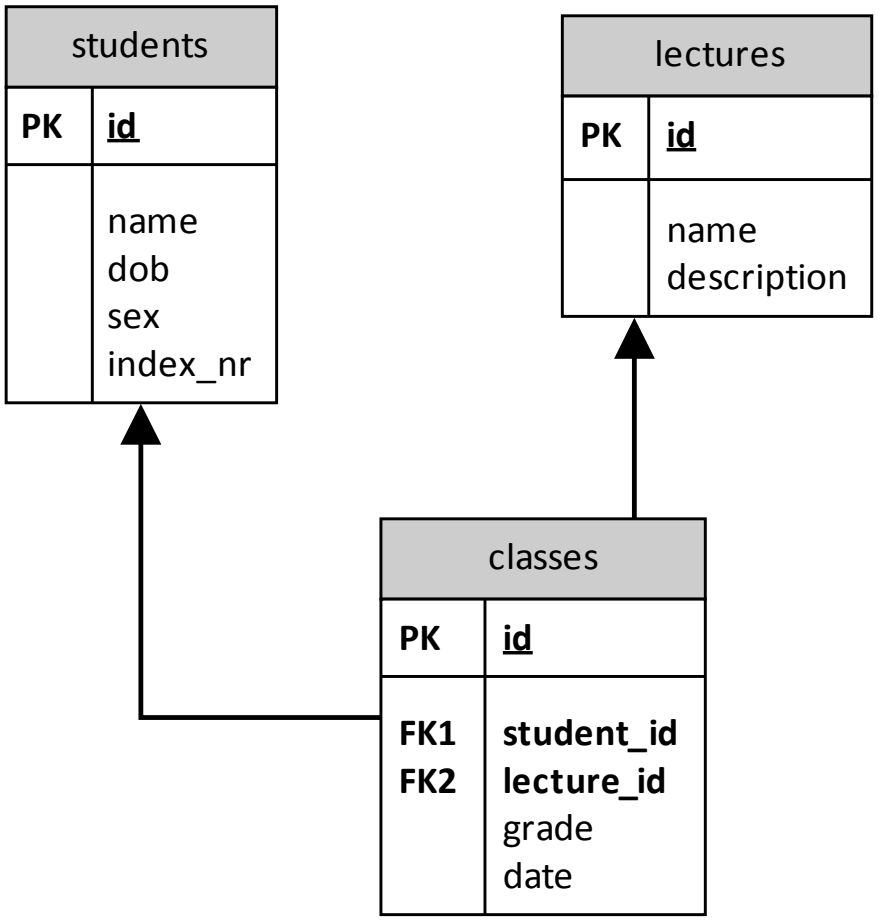
Oprogramowanie Systemów Medycznych

Wykład 5

Relacyjna baza danych (Relational Data Base Management System)

Podstawową strukturą bazy danych jest **tabela** zawierająca indeksowane **wiersze**. Wiersze tabeli mogą zawierać wartości lub indeksy do jednej lub wielu innych tabel umieszczonych w bazie danych. Dzięki temu możliwe jest tworzenie **relacji** pomiędzy danymi.

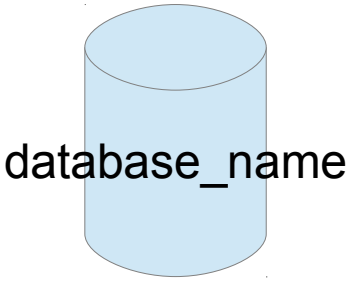


Język komunikacji z bazami danych

- SQL - **S**tructured **Q**uery **L**anguage - standaryzowany (ANSI/ISO/IEC) język zapytań i operacji na danych w relacyjnych bazach danych
- Zadania:
  - dodawania danych
  - odpytywanie bazy danych
  - aktualizacja i usuwanie danych
  - tworzenie i aktualizacja schematów trzymania danych
  - ograniczanie dostępu do danych

Tworzenie bazy danych

- **`CREATE DATABASE database_name`** - zakłada nową bazę danych



Tworzenie tabel

- **CREATE TABLE table_name ...**

```
mysql> CREATE TABLE students (
    id INTEGER PRIMARY KEY,
    name VARCHAR(256) NOT NULL,
    dob DATE NOT NULL,
    sex VARCHAR(1) NOT NULL,
    index_nr INTEGER(6) NOT NULL
)
```

```
mysql> describe students;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| name     | varchar(256) | NO   |     | NULL    |       |
| dob      | date         | NO   |     | NULL    |       |
| sex      | varchar(1)   | NO   |     | NULL    |       |
| index_nr | int(6)       | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> CREATE INDEX students_index_nr ON students(index_nr);
```

