program obsługi danych - przekształca modele logiczne na fizyczne;
- procesor schematu bazy danych.

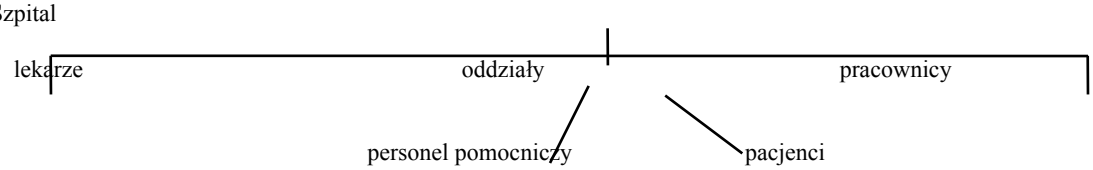
Modele danych:

- sieciowy
- hierarchiczny
- relacyjny
- obiektowy;

Model sieciowy:

Podstawową własnością tego modelu jest istnienie takich odwzorowań aby powstały sieci. CODASYL - 1971 - podaje jak stworzyć b. d. aby była to baza sieciowa.
Model hierarchiczny - wyróżniamy tu związki tworzone hierarchicznie. Przykład: szpital

Szpital



Struktury danych:

- sąsiedztwo fizyczne - obok korzenia wpisuje się wszystkie elementy (drzewo);
- łańcuchy i pierścienie, drzewo;
- mapy bitowe;
- odsyłacze (wskaźniki);

Model relacyjny - ujęcie danych w tabelę danych wiąże się je tylko wtedy gdy jest taka potrzeba. Tablice nie są wzajemnie powiązane. Tabele nazywa się relacjami; składają się one z krotek, czyli wierszy, rekordów, w nich są określone atrybuty, które należą do określonej dziedziny. Dziedzina określa zbiór wartości pola. Atrybuty - określona wartość pola. Rekordy nie mogą się powtarzać. Niektóre atrybuty mogą być wartościami odróżniającymi krotki - taki atrybut nazywamy kluczem.
Schemat formalny (pojęciowy) - zb. nazw atrybutów zapisujemy w postaci:
$S = \{A_1 ..... A_2\}$
A_i - atrybut relacji
R - relacja
D_i - dziedzina
S - schemat formalny.

def. Relację R rozpiętą na schemacie S nazywamy odwzorowanie $\{ k_1 ..... k_p \}$, takie że dla każdego $k \in R$ $K(A_i) \in D_i$. Odwzorowanie $k_1 ..... k_p$ nazywa się krotkami tej relacji. Relacja to zbiór krotek. Krotka określa wartość dla każdego atrybutu zgodnego z dziedziną atrybutu. pojęcie podkrotki w zbiorze ś wyróżnia się podzbiór B [$B \subseteq S$] krotki rozpięte na atrybutach tego zbioru.
Kluczem K w zbiorze S będziemy nazywać zbiór atrybutów, takich że $K \subseteq S$ i dla dowolnych krotek $k_1 .... k_p$ należących do R_s (relacja R rozpięta na schemacie S) wartości krotki są różne
$k_1 (K) \neq k_2 (K)$
Każdy podzbiór zbioru K będziemy nazywać kluczem desygnowanym, jeżeli w zb. kluczy desygnowanych wyróżnimy jeden klucz to będziemy go nazywać kluczem głównym. Kluczem właściwym w S będziemy nazywać taki klucz, którego żaden podzbiór właściwy nie jest kluczem. Wszystkie klucze które nie są desygnowane są kluczami niejawnymi (nie zostały określone).

OPERACJE WYKONYWANE NA RELACJACH

Jeżeli mamy dwie relacje to dla nich można utworzyć następną relację, która jest suma tych relacji.
Różnica dwóch relacji R_1, R_2: do relacji R_1 będą należeć tylko te krotki które nie należą do tej drugiej relacji.

Iloczyn dwóch relacji zawiera krotki wspólne dla tych dwóch relacji.
Selekcja:

$$\sum_w \left(R_s \right) = \{k \in R_s : w(k) = true\}$$

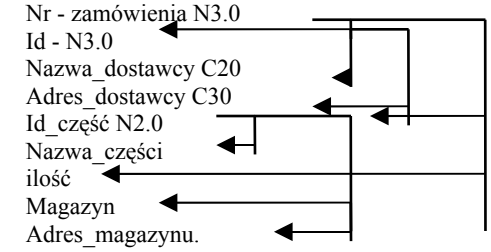
w - warunek selekcji buduje się przy użyciu AND, OR, NOT φ = G{ =, ≠, <, >, <=, >=, }w postaci K { A } φ α; K (A) φ K (B), gdzie α ∈ dan (A); A, B < S;
Projekcja - wypis tych krotek, które przyjmują różne wartości dla. .... pól.
$\Pi_s^{'}$ (R_s = { K(s'); K ∈ R_s ∩ S' ⊆ S }
Łączenie:
R'$_K$ (A) φ$_K$ (B) * R'' lub R' * R''
Dzielenie jest wyborem krotek jednej relacji względem drugiej relacji. Własności:
Πs' ($\Sigma_{A=a}$ (R * Πs (R')) jest relacją.
B. d. która jest relacją i zawiera atrybuty nierozerwalne, nierozróżnialne jest to b. d. pierwszej postaci normalnej. Atrybut B relacji jest wpełni funkcjonalnie zależnie od atrybutów X, jeżeli jest funkcjonalnie zależny od tego zbioru atrybutów i nie jest funkcjonalnie zależny od żądanego podzbioru zbioru X.
Relacja R jest w drugiej postaci normalnej, jeżeli jest pierwszej postaci normalnej i każdy atrybut tej relacji nie wchodzący w skład żądanego klucza potencjalnego. Test wpełni zależy funkcji od wszystkich kluczy potencjalnych.
Niech X Y Z będą trzema rozłącznymi podzbiorami atrybutów relacji R.
Podzbiór Z jest przechodnio funkcjonalnie zależny od podzbioru X jeżeli podzbiór Z jest funkcjonalnie zależny od Y, X nie jest funkcjonalnie zależny od Y, Y jest funkcjonalnie zależny od X, Y jest funkcjonalnie zależny od Z.
Relacja jest w B. .. jeżeli jest w 2. .. i każdy jej atrybut nie wchodzący do klucza potencjalnego nie jest przechodnie funkcjonalnie zależne od żadnego klucza tej relacji.

PROJEKTOWANIE BAZ DANYCH - PROBLEMY NORMALIZACJI

Początkowo, w procesie projektowania b. d. projektant dysponuje zbiorem wszystkich atrybutów potencjalnych relacji oraz informacjami o zależnościach między nimi. Teraz trzeba zaprojektować podział zbioru atrybutów na schematy relacji, tak by uzyskać pożądane właściwości b. d. Niewłaściwie zaprojektowany schemat relacji przechowywanych w b. d. jest przyczyną:

- dublowania się danych,
- niespójności danych,
- ..... podczas ich aktualizowania.

Takie zjawiska utrudniają lub uniemożliwiają poprawną eksploatację bazy danych. Trzeba wówczas najczęściej projektować bazę danych oraz programować aplikację rozpiętą na tej bazie danych. W ramach projektowania b. d. najistotniejszą czynnością jest normalizacji relacji. Normalizacja relacji polega na doprowadzeniu relacji do odpowiedniej postaci zwanej **postacią normalną.** Proces normalizacji polega na odpowiednim podziale relacji na mniejsze relacje gdy przechodzi się do wyższej postaci normalnej. Załóżymy, że mamy do dyspozycji relację w której każda wartość atrybutu w każdej krotce tej relacji jest wartością elementarną, czyli nierozkładalną. Powiemy, że taka relacja jest relacją w pierwszej postaci normalnej. Oznacza to, że nie ma w relacji dwóch identycznych krotek oraz, że każdy element relacji znajdujący się na przecięciu dowolnej krotki i dowolnego atrybutu jest pojedynczą wartością, a nie zbiorem wartości. można by powiedzieć, że pierwsza postać normalna jest podstawową cechą każdej relacji.
Relacja w pierwszej postaci normalnej jest jeszcze relacją, w której występują anomalie aktualizacji wyszukiwania danych.
Aby przedstawić drugą postać normalną należy wadzić kilka pojęć podstawowych, a mianowicie:
def. *Zbiorem identyfikacyjnym relacji nazywamy tak atrybutów tej relacji, że ich kombinacja wartości jest automatycznie identyfikuje każdą krotkę tej relacji.*
def. *Atrybut B relacji R jest funkcjonalnie zależny od atrybutu A tej relacji, jeżeli zawsze wartości a atrybutu A odpowiada nie więcej niż wartość b atrybutu B. (A→ B).*
Stwierdzenie, że B jest zależne funkcjonalnie od równoważne stwierdzeniu, że A identyfikuje B. Zależność funkcjonalna między dwoma atrybutami nie jest związana z przypadkowym układem w tych atrybutów, lecz wynika z charakteru zależności między tymi atrybutami nie jest związana z przypadkowym układem w tych atrybutów, lecz wynika z charakteru zależy m. tymi atrybutami. Dla wychwycenia istoty zależności funkcjonalnych rozważmy przykład relacji typu ZAMÓWIENIE. Relacja ta składa się:

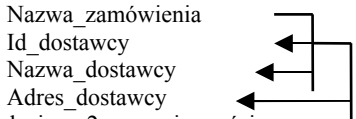


Z rysunku tego wynika, że Id_Dostawcy jest funkcjonalnie zależny od Nr_zamówienia, bo każdemu zw_zmówienia odpowiada nie więcej niż jeden dostawca (jest kierowane do jednego dostawcy), co oznacza, ze zw_zamówienia jednocześnie określa dostawcę.
Odwrotna zależność nie jest prawdziwa, gdyż jednemu dostawcy może odpowiadać wiele zamówień. Podobnie jest z pozostałymi atrybutami tej relacji. Dalej wprowadzimy kolejna definicję.
def. *Atrybut B relacji R jest w pełni funkcjonalnie zależny od zbioru atrybutów X, jeżeli funkcjonalnie zależny od żadnego zbioru X.*
W rozważonym przykładzie wszystkich zależności funkcjonalne są zależnościami pełnej funkcjonalnej umożliwia określenie drugiej postaci normalnej.
def. *Relacja R jest w drugiej postaci normalnej jest każdy atrybut tej relacji nie wchodzący w skład danego klucza potencjalnego jest w pełni zależności funkcjonalnie od wszystkich kluczy potencjalnych.*
W rozważanym [przykładzie kluczem potencjalnym Nr_zamówienia oraz Identyfikator części czyli Trybie Id_części. Atrybuty
Id_dostawcy

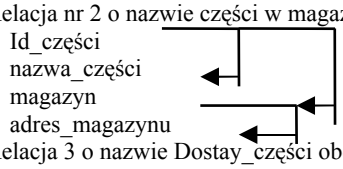
Nazwa_dostawcy
Adres_dostawcy
Nazwa_części

nie są w pełni zależne funkcjonalnie od klucze. W/w atrybuty są funkcjonalnie zależne od atrybutu Id_części, który jest również podzbiorem klucza. Oznacza to, że ta relacja nie jest w drugiej postaci normalnej. Aby drugą postać normalną należy podzielić całą relację na zbiór takich relacji, że wszystkie atrybuty tych relacji będą w pełni funkcjonalnie zależne od klucza. W rozważanym przypadku relacji wyjściową podzielić na 3 relacje:

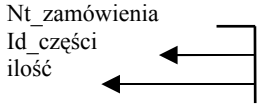
Relacja 1 o nazwie Dostawca_na_zamówieniu:



Relacja nr 2 o nazwie części w magazynie obejmuje następujące atrybuty:



Relacja 3 o nazwie Dostay_części obejmuje:



Relacja wynikowa Dostawca_na zamówieniu zawiera tylko atrybuty zależne od nr_zmówienia; dostawy_części zostały utworzone ze względu na atrybut „ilość", który jest funkcjonalnie zależny od Nr_zamówienia oraz Id_części. Wszystkie 3 relacje są w drugiej postaci normalnej, ponieważ ich klucze są pełnymi zależnościami funkcjonalnymi. Można uogólnić, że relacje, które są w pierwszej postaci normalnej (N1), których wszystkie klucze potencjalne są kluczami prostymi są w drugiej postaci normalnej (N2).

Analizując otrzymane wyniki należy zauważyć, że nadal zawierają one dublujące się dane. To dublowanie się danych wynika z tzw. przechodnich zależności funkcjonalnych między atrybutami. Normalizacja N2 nie usuwa zależności przechodnich funkcjonalnych, dlatego niezbędna jest trzecia postać normalna (N3).

def. *Niech X, Y i Z będą trzema rozłącznymi podzbiorami atrybutów danej relacji. Podzbiór atrybutów Z jest przechodnio funkcjonalnie zależny od podzbioru atrybutów X jeżeli podzbiór atrybutów Z jest funkcjonalnie zależny od Y a podzbiór Y jest funkcjonalnie zależny od podzbioru X, ale podzbiór X nie jest funkcjonalnie zależny od Yi Y nie jest funkcjonalnie zależny od Z, co można schematycznie zapisać*
$Y \rightarrow Z \wedge (Z \rightarrow Y) \wedge (Y \rightarrow X) \wedge (Z \rightarrow Y)$ *to* $X \dashrightarrow Z$ *gdzie* $\dashrightarrow$ *oznaczają przechodnią zależność funkcjonalną Z do X (linia przerywana).*

Przykładowo:
W relacji Dostawca_na_zamówieniu, mamy:
Id_dostawcy → Nazwa_dostawcy, Adres_dostawcy
Nr_zamówienia → Id_dostawcy
niezachodzi zależność funkcjonalna
Id_dostawcy - nr_zmówienia
bo do jednego dostawcy może być skierowane kilka zamówień oraz nie zachodzi relacja, że
nazwa_dostawcy - Id_dostawcy
adres_dostawcy - Id_dostawcy
bo kilku dostawców może mieć taką samą nazwę lub taki sam adres. Z powyższych zależności wynika, że Nr_zmówienia Nazwa_dostawcy, Adres_dostawcy.
W relacji Części_w_magazynie Adres_magazynu jest przechodnio- funkcjonalnie zależny od Id_części ponieważ
id_części → magazyn
magazyn → adrs_magazynu
adres_magazynu - magazyn
bo kilka magazynów może mieć ten sam adres
magazyn - id_części
bo kilka części jest co najmniej w jednym magazynie.
Zatem id_części adres_magazynu.

