**Wprowadzenie do baz danych**

**Baza danych** to uporządkowany zbiór danych z określonej dziedziny tematycznej, zorganizowany w sposób ułatwiający do nich dostęp.

**System zarządzania bazą danych** to program zarządzający danymi w bazie i umożliwiający ich przetwarzanie.

**System bazy danych** to baza danych i system zarządzania bazą danych.

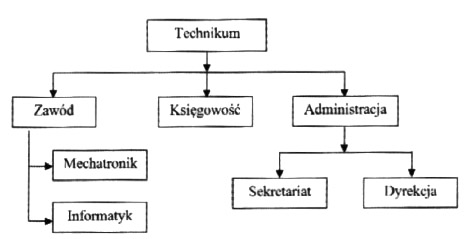
Zalety korzystania z komputerowych baz danych:

* szybkie wyszukiwanie informacji,
* łatwe wykonywanie obliczeń,
* możliwość przechowywania dużej ilości danych na małej powierzchni,
* szybkie porządkowanie danych

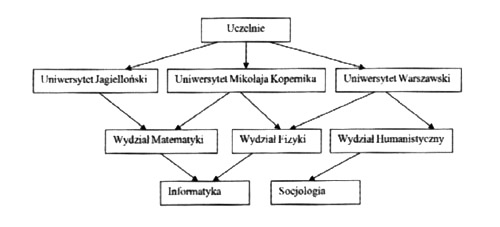
**Modele baz danych**

**Model hierarchiczny**

W tym modelu przechowywane dane są zorganizowane w postaci drzewa. Informacja jest zawarta w dokumentach oraz w **strukturze drzewa** (podobnej do drzewa folderów na dysku komputera).

****

**Model sieciowy**

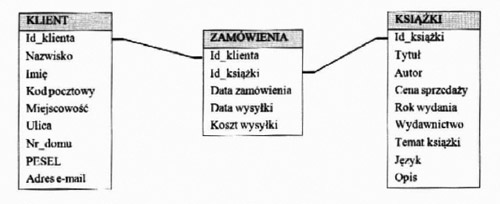
Połączenia między dokumentami tworzą sieć. Informacja jest zawarta w dokumentach oraz w przebiegu połączeń sieci.

**Model obiektowy**

Model obiektowy łączy cechy programów komputerowych tworzonych w językach **programowania obiektowego** z cechami aplikacji bazodanowych. Obiekt w bazie reprezentuje obiekt w święcie rzeczywistym.

**Model relacyjny**

Model relacyjny baz danych został oparty na matematycznym modelu organizacji danych i **pojęcia relacji**. W tym modelu dane są przestawiane w postaci relacji reprezentowanych przez tabelę. Składa się ona z rekordów (wierszy). Ze względu na funkcjonalność jest najczęściej wykorzystywany.



**Model relacyjny**

**Tabele**

W modelu relacyjnym baz danych wszystkie dane są przechowywane w dwuwymiarowych tabelach (relacjach). W skład bazy danych może wchodzić wiele relacji.

Tabela to zbiór powiązanych ze sobą danych. Jest to układ poziomych wierszy, nazywanych **rekordami** lub **krotkami**, i pionowych kolumn, nazywanych **polami rekordu** lub **atrybutami**. Tabela jest identyfikowana poprzez nazwę.

**Klucz podstawowy**

W każdej tabeli musi znaleźć się pole, które dla każdego rekordu będzie przyjmowało inną, niepowtarzalną wartość. Pole takie jest potrzebne do jednoznacznego zdefiniowania rekordu. To**klucz podstawowy**, nazywany również **kluczem głównym** lub **pierwotnym**.

**Klucz podstawowy jest to minimalna kombinacja pól identyfikująca każdy rekord w tabeli w sposób jednoznaczny.**

Klucz podstawowy pozwala w sposób efektywny przeszukiwać i odczytywać dane w bazie oraz łączyć dane zapisane w różnych tabelach.  
Klucz podstawowy nie może zawierać powtarzających się danych oraz nie może być pusty. Oznacza to, że w tabeli musi się znaleźć jedno lub kilka pól, które pozwolą odróżniać dane zapisane w jednym rekordzie od danych zapisanych w innym rekordzie.

