Работа с релационни бази данни

"Has everyone noticed that all the letters
of the word "database" are
typed with the left hand?
QWERTY
keyboard was designed for even use
of both hands.
It follows, therefore, that writing about
databases is
not only unnatural, but a lot harder than
it appears."

База данни

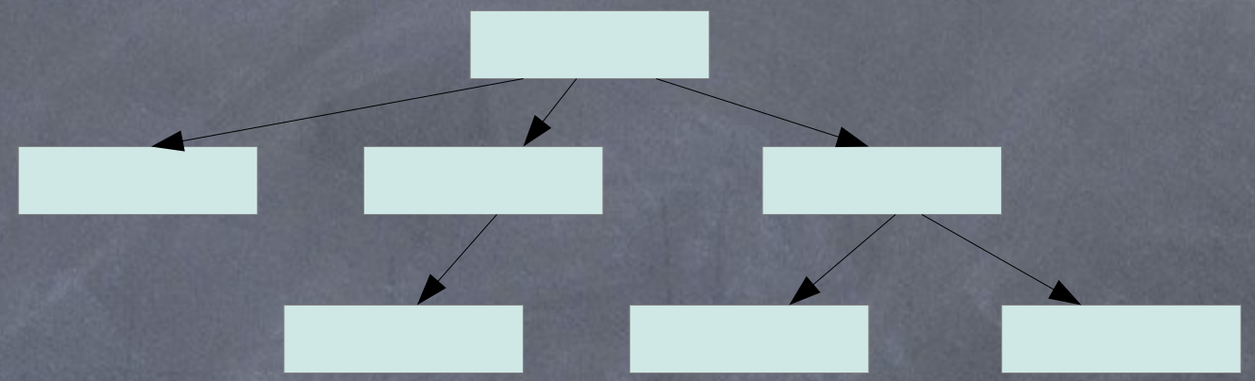
- Колекция от логически свързани данни
- Систематично подредени записи

Видове бази данни

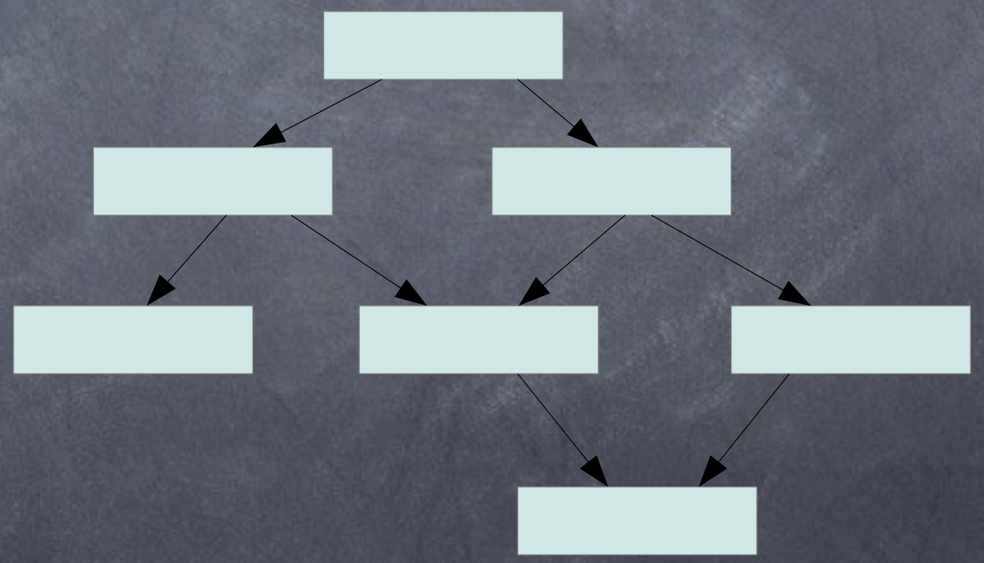
- Йерархични
- Мрежови
- Релационни
- Обектно-ориентирани

Видове бази данни

Йерархични



Мрежови



Релационни бази данни

- Съхранение на данни във вид на релации (връзки)