Korzystając z pojęcia przechodniej zależności funkcjonalnej można zdefiniować trzecią postaci normalnej (N3).

def. *Relacja jest w trzeciej postaci normalnej jeżeli jest ona N2 i każdy jej atrybut nie wchodzący do klucza potencjalnego nie jest przechodnio funkcjonalnie zależny od żadnego klucza potencjalnego tej relacji.*

Aby uzyskać N3 należy podzielić je na dwie relacje tak by wyrugować przechodnią funkcjonalną zależność.

Po normalizacji N3 mamy 6 relacji:
Zamówienie_do_dostawcy { nr_zamówienia, id_dostawcy }
dostawcy { id_dostawcy, dostawcy}
części_w_magazynie { id_części, magazyn }
części { id_części, nazwa_części }
magazyn {magazyn, adres_magazynu }

W praktyce może jednak występować potrzeba przeprowadzenia tzw. czwartej i piątej postaci normalizacji. Jednakże problemy związane z wykonaniem czwartej i piątej postaci normalizacji są bardziej subtelnej natury i rzadziej występują w praktyce. Część N4 normalizacji związana jest z wielowartościowymi zależnościami funkcjonalnymi, a N5 z połączeniowymi zależnościami funkcjonalnymi.

Przykład.

W bazie danych pamiętane są inf. o znajomości pracowników języków programowania i języków obcych.

nazwa_pracownika	język_programowania	język_obcy
Kowalski	Fortran	Angielski
Kowalski	Fortran	Francuski
Kowalski	Basic	Angielski
Kowalski	Basic	Francuski
Rowalski	Pascal	Angielski
Kowalski	Pascal	Francuski
Wałecki	Logo	Angielski
Wałecki	Logo	Włoski
Wałecki	Logo	Hiszpański
Wałecki	Pascal	Angielski
Wałecki	Pascal	Włoski
Wałecki	Pascal	Hiszpański
Nowak	PL/M	Angielski
Nowak	PL/m	Rosyjski
Nowak	C++	Angielski
Nowak	C++	Rosyjski

Występuje tu dublowanie się danych. Przyczyną dublowania się danych są tym razem wielowartościowe zależności funkcjonalne. Jedynej wartości atrybutu pracownika odpowiada wiele wartości atrybutu język_prog. oraz wiele atrybutu język_obcy. W omówionym przykładzie mamy do czynienia z dwoma wielowartościowymi zależnościami:

nazwisko_pracownika & język_prog

nazwisko_pracownika & język_obcy

Istotą N4 jest dopuszczenie tylko jednej wielowartościowej zależności funkcjonalnej. Doprowadzenie relacji wyjściowej do postaci N4 będzie polegało na rozkładzie tej relacji na dwie relacje zawierające odpowiednio nazwisko_pracownika oraz język_prog oraz drugą relację zawierającą nazwisko_pracownika - język_obcy.

def. *Niech R jest relacją a X iY oznaczają podzbiory atrybutów relacji R. Pod zbiór atrybutów Y jest wielowartościowo funkcjonalnie zależny od podzbioru X relacji R, jeżeli dla relacji R takich, że*

$k_1(x) = k_2(x)$ istnieje taka para krotek S_1 i s_2 taka, że

$s_1(x) = s_2(x) = k_1(x) = k_2(x)$

$s_1(y) = k_1(y)$

$s_1(R -X-Y) = k_1(R-X-Y)$

$s_2(Y) = k_2(Y)$ i $s_2(R-X-Y) = k_2(R-X-Y)$

Inaczej mówiąc, jeżeli dla dowolnej pary krotek k_1 i k_2 z relacji R, takich że wartości tych krotek dla atrybutów z podzbioru X są sobie równe, zamienimy na w tych krotkach wartości atrybutów z podzbioru Y, to otrzymamy w ten sposób krotki s_1 i s_2 również należące do relacji R

Przykład.

X ={ nazwisko_pracownika}

Y = { język_prog}

Wówczas R-X-Y = { język_obcy}Wówczas parze krotek

k_1 = { kowalski, fortran, angielski} - krotka 1

k_2 = {Kowalski, Pascal, francuski} - krotka 2

odpowiada para s_1, s_2 utworzona przez zmienną wartość atrybutu język_obcy będącą krótką 3 i 4.

def. *Trywialną wielowartościową zależnością funkcjonalną będziemy nazywać taki przypadek gdy $R= X \cup Y$ a X i Y są rozłącznymi podzbiorami relacji R.*

Niech X,Y Z będą niepustymi rozłącznymi podzbiorami atrybutów relacji R takimi, że

$R = X \cup Y \cup Z$

i Y jest nietrywialnie wielowartościowo zależny od X. Relacja R jest w 4 postaci normalnej gdy jest w N3 i wielowartościowa zależność Y i X pociąga za sobą funkcjonalną zależność wszystkich atrybutów tej relacji od X.

Innymi słowy, jeżeli w danej relacji występuje wielowartościowa zależność funkcjonalna atrybutów za zbioru Y od dowolnego podzbioru atrybutów X, to podzbiór X zawiera klucz relacji. Jeżeli w danej relacji postaci N3 występuje tylko trywialna zależność funkcjonalna to relacja ta jest w N4. W omawianym przykładzie występuje nietrywialna wielowartościowa zależność funkcjonalna zbioru X = { nazwisko_pracownika}} oraz zbiór y = { jezyk_prog} i podzbiór X nie zawiera klucza tej relacji. E celu doprowadzenia tej relacji do postaci N4 należy rozbić ją na:

{nazwisko_pracownika, język_prog}

{nazwisko_pracownika, język_obcy}

uzyskane relacje zawierają jedynie trywialne wielowartościowe zależności funkcjonalne i są w N4.

Piąta postać normalną relacji związana jest z tzw. **połączeniowymi zależnościami funkcjonalnymi.**

def. *Dekompozycją relacji R opartej na zbiorze atrybutów $\{A_1,. ...,A_n\}$ nazywamy zastąpienie jej zbiorem niekoniecznie rozłącznych relacji $\{R_1,. .., R_m\}$ takich, że R_i stanowi podzbiór atrybutów $\{ A_1,. ..., A_n \}$ i*

$$\bigcup_{i=1}^{m} R_i = R = \{A_1,....,A_n\}$$

def. *Mówimy, że w relacji R występuje połączeniowa zależność funkcjonalna co oznacza*

$*R [R_1,. .., R_m]$

wtedy i tylko wtedy gdy dla dowolnej relacji $r \subset R$ zachodzi

$r = \Pi_{R1}(r)$. ..... Π_{Rm}

gdzie Π oznacza operację projekcji (rzutowania) relacji r na podrelację R.

Inaczej mówiąc oznacza to, dekompozycję relacji na podrelację R_1,. ., R_m przez wykonanie sekwencji operacji a następnie można zrekonstruować relację pierwotną przez wykonanie sekwencji operacji łączenia relacji R_1,. ., R_m. Można pokazać, że wielowartościowa zależność funkcjonalna jest szczególnym przypadkiem połączeniowej zależności funkcjonalnej dla przypadku m=2.

JĘZYKI ZAPYTAŃ

Są to mechanizmy umożliwiające wyszukiwanie. do najważniejszych cech języka zapytań zaliczamy:

- siła ekspresji - określa zakres tego co w danym języku można zrobić;
- pełność - oznacza, że język zapytań musi mieć tę cechę, że w języku tum można zapisać taką kwarędę, która powoli odszukuje w b. d. dowolną informację jednostkową lub zestaw inf. jednostkowych w b. d. o ile ta dana się tam znajduje;
- rozszerzalność operacyjna języka - jt. wyposażenie języka zapytań w takie mechanizmy, które pozwalają:
  - zapisać różnorodne operacje arytmetyczne;
  - użycie funkcji agregujących; przydatne jest tu użycie operandów suma wartości atrybutów, wyznaczenie średniej itp.
  - możliwość korzystania z wyrażeń warunkowych,
  - możliwość nadawania nowopowstałym relacjom arbitralnych nazw,
  - prezentacja wskazanych relacji.
- prostota i łatwość języka zapytań;
- efektywność języka, dotyczy to po zadaniu pytania oraz dotyczy to również rozmiaru pamięci roboczej potrzebnej do obsługi kwarend.

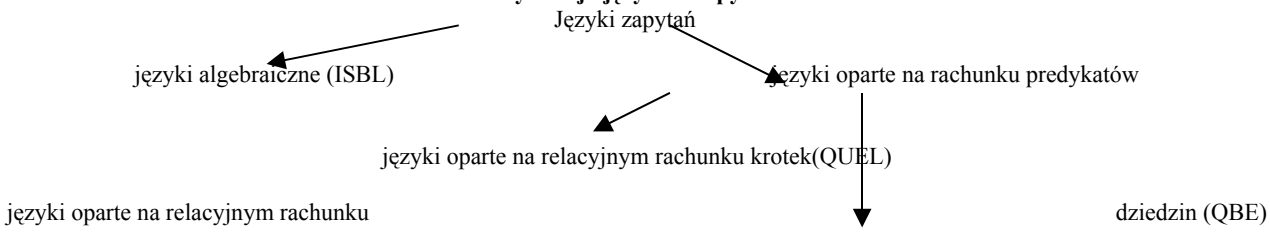
Cennym rozszerzeniem języka zapytań zwiększającym jego siłę ekspresji jest **zanurzenie tego języka w język programowania,** co pozwala profesjonalnemu użytkownikowi bazy dołączyć do zadań języka zapytań zapisanych w języku programowania.

Jednym z ważniejszych kryteriów języka zapytań jest proceduralność. W proceduralnych językach zapytań użytkownik buduje kwarędę w postaci procedury, którą można przedstawić jako ciąg kroków takich jak sekwencja, rzut, złączenie itd. Kwarenda jest tu zarówno opisem tego o co pytamy jak i metodą prowadzącą do uzyskania odpowiedzi.

Przeciwieństwem języków proceduralnych są **języki nieproceduralne.** Tu kwarenda zapisana jest żądaniem użytkownika. Każde jego żądanie składa się z dwóch części:

- opis żądanego schematu relacji wynikowej;
- opis warunków jakie muszą spełniać krotki tej relacji.

Klasyfikacja języków zapytań.



Języki oparte na rachunku predykatów to języki nieproceduralne w których kwarenda określa kształt relacji wynikowej oraz warunek, który musi być spełniony przez krotki składające się na relację wynikową.

Języki algebraiczne oparte są na algebrze relacji. Są to języki proceduralne. Języki oparte na rachunku krotek i dziedzin są językami **pełnymi.**

Przykładem języka opartego na algebrze relacji jest język ISB$. Przedstawicielem języka opartego na algebrze krotek jest język QUEL. QBE jest przedstawicielem języka opartego na relacyjnym rachunku dziedzin.

Kombinacja cech języka opartego na algebrze relacji i na algebrze krotek stanowi rozpowszechniony język zapytań SQL. Jest to najbardziej popularny język zapytań stosowany w relacyjnych b. d.

Język ISBL (Information System Base Language) jest językiem algebry relacji, do których zalicza się sumę, iloczyn i różnicę mnogościową oraz selekcję, projekcję i łączenie.

Język QUEL jest językiem opartym o relacyjny rachunek krotek. przedstawicielem systemu DBMS, który działa z takim językiem zapytań jest system INGRES. System ten działa w środowisku UNIX. Inspiracją dla tego języka zapytań był rachunek predykatów w abstrakcyjnym języku ALPHA zaproponowany przez Codda. Używa się tu fazy typu RANGE IS, która określa zmienna krotkę. Zmienna krokowa wskazuje te krotki, które należą do danej relacji i spełniają określone warunki. Używa się tu również fazy RETRIEVE INTO, która określa gdzie należy umieścić wyszukane krotki. W języku tym można zadawać pytania dotyczące dwóch i więcej relacji. W języku tym można również usuwać, dodawać, sortować krotki. W języku QUEL można używać również funkcje agregujące typu sumowanie, wyznaczanie wartości najmniejsze i największe. Można go zanurzać w procedurach pisanych w języku C.

Języki QBE (Query by Example) oparty jest na relacyjnym języku dziedzin. Język ten zyskał popularność ze względu na swój dialogowy i wizualny charakter. Jest to język oparty na relacyjnym rachunku dziedzin. Zakresem zmiennej jest wartość atrybutu a nie zbiór krotek. W tym języku można określić warunki wyszukiwania dla określonych atrybutów. Istotnym rozszerzeniem tego języka było wprowadzenie funkcji agregujących SUM, AVG, MAX, MIN, CNT. QBE jest wyposażony w mechanizmy typowe dla DML pozwalające zmienić zawartość tabel oraz mechanizmy pozwalające definiować nowe tablice, dodawać atrybuty lub je usuwać.

Język SQL jest językiem który łączy w sobie cechy języków algebraicznych i języków rachunku krotek. Język ten stał się standardem w system b. d. i obecnie jest używany w większości komercyjnych systemach b. d. Język ten zawiera w sobie część typu DML oraz DLL. Podstawową formułą:

SELCT R_{i1} A_1,....., R_{ik} A_k
FROM R_1. .... R_j
WHERE Φ;

W formule tej listy R_{i1}... określa atrybuty relacji wynikowej Φ zaś określa warunki wyszukiwania. Formuła ta może być osiągnięta z użyciem złożenia operacji relacyjnych projekcji z selekcją. SQL wyposażony jest w operatory agregacji. Ponadto ma on też dodatkowe mechanizmy do aktualizacji b. d., zmiany wartości atrybutów, dopisywanie krotek do relacji lub usunięcie ich. Typowymi zaś elementami dla DDL jest w standardzie SQL mechanizm definiowania schematów przy pomocy CREATE.

CO RÓŻNI SYSTEMY DBMS?

- model danych
  - utrzymują jeden model danych DBMSy = modelem sieciowym, hierarchicznym, relacyjnym =, obiektowym;
  - utrzymują wiele modeli danych (najczęściej dwa modele relacyjno-sieciowym, relacyjno-hirarchicznym);
- ilość użytkowników jednocześnie obsługiwanych
  - DBMSy dla jednego użytkownika
  - DBMSy dla wielu użytkowników - wielodostępne (wieloużytkowe).
- rozmieszczenie baz danych (fizyczne)
  - DBMSy umieszczone centralnie-globalnie w „host" komputerze lub na tzw. serwerze działające na zasadzie „klient - serwer"
  - DBMS, w których b. d. umieszczona jest w wielu miejscach - węzłach sieci (chociaż użytkownik niekoniecznie musi to wiedzieć) są to systemy rozproszonych b. d.
- ilość ostatnio utworzonych wersji b. d.
  - najczęściej systemy DBMS dokonują aktualizacji b. d. na bieżąco (pamiętana jest zawsze jedna wersja b. d.)
  - DBMSy welowersyjnymi bazami danych;
- zawartość b. d. - obok danych mogą być pamiętane reguły wnioskowania mówi się o bazach wiedzy systemy obsługujące bazy wiedzy to tzw. dedukcyjne b. d.;
- język zapytań użyty w DBMS-ie np. o bazie w systemie CODASYL;
- sposób zabezpieczenia bazy przed awarią;
- sposób obsługi transakcji, mówi się o systemach z blokowaniem relacji, krotek, z blokowaniem i dwufazową obsługą relacji;
- system operacyjny w środowisku, którego b. d. są implementowane.