**Klucz sztuczny**

W celu uniknięcia dodawania kolejnych pól do klucza bardzo często zastępuje się klucz podstawowy **kluczem sztucznym**. Najprostszym sposobem utworzenia klucza sztucznego jest dodanie do tabeli dodatkowego pola i umieszczenie w nim kolejnych numerów.

**Klucz sztuczny to pole zawierające unikatowy numer identyfikacyjny nadany w sposób sztuczny każdemu obiektowi umieszczonemu w tabeli**.

**Klucz obcy** to jedno pole lub więcej pól tabeli (kolumn), które odwołują się do pola lub pól klucza podstawowego w innej tabeli. Klucz obcy pokazuje, w jaki sposób tabele są powiązane. Jest **niezbędny do zdefiniowania połączenia między tabelami**.

**Relacje**  
Projektując bazę danych, dzielimy dane na wiele tabel tematycznych, tak aby każda informacja została zapisana tylko raz. Aby zestawić razem dane zapisane w różnych tabelach, tworzy się między nimi połączenia zwiane **relacjami**.

**Relacja jest to zdefiniowanie logicznego połączenia między tabelami bazy danych. W efekcie zmiana wiersza bieżącego w tabeli głównej powoduje automatyczną zmianę wiersza bieżącego w tabeli przyłączonej.**

**Typy relacji**

* **Relacja jeden do jednego**

W relacji „jeden do jednego” każdemu rekordowi z pierwszej tabeli może odpowiadać tylko jeden rekord z drugiej tabeli i każdemu rekordowi z drugiej tabeli może odpowiadać tylko jeden rekord z pierwszej tabeli.

* **Relacja wiele do jednego**

W relacji „wiele do jednego” każdemu rekordowi z pierwszej tabeli może odpowiadać najwyżej jeden rekord z drugiej tabeli, a każdemu rekordowi z drugiej tabeli może odpowiadać wiele rekordów z pierwszej tabeli. Jest to typ relacji najczęściej występujący w relacyjnych bazach danych.

* **Relacja wiele do wielu**

W relacji „wiele do wielu” każdemu rekordowi z pierwszej tabeli może odpowiadać wiele rekordów z drugiej tabeli i każdemu rekordowi z drugiej tabeli może odpowiadać wiele rekordów z pierwszej tabeli.

**Zasady projektowania baz danych**

Cały proces projektowania bazy danych możemy podzielić na kilka etapów:

* planowanie bazy danych,
* tworzenie modelu konceptualnego (**diagramu ERD**),
* transformacja modelu konceptualnego na model relacyjny,
* proces normalizacji bazy danych,
* wybór struktur i określenie zasad dostępu do bazy danych.

**Z punktu widzenia relacyjnej bazy danych świat rzeczywisty widzimy i analizujemy jako zestaw encji i związków zachodzących między nimi.**

**Encja**  
Encją jest każdy przedmiot, zjawisko, stan lub pojęcie, czyli każdy obiekt, który potrafimy odróżnić od innych obiektów (na przykład: osoba, samochód, książka, stan pogody)  
  
Encje podobne do siebie (opisywane za pomocą podobnych parametrów) grupujemy w zbiory encji. Projektując bazę danych, należy precyzyjnie zdefiniować encje i określić parametry, przy użyciu których będą opisywane.  
  
**Atrybut**Encje mają określone cechy wynikające z ich natury. Cechy te nazywamy atrybutami. Zestaw atrybutów, które określamy dla encji, zależy od potrzeb bazy danych.  
  
**Dziedzina**  
Atrybuty encji mogą przyjmować różne wartości. Projektując bazę danych, możemy określić, jakie wartości może przyjmować dany atrybut. Zbiór wartości atrybutu nazywamy dziedziną (**domeną**).

**Diagramy związków encji (diagramy ERD)**

**Konceptualne projektowanie bazy danych** to konstruowanie schematu danych niezależnego od wybranego modelu danych, docelowego systemu zarządzania bazą danych, programów użytkowych czy języka programowania.

Do tworzenia modelu graficznego schematu bazy danych wykorzystywane są **diagramy związków encji**, z których najpopularniejsze są **diagramy ERD** (ang. Entity Relationship Diagram). Pozwalają one na modelowanie struktur danych oraz związków zachodzących między tymi strukturami. Nadają się szczególnie do modelowania relacyjnych baz danych, ponieważ umożliwiają prawie bezpośrednie przekształcenie diagramu w schemat relacyjny.