Автомобили			
ID	**Марка**	**Рег. номер**	**Собственик**
123	25	C1244CC	12
333	25	0000001	88
244	55	CA1111CA	120

Марки	
ID	**Име**
25	BMW
34	Mercedes
55	Audi

Клиенти			
ID	**ЕГН**	**Име**	**Фамилия**
12	1234567899	Иван	Иванов
23	9876554321	Петър	Петров

Основни понятия при релационните бази данни

- Таблици
- Колони
- Записи (редове)
- Релации
  - Първичен ключ
  - Вторичен ключ

Системи за управление на бази от данни (DBMS)

- Компютърни програми
- Контролират създаването, обработката и използването на базите от данни

Системи за управление на бази от данни

- Oracle
- MySQL
- IBM DB2
- Microsoft SQL Server
- PostgreSQL
- SQLite

...

Structured Query Language (SQL)

- Език за работа с релационни бази от данни
- Най-използваният език за работа с бази от данни
- Стандарт, имплементиран от всяка RDBMS
- Няма преносимост между отделните RDBMS

SQL - Заявки (Query)

- Извличане на данни по зададени критерии

- SELECT
- FROM
- WHERE
- GROUP BY
- ORDER BY

```
SELECT *
FROM cars
WHERE registration_date > '2010-01-01'
GROUP BY owner_id
ORDER BY registration_date;
```

SQL - Извличане на данни от повече таблици

- JOIN

```
SELECT cars.reg_number,
   brands.name
   owners.first_name,
   owners.last_name
FROM cars
JOIN brands ON cars.brand = brands.id
JOIN owners ON cars.owner = owners.id
WHERE registration_date > '2010-01-01';
```

SQL - Манипулация на данни

- Създаване, редакция и изтриване на данни

- INSERT
- UPDATE
- DELETE

```
INSERT INTO cars (brand, owner, reg_number)
VALUES (12, 55, 'C 1111 CA');
```

```
DELETE FROM cars WHERE id = 45;
```

```
UPDATE cars SET owner = 22 WHERE id = 15;
```

SQL - Дефиниция на данни

- Управление на таблици и индекси

- CREATE
- ALTER
- TRUNCATE
- DROP

```
CREATE TABLE car_models (
    id INT,
    brand INT,
    name VARCHAR(50) NOT NULL,
    PRIMARY KEY (id)
);

ALTER TABLE cars ADD model INT NOT NULL;

TRUNCATE TABLE customers;

DROP TABLE payments;
```

SQL - Транзакции

- Набор от SQL заявки, извършващи логическа единица работа

- Завършват с:
  - COMMIT – изпълнения на заявките
  или
  - ROLLBACK – анулиране на промените

- Заключване на таблици

SQL - Процедури (Stored Procedures)

- Група от SQL твърдения, достъпвани по име
- Малки програми, които изпълняват определена задача
- Приемат входящи параметри
- Може да връщат резултат