Systemy DBMS początkowo tworzona na mini i duże komputery. Służyły one do obsługi dużych b. d. o wysokim stopniu niezawodności.

Nowe systemy operacyjne takie jak MS Windows NT, NetWare, OS/2 zapewniają już bezpieczeństwo, danych, oferują wysoką wydajność i możliwości stosunkowo łatwego korzystania z serwerów opartych na komputerach osobistych.

Test na niezawodność systemu: baza o zawartości 1GB, testowano tablice o ok. 1 mln. rekordów. Tylko takie narzędzia jak Oracle7 for NetWare i Sybase SQL for NetWare wersja 4.2 dają najbardziej zadawalające rezultaty testów. Taki system jak Gupta SQL Base Serwer for NetWare wersja 5.12 - będąca zgodnie z twierdzeniem firmy Gupta systemem obsługi b. d. mniejszych rozmiarów potrzebowała niewiarygodnie długiego czasu (60 godz.) do załadowania tablic a potem zawiesiła się na indeksowaniu.

Inny system taki jak INGRES Serwer for OS/2 wersja 6.4 spowodował w skutek błędu odłączenia klienta od serwera nie dał poprawnych testów badających tzw. wielodostęp.

Problemy z blokowaniem danych w teście wielodostępu wykazywał również system INFOMIX on line for NetWare wer. 4.1 powodując regularne zawieszanie się - powodując straty danych. Należy pamiętać, że testowanie wydajności systemów DBMS wykrywało problemy, które nie muszą wcale jawnie wystąpić w trakcie pracy w normalnym środowisku roboczym. Testy przeprowadzono w warunkach ekstremalnych. W końcu, po pewnym dostrojeniu i wyregulowaniu wszystkie produkty zdać mogą praktyczny egzamin poza systemami SQL base oraz INGRES Serwer.

Jednakże najbardziej niezawodnymi systemami okazały się Orecle7, Sybase, Serwer, Watcom SQL. Z testów również wynika, że takie systemy jak FoxPro lub Paradox przenoszące dane z serwera do klienta w celu ich przetwarzania są niewystarczające do tego by w systemach tych praktycznie realizować zadania o takiej skali złożoności. Przeprowadzając testy czasowe na ładowanie i indeksowanie b. d. u klienta uzyskano następujące wyniki:

Oracle7 - 3 jednostki czasu
Watcom SQL - 8
Ingras - 9
Sybase - 12
IBM DB2/2 - 17
Informix - 36

Podczas testu odczytów losowych - najlepsze wyniki daje Sybase. na pozycji 2 - Oracle7, dalej Informix, DB2/2 oraz WatCom. Podobnie w teście zapisu losowego. W teście zapytań ad hoc (zestaw mieszanych zapytań) najlepiej wypadł Oracle7 - trzy krotnie szybszy niż pozostałe. W teście mieszanym który obejmował: pracę bez interwencji innych, równoczesny odczyt losowy i zapis najlepsze wyniki dawał Oracl7 oraz Sbase.

Narzędzia do strojenia b. d.

Większość systemów b. d. typu SQL daje się elastycznie konfigurować, pozwalając administratorowi zmieniać wszystko. Tylko Oracl7 i Informix zawierają narzędzia do monitorowania b. d.

Systemami dającymi dobre rezultaty szczególnie dla małych i średnich aplikacji są systemy Gupta, FoxPro, Paradox.

Przeglądanie wybranych systemów zarządzania b. d. z punktu widzenia ich stosowalności do rozproszonych b. d.

Cechy systemu Oracle:

- współbieżna praca wielu użytkowników,
- blokowanie na poziomie rekordów, plików, blokady wew., blokada schematu b. d. (słowników),
- odporność na awarie,
- brak mechanizmów sprawdzania integralności danych - na poziomie pisania aplikacji należy pamiętać o zachowaniu tej integralności,
- umożliwia komunikację między różnymi protokółami komunikacyjnymi;
- umożliwia tworzenie zintegrowanych otwartych systemów rozproszonych b. d. rozmieszczonych na komputerach o różnej architekturze i różnych systemach operacyjnych,

- w zakresie osiągów użytkownicy oceniają system jako przeciętny nawet słaby w przypadku przetwarzania transakcji pracujących w trybie On-line na sprzęcie IBM; dobrze oceniana jest współbieżność, gorzej funkcjonalność b. d.

Cechy systemu Progress4:

- służy do działań w rozproszonym i niejednorodnym środowisku systemowym, może pracować praktycznie w każdym środowisku sieciowym,
- ma możliwości pracy wielowątkowej i wieloserwerowej,
- możliwe jest blokowanie danych na poziomie rekordów i plików,
- integralność danych jest uzyskiwana poprzez blokadę określonych operacji (z poziomu aplikacji), spójność logiczną na poziomie schematu b. d.,
- system umożliwia zabezpieczenie obrazu końcowego (więcej niż jedna wersja pliku końcowego),
- ma możliwość składania całej bazy lub tylko jej części bez przerywania pracy systemu lub z przerwaniem.

Uwagi o systemie Informix:

- nadaje się do pracy ze standardowymi protokółami sieci lokalnych,
- ma mechanizmy blokowania rekordów, plików i całych baz danych,
- aktualizacja b. d. odbywa się bez tworzenia kopii aktualnego fragmentu, co zwiększa czas trwania blokady i obniża efektywność systemu,
- utrzymuje plik transakcyjny na wypadek odtwarzania stanu bazy danych,
- tworzenie kopii transakcyjnych jest możliwe po włączeniu współbieżnego przetwarzania b. d. a więc praktycznie po unieruchomieniu wszystkich aplikacji na b. d.

Cechy systemu Ingress:

- różnorodne środowisko operacyjne wieloprocesorowe i sieciowe;
- wieloserwerowa i wielowątkowa architektura;
- możliwość tworzenia i zarządzania b. wiedzy;
- możliwość sterowania dostępem do obiektowych b. d.,
- praca z rozproszonymi b. oraz praca z b. d. innych systemów,
- zastosowano w tym systemie przy realizacji odwołań do bazy danych optymalizator oparty na bazie analizy kosztów i wykorzystano statystykę w słownikach danych,
- integralność danych zapewniona jest poprzez metodę składania danych, system blokad oraz system rejestracji użytkowników,
- system zawiera tylko mechanizmy blokowania dwufazowego, brak jest blokad na poziomie rekordów,
- efektywność dostępu do danych jest przez użytkowników oceniona na dobrą przy małych b. d. i średnią dla dużych systemów, wysoko oceniona jest funkcjonalność systemu, przenośność jako dobra.

INDEKSOWANIE

Podstawową techniką stosowaną w systemach DBMS z zakresu wyszukiwania danych i języków zapytań jest technika indeksowania. B. d. indeksuje się jednym lub wieloma kluczami. Dla każdego pojedynczego lub zagregowanego klucza określa się zbiory indeksowe pojedynczego lub zagregowanego indeksu. Niektóre zbiory b. d. przechowywane są w formie zindeksowanej dla b. d. mogą być tworzone zbiory indeksowe czasowo, w dowolnym momencie gdyż są one technikami użytecznymi. W niektórych systemach pamięć zajmowana przez indeksy jest > niż pamięć zajmowana przez dane. Ważne jest by organizacja indeksu była najefektywniejsza. Projekt musi dążyć do tego by organizacja indeksów była jak najefektywniejsza tzn. powinien dążyć do zminimalizowania obszaru pamięci zajmowanego przez indeksy oraz czasu potrzebnego do ich poszukiwania. Indeks powinien być tak zaprojektowany by możliwe było dołączenie i usunięcie pozycji, oraz by utrzymywanie indeksów nie było czasochłonne.

Indeks jest tablicą na której wykonuje się operacje przeglądania. Jesteśmy zainteresowani czasem przeszukiwania indeksów. Pole użyte do szukania nazywa się **argumentem**, a pole które otrzymamy z tablicy nazywa się **funkcją.** Z jednym argumentem może być powiązana jedna lub wiele funkcji.

Funkcja indeksu b. d. może przyjąć jedną z następujących postaci:

- adres rekordu - w większości przypadków indeks podaje maszynowy adres rekordu; często zależnie od rekordu wymaga przeszukiwania kilku poziomów indeksów;
- względny adres rekordu - stało się to ważne z chwilą upowszechnienia techniki stronicowania
- symboliczny adres rekordu - jeżeli indeks podaje pozycję określonych rekordów to jest on w efekcie tablicą wskaźników, które mogą być adresami maszynowymi, względnymi lub symbolicznymi. W indeksach wtórnych i skorowidzach używa się niekiedy wskaźnika symbolicznego w celu oddzielenia struktury indeksu wtórnego od fizycznego rozmieszczenia danych. Indeks wtórny wiąże bezpośrednio z kluczami wtórnymi. Gdy utworzony zostanie zbiór indeksów dla takiego klucza może on wskazywać np.: na pierwsze takie wystąpienie w b. d. dla którego określony klucz ma taką a nie inna wartość. Zbiory indeksów wtórnych mogą być czasami b. duże i przepisywanie ich po każdej reorganizacji pliku byłoby czasochłonne. Dlatego też indeks może podawać klucz główny szukanego rekordu za pomocą dowolnej metody adresowania pliku będzie się znajdowało żądany rekord. Używanie indeksów symbolicznych wydłuża czas potrzebny do znalezienia rekordu, lecz ułatwia utrzymanie indeksu.
- umiejscowienie porcji - niektóre indeksy nie wskazują pojedynczego rekordu lecz obszar, w którym mieści się większa liczba rekordów. Obszar ten zwany niekiedy porcją może być ścieżka dysku lub obszarem kontrolowanym lub obszarem dobranym jako odpowiedni z punktu widzenia zastosowanej techniki indeksowania. Po uzyskaniu dostępu do porcji trzeba porcję te przeszukać aby odnaleźć wymagany rekord. Liczba rekordów w porcji, na które wskazuje indeks, charakteryzuje rozdzielczość indeksu. Indeks wskazujący pojedyncze porcje określa się mianem **indeksu porcji**. Indeks wskazujący pojedyncze rekordy nazywa się **Indeksem rekordu.** Indeks porcji jest mniejszy i jego przeglądanie trwa krócej. Indeks porcji może być używany wtedy jeśli rekordy mogą być grupowane w tych samych porcjach oraz wtedy gdy zapis rekordów odbywa się sekwencyjnie a nie uporządkowaniem losowym. Wskaźniki zapisane w indeksie porcji mogą być adresami: maszynowymi, względnymi, symbolicznymi. Symbolicznym wskaźnikiem indeksu może być klucz główny pierwszego rekordu tej porcji.
- adres łańcucha - gdy używane jest w b. d. łączenie łańcuchowe

- wartości atrybutów - niektóre indeksy wtórne podają zmiany adresów wartości atrybutów. W wielu przypadkach pozwala to udzielić odpowiedzi na zapytania dotyczące kluczy wtórnych bez konieczności przeglądania rekordów danych. Wartość atrybutu dostarczana przez indeks wtórny po dodatkowych operacjach może być przekształcona w adres rekordu danych. Niekiedy indeks składa się z wierszy kluczy, z których jeden będący kluczem głównym pozwala poprzez dalsze operacje adresowania odnaleźć rekord danych. W niektórych systemach różnica między rekordami indeksu a rekordami danych staje się nieuchwytna. Rekordy indeksu zawierają dane, które nie muszą być powtarzane w rekordach danych.
- wyjście wielokrotne - każdy zapis w indeksie głównym podaje pojedynczy wskaźnik. Zapis może określać wiele wskaźników. Mogą one wskazywać rekordy porcje lub fragmenty łańcucha, będące adresami maszynowymi, względnymi, symbolicznymi lub wartościami atrybutów.

Jeżeli zapis w indeksie może przyjmować jedną zwartości skończonego zbioru, to przedstawiciele każdej wartości zapisu w postaci binarnej pozwala zaoszczędzić obszar pamięci. Ograniczeniem przechowywania zapisów w formie liczb binarnych jest to, że potrzeba dodatkowej tablicy w celu zapamiętania wartości w pierwotnej formie. Jeśli tablica wartości jest mała to może być przechowywana w pamięci operacyjnej lub w ostateczności może być dostępna poprzez operację stronicowania. W niektórych indeksach każdy argument może mieć wiele f-cji ze stronicowego zbioru funkcji. Konieczna jest wtedy minimalizacja liczby bitów koniecznych do zapamiętania tych f-cji. Są tu duże metody ich zapamiętania.

Indeks gęsty to taki indeks, który przyjmuje wszystkie możliwe wartości klucza. Indeksy wtórne są na ogół gęste, tzn. zawierają wszystkie możliwe wartości klucza głównego. Indeks główny nie musi być indeksem gęstym jeśli rekordy ułożone są w sekwencji swoich kluczy głównych; indeks może wskazać ścieżkę lub obszar pliku, który musi być przeszukany; w tym przypadku indeks nazywa się **indeksem rzadkim.** Za pomocą indeksu znaleźć można tylko konkretnie wybrane wartości. Indeks o takich cechach nazywa się **indeksem znaczonym.**

Wiele atrybutów indeksowych przyjmuje wartości dyskretne. Niekiedy **rozkład wartości atrybutów** jest ciągły. Wtedy rozkład wartości może zostać podzielony na odpowiednie odcinki, podobnie jak podziałka linijki. Operacje taką nazywa się **kwantyzację**. Redukuje ona liczbę wymaganych zapisów w indeksie. Argument indeksu może się odnosić do konkretnej wartości lub przedziału wartości. Taki indeks nazywa się **indeksem zakresu.**

Podczas projektowania plików wielokluczowych, przeznaczony do obsługi zapytań spontanicznych projektant musi postawić pytanie: dla których argumentów tworzyć indeks. Jednocześnie pożądane jest, by indeks nie był zbyt obszerny. Projektant może preferować indeksowanie atrybutów mających mały zbiór wartości, gdyż wówczas lista będzie składać się z małej liczby zapisów. Niestety atrybuty o małej liczbie wartości nie są zbyt selektywne. Np. w bibliotece wartość atrybutu „fantastyka" może się odnosić do dużej części książek, a tym samym indeks tej wartości jest w procesie indeksowania mało pomocy. Odwrotnie atrybuty dostarczając małą ilość pozycji na zapytanie będzie prawdopodobnie miał dużą liczbę indeksowanych wartości. Wyjściem z tego dylematu jest łącznie w indeksie pewnych atrybutów, pod kątem ich łącznego występowania w kryteriach przeszukiwania. Z powodu wielkości indeksów, w szczególności indeksów wtórnych ważne są techniki **kompresji** zapisów w indeksie. Wielkość niektórych indeksów ulegnie zmniejszeniu, jeśli przedstawi się je w postaci **binarnej**. Inne można zmniejszyć przez usunięcie nieznaczących części kluczy. Przykładem takiego postępowania jest odcięcie końcowej części klucza w **indeksie wielopoziomowym**.