**Diagramy ERD składają się z trzech rodzajów elementów:**

* zbiorów encji,
* atrybutów encji,
* związków zachodzących między encjami.

**Transformacja modelu konceptualnego do modelu relacyjnego**

Reguły transformacji diagramów ERD do tabel:

-Do każdego zbioru podobnych encji stosuje się osobną tabelę

-Jednej encji odpowiada jeden wiersz

-Atrybutowi odpowiada kolumna

-Dla każdego atrybutu określa się typ informacji

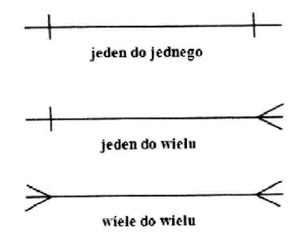
**Notacja Chena**

Została zaprezentowana w 1976 roku. tylko w notacji Chena przewidziany jest oddzielny symbol reprezentujący związek w taki sposób, który pozwala modelować istniejące niejednoznaczne zależności pomiędzy encjami.

Notacja Chena pełni w chwili obecnej rolę historyczno-dydaktyczną. Żadne współczesne narzędzia jej nie wykorzystują.

**Notacja Martina** **- James Martin**

(zwana też notacją kurzych łapek) Swoją potoczną nazwę zawdzięcza rozgałęzieniu oznaczającym związek "do wielu", które przypomina trójpalczastą ptasią nogę. W notacji tej związki wymagalne oznacza się pionową kreską "I" (lub w wersji zmodyfikowanej - dwiema kreskami "II"), natomiast opcjonalne kołem "o".



**Związek obligatoryjny** – wymagany. Bez niego encja nie może istnieć.

-opcjonalny

-wymagany

**Diagramy ERD spotyka się w wielu notacjach:**

* Chena
* Martina
* Bachmana
* IDEF1X
* Barkera

**Normalizacja tabel**

Normalizację tabel stosuje się, aby sprawdzić czy zaprojektowane tabele mają prawidłową strukturę.

Proces normalizacji rozpoczynamy, gdy zostanie utworzony wstępny projekt tabel.

Normalizacja umożliwia określić, czy informacje przewidziane

w projekcie bazy zostały przydzielone do właściwych tabel.

Normalizacja nie odpowiada na pytanie, czy projekt bazy danych jest prawidłowy.

Podczas normalizacji zmienia się strukturę tabel, tworzy nowe tabele i określa łączące je relacje, ale nie usuwa się ani nie modyfikuje przechowywanych w bazie informacji.

Normalizacja polega na rozbiciu dużych tabel na mniejsze. Zmniejsza to wydajność bazy danych, dlatego w niektórych przypadkach nie normalizuje się tabel (systemy nie korzystające z modelu relacyjnego).

Korzyści płynące z normalizacji tabel:

* Wyeliminowanie problemu powtarzania danych (redundancja)
* Optymalizacja objętości bazy danych
* Optymalizacja efektywności obsługi bazy danych
* Minimalizacja zagrożenia błędami przy wprowadzaniu danych

**Edgar Frank Codd** zdefiniował trzy postacie normalne i chociaż później zostały dodane kolejne trzy to prawie wszystkie bazy danych doprowadza się do 3 postaci normalnej, a ponad 75% baz danych normalizowanych jest do czwartej postaci normalnej. Baza danych znajdująca się w wyższej postaci normalnej musi spełniać wymogi wszystkich niższych postaci normalnych. Stosowanych jest pięć reguł normalizacji, ale w większości projektów baz danych wystarczy sprawdzić trzy pierwsze.

Dla każdej z nich stosowane są określenia:

* Pierwsza postać normalna (1PN 1NF)
* Druga postać normalna (2PN 2NF)
* Trzecia postać normalna (3PB 3NF)
* Postać normalna Boyce-Codd’a (określana również mianem 3.5NF)
* Czwarta postać normalna 4NF
* Piąta postać normalna 5NF

Czwarta i piąta postać normalna są używane wyłącznie przy okazji rozważań teoretycznych.