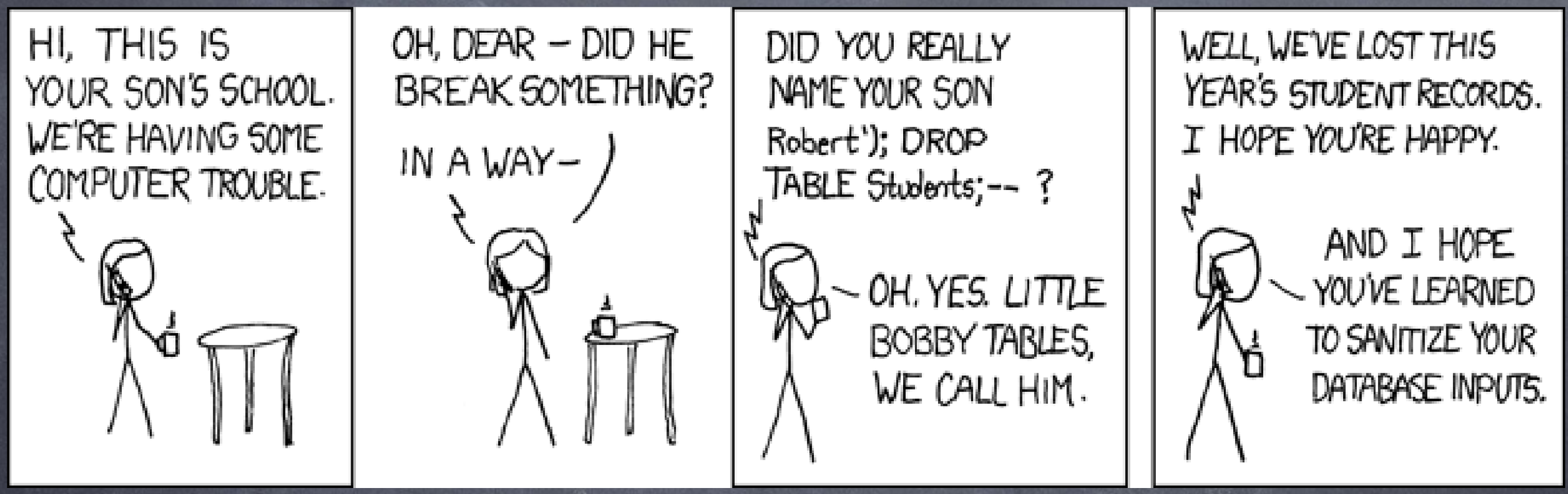
Метаданни

- Данни за данните
- Съдържат се в системни таблици
- Данни за:
  - Имена на таблиците
  - Колони във всяка таблица
  - Процедури
  - Първични и вторични ключове

  ...

Result Sets и Курсори

- Result Set – резултат от изпълнението на дадена заявка; редовете, които удовлетворяват условието на заявката;
- Курсор – указател към редовете на резултата


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY-
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

JDBC

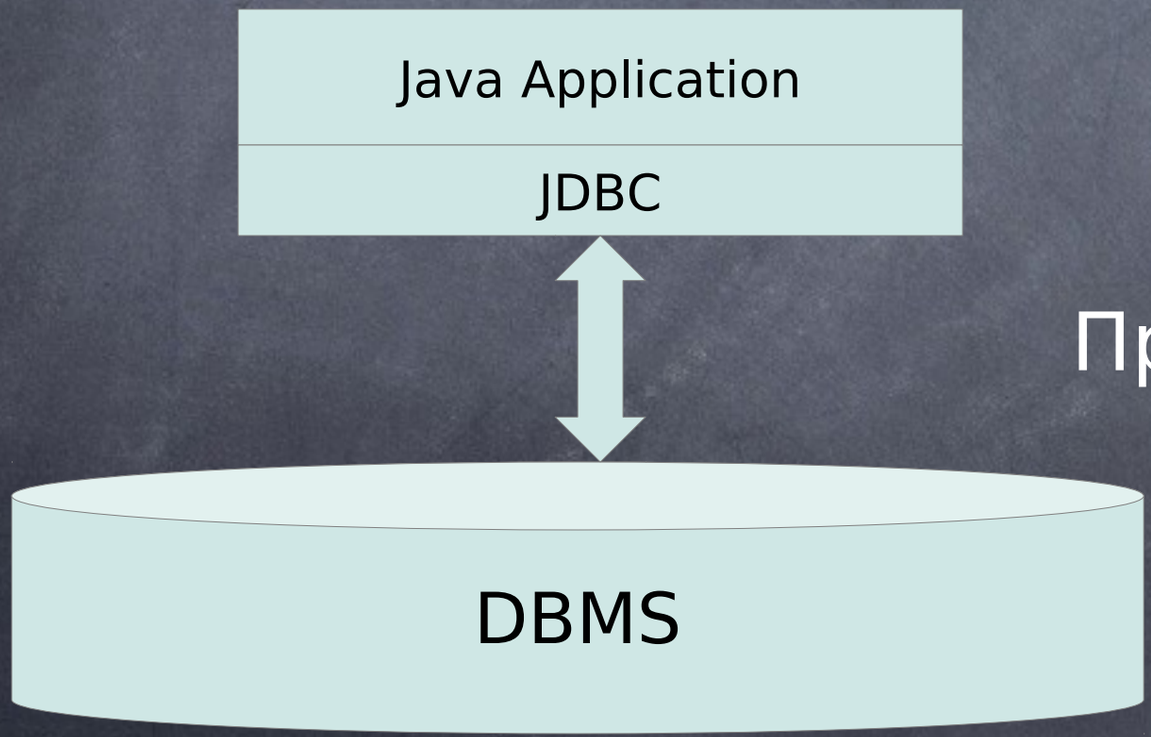
- Java API за работа с бази данни
  - Свързване с база данни
  - Изпращане на заявки и UPDATE statements
  - Извличане и обработка на резултати от заявките