Do zlokalizowania rekordu konieczne jest by najbardziej znaczące znaki jego klucza różniły się od najwyższego klucza z następnej porcji.

Jeszcze większą kompresję można uzyskać za pomocą techniki zwanej **rozbiorem.**

Przy rozbiorze najwyższy poziom indeksu zawiera część klucza. Następny poziom indeksu zawiera jeden lub kilka następnych znaków klucza, jednak bez powtarzania tych, które już zostały zapisane na najwyższym poziomie. Trzy najwyższe poziomie indeksu składają się z zapisów jednoliterowych. Odpowiadają one trzem pierwszym literom. Czwarty poziom nie powtarza już tych trzech liter. Nie trzeba żadnych dodatkowych oznaczeń wskazujących początek i koniec zapisu. Jeśli porcje wskazywane przez czwarty poziom indeksu miałby zmienną pojemność, to ten właśnie poziomu indeksu mógłby mieć stałą długość. Wadą tego sposobu jest to, że niektóre bloki indeksu zawierać będzie więcej zapisów niż dziesięciozapisowe bloki w poprzedniej technice. W tym przypadku gdy kliki tworzą trzy poziomowe indeksy, to mogą być przeszukiwanie binarne. W tej technice może być opłacane dodanie dodatkowego poziomu indeksu, składającego się z zapisu jednoliterowego, pomiędzy poziomu 3 i 4. Zaleta metody może być to, że plik może być bardzo zmienny, a potrzeba będzie bardzo niewiele zmian w trzech górnych poziomach indeksu. Mechanizm dołączenia i usuwania elementów zb. indeksowego koncentrują się przede wszystkim od poziomu 4.

TECHNIKI PRZESZUKIWANIA INDEKSÓW.

Indeks b. d. to również pliki z danymi. Wyróżnia go prostota i możliwość większej kontroli. Zakres stosowania techniki kompresji danych i optymalizacji jest w indeksie znacznie większy niż w głównych plikach b. d. Musi być on tak zorganizowany, by można było dołączyć i usuwać pozycje lub by utrzymywanie pliku indeksowego nie było pracochłonne.

Podstawowym problemem jest wybór odpowiedniej metody szybkiego określenia pozycji na podstawie indeksu w zb. podstawowym b. d. Może koniecznym być czytanie kilku pozycji w zbiorze indeksów by określić pozycję w zbiorze podstawowym. Czytanie jednej pozycji indeksu nazywać będzie próbą. W obliczeniach na podstawie których opracowano wykres założono, że prawdopodobieństwo szukania dowolnej pozycji jest dal wszystkich pozycji równe.

Niech:

N_I - liczba zapisów;

N_P - liczba prób podczas przeszukiwania indeksu.

Efektywność technik wyszukiwania i przeszukiwania indeksu określa stosunek:

średnia liczba prób/liczba zapisów w indeksie = $E(N_P)/N_I$

Nie jest to jednak pełna ocena efektywności przeszukiwania, ponieważ różne próby wymagają różnej ilości czasu, a różnie w czasie zależeć będą od zastosowanego sprzętu, pojemności pamięci operacyjnej, itp. Dla zlokalizowania pozycji w indeksie (zbiorze indeksowym) stosuje się następujące techniki:

1. Przeglądanie indeksu - sekwencyjne.
2. Przeszukiwanie blokowe.
3. Przeszukiwanie binarne.
4. Przeszukiwanie drzewa binarnego.
5. Przeszukiwanie indeksu w postaci zrównoważonego drzewa z seryjnym przeglądaniem węzłów.
6. Przeszukiwanie indeksu w postaci zrównoważonego drzewa z blokowym przeszukiwaniem węzłów.
7. Przeszukiwanie indeksu w postaci zrównoważonego drzewa z binarnym przeszukiwaniem węzłów.

8. Technika z operacyjnym zbiorem wskaźników.
9. Technika bufora śladowego.
10. Techniki oparte o drzewa niezrównoważone.
11. Indeksowanie algorytmiczne
12. Indeksowanie za pomocą kodowania rozpraszającego.

Można wykazać, że dla przeszukiwania sekwencyjnego zbioru indeksowego średnia liczba prób wynosi

$E(N_P) = (N_I + 1)/2$

Przy przeszukiwaniu blokowym,

$E(N_P) \approx \sqrt{N_I}$

Przeszukiwanie blokowe rzadko stosuje się do całego indeksu, lecz jest to ważna technika dla przeszukiwania wycinak indeksu. Optymalny rozmiar bloków wynosi

$N_B = \sqrt{N_P}$.

Plik b. d. składa się z rekordów uporządkowanych wg. klucza K. Szukamy w zbiorze indeksowym wartości klucza K_S. Rekordy w bazie indeksów grupuje się w bloki o wielkości N_B rekordów. W momencie natrafienia na rekord o numerze klucza większym od zadanego następuje dokładne przeszukanie ostatnio opuszczonej grupy obejmującej odpowiednią ilość rekordów - rekordy grupowane są w bloki.

Przeszukiwanie binarne.

W tym przypadku zaczynamy od środka obszaru przeszukiwanego i porównujemy klucz. Następnie obszar przeszukiwania dzielimy na dwie części i powtarzamy proces. Średnia liczba prób w przypadku dużego N_I może być aproksymowana przez liczbę $\log_2 N_I - 1$. Jest to liczba prób znacznie mniejsza niż w przypadku poszukiwania seryjnego lub blokowego. Im większy jest indeks tym większa jest liczba prób. Niestety wiele indeksów z którymi mamy do czynienia praktycznie nie mieści się w całości w pamięci głównej: przechowuje się je w pamięciach masowych. Ta technika jest niedogodna, ponieważ mechanizm dostępu jest nieustannie przesuwany.

Przeszukiwanie binarne ma jeszcze inną wadę. Najczęściej takie binarne przeszukiwanie wyklucza użycie technik kompresji indeksu, ponieważ usuwają one części klucza, które są wymagane w całości w przekształceniach binarnych, a kompresja jest wartościową techniką przechowywania większości indeksów. Najczęściej projektant technik indeksowych dokonuje wyboru między kompresją a przeszukiwaniem binarnym bloków w pamięci operacyjnej. Inną wadą tej metody jest trudność w dołączeniu i kasowaniu pozycji. Konieczne będzie tu przesuwanie pozycji zapisanych wcześniej. Można tego uniknąć jeśli zastosuje się przeszukiwanie drzewa binarnego. Jeżeli pozycje, które będą przeszukiwane w pliku indeksowym ułożone są w formie drzewa binarnego połączonego wskaźnikami, to możliwe jest dołączenie nowych pozycji bez przemieszczania istniejących już pozycji. Obszar do zapisu drzewa binarnego jest przypuszczalny np. na 16 pozycji. Aktualnie w przykładzie zapisano 12 pozycji. Każda z pozycji ma dwa krótkie wskaźniki do innych miejsc w bloku. Pierwszą zapisaną pozycją jest pozycja od której rozpoczyna się, które będą przeszukiwane w drugiej kolejności. Nie potrzeba obliczać, która pozycja ma być sprawdzona jako następna. Kolejność sprawdzania określa jeden z dwóch wskaźników umieszczonych w każdej pozycji. Wskaźniki są śledzone aż do momentu znalezienia własnej pozycji. Średnia liczba prób jest taka sama jak w tradycyjnym przeszukiwaniu binarnym. Elementy drzewa binarnego mogą być ułożone w różnej sekwencji. Wskaźniki w zapisie drzewa mogą być reprezentowane przez jeden bit, 4 itp.

Dołączenie nowej pozycji polega na zapisanie tej pozycji jako element ostatni, wraz z aktualizacją wskaźnika dla elementu I - który ma pusty wskaźnik (tzw. szczelinę). Usunięcie elementu - to zaznacza się tylko, że pozycja jest formalnie usunięta. Dołączenie dalszych pozycji po zapełnieniu bloku, wymagać będzie jego reorganizacja a nawet podziału bloku można przeprowadzić korzystając ze wskaźników nadmiarowych lub rozproszonych obszarów wolnych.

Jednobitowy wskaźnik może być tylko wtedy, gdy pozycje w bloku mają stałą długości. Jeśli długość pozycji jest stała, to kompresja nie może być zastosowana w obrębie bloku. Przy pozycjach o zmiennej długości można zastosować kompresję przedniej części, ponieważ zapis nie musi powtarzać znaków które są identyczne jak w zapisie wskazywanym.

Jeśli w drzewie binarnych odłączymy i usuwamy wiele pozycji, to czas potrzeby do jego przeszukiwania wzrasta. Jeżeli zmiany te mają charakter ciągły i losowy, to średnia liczba prób potrzebna do przeszukiwania drzewa wynosi 1.386 ln N, gdzie N - ilość pozycji w drzewie. Jeśli drzewo zostanie zreorganizowane to czas przeszukiwania zmniejszania się do liczby ln N-1 prób.

Wadą przeszukiwania binarnego z postacią drzewa jest to, że w każdy zapisie potrzebna jest dodatkowa pamięć na wskaźniki oraz że nie można zastosować głębszej kompresji. Jeśli pamięć zajmowana przez indeks musi być mała, to preferowane będzie przeszukiwania blokowe wykorzystujące rozbiór. przeszukiwanie drzewa binarnego jest również nieodpowiednie, jeśli wymagane jest występowanie pozycji w kolejności klucza.

Większość indeksów zbudowana jest z więcej niż jednego poziomu ma postać drzewa.

Jako pierwszy przeszukiwany jest węzeł z najwyższego poziomu wskazując na węzeł na następnym poziomie. Węzeł tego poziomu jest poszukiwany i w wyniku tego wskazuje węzeł z poziomu jest przeszukiwany i w wyniku tego wskazuje węzeł z poziomu najniższego. Przeszukanie dowolnego węzła może być dokonane za pomocą którejkolwiek z wcześniej omówionych technik - seryjnej, binarnej, drzewa binarnego.

Indeks w postaci takiego drzewa zajmuje więcej miejsca niż indeks jednopoziomowy, ponieważ największy (lub najniższy) klucz w każdym węźle powtarzany jest w kolejnym, wyższym węźle. indeks taki może być przeszukiwany szybciej niż mający postać drzewa z mniejszą liczbą odwołań do pamięci zewnętrznej. Istotne jest dobranie optymalnej liczby poziomów drzewa oraz optymalnej ilości zapisów w węźle.

Niech

N_B - ilość zapisów indeksowych w węźle;
L - liczba poziomów;
N_I - liczba zapisów w indeksie;
to

$$L = \left[\frac{\ln N_I}{\ln N_B} \right]$$

$$E(N_P) = L \frac{N_B + 1}{2}$$

$$\frac{E(N_P)}{\ln N_I}$$

By zminimalizować, czyli zwiększyć efektywność techniki należy N_B dobrać stosunkowo małe.

Blok złożony z 20-30 pozycji daje zadawalający kompromis między oszczędnością pamięci zajmowanej przez indeks a minimalizacją liczby prób. W praktyce liczba zapisów w bloku może być dostosowana do sprzętu.

Jeżeli czynnik czasu jest czynnikiem nadrzędnym to przeszukiwanie binarne indeksu dopasowanego do bloku w takim sposób by uniknąć ruchu głowicy odczyt zapis jest najlepszą, techniką przeglądania tablic. Proste przeszukiwanie binarne nie może być zastosowane z kompresją zapisów w indeksie co ma zasadniczy wpływ na oszczędności pamięci zajmowanej przez indeks. dołączenie nowej pozycji do bloku, który będzie przeszukiwany binarnie (bez drzewa binarnego) jest wygodne, ponieważ wymaga przesuwania pozycji zapisanych wcześniej. Z tych to powodów w wielu pakietach b. d. stosuje się raczej wielopoziomowe przeszukiwanie binarnego indeksu nie jest dużo mniejsza od liczby prób stosowanej w przeszukiwaniu blokowym tego indeksu przedstawionego w postaci drzewa.

Dolny poziom indeksów w formie drzewa (poziom 1) nazywa się **zbiorem wskaźników.** Zapisy argumenty w tym zbiorze ułożone są w ścisłej sekwencji. Operacje indeksowe mogą być przeprowadzone albo idąc od korzenia drzewa w dół do zbioru wskaźników lub od razu na poziomie zbioru wskaźników. Miejsce od którego należy zacząć przeszukiwanie tego zbioru; można określić za pomocą dowolnej techniki i metod, które teraz omawiamy. Zbiór wskaźników jest używany przy bezpośrednim dostępie do pliku sekwencyjnego (liczba prób dla trzech typów indeksów postaci drzewa).

Przy przeglądaniu zbioru wskaźników stosuje się łańcuchy jednokierunkowe lub dwukierunkowe, pozwalające przeglądać zbiór wskaźników od przodu i do tyłu. Zbiór wskaźników indeksu może być sam zorganizowany jak plik indeksowo-sekwencyjny. Jego organizacja musi uwzględnić dogodne metody dołączenia i usuwania rekordów. Można w tym celu wykorzystać **rozproszone obszary wolne,** do których dopisuje się nowe indeksy. Te obszary wolne są tworzone dla pewnej liczby nowych indeksów; można również pozostawić bloki wolne. Indeksy do obszaru wolnego można zapisów w dwojaki sposób:

- można zapisać indeksy bez przeszukiwania,
- z chwilą dołączenia nowego indeksu, indeksy już zapamiętane zostają przesunięte tak, że grupa jest uporządkowana wg. klucza - w tym przypadku dołączenie wymaga nieco więcej czasu, ale skraca czas czytania.

Jeżeli czytanie indeksów występuje częściej niż dołączanie to ten drugi sposób jest lepszy. używa się dwóch rodzajów rozproszonych obszarów wolnych. po pierwsze każdy **przedział kontrolny** nie jest całkowicie zapełniony podczas tworzenia pliku. Po drugie, niektóre przedziały kontrolne w obrębie obszaru kontrolnego w całości pozostaje puste. Każdy element zbioru wskaźników wskazuje jeden przedział kontrolny, który jest pusty lub nie.