Klucze i indeksy

- Wbudowane w silnik baz danych mechanizmy:
  1. Indeksy - struktury wspomagające wyszukiwanie
     - zakładane na polach, lub grupach pól po których tablica będzie przeszukiwana najczęściej
     - typowa implementacja - drzewa zbalansowane (b-tree)
  2. Klucze - reguły ograniczające sposób wstawiania danych
     - PRIMARY - unikalny klucz główny
     - UNIQUE - unikalna wartość kolumny
     - FOREIGN KEY - klucz obcy

Wstawianie wierszy do tabeli

- **INSERT INTO table [(*columns*)] VALUES (*values*)**

```
mysql> INSERT INTO students
       VALUES (1, 'Jan Kowalski', '1981-05-20', 'M', 123456);
```

```
+----+--------------+------------+-----+----------+
| id | name         | dob        | sex | index_nr |
+----+--------------+------------+-----+----------+
|  1 | Jan Kowalski | 1981-05-20 | M   |   123456 |
|  2 | Roman Kluska | 1981-07-20 | M   |   123457 |
+----+--------------+------------+-----+----------+
2 rows in set (0.00 sec)
```

Zapytania

- **`SELECT`** - przeszukiwanie i wybieranie wierszy z tabeli
  - może być ograniczone warunkami
  - może podlegać grupowaniu
  - może wykorzystywać funkcje agregacyjne i statystyczne
  - może wykorzystywać relacje pomiędzy tabelami
- Ogólna struktura zapytania:

SELECT ***jakie dane?***

FROM ***z których tabel?***

WHERE ***pod jakim warunkiem?***

Proste zapytania

```
mysql> SELECT * FROM students;
+----+--------------+------------+-----+----------+
| id | name         | dob        | sex | index_nr |
+----+--------------+------------+-----+----------+
|  1 | Jan Kowalski | 1981-05-20 | M   |   123456 |
|  2 | Roman Kluska | 1981-07-20 | M   |   123457 |
+----+--------------+------------+-----+----------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT name, sex FROM students;
+--------------+-----+
| name         | sex |
+--------------+-----+
| Jan Kowalski | M   |
| Roman Kluska | M   |
+--------------+-----+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM students WHERE index_nr = 123456;
+----+--------------+------------+-----+----------+
| id | name         | dob        | sex | index_nr |
+----+--------------+------------+-----+----------+
|  1 | Jan Kowalski | 1981-05-20 | M   |   123456 |
+----+--------------+------------+-----+----------+
1 row in set (0.00 sec)
```

Eliminowanie wartości powtarzalnych

```
mysql> SELECT DISTINCT sex FROM students;
+-----+
| sex |
+-----+
| M   |
| F   |
+-----+
2 rows in set (0.00 sec)
```

Wykorzystanie funkcji

- silniki baz danych dostarczają funkcji, które można wykorzystać do manipulacji na danych w czasie wykonywania zapytań

```
mysql> SELECT name, (YEAR(CURDATE())-YEAR(dob)) AS age, dob FROM
students;
+-----------------------+------+------------+
| name                  | age  | dob        |
+-----------------------+------+------------+
| Jan Kowalski          |   31 | 1981-05-20 |
| Roman Kluska          |   31 | 1981-07-20 |
| Aleksandra Malinowska |   31 | 1981-04-04 |
+-----------------------+------+------------+
3 rows in set (0.00 sec)
```