Компоненти на JDBC

- JDBC API
  - java.sql
  - javax.sql
- JDBC Driver Manager
  - класът DriverManager
  - класът DataSource
- JDBC Test Suite
- JDBC-ODBC Bridge

JDBC - Двуслойна (two-tier) архитектура

- Приложението си "говори" директно с базата данни



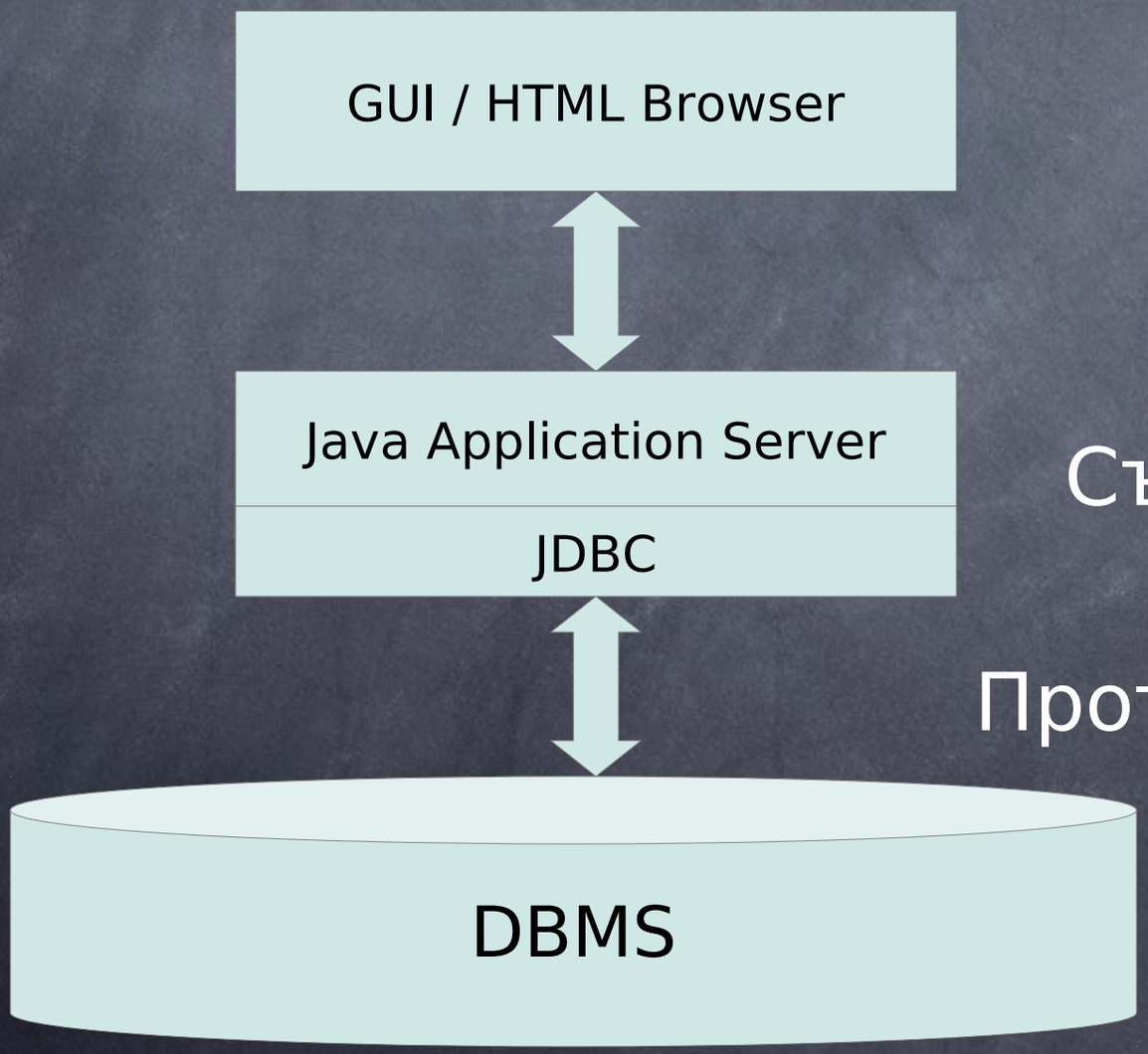
Клиент

Протокол за връзка с БД

DB сървър

JDBC - Трислойна (three-tier) архитектура

- "Междинен слой"



Клиент

Сървър (бизнес логика)

Протокол за връзка с БД

DB сървър

JDBC

- Инсталирайте си DBMS
- Примерите в лекцията са с MySQL
- Инсталирайте си JDBC driver за DBMS (за MySQL това е Connector/J)
  - http://www.mysql.com/downloads/connector/j/
  - Свалете си архива и го сложете в някое от:
    - като настройка на IDE-то
    - в Classpath
    - в ext директорията на JDK

Последователност от стъпки при JDBC

- Установяване на връзка с базата данни
- Създаване на SQL statement
- Изпълнение на заявката
- Обработка на резултата от изпълнение на заявката – обектът ResultSet
- Затваряне на връзката към базата

Връзка с базата данни

- Чрез класа DriverManager
  - методът getConnection()
  - connection URL – има различен формат в зависимост от конкретната СУБД
- Чрез DataSource
  - Connection Pool

Пример

```
public Connection getConnection() throws SQLException {
    Connection conn = null;
    Properties connectionProps = new Properties();
    connectionProps.put("user", this.userName);
    connectionProps.put("password", this.password);

    String currentUrlString = null;

        if (this.dbms.equals("mysql")) {