BUFOR ŚLADOWY

W wielu systemach odwołań do pliku grupują się często wokół określonych rekordów. Gdy odwołujemy się do rekordu o określonym kluczu, prawdopodobne jest, że poprzednie odwołanie dotyczyło rekordu o tym samym lub zbliżonym kluczu. W niektórych przypadkach właściwość tę powoduje przetwarzanie po prostu pewna grupa rekordów jest przetwarzana znacznie częściej niż pozostałe. I tak np. w systemie rezerwacji lotniczej odwołania do rekordów dotyczących najbliższych lotów są najczęstsze. W systemie hurtowni większość zapytań dotyczyć będzie pewnej grupy poszukiwanych towarów, lecz w następnym okresie zapotrzebowanie może się zmienić. Bez względu na przyczyny należy tę wiadomość uwzględnić podczas organizacji plików. Gdy stosowana jest technika stronicowania, algorytm wymiany stron może pozostawić najczęściej wywołane pozycje w pamięci głównej lub innej pamięci o szybkim dostępie. Przy stosowaniu techniki indeksowej w pamięci głównej numerów ostatnio używanych indeksów. Bufor śladowy dla indeksu drzewiastego może obsługiwać kilka lub wszystkie poziomy drzewa. W celu odszukania rekordu sprawdzany jest najpierw bufor śladowy aby stwierdzić czy zadany zapis indeksu jest w nim umieszczony.

(rys. bufory śladowe)

Bufory śladowe dla trzech poziomów znajdujących się w pamięci głównej. Określono zapotrzebowanie np. na rek. 721. Sprawdza się bufor śladowy dla pierwszego poziomu i znajduje się tam zapis wskazujący odpowiedzi blok w zbiorze wskaźników. Z kolei żądany jest rekord 801. W pamięci głównej brak takiego zapisu w buforze pierwszego poziomu i dlatego przechodzi do sprawdzania bufora poziomu drugiego. Bufor ten zawiera zapis wskazujący żądany blok; znajduje się on na drugim poziomie indeksu.

Jest wiele sposobów organizowania buforów śladowych. W szczególności istnieje dużo algorytmów wspomagających określenie, które zapisy będą przechowywane w buforach. W buforach należy przechowywać najczęściej ostatnio używane zapisy indeksów. W buforach śladowych mogą być zapisane najczęściej używane zapisy w ciągu określonego okresu (miesiąca roku, itp.). Można też w buforach śladowych zapisywać indeksów. Efektywność algorytmów śladowych jest zmienna i zależy od systemu i sposobu użytkowania pliku.

Drzewa niezrównoważone.

Dotychczas rozważaliśmy tzw. drzewa zrównoważone indeksów. Charakteryzowały się one równą liczbą zapisów w blokach. Struktury drzewiaste mogą nie być zrównoważone. Jedna z wad przypisywanych drzewom zrównoważonym i indeksom w postaci tych drzew jest konieczność czytania wszystkich poziomów indeksów w celu uzyskania dostępu do rekordu. Rekordy czytane często lub takie do których jest szybki dostęp nie są traktowane w sposób uprzywilejowany. Preferencje dla tych rekordów można uzyskać w indeksie o strukturze drzewiastej niezrównoważonej. (rys 30.9 Niezrównoważone drzewo indeksowe). Niewielką liczbę rekordów można tylko odszukać z najwyższego poziomu drzewa. Większą liczbę rekordów można znaleźć korzystając z dwóch poziomów drzewa. Dotarcie do większości rekordów wymaga sprawdzenia wszystkich poziomów. Najbardziej popularną pozycją w pliku z równoważnego przykładu jest BETTY i JUNE. Indeksy dla tych przypadków znajdują się w górnej części drzewa. Dalej umieszczone są następne pod względem popularności imiona. Częstość odwołania się do poszczególnych pozycji może się zmieniać w czasie. Jeśli b. d. ma być reorganizowana na skutek popularności imion, to zmianę tę na skutek spadku popularności imion, to zmianę tę można dokonać w indeksie bez przemieszczania rekordów danych. Na kolejnym rys 30.10 pozycja najczęściej poszukiwanymi stały się np. CHLOE i SCARLETT i dlatego ich zapisy indeksowe umieszczono na szczycie drzewa. Pozostałe zapisy w indeksie są również odpowiednio przedstawione. Wadą drzewa niezrównoważonego jest to, że wymaga większej liczby poziomów

niż drzewo zrównoważone. Rekordy, które na rys 30.10 wymagały 4 poziomów indeksu, wymagały tylko 3 gdyby użyte zostało drzewo zrównoważone z taką samą liczbą zapisów w węzłach. Jednak okazuje się, że nawet dla bardzo dużych plików liczba poziomów indeksu nie przekracza 5 lub 6, a więc niezbyt wiele czasu można zaoszczędzić stosując drzewa niezrównoważone chyba, że krótsza droga dostępu może ograniczać fizyczny dostęp, który byłby konieczny w przeciwnym razie.

Szacunkowe obliczania adresu.

Przeszukiwanie indeksu może być skrócone, jeśli potrafimy w przybliżeniu określić pozycję zapisu w indeksie. Jeśli ten szacunek będzie odpowiednio dokładny i ograniczy przeszukiwanie do 1% indeksów, to zmniejszenie liczby prób może być znaczące. Możliwość przeprowadzenia takiego szacunku zależy w znacznym stopniu od natury zastosowania. Tablica z następnej strony dotyczy znajdywania nazwisk poindeksowanych w porządku alfabetycznym. W drugiej kolumnie tej tabeli informuje jaką część indeksu licząc od początku można pominąć, przy czym ta kolumna odnosi się do pierwszej litery nazwiska. Trzecia kolumna tej tabeli ma podobną interpretację do kolumny drugiej ale odnosi się do drugiej litery nazwiska. Pozycje zaczynające się na K będą znajdować się w odległości równej 0,4554 od początku indeksu. Pozycje zaczynające się na KE będą się znajdować w odległości 0,4554 + 0,1613 (0,4968 - 0,4554) długości całego indeksu. Znajdując liczbę zapisów w indeksie można oszacować wielkość żądanego zapisu indeksu.

Indeksowanie algorytmiczne.

Oparte jest ono o różne algorytmy wykorzystujące sposób adresowania pliku. Stwarza to możliwość szybkiego dotarcia do żądanych rekordów, szybciej niż indeksowanie.

	Pierwsza litera	Druga litera		Pierwsza litera	Druga litera
A	0	0	N	0,6323	0,5294
B	0,500	0,0781	O	0,6608	0,6022
C	0,1285	0,0909	P	0,6753	0,6843
D	0,2005	0,1202	Q	0,7220	0,7058
E	0,2425	0,1613	R	0,7237	0,7072
F	0,2661	0,2918	S	0,7952	0,7736
G	0,3108	0,3206	T	0,8833	0,8382
H	0,3688	0,3345	U	0,9167	0,9284
I	0,4220	0,3930	V	0,9269	0,9561
J	0,4333	0,4607	W	0,9403	0,9661
K	0,4554	0,4630	X	0,9871	0,9810
L	0,4968	0,4672	Y	0,9871	0,9840
M	0,5495	0,5032	Z	0,9925	0,9991

w postaci drzewa, ponieważ dostęp do bloku indeksu uzyskuje się poprzez pojedyncze szukanie - odpowiada to jednemu poziomowi drzewa. Z drugiej strony ta technika nie jest tak szybka jak adresowanie bezpośrednie bez indeksu.

Kombinacje algorytmu i indeksu:

- metoda może wykorzystywać wiele algorytmów nie dość precyzyjnych do wskazywania rekordu lecz mogą one zlokalizować indeksową grupę rekordu;
- metoda eliminuje główną wadę adresowania bezpośredniego - tj. brak niezależności między fizycznym lokowaniem rekordu w pliku a jego kluczem logicznym. Fizycznej pozycji rekordu nie można zmieniać niezależnie od algorytmu adresującego.

Przykładem indeksowania algorytmicznego jest lokalizacja rekordów nazwisk pasażerów, w systemie rezerwacji pasażera i mają długość kilkaset znaków. Do rekordów tych trzeba się szybko odwoływać mieć szybki dostęp. Plik jest duży i szybko zmienny. Ze względu na tę zmienność potrzebny jest schemat adresowania szybszy niż indeks w postaci drzewa. Można tu przyjąć algorytm, który bazuje na prostych przeliczeniach numeru lotu i daty oraz tablicy wskaźników zamieszczonej w pamięci operacyjnej, można określić jednoznaczny adres w pliku rekordu tego lotu. Rekord ten zajmuje stałe miejsce w części pamięci o bardzo szybkim dostępie. Składa się on z indeksu lub zbioru wskaźników wskazujących rekordy nazwisk pasażerów tego lotu, które ze względu na swą zmienność, mogą być umieszczane w dowolnym miejscu pamięci. W tym przykładzie indeks zapisany jest w segmentach, między rekordami danych. Taki indeks często nazywa się **osadzonym.** Zmienne listy wskaźników, służące do reprezentacji struktur drzewiastych i sieciowych są przykładem struktur osadzonych.

Indeks za pomocą kodowania rozpraszającego.

Kodowanie to używa się najczęściej do bezpośredniej lokalizacji rekordu, za pomocą tego kodowania można lokalizować blok indeksu eliminując potrzebę posiadania wielopoziomowych indeksów. Kombinacja indeksowania i kodowania rozpraszającego może być lepsza niż samo indeksowanie ponieważ wymaga mniejszej liczby prób lub mniejszej liczby dostępów. Główna zaleta kodowania rozpraszającego, którą jest znajdywania rekordów w wyniku jednego szukania jest utracona. Jednak indeksowanie kodem rozpraszającym może być bardziej zalecane niż adresowanie wyłączenia kodem rozpraszającym, ponieważ pozwala przechowywać rekordu w kolejności wynikającej z innych uwarunkowań. Rekordy można przechowywać np. w porządku klucza głównego lub razem z ich poprzednikami w strukturze drzewiastej. Te rekordy, które są najczęściej używane mogą być umieszczane w obszarach pamięci, do której dostęp jest najszybszy w niektórych przypadkach cenną własnością może być możliwość zmiany rozmieszczenia rekordów, bez zmiany algorytmu kodowania rozpraszającego. Co więcej puste obszary w porcjach charakterystyczne dla kodowania rozpraszającego są obszarami w indeksie a nie obszarami w danych. **Indeksowanie kodowania rozpraszających daje mniejsze straty pamięci** niż adresowanie rekordów czystą techniką kodowania rozpraszającego. Rekordy nadmiarowe, które występują w kodowaniu rozpraszającym i nie są do uniknięcia mogą być utrzymywane w indeksie a nie w rekordach danych gdzie powodują czasochłonność szukania. Rozmiary porcji mogą być wystarczająco duże by liczba rekordów nadmiarowych była mała.

Jak rozmieszczać bloki indeksu?

Gdy używa się indeksów w postaci drzewa, to jest wiele sposobów w jaki mogą być rozmieszczone bloki. Możemy rozważyć ten problem np. na przykładzie trójpoziomowego indeksu.

SYSTEMY PLIKÓW ODWRÓCONYCH

W systemie plików odwróconych rekordy adresowane są zwykle na podstawie danych z b. d. czyli realizacji zapytań czyli realizacji zapytań użytkownika istnieje konieczność tworzenia wielu indeksów tzw. wtórnych i wówczas przestrzeń zajmowana przez indeksy może okazać się większa od przestrzeni zajmowanej przez dane. Aby osiągnąć zadawalający czas odpowiedzi, ważne jest by umożliwić przeszukiwanie plików bez śledzenia łańcuchów lub wskaźników umieszczonych w pierścieniach w danych bezpośrednio umieszczonych. Można wyróżnić 3 odmiany systemów wykorzystujących indeksy odwrócone:

- systemy z indeksem wtórnym,
- systemy z plikami częściowo odwróconymi,
- systemy z plikami całkowicie odwróconymi,

Systemy z indeksem wtórnym.

Rekordy lub segmenty danych w b. d. są sekwencyjnie uporządkowane na podstawie klucza głównego, a indeksy wtórne projektuje się dla określonych typów zastosowań.

Systemy z plikami częściowo odwróconymi - w tym przypadku rekordy lub segmenty danych w b. d. mogą tworzyć zupełnie dowolną sekwencję. Nie występuje tu indeks główny tylko jedynie indeksy wtórne. Dołączenie nowych rekordów do pliku nie stanowi problemu. Można je dopisywać w dowolnym miejscu. Natomiast indeksy wtórne muszą być uaktualniane. Fizyczny adres rekordu podawany jest bezpośrednio przez indeks wtórny, bez odsyłacza do klucza głównego.

System plików całkowicie odwrócony - można tworzyć pliki, w których wartości tworzące określony rekord nie muszą znajdować się fizycznie obok siebie. Aby stwierdzić wystąpienia określonej wartości danej bada się indeks wystąpień a w celu odszukania danych związanych z tym wystąpieniem trzeba wprowadzać oddzielny indeks danych.

Czterema zasadniczymi elementami systemów plików całkowicie odwróconych są:

- słownik;
- indeks wystąpień;
- indeks danych;
- wartości danych.

Do wykazania odpowiedniego miejsca w indeksie wystąpień służy słownik. W indeksie wystąpień odszukuje się poprzez słownik listę wystąpień. Listę tę sprawdza się i porównuje po to by indeks danych znaleźć żądane wartości danych. Przykładową strukturę plików odwróconych przedstawia następujący rys. (ksero - orangutan). Dialog sterujący poszukiwaniem systemu plików odwróconych może przebiegać w trzech etapach. Najpierw sprawdza się słownik a następnie indeks wystąpień. indeks wystąpień może być przeszukiwany wielokrotnie przed dotarciem do indeksu danych. System plików całkowicie odwróconych ma szczególne zastosowanie w konwersacyjnym przeszukiwaniu b. d. w przypadku konwencjonalnego przetwarzania danych, np. przy tworzeniu listy płac, gdzie z góry określona grupa danych elementarnych jest razem odczytywana, ten system może się okazać wolny i nie wygodny.