Porównywanie wartości

- wyrażenia boolowskie (AND, OR, NOT)
- IN (warość1, wartość2, ...)
- BETWEEN wartość1 AND wartość2
- LIKE %_

```
mysql> SELECT * FROM students WHERE MONTH(dob) IN (5, 4);
+----+----------------------+------------+-----+----------+
| id | name                 | dob        | sex | index_nr |
+----+----------------------+------------+-----+----------+
|  1 | Jan Kowalski         | 1981-05-20 | M   |   123456 |
|  3 | Aleksandra Malinowska | 1981-04-04 | F   |   123453 |
+----+----------------------+------------+-----+----------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM students WHERE MONTH(dob) BETWEEN 6 AND 12;
+----+--------------+------------+-----+----------+
| id | name         | dob        | sex | index_nr |
+----+--------------+------------+-----+----------+
|  2 | Roman Kluska | 1981-07-20 | M   |   123457 |
+----+--------------+------------+-----+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM students WHERE name LIKE 'Aleks%';
+----+-----------------------+------------+-----+----------+
| id | name                  | dob        | sex | index_nr |
+----+-----------------------+------------+-----+----------+
|  3 | Aleksandra Malinowska | 1981-04-04 | F   |   123453 |
+----+-----------------------+------------+-----+----------+
1 row in set (0.00 sec)
```

Sortowanie

- określenie kolejność sortowania
  - ORDER BY pole1, pole2, ... ASC - rosnąco
  - ORDER BY pole1, pole2, ... DESC - malejącą

```
mysql> SELECT * FROM students ORDER BY name, sex ASC;
+----+-----------------------+------------+-----+----------+
| id | name                  | dob        | sex | index_nr |
+----+-----------------------+------------+-----+----------+
|  3 | Aleksandra Malinowska | 1981-04-04 | F   |   123453 |
|  1 | Jan Kowalski          | 1981-05-20 | M   |   123456 |
|  4 | Marta Kosińska        | 1981-07-12 | F   |   123985 |
|  6 | Małgorzata Lis        | 1981-01-11 | F   |   123981 |
|  5 | Michał Kubicki        | 1981-08-12 | M   |   123982 |
|  2 | Roman Kluska          | 1981-07-20 | M   |   123457 |
+----+-----------------------+------------+-----+----------+
mysql> SELECT * FROM students ORDER BY sex, name ASC;
+----+-----------------------+------------+-----+----------+
| id | name                  | dob        | sex | index_nr |
+----+-----------------------+------------+-----+----------+
|  3 | Aleksandra Malinowska | 1981-04-04 | F   |   123453 |
|  4 | Marta Kosińska        | 1981-07-12 | F   |   123985 |
|  6 | Małgorzata Lis        | 1981-01-11 | F   |   123981 |
|  1 | Jan Kowalski          | 1981-05-20 | M   |   123456 |
|  5 | Michał Kubicki        | 1981-08-12 | M   |   123982 |
|  2 | Roman Kluska          | 1981-07-20 | M   |   123457 |
+----+-----------------------+------------+-----+----------+
```

Grupowanie

- GROUP BY - pozwala skleić wiersze które spełniają określony warunek
- grupowanie jest stosowane najczęściej z funkcjami agregującymi takimi jak np. COUNT()

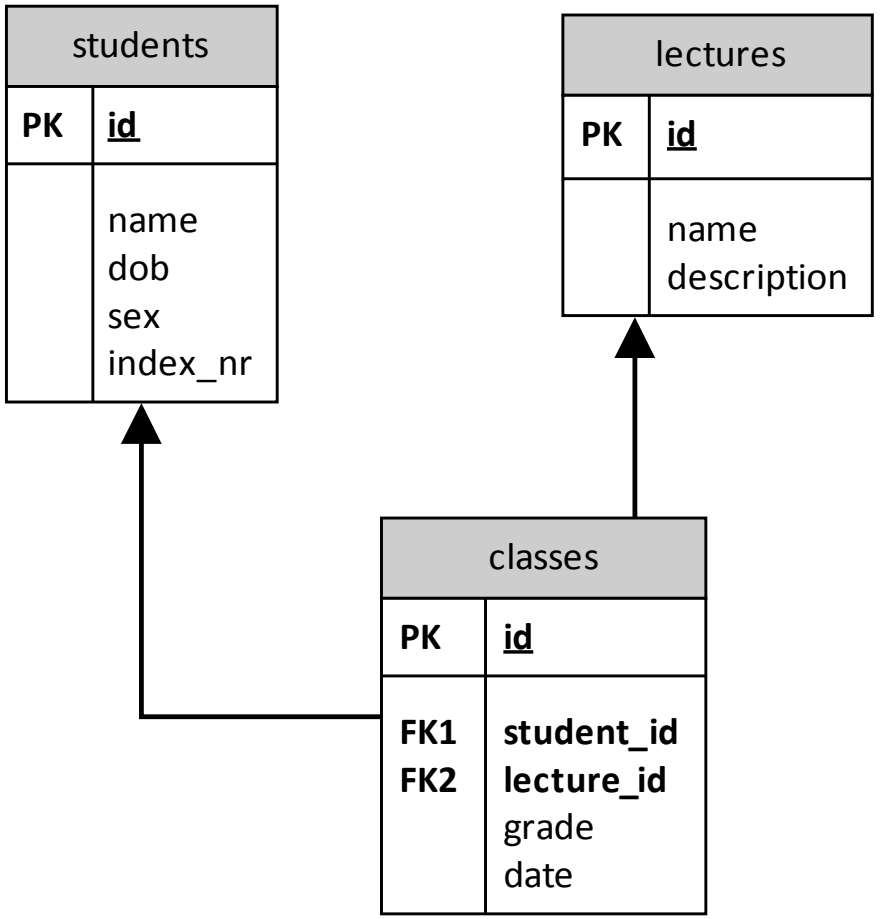
```
mysql> SELECT sex, COUNT(*) AS total FROM students GROUP BY sex;
+-----+-------+
| sex | total |
+-----+-------+
| F   |     3 |
| M   |     3 |
+-----+-------+
2 rows in set (0.00 sec)
```