Celem normalizacji jest doprowadzenie tabeli do postaci normalnej zazwyczaj jest to postać normalna trzecia.

Pierwsza postać normalna (1PN)

Tabela jest w pierwszej postaci, jeżeli każdy wiersz w tabeli przechowuje informacje o pojedynczym obiekcie, a każde pole tabeli zawiera informację elementarną (atomową).

Oznacza to, że w komórce tabeli nie może wystąpić lista wartości.

Tabela jest w drugiej postaci normalnej, jeżeli jest w pierwszej postaci normalnej oraz każde z pól niewchodzących w skład klucza podstawowego zależy od całego klucza a nie od jego części.

Ta postać mówi o tym, że każda tabela powinna przechowywać dane dotyczące tylko konkretnej klasy obiektów.

Tabela jest w trzeciej postaci normalnej (III PN), jeżeli jest w pierwszej i drugiej postaci normalnej oraz każde z pól niewchodzących w skład klucza podstawowego niesie informacje bezpośrednio o kluczu i nie odnosi się do żadnego innego pola.

Ta postać normalna głosi, że kolumna nie należąca do klucza zależy bezpośrednio tylko od klucza głównego a nie od innej kolumny.

Jeżeli mamy klucz pojedynczy to wymaganie drugie i trzecie możemy potraktować jako jedno. Jeżeli klucz jest złożony (kilka kolumn) wtedy trzeba dokonać normalizacji do trzeciej postaci normalnej (III PN).

**Anomalie baz danych**

**Anomalia** **(nieprawidłowość)** – odchylenie od wartości typowej lub średniej.

**Anomalia** – problemy mogące wystąpić w nieodpowiednio zaplanowanej, nieznormalizowanej bazie (dane przechowywane w jednej tabeli).

Anomalnie mogą doprowadzić do utraty danych.

Rodzaje anomalii:

* modyfikacji – informacja jest modyfikowana tylko w części danych, pozostają dane które nie zostały zmodyfikowane (nie jesteśmy w stanie wskazać prawdziwą informacje)
* usuwania – usuwanie części informacji powoduje utratę innych informacji, których nie chcemy stracić
* dołączania – wprowadzenie pewnej informacji jest możliwe tylko wtedy, gdy jednocześnie wprowadzamy jakąś inną informację, która może być obecnie niedostępna

**Cechy relacyjnej bazy danych**

Baza danych powinna charakteryzować się następującymi cechami:

* trwałość danych
* integralność danych
* bezpieczeństwo danych
* współdzielenie danych
* abstrakcja danych
* niezależność danych
* integracja danych

**Integralność danych** (spójność, poprawność danych) oznacza, że dane muszą:

* wiernie odzwierciedlać dane rzeczywiste (dane są prawdziwe oraz są aktualizowane, gdy ulega zmianie rzeczywistość).
* spełniać ograniczenia nałożone przez użytkowników (system kontroli danych wejściowych)

**Trwałość danych** zapisanych w bazie jest podstawową cechą baz danych. Trwałość oznacza, że dane zostały zapisane w bazie w sposób nieulotny.

Wszystkie współczesne systemy muszą spełniać ten wymóg.

Trwałość danych jest niezależna od działania aplikacji oraz od platformy sprzętowej i programowej. Dane gromadzone w bazie danych są przechowywane w pamięci zewnętrznej (dyskowej).Powinny być przechowywane w pamięci dopóty, dopóki wymagają tego użytkownicy systemu baz danych.

Na poziomie danych wyróżniamy dwa rodzaje integralności danych:

* **semantyczna (spójność logiczna)** – oznacza spójność danych z rzeczywistością, czyli poprawność odwzorowania rzeczywistości.
* **bazowa** – oznacza poprawność procesów zachodzących w bazie

Na poziomie struktury bazy danych można wyróżnić następujące rodzaje integralności:

* **referencyjna** – odnosi się do powiązań między tabelami i oznacza, że każdej wartości klucza obcego odpowiada dokładnie jedna wartość klucza podstawowego.
* **encji** – odnosi się do schematu bazy danych i oznacza, ze każda encja musi posiadać klucz podstawowy (w danej tabeli nie mogą istnieć dwa identyczne wiersze).
* **atrybutu** – wartość zależy od jego dziedziny.