Ważna kategorią systemów plików odwróconych jest system wyszukiwania dokumentów lub system wyszukiwania tekstu. Na rysunku przedstawiono typowa strukturę pliku w interakcyjnym systemie wyszukiwania dokumentu. Jest to struktura zgodna z systemem STAIRS firmy IBM. Po prawej stronie rysunku przedstawiono plik dokumentów. Dokumenty są numerowane poprzez system. Każdy dokument podzielony jest na paragrafy i zadania, które są też numerowane. Każdy paragraf ma kod określający jego typ np. „tytuł", „autor", „streszczenie", „tekst", lub inne. Po lewej stronie rys. znajduje się słownik wyrazów. Podaj e on w jęz. ang. lub innym wyrazy występujące w dokumentach. Słownik wyrazów może być obszerny i może wymagać indeksu wyższego poziomu. Tu wykorzystano macierz 2-znakowa. Każda para znaków ma znacznik do bloku słownika zawierającego wyrazy zaczynające się od tych dwóch znaków. Znakami tu mogą być: litery, cyfry, znaki specjalne. Drugi znakiem może być również spacja. liczba zapisów tej macierzy = 37*38 = 1406. Słownik zawiera zapisy o zmiennej długości. Każdy zapis w grupie przeznaczony jest dla wyrazów zaczynających się od dwóch takich samych znaków (pierwsze dwa znaki nie są powtarzane w zapisie słownika). Niektóre słowa w słowniku mogą mieć identyczne znaczenie i wówczas łączy je wskaźnik synonimów (linia przerywana na rys.). Każdy zapis w słowniku zawiera wskaźnik do listy wystąpień, określającej każde wystąpienie danego wyrazu w pliku dokumentów. Lista wystąpień ma pola nagłówkowe. Pola te podają liczbę dokumentów zawierających dany wyraz oraz całkowitą ilość wystąpień wyrazu w obrębie pliku dokumentów. Liczby te są podawane użytkownikowi, przeprowadzającemu wyszukiwanie. Jeśli liczby te są zbyt duże, to użytkownik może zrezygnować z danego wyrazu i znalezienia odpowiedniejszego, jako podstawy wyszukiwania. System przydziela wszystkim dokumentom jednoznaczny nr, rozmiarze tak małym jak to jest możliwe nie jest to związane z jakimkolwiek nr zew., za pomocą którego użytkownik może się odwołać do dokumentu. Lista wystąpień zawiera ten wew. nr dokumentu dla każdego wystąpienia. Gdy wyszukiwanie ma polegać na znalezieniu wszystkich dokumentów, zawierających np. dwa określone wyrazy stosuje się tzw. „scalanie list" tworząc najpierw listę dla jednego oraz oddzielna listę dla drugiego wyrazu. Istnieje wiele innych kryteriów wyszukiwania. Przyjmuje się, że dwa lub więcej muszą występować obok siebie, w określonym kontekście. Z tego powodu w liście w wystąpień zapisuje się pozycje wyrazów w zdaniach. Użytkownik może zawężać poszukiwania. Może również określić, że poszukiwane słowo znajduje się w: "tytule", „nazwisku autora" itp. Lista wystąpień mogła by wskazywać dokumenty w sposób bezpośredni ale taki wskaźnik zawierałby więcej bajtów niż krótki wew. nr dokumentu. Listy zawierające setki tys. dokumentów scalane są w procesie wyszukiwania dlatego należy używać jak najkrótszych nr Z tego powodu indeksy dokumentów przechowuje się oddzielnie. Indeks dokumentu może zawierać obok adresu dokumentu jego zew. nr, znaki kasowania (cały lub część dokumentu usuniętą), kod tajności - dostępu do dokumentu. Wyszukiwanie zaczyna się od słownika i nie może być ono kontynuowane dopóki nie zmodyfikował użytkownik kryterium wyszukiwania. Dalej poprzez indeks wystąpień kończy się przeglądaniem dokumentu. To następstwo poziomów wyszukiwania sprzyja trybowi konwersacyjnemu. Kluczem do osiągnięcia sukcesów w projektowaniu operacji na plikach odwróconych jest dobranie właściwej struktury następstwa poziomów wyszukiwania, ograniczając je tam gdzie to jest możliwe do sekcji pamięci szybszego dostępu. Ważną sprawą w tej tech. jest również właściwy dobór słów do słownika. Częstym błędem jest umieszczanie w słowniku zbyt wielu słów lub stosowaniem słów które są zbyt powszechne. Gdy słowo będzie powszechne, to lista wystąpień będzie długa, a potrzebne w tym przypadku operacje scalania zbyt czasochłonne. Użycie nieodpowiednich zapisów słownika zmniejsza wydajność systemu.

Niektóre systemy plików odwróconych ograniczają się tylko do struktur danych w postaci drzewa (struktur hierarchicznych). W wyszukiwaniu dokumentów mamy do obróbki słownik składający się ze słów zapisanych w dokumentach, natomiast w uniwersalnych b. d. wykorzystuje się wiele typów danych elementarnych. Wymaga to utrzymywania słowników typu tych danych.

Pierwszym ważnym systemem zarządzania plikami odwróconymi dla interakcyjnych zastosowań był TDMS (Timeshared Data Management System) - system zarządzania danymi z podziałem czasu, opracowany przez system Development Corporation (późniejsze CSDC). Sys-

tem TDMS może udzielać odpowiedzi na większość wielokluczowych zapytań, zadawanych przez użytkowników końcowych. W rozwiązaniach TDMS wykorzystuje się tech. częściowego lub całkowitego odwrócenia.

Przykład.

Zapisy uwidocznione na rys. odpowiadają zaetykietowanym blokom z rys. poprzedniego. Przechowywane dane ujęto w dwóch kolumnach po prawej stronie rys.

Jeżeli długość żądanej danych elementarnych nie przekroczyła 4B, to wszystkie dane elementarne mogły by być zapisane w tablicy danych elementarnych. W przeciwnym przypadku, wszystkie wartości danych elementarnych których długość przekracza 4B zapisuje się w oddzielnej tablicy wartości. Powodem takiego postępowania jest to że w większości danych ta sama wartość danej elementarnej występuje wielokrotnie. Jeśli wartość ta jest „długa" to można zaoszczędzić miejsce w pamięci zapisując ją teraz raz i stosując odpowiedni wskaźnik. Takie postępowanie czasami jest opłacalne a czasami nie. I tak np. wartość LONDON w pliku byłaby wskazywana wielokrotnie i dlatego jej umieszczenie w oddzielnej tablicy zaoszczędzi miejsce w pamięci.

Natomiast wartość '128 MARINE PARADE SOUTH' prawdopodobnie pojawi się raz a więc zapisanie je w oddzielnej tablicy poza rekordem, którego jest częścią byłoby stratą przestrzeni i czasu zapisy w „tablicy danych elementarnych" oraz w „tablicy wartości" są rozmieszczane w kolejności w jakiej się pojawiły. Nowe zapisy dołącza się na koniec pliku. Aktualizacja tablic indeksowych jest jednak operacją złożoną. Tablica znajdująca się na rys. drugim jest słownikiem danych elementarnych który definiuje typ danych. Tablica ta podaje nazwy każdego typu danych elementarnych przynależność do określonego poziomu w strukturze drzewiastej oraz jego nr wew. (nr pola). Jest ona indeksowana za pomocą nr pola. Słownik ten może również zawierać inne inf. nie pokazane na rys. 2 taki jak synonimy, typ i max rozmiar danych elementarnych. Niekiedy znajdują się tam inf. obejmujące kontrolę zastrzeżonego dostępu do danych. W TDMS słownik danych elementarnych zawiera również dla każdego typu danych elementarnych, wskaźnik do tablicy zgodności, ujmującej zapisane wartości danych elementarnych. W niektórych przypadkach lista wartości w tablicy zgodności jest kwantyfikowalną. I tak np. nie zapisuje się każdej możliwej wartości LICZBA POKOI; w sekwencji rosnącej przechowuje się np. co piątą możliwą wartość. W przypadkach „długich" danych elementarnych nie zapisuje się całej wartości lecz tylko wyróżniona ilość znaków (na rys. 2. 5 znaków) i wskaźnik do tablicy wartości, w którą ujęta jest cała wartość. Jeżeli dwie różne wartości mają takie same znaki, to zapisuje się wskaźnik duplikacji. W tablicy zgodności liczba zapisów dla jednego typu danych elementarnych może być tak duża, że przeszukiwanie binarne (podział na dwa) byłoby opóźnione z powodu ciągłego przesuwania się ramienia mechanizmu dostępu.

Dlatego do tablic zgodności wykorzystuje się indeks wyższego poziomu w hierarchii. Daje to możliwości wczytania do pamięci operacyjnej bloków zgodności wystarczająco małych aby można je przeszukiwać binarnie. Tablica zgodności zawiera wskaźnik do listy wystąpień, w której ujęto wszystkie wystąpienia dla każdej danej elementarnej. Tablica zgodności jest wiec indeksem listy wystąpień i spełnia tę samą rolę co słownik wyrazów w systemie STAIRS. Jeśli wartość danej elementarnej występuje tylko raz, to nie pojawia się ona w liście wystąpień. W tym przypadku tablica zgodności w zamian zawiera wskaźnik do następnej tablicy np. do skorowidza. Każda pozycja na liście wystąpień zawiera wskaźnik do skorowidza. Skorowidz wykorzystuje wskaźnik na poprzednik i na rekord podobny oraz wskaźnik na grupę powtórzeń, które wespół z ułożeniem sekwencyjnym wskazują powiązania w strukturze drzewiastej oraz pozycje w tablicy danych elementarnych. Zapisy w skorowidzu ułożone są w sekwencji drzewa typu góra - dół - lewy - prawy.

Tablice znajdujące się po lewej stronie tablicy danych elementarnych są więc łączone, aby utworzyć odpowiedni zbiór indeksów odwróconych oraz skorowidz do danych elementarnych.

Przykład użycia systemu TDMS.

Droga oznaczona na rys. 2 pokazuje kroki podejmowane dla udzielenia odpowiedzi na zapytanie:

„W JAKICH MIASTACH ZNAJDUJĄ SIĘ HOTELE O STANDARDZIE ***** I LICZBIE POKOI > 140 ORAZ JAKIE SĄ NAZWY TYCH HOTELI."

Zapytanie to można w języku TDMS zapisać w postaci:

PRINT NAZWA HOTELU AND MIASTO WHERE STANDARD EQ 5 GWIAZDEK AND LICZBA POKOI GR 140.

Udzielenie odpowiedzi rozpoczyna się od słownik danych elementarnych. Znajduje się w nim

NAZWĘ HOTELU
MIASTO
STANDARD
LICZBĘ POKOI

Osoba zapytująca może skorzystać z innej nazwy, np. POJEMNOŚĆ HOTELU bowiem słownik zawiera wszystkie dopuszczalne synonimy nazw danych elementarnych. Słownik wskazuje także, że

NAZWĘ HOTELU
STANDARD
LICZBĘ POKOI

są umieszczone w rekordzie tego samego typu (poziom 3) a rekord MIAST jest w strukturze drzewiastej ich poprzednikiem. Słownik zawiera wskaźnik do indeksu tablicy zgodności. Znajdują się w nim zapisy odpowiadające wartości STANDARD = 5 Gwiazdek, Liczba pokoi > 140. Tablica zgodności zawiera wskaźniki do listy wystąpień, jest tych wystąpień wiele; tyle ile jest takich hoteli. W nagłówku każdej listy wystąpień jest zapis, który określa liczbę pozycji na tej liście. Jeśli lista jest b. długa to użytkownik zostanie poinformowany o tym i zostanie mu postawione pytanie czy pragnie wykonywać (kontynuować) wyszukiwanie. W celu znalezienia identycznych zapisów scalane są listy wystąpień. W naszym przykładzie zapis 605 pojawi się w obu listach (ze względu na standard i ilości pokoi). 605 jest wew. nr zapisu w skorowidzu. Zapis ten wskazuje pozycję NAZWA HOTELU w tablicy danych elementarnych, która z kolei określa wartość umieszczoną w tablicy wartości. Zapis 605 umieszczony w skorowidzu wskazuje również rekordy poprzednika, który sam zawiera wskaźnik do tablicy danych elementarnych. W ten sposób odszukuje się zapis MIAST. Dowolne dane elementarne związane z hotelem lub miejscowością można znaleźć na podstawie skorowidza i tablicy danych elementarnych.

We wszystkich systemach plików odwróconych list wystąpień poważnych problemem jest scalanie list. Niekiedy listy scalane mogą być długie. Konieczne jest wiec takie projektowanie list wystąpień by zajmowały jak najmniej przestrzeni. Dlatego listy wystąpień oddzielone są od pozostałych tablic i zawierają zawarte wskaźniki binarne do środowiska. Gdy wartości elementarne się zmieniają to musimy dołączyć do listy wystąpień nowy rekord lub usunąć stary. Dlatego listy wystąpień są zmienne, nawet gdy plik nie zmienia swoich rozmiarów. Dla bardzo zmiennego pliku utrzymywanie tablic zgodności skorowidza i listy wystąpień staje się złożonym problemem.

Utrzymywanie takich list wymagać będzie również częstych mniej lub bardziej operacji sortowania, oraz zmiany dużej ilości wskaźników. Dlatego w przypadku gdy struktura odwrócona aktualizowana jest w czasie rzeczywistym, to w tablicach należy utrzymywać rozproszone obszary wolne o dużych rozmiarach lub utrzymywać odpowiednio duże obszary nadmiarowe. Ze względu na duży koszt i złożo-

ność operacji utrzymywania plików odwróconych nie próbuje się wprowadzać „najświeższych danych”. Są one aktualizowane w trybie „off-line” i jest to możliwe niezbyt często. System może być aktualizowany okresowo danymi gromadzonymi w tym celu przez system operacyjny. Struktury odwrócone, aczkolwiek doskonałe dla pewnych zastosowań, daleki są od ideałów. Są zbyt słabe dla przetwarzania wsadowego. W systemie STAIRS korzysta się np. z prostej tech. opóźniania rekonstrukcji tablic. Okresowo, dołącz się do systemu nową porcję dokumentów z tych nowych dokumentów tworzy się odwrócone pliki odrębne. Te nowe odrębne tablice - pliki dołącza się do już istniejących.

SYSTEMY SZYBKICH ODPOWIEDZI

Systemy pracujące w czasie rzeczywistym powinny być tak zaprojektowanie aby szybko udzielał odpowiedzi na zapytanie. Szybkość systemu warunkują:

- natura zapytań i zastosowań,
- dobrany sprzęt i środki tech.

Wymóg szybkich dostępów, szybkich odpowiedzi na zapytania wpływa na projektowanie:

- systemu komunikacji - teleprzetwarzania,
- systemu b. d.