Korzystanie z relacji

```
mysql> SELECT * FROM students ORDER BY sex, name ASC;
+----+----------------------+------------+-----+----------+
| id | name                 | dob        | sex | index_nr |
+----+----------------------+------------+-----+----------+
|  3 | Aleksandra Malinowska | 1981-04-04 | F   |   123453 |
|  4 | Marta Kosińska       | 1981-07-12 | F   |   123985 |
|  6 | Małgorzata Lis       | 1981-01-11 | F   |   123981 |
|  1 | Jan Kowalski         | 1981-05-20 | M   |   123456 |
|  5 | Michał Kubicki       | 1981-08-12 | M   |   123982 |
|  2 | Roman Kluska         | 1981-07-20 | M   |   123457 |
+----+----------------------+------------+-----+----------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM classes;
+----+------------+------------+-------+------------+
| id | student_id | lecture_id | grade | date       |
+----+------------+------------+-------+------------+
|  1 |          1 |          1 |   4.5 | 2012-04-04 |
|  2 |          2 |          1 |     4 | 2012-04-04 |
|  3 |          3 |          1 |   3.5 | 2012-04-04 |
|  4 |          4 |          1 |   3.5 | 2012-04-04 |
|  5 |          5 |          1 |     5 | 2012-04-04 |
|  6 |          6 |          1 |     3 | 2012-04-04 |
+----+------------+------------+-------+------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM lectures;
+----+------+-------------------------------------+
| id | name | description                         |
+----+------+-------------------------------------+
|  1 | OSM  | Oprogramowanie systemów medycznych |
+----+------+-------------------------------------+
1 row in set (0.00 sec)
```