            currentUrlString = "jdbc:"+this.dbms+"://" + this.serverName+":"
+this.portNumber+"/";
            conn = DriverManager.getConnection(currentUrlString, connectionProps);

            this.urlString = currentUrlString + this.dbName;
            conn.setCatalog(this.dbName);

            System.out.println("Connected to database");
        }

        return conn;
    }
```

SQL Statements

- Създаваме Statement обекти от Connection
- Изпълняваме Statement
- Резултатът от изпълнението е ResultSet

- 3 типа:
  - Statement – прост SQL без параметри
  - PreparedStatement – SQL твърдение, което може да приема параметри
  - CallableStatement – за изпълнение на stored процедури

```
Statement stmt = con.createStatement();
```

Изпълнение на заявки

- Изпълнява се Statement с някой от методите:
  - execute – изпълнява заявката и връща true, ако има резултат от изпълнението
  - executeQuery – връща резултата като ResultSet
  - executeUpdate – връща брой редове, променени от INSERT, UPDATE или DELETE.

```
ResultSet rs = stmt.executeQuery(query);
```

Обработка на ResultSet

- Обработка с курсор
- Този курсор е различен от курсора на БД!

```
try {
        stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            String brandName = rs.getString("brand_name");
            String modelName = rs.getString("model_name");
            int carId = rs.getInt("car_id");
            System.out.println(carId + "\t"
                + brandName + "\t" + modelName);
        }
}
```

