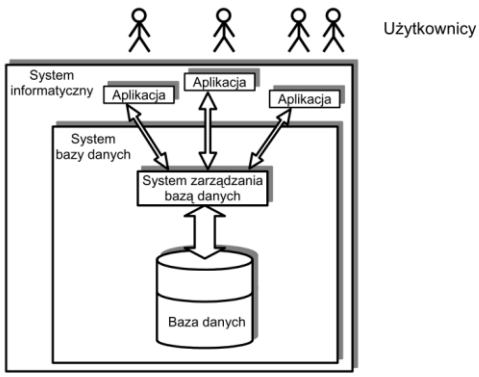
Baza danych jest zbiorem danych oraz narzędzi systemu **DBMS** (**Database Management System - System Zarządzania Bazą Danych. SZBD**) przeznaczonego do zarządzania bazą danych oraz gromadzenia, przekształcania i wyszukiwania danych.

Baza danych to zbiór danych, który dotyczy rzeczywistości - a konkretnie określonego jej fragmentu, który reprezentuje. Fragment ten określamy mianem **obszaru analizy**.



Baza danych ma takie cechy charakterystyczne, jak:

Trwałość danych - oznacza możliwość przechowywania danych w pamięci masowej (trwałej) komputera. Dane tymczasowe mogą być przechowywane w pamięci komputera i tracone po jego wyłączeniu. **Niezależność danych** - pozwala osiągnąć większą elastyczność, **ponieważ programy wymieniające informacje z bazą danych są niezależne od przechowywania danych na dysku** i szczegółów reprezentacji danych na dysku. Niezależność dotyczy również posługiwania się danymi. **Użytkownicy są zabezpieczeni przed logicznymi zmianami (program obsługujący bazę danych jest zabezpieczony przed modyfikacją struktury tabel bazy danych).** DBMS - gwarantujący niezależność fizyczną - przejmuje na siebie zadanie określenia, w jakim formacie i jak dane będą przechowywane na dysku.

Ochrona danych - baza danych oferuje mechanizmy kontroli dostępu do danych w sposób umożliwiający użytkowanie danych wyłącznie przez uprawnionych do tego użytkowników.

Integralność danych - zgodność z rzeczywistością. Dane w bazie danych są odwzorowaniem rzeczywistości. Jeśli modelowany fragment rzeczywistości ulegnie zmianie, baza danych również musi się zmienić.

Baza danych składa się z części **intensjonalnej** oraz **ekstensjonalnej**.

Część intensjonalna bazy danych jest zbiorem definicji, które opisują strukturę danych.

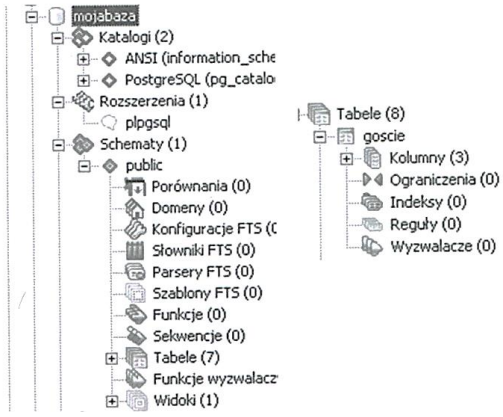
Z kolei **część ekstensjonalna** jest łącznym zbiorem danych w bazie danych. Intensjonalną stronę bazy danych nazywamy schematem bazy danych.

System zarządzania bazą danych SZBD (**DBMS - Database Management System**) obsługuje użytkowników bazy danych, umożliwiając im eksploatacje oraz tworzenie baz danych. By stworzyć i zaprojektować bazę danych, należy ją zdefiniować, a do tego konieczne jest określenie (zdefiniowanie) typów przechowywanych w niej danych. Istotną rolę odgrywa również wyznaczenie użytkowników oraz ich praw dostępu.
SZBD pełni funkcje, które określane są mianem **właściwości baz danych**. Zaliczamy do nich:

- tworzenie struktur baz danych.
- wykonywanie operacji CRUD [Create. Read, Update, Delete),
- obsługa zapytań (selekcjonowanie danych).
- generowanie raportów i zestawień.
- administracja bazą danych.