Wybieranie danych z wielu tabel

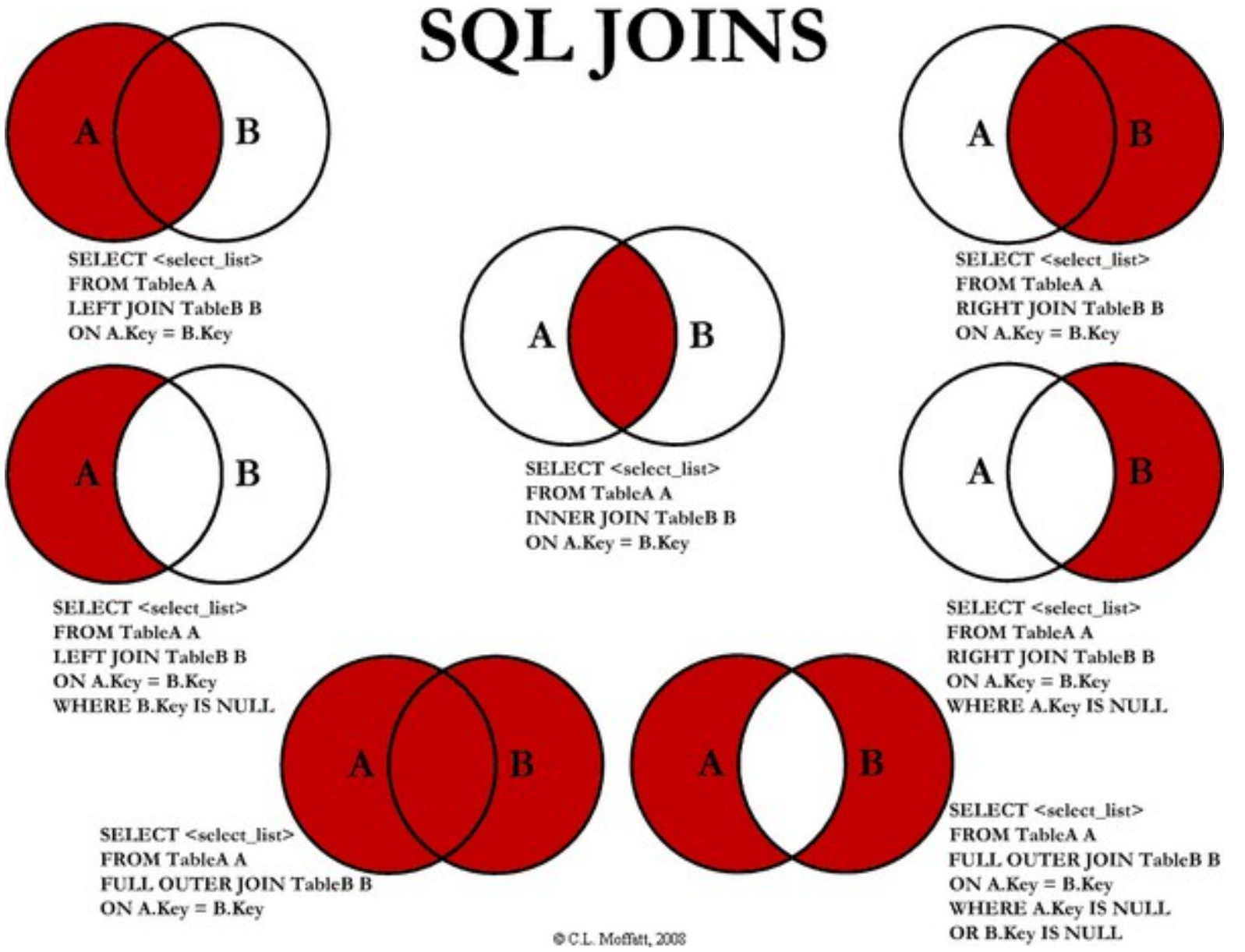
```
mysql> SELECT
          a.grade AS ocena, a.date AS data,
          b.name AS przedmiot,
          c.name AS student
       FROM classes a
       LEFT JOIN lectures b ON a.lecture_id=b.id
       LEFT JOIN students c ON a.student_id=c.id;

+-------+------------+-----------+----------------------+
| ocena | data       | przedmiot | student              |
+-------+------------+-----------+----------------------+
|   4.5 | 2012-04-04 | OSM       | Jan Kowalski         |
|     4 | 2012-04-04 | OSM       | Roman Kluska         |
|   3.5 | 2012-04-04 | OSM       | Aleksandra Malinowska |
|   3.5 | 2012-04-04 | OSM       | Marta Kosińska       |
|     5 | 2012-04-04 | OSM       | Michał Kubicki       |
|     3 | 2012-04-04 | OSM       | Małgorzata Lis       |
+-------+------------+-----------+----------------------+
```

Złączenia tabel

- **INNER JOIN** - zwraca tylko odpowiadające sobie wiersze występujące zarówno w tabeli A i B
- **LEFT JOIN** - zwraca **wszystkie** wiersze tabeli A, oraz odpowiadające wiersze z tabeli B, przy czym nie jest konieczne występowanie odpowiadających wierszy w tabeli B
- **RIGHT JOIN** - zwróci tylko wiersze z tabeli A, które mają wiersze odpowiadające w tabeli B oraz zwróci wszystkie wiersze z tabeli B
- **OUTER JOIN** - zwróci wszystkie wiersze z obydwu tabel