Wymóg maksymalizacji prędkości dostępów jest tu znaczący. udzielenie odpowiedzi często wymaga dostępu do pliku b. d. lub też dostępu do wielu rekordów. Zatem czasy dostępów do elementów b. d. muszą być krótkie. Potrzeba szybkiego dostępu nabiera szczególnego znaczenia w systemach wielotransakcyjnych, wielodostępnych. W takich systemach trzeba korzystać szczególnie szybkich tech. Tech. szybkiego wyszukiwania to takie tech. fizycznej organizacji pamięci, w których wszystkie dostępy do pojedynczego rekordu odbywają się poprzez jego klucz główny. o wiele trudniej znaleźć jest tech. szybkiego wyszukiwania, jeśli przeszukiwanie pliku lub dostęp do tego pliku odbywa się przez wiele kluczy. Aktualnie bardzo szybka „maszyna poszukiwania” czeka do zbudowania choć w tym względzie ostatnio poczyniono pewien sposób stosując tzw. pamięć asocjacyjną oraz chodźby częściowe rozwiązania hardware’owe, gdzie buduje się „wszyte” algorytmy wyszukiwania danych. Ogólnie, tech. pozwalające na osiągnięcie krótkich odpowiedzi są kombinacja następujących sposobów:

- Gromadzenia często wywoływanych danych na szybkich urządzeniach;
- lokowania danych tak, by tam gdzie jest to możliwe, eliminować długie wyszukiwania;
- doboru schematów adresowania i przeszukiwania tak aby eliminować „długie” wyszukiwania;
- stosowania wielu operacji równolegle (współbieżność).

W poniższej tabeli przedstawiono wcześniej rozważane techniki z uwagami które z nich przyczyniają się do osiągnięcia szybkich czasów odpowiedzi, czy też nie.

Techniki nieprzydatne	techniki szybkie
łańcuch osadzone i pierścieniowe	powiązania oddzielone od danych - struktury list odwróconych
Indeksy projektowane w celu optymalizacji pamięci (lub zmniejszania kosztów pamięci)	indeksy w pamięci operacyjnej adresowanie algorytmiczne lub rozpraszające
przeszukiwanie rozległych sekcji	przeszukiwanie binarne w pamięci operacyjnej
pojedyncza obsługa kolejek w kanale lub mechanizmie dostępu	łożenie plików pozwalające na równoczesne wykonanie wielu operacji
organizacja seryjna	organizacja równoległa
organizacje niezależne od komputera	organizacje dopasowane do sprzętu
organizacje niezależne od zastosowań	organizacje dopasowane do zastosowań
wymóg podobnego zachowania dla wszystkich odpowiedzi	dialog użytkownik - komputer dopasowany do organizacji pamięci w celu ograniczenia skutków czasochłonnych operacji
aktualizacja w czasie rzeczywistym	aktualizacja odkładana okres późniejszy
dołączenie i usuwanie w czasie rzeczywistym	dołączenie odłożone i przeprowadzone w trybie of-line
niezbyt dobre techniki dla dołączenia nowych rekordów (zwłaszcza w przypadku dołączeń grupowych)	b. d. oddzielona od siebie w zakresie systemu informacyjnego i operacyjnego
	Powielanie (redundancja) pewnych pozycji w celu minimalizacji czasu odpowiedzi
	organizacje adaptacyjne
	obsługa dialogu z wykorzystaniem peryferiałów
	sprzęt zaprojektowany pod kątem przetwarzania równoległego

Wymienione tutaj techniki są często dobre również dla plików dynamicznych tj. takich w których dołączenie nowych i usuwanie starych rekordów jest bardzo częste (szybko zmienne). Jak wcześniej omówiono, nie możliwe jest osiągnięcie krótkich czasów odpowiedzi jeśli dane w b. d. są przechowywane w postaci długich łańcuchów rekordów, a między operacjami czytania tych rekordów są operacje szukania (łańcuch osadzone). Zamiast używać wskaźników osadzonych lepiej jest zapisywać oddzielnie powiązania m. danymi miejscu, w którym mogą być szybko dostępne. Dla wyszukiwania wielokluczowego używa się plików z listami odwróconymi zamiast plików łańcuchowych. Struktury list odwróconych wymagają zwykle więcej pamięci i należy dokonać rozsądnego rozstrzygnięcia m. szybkością wyszukiwania a wykorzystaniem pamięci.

Wskaźniki osadzone w strukturach łańcuchowych lub pierścieniowych wskazują na wzajemne powiązanie rekordów w strukturze i są umieszczane zawsze z danymi. Pełnią rolę wskazań na poprzedniki, następniki, obszary nadmiarowe, itp. Dla wyszukiwania wielokluczowego stosuje się raczej pliki z listami odwróconymi zamiast plików łańcuchowych.

Indeksy powinny być tak projektowane by zajmowały minimalnie pamięć oraz czas wyszukiwania minimalizowały.

PROBLEM NORMALIZACJI

Początkowo, w procesie projektowania bazy danych dysponuje się zbiorem wszystkich atrybutów potencjalnych relacji oraz informacjami o zależnościach między nimi. Trzeba zaprojektować podział zbioru atrybutów na schematy relacji, tak by uzyskać pożądane właściwości bazy danych. Niewłaściwie zaprojektowany schemat relacji przechowywanych w bazie jest przyczyną dublowania oraz niespójności danych.

Takie zjawiska utrudniają lub uniemożliwiają poprawną eksploatację bazy danych. Trzeba wówczas najczęściej przeprojektować bazę danych oraz aplikację rozpiętą na tej bazie danych. W ramach projektowania bazy danych najistotniejszą czynnością jest normalizacji relacji. Normalizacja relacji polega na doprowadzeniu relacji do odpowiedniej postaci zwanej *postacią normalną***.** Proces normalizacji polega na odpowiednim podziale relacji na mniejsze relacje gdy przechodzi się do wyższej postaci normalnej. Do omówienia czwartej i piątej postaci normalnej celowe wydaje się przedstawienie pierwszej, drugiej i trzeciej postaci normalnej wraz z pojęciami z nimi związanymi.

Cele normalizacji.

Normalizacja relacji ma na celu takie jej przekształcenie, by nie posiadała ona cech niepożądanych. Weźmy dla przykładu relację opisującą zajęcia odbywające się na uczelni w jednym semestrze. Relacja ta może zawierać następujące atrybuty:

- nazwa przedmiotu
- imię
- nazwisko
- adres prowadzącego

Przykładowa krotka takiej relacji mogłaby mieć postać:
(język niemiecki, Elżbieta, Kowalska, ul. Kasprowicza 25/1 Gorzów Wlkp.)

Jednak z tak zaprojektowaną relacją związane są następujące problemy:

- adres składający się z kilku części nie został podzielony w związku z tym nie jest możliwe sprawdzenie ile osób mieszka w Gorzowie i nie potrzebuje hotelu;
- jeden prowadzący może mieć zajęcia z kilku przedmiotów, w związku z tym występuje redundancja danych;
- zmiana jednej z informacji o prowadzącym (np. adresu) powoduje konieczność zmiany wszystkich krotek zawierających te dane w celu zachowania integralności;
- nie jest możliwe wprowadzenie informacji o prowadzącym, który w aktualnym semestrze nie ma żadnych zajęć;
- usunięcie przedmiotu może spowodować również usunięcie wszelkich informacji o osobie, która go prowadziła

Utrzymanie integralności takiej bazy nie jest więc proste. Jednak opisaną relację można zamienić na dwie inne, które nie będą posiadały tych wad:

- relacja Prowadzący: (identyfikator, imię, nazwisko, kod pocztowy, miejscowość, ulica, nr_domu, nr_mieszkania)
- relacja Zajęcia (nazwa, id_prowadzącego)

Każda krotka relacji "Zajęcia", jest powiązana z krotką relacji "Prowadzący" za pomocą klucza obcego o nazwie "id_prowadzącego". Te dwie relacje nie posiadają opisanych wcześniej cech niepożądanych ponieważ:

- adres jest zdekomponowany na części składowe, w związku z czym możliwe jest wyszukiwanie danych np. według miejscowości zamieszkania prowadzącego;
- zmiana informacji o prowadzącym (np. adresu) nie powoduje konieczności zmian danych w relacji "Zajęcia". Zmiana ta odbywa się tylko w jednym miejscu;
- możliwe jest wprowadzenie informacji o osobie, która nie ma zajęć w aktualnym semestrze, ale być może będzie je miała w semestrze następnym;
- usunięcie przedmiotu nie powoduje usunięcia informacji o osobie, która go prowadziła;

Jednak taka reprezentacja danych posiada wady podobne do opisanych wcześniej, ale dotyczące przedmiotów. Dlatego w dobrze zaprojektowanej bazie danych konieczne jest wydzielenie trzeciej tabeli, która będzie zawierała spis przedmiotów.

Przekształcenie relacji do kolejnych postaci normalnych wiąże się najczęściej ze zmniejszeniem ilości pamięci potrzebnej do przechowania informacji.

Proces normalizacji ma na celu takie przekształcenie relacji, by uniknąć dublowania informacji. Unikanie powtórzeń pozwala na łatwiejsze i w wielu przypadkach szybsze posługiwanie się bazą danych.

Definicje pomocnicze.

Aby ułatwić przekształcanie relacji do postaci optymalnej wprowadzono pojęcie postaci normalnej. Przed omówieniem procesu normalizacji konieczne jest jednak wprowadzenie kilku pojęć:

- *Uniwersalny schemat relacji* $R = \{A_1, A_2, .., A_n\}$ jest zbiorem atrybutów tworzących relację.
- ***Zbiorem identyfikującym relacji*** $R = \{A_1, A_2, .., A_n\}$ nazywamy zbiór atrybutów S R, który jednoznacznie identyfikuje wszystkie krotki relacji o schemacie R. Inaczej mówiąc w żadnej relacji o schemacie R nie mogą istnieć dwie krotki t_1 i t_2 takie, że $t_1[S] = t_2[S]$.
- *Kluczem* K schematu relacji R nazywamy minimalny zbiór identyfikujący, tzn. taki, że nie istnieje K' K będące zbiorem identyfikującym schematu R. Klucze dzielą się na klucze proste i złożone.
- Klucz nazywamy ***kluczem prostym***, jeżeli zbiór atrybutów wchodzących w jego skład jest zbiorem jednoelementowym; w przeciwnym wypadku mamy do czynienia z *kluczem złożonym*. Najczęściej w relacji można wyróżnić wiele kluczy, które nazywamy *kluczami potencjalnymi*. Jeden (wybrany) klucz spośród kluczy potencjalnych nazywamy *kluczem głównym* (primary key), natomiast pozostałe kluczami drugorzędnymi (secondary key).

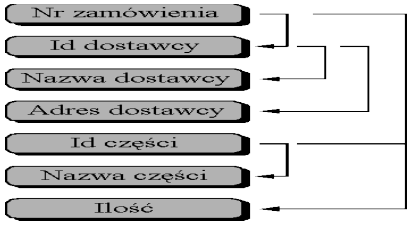
Dla przykładu w relacji "Zamówienia" jedynym kluczem potencjalnym jest para atrybutów (Nr zamówienia, Id części). Należy zauważyć, że sam numer zamówienia nie jest kluczem, ponieważ jedno zamówienie może dotyczyć wielu części.

- *Atrybut* relacji nazywamy *podstawowym*, jeżeli należy do dowolnego z kluczy tej relacji.
- *Atrybut* relacji nazywamy *wtórnym*, jeżeli nie należy do żadnego z kluczy tej relacji.

- Atrybut B relacji R jest *funkcjonalnie zależny* od atrybutu A tej relacji (co określa się również słowami, że A *identyfikuje* B i oznacza A B), jeśli dowolnej wartości a atrybutu A odpowiada nie więcej niż jedna wartość b atrybutu B.

Zależność funkcjonalna między dwoma atrybutami A i B nie jest związana z przypadkowym układem ich wartości, ale wynika z charakteru zależności między tymi atrybutami w modelowanej rzeczywistości.

Informację, że atrybut A identyfikuje B (tzn. że B jest funkcjonalnie zależny od A) zaznacza się na rysunkach strzałką biegnącą od A do B. W ten sposób schemat zależności funkcjonalnych dla przykładowej relacji "Zamówienia" można narysować następująco:



Pierwsza postać normalna.

Relacja jest w pierwszej postaci normalnej, jeśli wartości atrybutów są elementarne tzn. są to pojedyncze wartości określonego typu, a nie zbiory wartości.
Pierwsza postać normalna jest konieczna aby, tabelę można było nazwać relacją. Większość systemów baz danych nie ma możliwości zbudowania tabel nie będących w pierwszej postaci normalnej. Przekształcenie z postaci nieznormalizowanej do pierwszej postaci normalnej ilustruje rysunek:

Postać nieznormalizowana.

Zamówienia

Nr zamówienia	Id dostawcy	Nazwa dostawcy	Adres dostawcy	Id części	Nazwa części	Ilość
001	010	Seagate	Borsucza 8	054	Dysk twardy	30
				055	Sterownik I/O	50
002	020	Toshiba	Wilcza 3	070	Napęd CD	10
003	010	Seagate	Borsucza 8	054	Dysk twardy	40
				070	Napęd CD	15

Po przekształceniu do pierwszej postaci normalnej.

Zamówienia

Nr zamówienia	Id dostawcy	Nazwa dostawcy	Adres dostawcy	Id części	Nazwa części	Ilość
001	010	Seagate	Borsucza 8	054	Dysk twardy	30
001	010	Seagate	Borsucza 8	055	Sterownik I/O	50
002	020	Toshiba	Wilcza 3	070	Napęd CD	10
003	010	Seagate	Borsucza 8	054	Dysk twardy	40
003	010	Seagate	Borsucza 8	070	Napęd CD	15

Druga postać normalna.

Relacja jest w drugiej postaci normalnej, jeżeli każdy atrybut wtórny (tzn. nie wchodzący w skład żadnego klucza potencjalnego) tej relacji jest w pełni funkcjonalnie zależny od wszystkich kluczy potencjalnych tej relacji.
Można zauważyć, że relacja "Zamówienia" nie jest w drugiej postaci normalnej, ponieważ atrybuty "Id dostawcy", "Nazwa dostawcy", "Adres dostawcy" i "Nazwa części" nie są w pełni funkcjonalnie zależne od jedynego klucza potencjalnego - pary ("Nr zamówienia", "Id części").