Tworzenie struktur baz danych
Aby utworzyć strukturę bazy danych, należy posłużyć się wcześniej sporządzonym projektem. Struktura to szkielet bazy danych, przeniesienie koncepcji tabel, powiązań na obszar systemu zarządzania bazą danych.

Strukturę bazy danych możemy utworzyć po podłączeniu do serwera bazy danych. Na taką strukturę składają się: **tabele**, **widoki**, **powiązania pomiędzy tabelami**, **domeny**, **funkcje, indeksy, wyzwalacze, ograniczenia**. Np. w SZBD PostgreSQL strukturę bazy danych możemy poznać, oglądając menu programu pgAdminIII. Wszystko to stanowi logiczną organizację danych.



Kolejną właściwością bazy danych jest przeprowadzanie operacji **CRUD** (zapisu(utworzenie), odczytu, aktualizacji i usuwania). Może zajść potrzeba modyfikowania tabel, widoków oraz aktualizacji danych przechowywanych w tabelach.

Baza danych powinna być tak zaprojektowana, by wykonywanie aktualizacji na danych, usuwanie danych czy wprowadzanie nowych informacji do bazy danych nie spowodowało utraty spójności. **Spójność** bazy danych to **poprawność** umieszczonych w niej informacji.

Baza danych powinna mieć mechanizmy umożliwiające uzyskanie **szybkiego dostępu** do danych i ich **selekcjonowanie**. W relacyjnych bazach danych do uzyskiwania dostępu do danych służą zapytania.

Zapytania to instrukcje napisane przeważnie w języku SQL
Oprócz uzyskania dostępu do informacji i danych, ich sortowania, selekcjonowania i przeszukiwania baza danych powinna oferować mechanizmy umożliwiające **drukowanie wykazów** czy **zapisywanie** ich poza bazą danych. Funkcje takie spełniają **raporty** i **zestawienia**, które mogą być generowane z baz danych.

Baza danych powinna umożliwiać administrację swoimi zasobami. **Administracja** może mieć charakter nie tylko projektowania i implementowania, lecz także optymalizacji i dostosowywania do potrzeb użytkowników.

Cechy bazy danych

Trwałość danych
Niezależność danych
Ochrona danych
Integralność danych

Tworzenie bazy danych.

W systemie zarządzania bazą danych możemy tworzyć bazy danych, w których obrębie możemy utrzymywać tabele. W SZBD MySQL bazy danych mają formę katalogów, a tabele są plikami znajdującymi się w katalogach, których nazwy pokrywają się z nazwami baz danych w MySQL.

Aby utworzyć bazę danych, np. **auta** używamy polecenia

CREATE DATABASE auta;

c:\xampp\mysql\bin
mysql -u root -p

```
mysql> CREATE DATABASE auta;
Query OK, 1 row affected (0.00 sec)
```

Tworzymy bazy: **baza2** i **mojabaza** a następnie wyświetlamy wszystkie bazy danych poleceniem **SHOW DATABASES**

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| auta               |
| baza2              |
| drupal             |
| employees          |
| mojabaza           |
| mydb               |
| mysql              |
| performance_schema |
| test               |
+--------------------+
10 rows in set (0.07 sec)

mysql>
```

Jeśli w bazie danych, w której aktualnie pracujemy, znajdują sic tabele, ich listę możemy odczytać za pomocą polecenia **SHOW TABLES**.

Zanim jednak użyjemy tego polecenia, musimy wskazać, której bazy danych ono dotyczy.

Możemy to zrobić, przełączając się do pożądanej bazy danych za pomocą polecenia **USE**.

Oba polecenia powinny być wykonane w odpowiedniej kolejności - najpierw przełączenie się do określonej bazy danych, np. **auta**, a następnie wyświetlenie tabel:

```
USE auta;
mysql> use auta;
Database changed
mysql>
```

```
SHOW TABLES;
mysql> show tables;
Empty set (0.00 sec)
```

Może się zdarzyć, że w wybranej bazie danych nie ma tabel. Zwłaszcza gdy dopiero utworzyliśmy bazę danych, prawdopodobnie jest ona pusta. Aby utworzyć tabelę, używamy polecenia **CREATE TABLE**. Polecenie wymaga, aby konto osoby chcącej tworzyć bazy danych i tabele miało odpowiednie uprawnienia.

Przyjmijmy, że utworzymy przykładową tabelę na potrzeby komisu samochodowego. Tabela będzie nosić tytuł **pojazdy**, a **atrybutami aut** będą: **numer**, **marka**, **rok_produkcji**, **cena_przedaży**, **cena_zakupu**.

Instrukcja SQL, której zadaniem będzie utworzenie takiej tabeli, będzie miała postać:

Utworzenie tabeli pojazdy za pomocą instrukcji SQL **CREATE TABLE**

```
mysql> CREATE TABLE `auta`.`pojazdy` (
    -> `numer` INT NOT NULL ,
    -> `marka` VARCHAR(45) NULL ,
    -> `rokProdukcji` YEAR NULL ,
    -> `cenaSprzedaży` INT(6) NULL ,
    -> `cenaZakupu` INT(6) NULL ,
    -> PRIMARY KEY (`numer`) );
Query OK, 0 rows affected (0.01 sec)
```

Jak łatwo zauważyć, zaraz po identyfikatorach atrybutów (kolumn) występuje określenie typu danych instancji atrybutu, **AUTO_INCREMENT** oznacza automatyczne wypełnianie wartości kolumny po dodaniu kolejnej krotki (automatycznie numerująca się kolumna). **PRIMARY KEY** oznacza, że atrybut numer będzie kluczem głównym tabeli auta. Pozostałe typy danych i ich opis znajdują się w rozdziale 14.

Po utworzeniu tabeli należy wypełnić ją danymi. Używamy do tego polecenia SQL **INSERT**, po którym znajduje się **INTO** (predykat wskazujący) wskazujący na tabelę pojazdy, do której dane będą wstawiane. Polecenie SQL wstawia dane kolejnych krotek (wierszy tabeli).

DESCRIBE – wyświetlanie struktury tabeli

Taka instrukcja utworzy tabelę w bieżącej bazie. Nazwa bazy może być jednak podana jawnie. Należy wtedy użyć konstrukcji

nazwa_bazy.nazwa_tabeli.

Utworzenie prostej tabeli

```
CREATE TABLE klient
(
  id INTEGER,
  nazwa VARCHAR(20)
);
```

Utwórz tabelę, która będzie zawierała dwie kolumny typu INTEGER: pierwszą o nazwie id i drugą o nazwie znacznik.

```
CREATE TABLE test
(
  id INTEGER,
  znacznik INTEGER
);
```

Kolumna z wartościami rzeczywistymi o określonej precyzji

Utwórz tabelę zawierającą kolumnę przechowującą wartości z separatorem dziesiętnym, z szerokością wyświetlania określoną na 6 cyfr znaczących, z trzema miejscami po przecinku.

```
CREATE TABLE test_d
(
  wartosc DECIMAL(6, 3)
);
```

SHOW COLUMNS FROM nazwa_tabeli;

Tabela z kolumnami przechowującymi krótkie dane tekstowe

Utwórz tabelę, która będzie zawierała następujące kolumny: id typu INTEGER, imie typu VARCHAR i nazwisko typu VARCHAR.