Złączenia tabel - interpretacja graficzna



```
mysql> SELECT c.sex AS sex, AVG(a.grade) AS mean
       FROM classes a
       LEFT JOIN lectures b ON a.lecture_id=b.id
       LEFT JOIN students c ON a.student_id=c.id
       WHERE b.name='OSM'
       GROUP BY c.sex
       ORDER BY mean DESC;

+------+------------------+
| sex  | mean             |
+------+------------------+
| M    |              4.5 |
| F    | 3.333333333333333 |
+------+------------------+
2 rows in set (0.00 sec)
```

Aktualizowanie i usuwanie wierszy

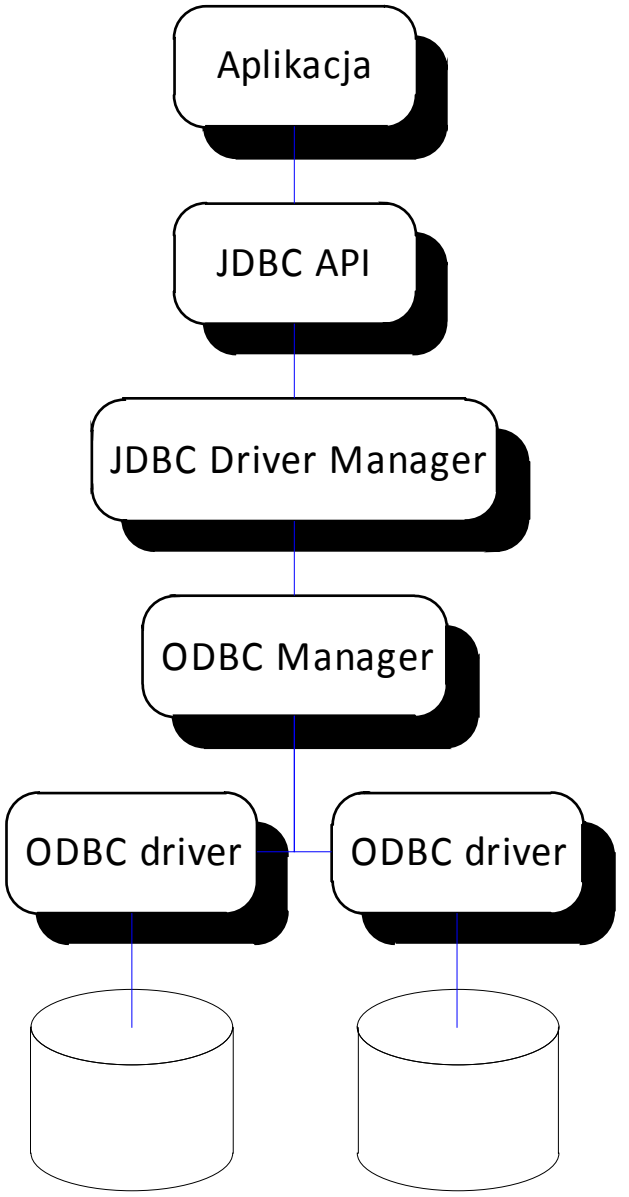
- UPDATE - aktualizuje wartości w wierszach
- DELETE - usuwa wiersze
- TRUNCATE - usuwa wszystkie wiersze z bazy danych, ale pozostawia schemat, indeksy itd.

```
mysql> SELECT * FROM students WHERE index_nr = 123456;
+----+--------------+------------+-----+----------+
| id | name         | dob        | sex | index_nr |
+----+--------------+------------+-----+----------+
|  1 | Jan Kowalski | 1981-05-20 | M   |   123456 |
+----+--------------+------------+-----+----------+
1 row in set (0.00 sec)
```

```
mysql> UPDATE students SET name='John Kowalski' WHERE id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Bazy danych i Java

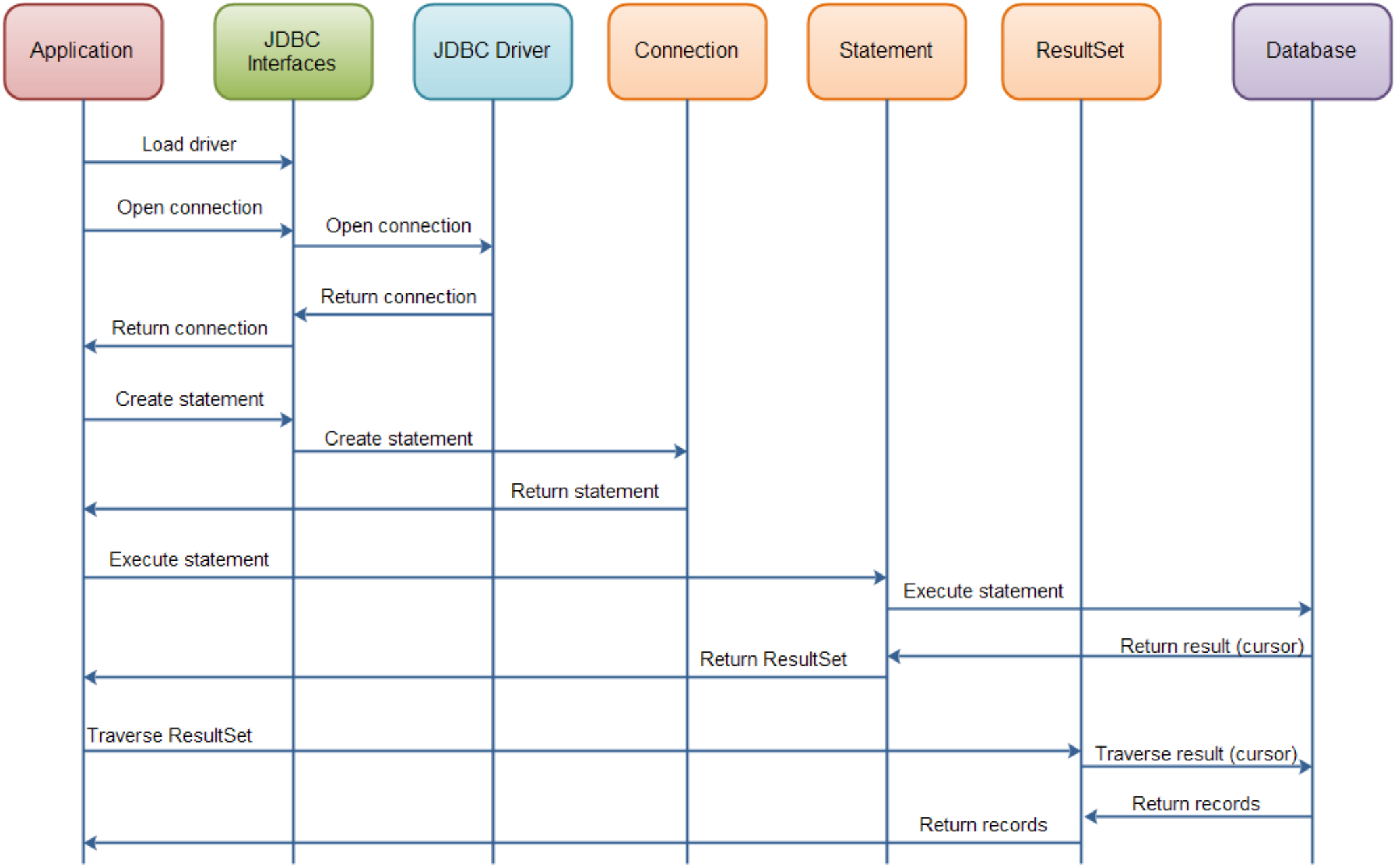
Platforma Java zapewnia standardowy mechanizm łączenia się z relacyjnymi bazami danych, wykonywania zapytań oraz odbierania rezultatów poprzez JDBC (Java Database Connectivity). Dzięki niemu aplikacja może prawie całkowicie uniezależnić się od rodzaju silnika bazy danych (np. MySQL, Java DB, etc.), pod warunkiem dostarczenia odpowiedniego sterownika.