**Współdzielenie danych**

Współdzielenie danych oznacza, że istnieje możliwość równoczesnej pracy wielu użytkowników z tą samą bazą danych

**Abstrakcja danych**

Abstrakcja to uogólnienie (uproszczenie) rozpatrywanego problemu, które polega na wyodrębnieniu wspólnych jego cech.

Bazy danych powinny opisywać wyłącznie istotne cechy obiektów świata rzeczywistego. Baza danych to abstrakcyjny model pewnego wycinka rzeczywistości, a jej struktura powinna poprawnie odzwierciedlać obiekty świata rzeczywistego i powiązania między obiektami.

**Abstrakcja danych**

Wyodrębniamy trzy poziomy abstrakcji:

* **poziom wewnętrzny** – określa sposób pamiętania danych
* **poziom pojęciowy (konceptualny)** – wyższy poziom abstrakcji danych. Definiuje dane oraz związki zachodzące między nimi.
* **poziom zewnętrzny** – najwyższy poziom abstrakcji danych. Opisuje w jaki sposób dane są widziane przez użytkownika

Dla tej samej bazy danych istnieje jeden model pojęciowych i wiele schematów zewnętrznych.

**Prawidłowy projekt bazy danych**

Prawidłowy projekt bazy danych jest istotny dla efektywnej pracy. Właściwy projekt nie powinien zawierać powtarzających się danych. Cel ten możemy osiągnąć poprzez podzielenie danych na wiele tabel. Kolejny krok to zdefiniowanie połączeń między tabelami.

Proces projektowania bazy danych składa się z następujących kroków:

* określenie celu, jakiemu ma służyć baza danych (baza klientów siłowni)
* określenie zakresu potrzebnych informacji (określenie przechowywanych informacji: imię, nazwisko, nr\_tel, rodzaj karnetu, data\_zakupu\_karnetu, cena\_karnetu, płatność)
* podzielenie informacji na tabele (Tabela: klient, karnet, zakup)
* podzielenie informacji na kolumny (jakie informacje mają być przechowywane w tabelach Klient: imię, nazwisko, nr\_tel)
* wybranie kluczy podstawowych (należy wybrać klucz podstawowy dla każdej tabeli)
* zastosowanie reguł normalizacji (za pomocą reguł normalizacji sprawdzamy czy tabele mają prawidłową strukturę)
* poprawienie projektu (po sprawdzeniu tabel, jeżeli to konieczne należy zmodyfikować projekt bazy danych)
* utworzenie relacji pomiędzy tabelami (należy przejrzeć projekt i wybrać odpowiednie relacje dla bazy danych)

**Typy danych**

Każda kolumna tabeli w bazie danych ma przypisany typ, który określa rodzaj danych, jakie mogą być w niej przechowywane.

Występujące w MySQL typy danych możemy podzielić  na trzy grupy:

* Liczbowe
* Daty i czasu
* łańcuchowe

Typy liczbowe można podzielić  na dwa rodzaje:

* typy całkowite (ang. Integer types)
* typy zmiennoprzecinkowe (ang. Floating point types)

Zgodnie  z nazwami służą one do prezentacji wartości całkowitych i zmiennoprzecinkowych.

Przy definiowaniu typu zastosowanie modyfikatora UNSIGNED oznacza, że wartość ma być traktowana jako liczba bez znaku (niedopuszczalne wartości ujemne).

Przykład: **TINYINT UNSIGNED** (wartości od 0 do 255)

**Liczby zmiennoprzecinkowe**

**Decimal** – wolniejsze działanie, ale precyzyjniejsze. Wykonywanie sumowanie, opera na walutach

**Float** i **Double** – szybsze działanie, obliczenia mniej precyzyjne, zajmuje mniej miejsca. Wykorzystywany w dużych obliczeniach naukowych

**Typy łańcuchowe** służą do przechowywania  zarówno ciągów znaków jak i danych binarnych. Można je podzielić na cztery grupy

* CHAR, VARCHAR
* BINARY, VARBINARY
* BLOB, TEXT
* ENUM, SET

**Każdy znak zajmuje jeden bajt.**

Typ **CHAR** i **VARCHAR** służą do przechowywania łańcuchów znakowych czyli tekstów. Oba wymagają podania długości łańcucha za nazwą typu po nawiasie.