W celu sprowadzenia relacji do drugiej postaci normalnej, należy podzielić ją na takie relacje, których wszystkie atrybuty będą w pełni funkcjonalnie zależne od kluczy. W tym celu przykładową relację "Zamówienia" należy podzielić na trzy relacje: "Dostawca na zamówieniu", "Zamówione dostawy", "Części" w następujący sposób:

Dostawca na zamówieniu

Nr zamówienia	Id dostawcy	Nazwa dostawcy	Adres dostawcy
001	010	Seagate	Borsucza 8
002	020	Toshiba	Wilcza 3
003	010	Seagate	Borsucza 8

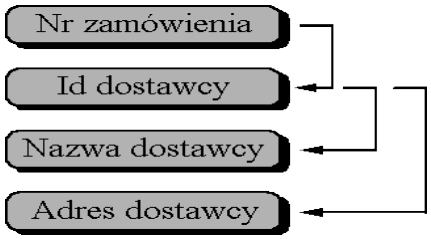
Części

Id części	Nazwa części
054	Dysk twardy
055	Sterownik I/O
070	Napęd CD

Jak widać wszystkie te trzy relacje są w drugiej postaci normalnej, ponieważ klucze relacji "Dostawca na zamówieniu" oraz "Części" są kluczami prostymi, natomiast atrybut "Ilość" w relacji "Zamówione dostawy" jest w pełni funkcjonalnie zależny od klucza złożonego ("Nr zamówienia", "Id części").

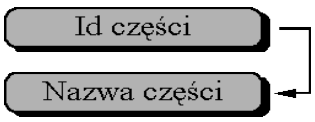
Należy zauważyć, że relacja będąca w pierwszej postaci normalnej, jest równoczesnie w drugiej postaci normalnej, jeśli wszystkie jej klucze potencjalne są kluczami prostymi.
Po przekształceniu relacji "Zamówienia" do drugiej postaci normalnej otrzymujemy następujące zależności funkcjonalne:

- Dostawca na zamówieniu



- Zamówione dostawy

- Części



Czwarta postać normalna

Istotą czwartej postaci normalnej jest dopuszczenie tylko jednej wielowartościowej zależności funkcjonalnej.

Niech R jest relacją a X i Y oznaczają podzbiory atrybutów relacji R. Pod zbiór atrybutów Y jest wielowartościowo funkcjonalnie zależny od podzbioru X relacji R, jeżeli dla relacji R takich, że
$k_1(x) = k_2(x)$ istnieje taka para krotek s_1 i s_2 taka, że
$s_1(x) = s_2(x) = k_1(x) = k_2(x)$
$s_1(y) = k_1(y)$
$s_1(R – X - Y) = k_1(R – X - Y)$
$s_2(Y) = k_2(Y)$ i $s_2(R – X - Y) = k_2(R – X - Y)$

Inaczej mówiąc, jeżeli dla dowolnej pary krotek k_1 i k_2 z relacji R, takich że wartości tych krotek dla atrybutów z podzbioru X są sobie równe, zamienimy na w tych krotkach wartości atrybutów z podzbioru Y, to otrzymamy w ten sposób krotki s_1 i s_2 również należące do relacji R

Trywialną wielowartościową zależnością funkcjonalną będziemy nazywać taki przypadek gdy $R= X \cup Y$ a X i Y są rozłącznymi podzbiorami relacji R.

Niech X, Y, Z będą niepustymi rozłącznymi podzbiorami atrybutów relacji R takimi, że
$R = X \cup Y \cup Z$
i Y jest nietrywialnie wielowartościowo zależny od X. Relacja R jest w 4 postaci normalnej gdy jest w N3 i wielowartościowa zależność Y i X pociąga za sobą funkcjonalną zależność wszystkich atrybutów tej relacji od X.

Innymi słowy, jeżeli w danej relacji występuje wielowartościowa zależność funkcjonalna atrybutów za zbioru Y od dowolnego podzbioru atrybutów X, to podzbiór X zawiera klucz relacji. Jeżeli w danej relacji postaci N3 występuje tylko trywialna zależność funkcjonalna to relacja ta jest w N4.

Dana jest relacja o schemacie R oraz trzy parami rozłączne i niepuste podzbiory X, Y, Z atrybutów z R takie, że X Y Z = R i podzbiór Y jest nietrywialnie wielowartościowo zależny od X.

Dana relacja R jest w czwartej postaci normalnej wtedy i tylko wtedy, gdy jest w trzeciej postaci normalnej i wielowartościowa zależność zbioru Y od X pociąga za sobą funkcjonalną zależność wszystkich atrybutów tej relacji od X.

Łatwo zauważyć, że tabela "Pracownicy" z definicji wielowartościowej zależności funkcjonalnej jest w trzeciej postaci normalnej, ale nie jest w czwartej postaci normalnej, ponieważ atrybuty "Dziecko" i "Wykład" nie są funkcjonalnie zależne od atrybutu "Nazwisko".

Pracownicy

Nazwisko	Dziecko	Wykład
Kowalski	Agnieszka	Język C
Kowalski	Agnieszka	Systemy operacyjne
Kowalski	Magda	Język C
Kowalski	Magda	Systemy operacyjne
Nowak	Jarosław	Bazy danych
Nowak	Jarosław	Teoria kompilatorów
Nowak	Jan	Bazy danych
Nowak	Jan	Teoria kompilatorów
Nowak	Aleksander	Bazy danych
Nowak	Aleksander	Teoria kompilatorów

Jak wynika z definicja relacja, która zawiera trywialną wielowartościową zależność funkcjonalną jest w czwartej postaci normalnej. Stąd wniosek, że relację zawierającą nietrywialną wielowartościową zależność funkcjonalną należy podzielić na takie relacje, które będą zawierać tylko zależności trywialne.

W opisywanym przykładzie relację "Pracownicy" można podzielić na dwie relacje: "Dzieci" i "Wykłady", które będą zawierać tylko trywialną wielowartościową zależność funkcjonalną:

Dzieci

Nazwisko	Dziecko
Kowalski	Agnieszka
Kowalski	Magda
Nowak	Jarosław
Nowak	Jan
Nowak	Aleksander

Wykłady

Nazwisko	Wykład
Kowalski	Język C
Kowalski	Systemy operacyjne
Nowak	Bazy danych
Nowak	Teoria kompilatorów

Piąta postać normalna

Dana relacja r o schemacie R jest w piątej postaci normalnej wtedy i tylko wtedy, gdy jest w czwartej postaci normalnej i w przypadku występowania w niej połączeniowej zależności funkcjonalnej *R[R_1,. .., R_m] zależność ta wynika z zależności atrybutów od klucza.

Wynika z tego, że w celu doprowadzenia pewnej relacji do piątej postaci normalnej konieczne jest podzielenie jej na takie relacje, które spełniać będą podany wyżej warunek.

Piąta postać normalną relacji związana jest z tzw. ***połączeniowymi zależnościami funkcjonalnymi.***

Dekompozycją relacji R opartej na zbiorze atrybutów {A_1,. ...,A_n} nazywamy zastąpienie jej zbiorem niekoniecznie rozłącznych relacji {R_1,. .., R_m } takich, że R_i stanowi podzbiór atrybutów { A_1,. ..., A_n } i

$$\bigcup_{i=1}^{m} R_i = R = \{A_1, \ldots, A_n\}$$

Mówimy, że w relacji R występuje połączeniowa zależność funkcjonalna co oznacza

∗R [R_1,. .., R_m]

wtedy i tylko wtedy gdy dla dowolnej relacji r ⊂ R zachodzi

r = Π_{R1}(r). ..... Π_{Rm}

gdzie Π oznacza operację projekcji (rzutowania) relacji r na podrelację R.

Inaczej mówiąc oznacza to, dekompozycję relacji na podrelację R_1,. ., R_m przez wykonanie sekwencji operacji a następnie można zrekonstruować relację pierwotną przez wykonanie sekwencji operacji łączenia relacji R_1,. ., R_m. Można pokazać, że wielowartościowa zależność funkcjonalna jest szczególnym przypadkiem połączeniowej zależności funkcjonalnej dla przypadku m = 2.

MODELE BAZ

Model hierarchiczny był pierwszym modelem baz danych skonstruowany na bazie rekordów, dzięki hierarchii szybsze wyszukiwanie, rekordy zmiennej długości o specjalnym znaczniku końca, najczęściej w formie drzew zrównoważonych, stosowane mechanizmy nizania (poruszanie się w dwóch kierunkach drzewa), wykorzystywane np. w budowie sprzętu – baza danych opisująca części
Np. IMS

Cechy podstawowe

- struktura danych ma postać drzewa
- węzły - typy opisywanych obiektów
- łuki - związki typu ojciec-syn
- drzewo jest uporządkowane, tj. na każdym poziomie kolejność węzłów jest określona

- opis obiektu (rekord) zbudowany z pól zawierających dane opisujące obiekt
- związki zrealizowane jako wskaźniki

Model sieciowy – rekordy lub paczki rekordów w postaci sieci albo listy jedno lub dwukierunkowej zamkniętej w pierścień. Powstały w 1971 roku, organizacja zajmująca się tym modelem CODASYL
Np. IMAGE (IBM)

Cechy podstawowe

- struktura danych ma postać grafu (sieci)
- wierzchołki grafu - typy obiektów
- łuki w grafie - wiązania między typami
- opis obiektu (rekord) zbudowany z pól zawierających dane opisujące obiekt
- reprezentacja wiązań (wskaźniki):
- odesłanie bezpośrednie (jednowart.)
- odesłanie inwersyjne (wielowart.)
- wiązanie codasylowe

Model relacyjny – wykorzystywany obecnie w 95% aplikacji bazo-danowych, dane przechowywane są w tablicach rekordów, rekordy rozróżniane są za pomocą klucza – czyli wybranego pola o różnych wartościach w danym rekordzie, w tablicy baz danych nie może być dwóch identycznych rekordów – muszą się różnić przynajmniej wartością głównego klucza
Np.ORACLE, PROGRESS, INGRES, SQL

Cechy podstawowe

- dane zawarte w tabelach
- tabele składają się z kolumn
- liczba kolumn i typy - stałe
- liczba wierszy zmienna
- wiersze nie mają tożsamości innej niż wynikająca z zawartości kolumn
- związki pomiędzy wierszami tabel - zdefiniowane poprzez zależności między wartościami wybranych kolumn, tzw. kluczy (nie ma wskaźników)
- z punktu widzenia teorii model można opisać algebr± relacji między zbiorami atrybutów

Model obiektowy – kolekcja obiektów, do rekordów danych dołączane są mechanizmy obsługujące dane, obiekt jest pakietem danych i procedur, dane są trzymane w atrybutach obiektu, procedury są definiowane za pomocą metod obiektu, metody są uaktywniane przez komunikaty przekazywane między obiektami.
Np. ORION, INIS

Cechy podstawowe

- obiekt w bazie reprezentuje obiekt w świecie zewnętrznym
- typ obiektowy (klasa):
- definicja złożonego typu danych (może zawierać inne typy obiektowe lub ich kolekcje)
- procedury (metody) i operatory do manipulowania tymi danymi
- tożsamość obiektu jest niezależna od zawartości danych
- dziedziczenie (*inheritance*):
- strukturalne: potomek dziedziczy strukturę danych
- behawioralne: potomek dziedziczy metody i operatory

POSTACIE NORMALNE

1NF

Przed przejściem do 1NF nieznormalizowany zbiór dzielimy na tabele z atrybutami funkcyjnie zależnymi i niezależnymi (1 tabela – klucz główny jednoznacznie identyfikuje każdy rekord, a 2 tabela – klucz złożony)

Relacja jest w pierwszej postaci normalnej wtedy i tylko wtedy, gdy każdy atrybut niekluczowy jest funkcyjnie zależny od klucza głównego.

2NF

Pierwszą tabelę pozostawiamy bez zmian (klucz główny jednoznacznie identyfikuje każdy rekord), a drugą rozkładamy na tabele zależne od części klucza złożonego

Relacja jest w drugiej postaci normalnej wtedy i tylko wtedy, gdy jest w 1NF i każdy atrybut niekluczowy (nienależący do żadnego klucza) jest w pełni funkcyjnie zależny od klucza głównego.

3NF
Usuwamy przechodniość między danymi (dla każdej pary niekluczowych elementów A zależy od B i odwrotnie).

Rozkładamy wszystko na pojedyncze zależności od klucza głównego lub złożonego.

Relacja jest w trzeciej postaci normalnej wtedy i tylko wtedy gdy jest w 2NF i każdy niekluczowy atrybut jest bezpośrednio zależny (a nie przechodnio zależny) od klucza głównego.

4NF
Rozkładamy na osobne tabele zależności wielowartościowe od klucza głównego (zależności jeden do wielu
czyli każda zależność w innej tabeli). Atrybuty niekluczowe nie zależą od siebie nawzajem.

5NF
Relacja jest w piątej postaci normalnej jeśli jest w 4NF i jeżeli nie istnieje rozkład odwracalny na zbiór mniejszych tabel.
Niekluczowe atrybuty są zależne od siebie nawzajem (zależność złączenia) i nie można relacji rozłożyć na zbiór mniejszych tabel.

RELACJA

jest to jedyna struktura danych w relacyjnym modelu danych. W związku z tym, że pojęcie relacji jest matematyczną konstrukcją, relacja jest tabelą, dla której jest spełniony następujący zbiór zasad:

1. Każda relacja w bazie danych ma jednoznaczną nazwę. Według Codda dwuwymiarowa tabela jest matematycznym zbiorem, a matematyczne zbiory muszą być nazywane jednoznacznie.
2. Każda kolumna w relacji ma jednoznaczną nazwę w ramach jednej relacji. Każda kolumna relacji jest również zbiorem i dlatego powinna być jednoznacznie nazwana.
3. Wszystkie wartości w kolumnie muszą być tego samego typu. Wynika to z p. 2.

4. Porządek kolumn w relacji nie jest istotny. Schemat relacji - lista nazw jej kolumn - jest również matematycznym zbiorem. Elementy zbioru nie są uporządkowane.

5. Każdy wiersz w relacji musi być różny. Innymi słowy, powtórzenia wierszy nie są dozwolone w relacji.

6. Porządek wierszy nie jest istotny. Skoro zawartość relacji jest zbiorem, to nie powinno być określonego porządku wierszy relacji.

7. Każde pole leżące na przecięciu kolumny/wiersza w relacji powinno zawierać wartość atomową. To znaczy, zbiór wartości nie jest dozwolony na jednym polu relacji.

Kolumny tabeli to atrybuty, wiersze to krotki.
Klucz główny to jedna lub więcej kolumn tabeli w której wartości jednoznacznie identyfikują każdy wiersz tabeli.

Atrybut lub zestaw atrybutów relacji, które jednoznacznie identyfikują każdą krotkę relacji nazywa się **kluczem relacji**. Z punktu widzenia możliwości wyszukiwania rekordów z relacji interesujące są takie klucze, które zawierają minimalną ilość atrybutów. Takie klucze to **klucze kandydujące**. Jeśli w relacji jest wiele kluczy to jeden z nich ustala się jako **klucz główny**, pozostałe to **klucze wtórne**.

Indeksowanie bd. polega na utworzeniu zbioru wskazań do rekordów uporządkowanego ze względu na wartości danej grupy atrybutów.