Komponenty JDBC

- **JDBC Driver** - sterownik bazy danych, który jest pośrednikiem (proxy) pomiędzy aplikacją a obsługującym silnikiem bazy danych - standardowy interfejs tłumaczy na specyficzną implementację obsługiwanego silnika. Każdy sinik bazy danych udostępnia własne sterowniki JDBC
- **Connection** - ponieważ baza danych sama w sobie jest oprogramowaniem typu serwer, obiekt Connection reprezentuje ustanowione połączenie (wykorzystujące wybrany sterownik)
- **Statement** - obiekt, wykorzystywany do wykonywania zapytań do bazy danych. Każdy obiekt klasy Statement odpowiada jednemu zapytaniu
- **ResultSet** - obiekt zawierający rezultat wykonanego zapytania, typowo zapewnia możliwość przejścia przez wszystkie wiersze zwrócone przez bazę danych w wyniku wykonania zapytania

Schemat działania



RDBMS w Javie

- JavaDB - implementacja silnika bazy danych

  http://www.oracle.com/technetwork/java/javadb/overview/index.html

- Tryby pracy:
  - **embedded** - prosty dostęp do bazy danych z poziomu aplikacji. Tylko jeden proces może korzystać z bazy danych w tym samym czasie
  - **standalone** - dostęp do bazy danych poprzez połączenie TCP/IP. Wymaga uruchomienia osobnego procesu dla bazy danych. Dostęp może być współdzielony pomiędzy wiele aplikacji

Przykład