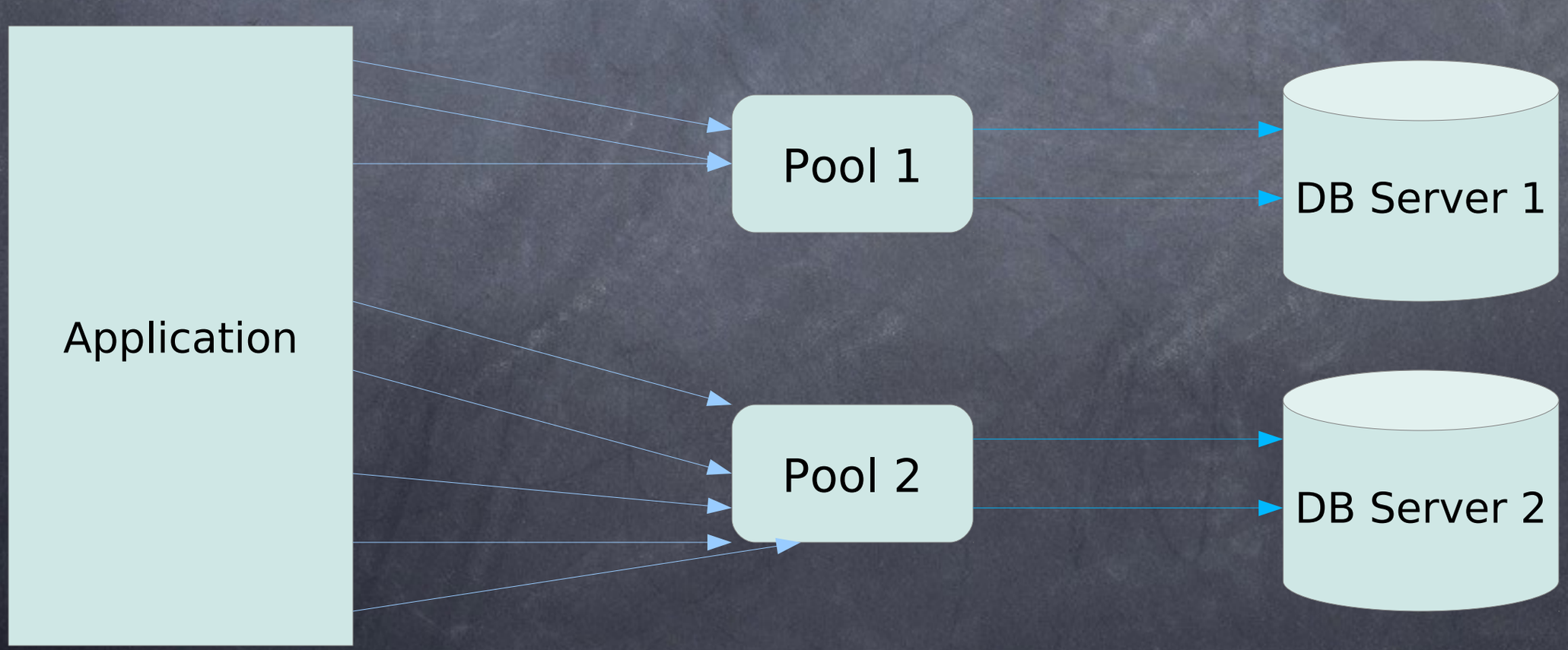
Затваряне на връзката с БД

- Затваряне на връзката в блока finally

```
try(Statement stmt = con.createStatement()) {
      ResultSet rs = stmt.executeQuery(query);
      while (rs.next()) {
            // ...
      }
} catch (SQLException e) {
      // ...
} finally {
      if (stmt != null) {
            stmt.close();
      }
}
```

Connection Pool

- Кеш на конекции към бази данни
- Конекциите могат да бъдат преизползвани
- Ако наличните конекции са заети, се създава нова
- Ускорява получаването на връзка към базата данни



DataSource

- Предпочитаният начин за връзка с бази данни с JDBC
- Connection pooling

- Създава се инстанция на DataSource
- Задават се настройки за достъп до базата данни
- Регистрира се този DataSource
- При използване, се търси регистрирания DataSource

Създаване и регистриране на DataSource

```
com.dbaccess.BasicDataSource dataSource =
    new com.dbaccess.BasicDataSource();
dataSource.setServerName("productionserver");
dataSource.setDatabaseName("leasing_db");
dataSource.setDescription("Leasing Database");
dataSource.setUserName("user");
dataSource.setPassword("secret");

Context ctx = new InitialContext();
ctx.bind("jdbc/leasingDB", dataSource);
```

Използване на DataSource

```
Context ctx = new InitialContext();
DataSource dataSource =
  (DataSource)ctx.lookup("jdbc/leasingDB");

Connection connection = dataSource.getConnection();
```

Благодаря за вниманието!