```
CREATE TABLE test_v`
(
  id INTEGER,
  imie VARCHAR(20),
  nazwisko VARCHAR(30)
);
```

W takiej tabeli w kolumnie imie będzie można przechowywać ciągi o długości do 20 znaków, a w kolumnie nazwisko — ciągi nie dłuższe niż 30 znaków.

Atrybuty kolumn

Każda kolumna może mieć dodatkowe atrybuty. Najczęściej spotykane, które często przydają się przy pracy z bazą danych, to:

PRIMARY KEY,
NOT NULL,
DEFAULT,
UNIQUE.

Atrybuty kolumn

Atrybut PRIMARY KEY oznacza, że dana kolumna będzie kluczem podstawowym (głównym). Kolumny z atrybutem PRIMARY KEY są unikatowe (tzn. każdy wiersz takiej kolumny musi mieć inną wartość). Jeśli kolumna ma być kluczem podstawowym, za jej definicją należy umieścić słowa PRIMARY KEY według schematu:

```
CREATE TABLE nazwa_tabeli
(
  nazwa_kolumny typ_kolumny PRIMARY KEY,
  definicje pozostałych kolumn
);
```

Atrybuty kolumn

Utworzenie klucza podstawowego

Utwórz tabelę zawierająca dwie kolumny: id typu INTEGER oraz nazwa typu VARCHAR. Kolumna id ma być kluczem podstawowym.

```
CREATE TABLE test
(
  id INTEGER PRIMARY KEY,
  nazwa VARCHAR(20)
);
```

Atrybuty kolumn **Utworzenie klucza podstawowego**

W przypadku gdyby klucz podstawowy miał się składać z więcej niż jednej kolumny, jego definicja będzie wyglądała inaczej. Schemat tego typu konstrukcji ma postać:

```
CREATE TABLE nazwa_tabeli
(
 nazwa_kolumny_1 typ_kolumny_1 [atrybuty],
 nazwa_kolumny_2 typ_kolumny_2 [atrybuty],
 ...
 nazwa_kolumny_n typ_kolumny_n [atrybuty],
 PRIMARY KEY (kolumna_x, kolumna_y, ..., kolumna_z)
);
```

Atrybuty kolumn

Tabela, w której kolumny id typu INTEGER oraz nazwa typu VARCHAR tworzą klucz podstawowy.

```
CREATE TABLE test
(
 id INTEGER,
 nazwa VARCHAR(20),
 PRIMARY KEY(id, nazwa)
);
```

Atrybuty kolumn

Atrybut NOT NULL oznacza, że w danej kolumnie nie mogą znajdować się wartości puste, czyli że każdy wiersz w tej kolumnie musi zawierać jakąś wartość. Próba zapisania wartości pustej zakończy się niepowodzeniem.

Utwórz tabelę zawierającą kolumnę id typu INTEGER oraz nazwa typu VARCHAR. Kolumna nazwa powinna być zdefiniowana w taki sposób, aby nie mogła zawierać wartości pustych.

```
CREATE TABLE test
(
 id INTEGER,
 nazwa VARCHAR(20) NOT NULL
);
```

Atrybuty kolumn

Wartość domyślna kolumny
Utwórz tabelę zawierającą kolumny id typu INTEGER oraz nazwa typu VARCHAR. Kolumna nazwa powinna mieć wartość domyślną zdefiniowaną jako ciąg znaków **brak**.

```
CREATE TABLE test
(
 id INTEGER,
 nazwa VARCHAR(20) DEFAULT 'brak'
);
```

Atrybuty kolumn

Kolumna bez duplikatów danych

Utwórz tabelę zawierającą kolumnę nazwa typu VARCHAR, tak aby wprowadzane do tej kolumny wartości nie mogły się powtarzać:

```
CREATE TABLE test
(
  id INTEGER,
  nazwa VARCHAR(20) UNIQUE
);
```

Atrybuty kolumn

Kolumna automatycznie zwiększająca wartość

Utwórz tabelę zawierającą dwie kolumny: id typu INTEGER oraz nazwa typu VARCHAR. Pierwsza kolumna powinna być kluczem podstawowym (głównym), a jej wartości powinny być zwiększane automatycznie podczas dodawania danych.
W przypadku bazy MySQL wystarczy do kolumny typu INTEGER dodać atrybut AUTO_INCREMENT. Instrukcja tworząca tabelę będzie wtedy miała postać:

```
CREATE TABLE test
(
  id INTEGER PRIMARY KEY AUTO_INCREMENT,
  nazwa VARCHAR(20)
);
```