```
// install JDBC driver
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

// open a connection to embedded data base
Connection connection
    = DriverManager.getConnection("jdbc:derby:DataBaseName;");

// SQL query string
String sql = "SELECT * FROM students WHERE dob > ?";

// create a Statement object for sending SQL
PreparedStatement statement = connection.prepareStatement( sql );

// prepare the statement
statement.setDate( 1, java.sql.Date.valueOf( "1981-05-19" ) );

// execute querry and obrain a handle to a result
ResultSet result = statement.executeQuery();

// iterate result rows
while ( result.next() )
{
    int    id    = result.getInt( "id" );
    String name  = result.getString( "name" );
    String sex   = result.getString( "sex" );
    Date   bdate = result.getDate( "dob" );

    System.out.printf ( "%d| %11s | %-20s | %10s \n",
            id, name, sex, bdate.toString() );
}

// close the connection
connection.close();
```

Data Access Object

- wzorzec projektowy rozdzielający obiekty od sposobu w jaki zapewniona jest ich "trwałość" (persistance)

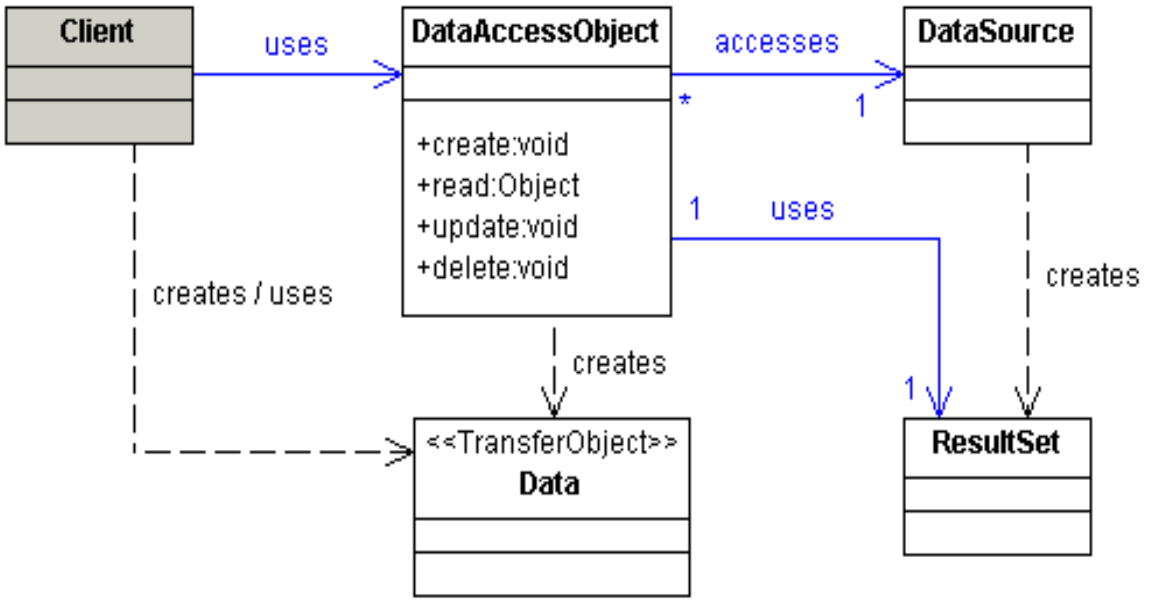


Diagram sekwencji DAO