Jeżeli planujemy utworzyć kolumnę, która będzie mogła przechowywać do 40 znaków należy użyć zapisu:

CHAR(40)

VARCHAR(40)

W przypadku CHAR cała kolumna w bazie danych będzie miała stałą długość wskazaną określoną parametrem długości. Jeśli zapisane dane będą miały mniej znaków, pozostałe miejsca zostaną uzupełnione spacjami z prawej strony. Spacje te będą usuwane pobierania danych. Parametr długości może przyjmować wartość od 0 do 255.

W typie VARCHAR każdy wiersz kolumny ma zmienną długość. Parametr długości od 0 do 65535 znaków.

W przypadku próby zapisania większej liczby znaków, niż wynika to z wartości parametru długości wpis zostanie obcięty.

Typy **BLOB** i **TEXT**  służą do przechowywania dużych ilości danych.

**BLOB** – (ang. Binary Large Object) – służy do przechowywania ciągów binarnych

**TEXT** – służy do przechowywania ciągów tekstowych (65535 znaki), stały rozmiar, nie można deklarować innych długości.

Oba te typy dzielą się na cztery podtypy różniące się od siebie wielkością danych, które mogą być za ich pomocą zapisane

**Typy ENUM i SET**

Typ **ENUM** jest typem wyliczeniowym pozwalający ograniczyć zbiór wartości, który będzie mógł być przechowywany w danej kolumnie.

Typ **SET** jest typem wyliczeniowym zdefiniowany analogicznie jak  typ ENUM

SET (’czerwony’ ‚’czarny’)

Każdy wiersz będą mogły być wartości: czerwony, czarny a także  czerwony i czarny.







**SQL**

SQL (and. Standard Query Language) to strukturalny język zapytań stosowany w systemach relacyjnych baz danych do o\komunikowania się z bazą

* SQL to podstawowy język programowanie\a bez danych, pozwalający na tworzenie i modyfikowanie obiektów baz danych
* Język SQL umożliwia wydajne przetwarzanie dużej ilości danych
* Język SQL jest językiem deklaratywnym – definiuje się warunki jakie musi spełniać końcowy wynik, natomiast nie podaje się sposobu w jaki ten wynik zostanie osiągnięty
* Instrukcja języka SQL są interpretowane a nie kompilowane, niektóre błędy (np. próba odwołania się do nieistniejącej tabeli, czy próba wstawienia danych tekstowych do kolumny typu liczbowego) zostaną wykryte dopiero podczas ich wykonywania. Problem ten nie występuje w językach kompilowanych
* Podczas interpretacji przeprowadzana jest optymalizacja polegająca na znalezieniu najlepszego (w przypadku Serwera SQL – jak najtańszego) planu wykonania
* Każdy dostępny na rynku systemów bazodanowych zawiera specyficzne niedostępne w innych systemach elementy
* Mogą być stosowane różne wersje języka SQL
* W celu ujednolicenia wersji języka SQL Amerykański Narodowy Instytut Standardów (and. American National Standards Institute ANSI) oraz Międzynarodowa Organizacja Normalizacyjna (ang. International Organization for Standardization ISO) opracowują i publikują standardy języka SQL
* W 1999 roku został przyjęty standard ANSI SQL99 (SQL3) w chwili obecnej większość producentów bazodanowych tworzy systemy zgodne z tym standardem. Standard SQL99 nie definiuje wielu rozszerzeń języka SQL, dlatego w roku 2003 został opublikowany czwarty standard języka SQL, który jest rozszerzeniem SQL3.

Mimo przyjętych standardów systemy zarządzania bazą danych nadal dodają własne rozszerzenia.

Wszystkie typowe operacje wykonywania są tak samo, niezależnie od używanego systemu zarządzania bazą danych. Są one nazywane mianem dialektu.

Standard języka SQL92 wprowadził klasyfikację poleceń ze względu na ich przeznaczenie:

* Instrukcje DDL (ang. Data Definition Language) – służą do tworzenia, modyfikowania i usuwania obiektów baz danych

CREATE, ALTER, DROP

Instrukcje DML (ang. Data Manipulation Language) – służą do odczytywania i modyfikowania danych

SELECT, INSERT, UPDATE, DELETE

Instrukcje DCL (ang. Data Control Language) – nadawanie i odbieranie praw użytkownikom

GRANT, REVOKE

Występuje 5 głównych kategorii

(zbiór wyrażeń tak dobranych, że gdy w jakimś zdaniu lub zwrocie sensownym dane wyrażenie zastąpimy wyrażeniem należącym do tej samej kategorii semantycznej lub syntaktycznej, całość pozostanie sensowna):

* Identyfikator (nazwy obiektów)
* Literały (stałe)
* Operatory (spójniki)
* Słowa kluczowe (wyrazy interpretowane przez serwer bazodanowy)
* Komentarze (ignorowane przez serwery bazodanowe)

**Optymalizacja zapytań i operacje wykonywane w bazie danych**

Podczas wykonywania zapytania, oprócz uzyskania interesujących nas informacji, istotna jest szybkość wykonywania zapytania.

W systemach zarządzania baza danych występuje narzędzie służące do optymalizacji zapytań (query optimizer).

Aby zapewnić wydajność bazy danych optymalizator analizuje zapytania i tworzy możliwie optymalny plan jego wykonania (execution plan).

Plan ten zawiera opis użytych indeksów, sposób dostępu do danych z tabeli, sposób i kolejność łączenia tabeli, uwzględnienie warunków z sekcji WHERE.

Aby przyspieszyć wykonywanie zapytać, należy je konstruować tak, by były wykonywane na jak najmniejszej ilości danych.

Zmniejszenie ilości przetwarzanych danych zmniejsza ilość potrzebnych zasobów oraz zwiększa efektywność działania indeksów.

Nie powinno się stosować porządkowania danych, jeżeli nie jest to konieczne (klauzula ORDER BY).

Dane należy pobierać w takiej kolejności w jakiej są zapisane w bazie.

Zaleca się unikanie zagnieżdżonych zapytań i klauzuli GROUP BY.

**Funkcje**

Podobnie jak w każdym innym języku programowania SQL obsługuje funkcje umożliwiające modyfikację danych. W zasadzie tylko kilka podstawowych funkcji obsługiwanych jest tak samo przez główne bazy danych. Nazewnictwo funkcji zależy od systemu zarządzania baza danych.

Funkcje to na ogół działania przeprowadzane na danych w celu ich konwersji lub manipulacji nimi.

Funkcje możemy podzielić na:

* Funkcje tekstowe operujące na ciągach znaków
* Funkcje liczbowe operujące na liczbach
* Funkcje daty i czasu operujące na danych typu data i godzina
* Funkcje konwersji służące do zmiany typu danych
* Na specjalne wyróżnienie zasługują funkcje kryptologiczne, które pozwalają zaszyfrować, odszyfrować i podpisać wiadomość oraz sprawdzić jej autentyczność

Funkcje tekstowe argumentem funkcji tekstowych są ciągi znaków (dane typów char, varchar lub text).

Typ danych zwracanych przez funkcje tekstowe jest podstawą do ich dalszego podziału:

* Wartość znakową
* Funkcje tekstowe zwracające liczbę

**Transakcje**

W bazach danych transakcja to zbiór wykonywanych operacji, które stanowią całość. Muszą być wykonane wszystkie operacje wchodzące w skład transakcji lub nie zostanie wykonana żadna z nich. Przykładem transakcji jest wykonanie przelewu z jednego konta na drugie.

Operacja przelewu musi zostać wykonana w całości, a jeżeli nie jest to możliwe, należy powrócić do stanu sprzed rozpoczęcia wykonywania operacji przelewu.

Pomiędzy jedną a drugą instrukcją może zdarzyć się anomalia. Dlatego też tego typu operacje powinny być wykonywane jako pojedyncza transakcja.

Transakcja składa się z 3 etapów:

* Rozpoczęcie transakcji
* Wykonanie instrukcji
* Zakończenie transakcji

Do rozpoczynania transakcji służy polecenie

* START TRANSACTION

lub

* BEGIN WORK

Aby zatwierdzić transakcję należy użyć instrukcji

* COMMIT

**Schemat transakcji:**

START TRANSACTION

*Instrukcje wchodzące w skład transakcji*

*Instrukcje wchodzące w skład transakcji*

COMMIT