

DBVS ir DB taikomųjų programų architektūroje

DBVS

TAIKOMŲJŲ PROGRAMŲ ARCHITEKTŪRA

DBVS IR TAIKOMŲJŲ PROGRAMŲ KOMUNIKACIJA

DBVS atsiradimo šaltiniai

Techninės įrangos pardavėjai – siekiant padidinti mainframe tipo kompiuterių pardavimus, nes jie buvo parduodami su juos lydinčia programine įranga.

IBM: IMS, SQL/DS, DB2; **HP:** Image, Allbase ;

GE / Honeywell: IDS; **DEC:** DBMS-10, RDB ; **Sperry:** DMS 1100

Universitetinės ir valstybinės tyrimų laboratorijos – kuriama programinė įranga, kaip techninių tyrimų projektų dalis.

UC, Berkley: INGRES; **Univ. of Texas:** TDMS, System 2000 (Intel Corp.) ;

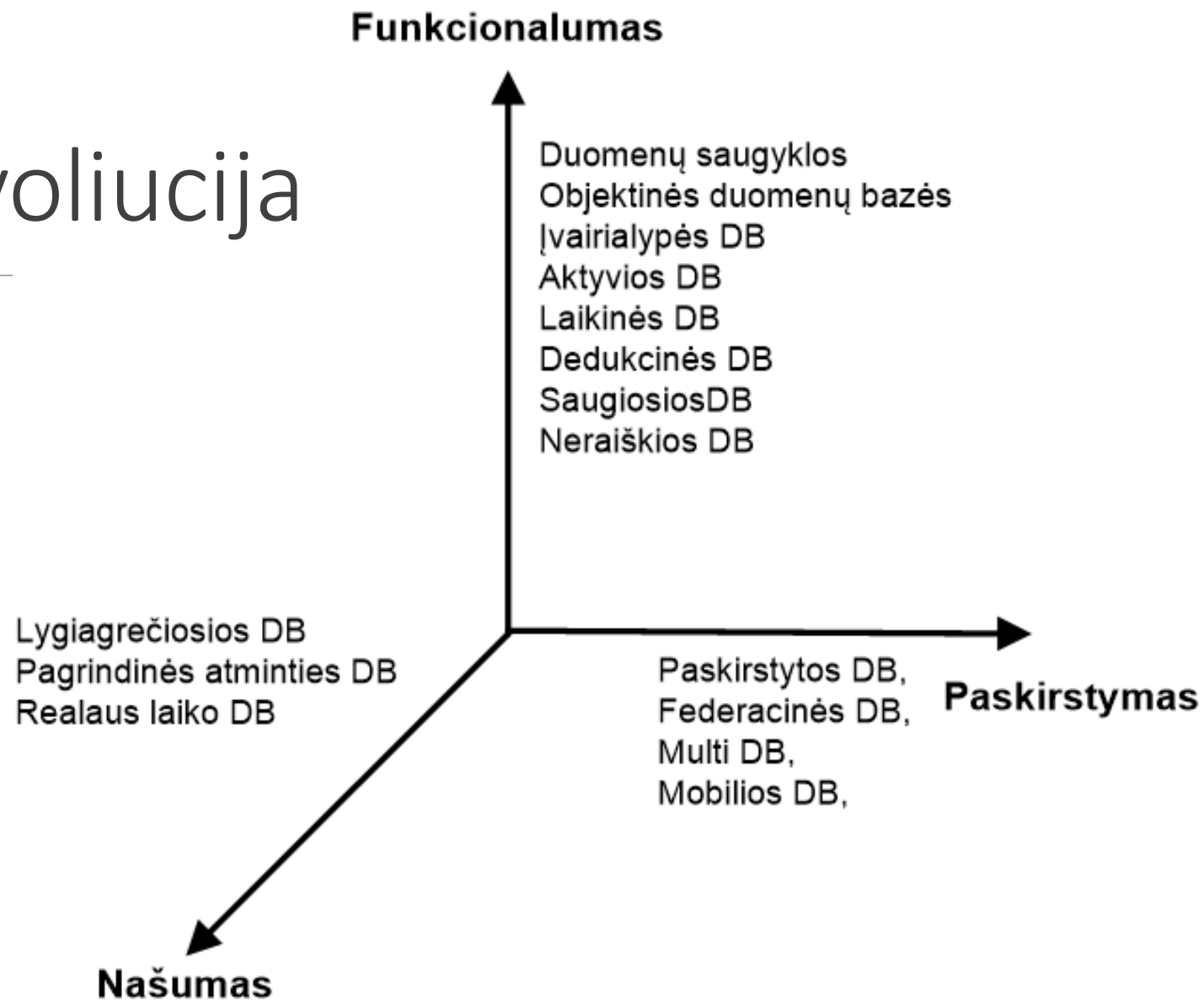
NASA: GUAM (IBM), RIM (Boeing Aerospace);

Programinės įrangos kūrėjai – kaip pagrindinė jų veiklos sritis.

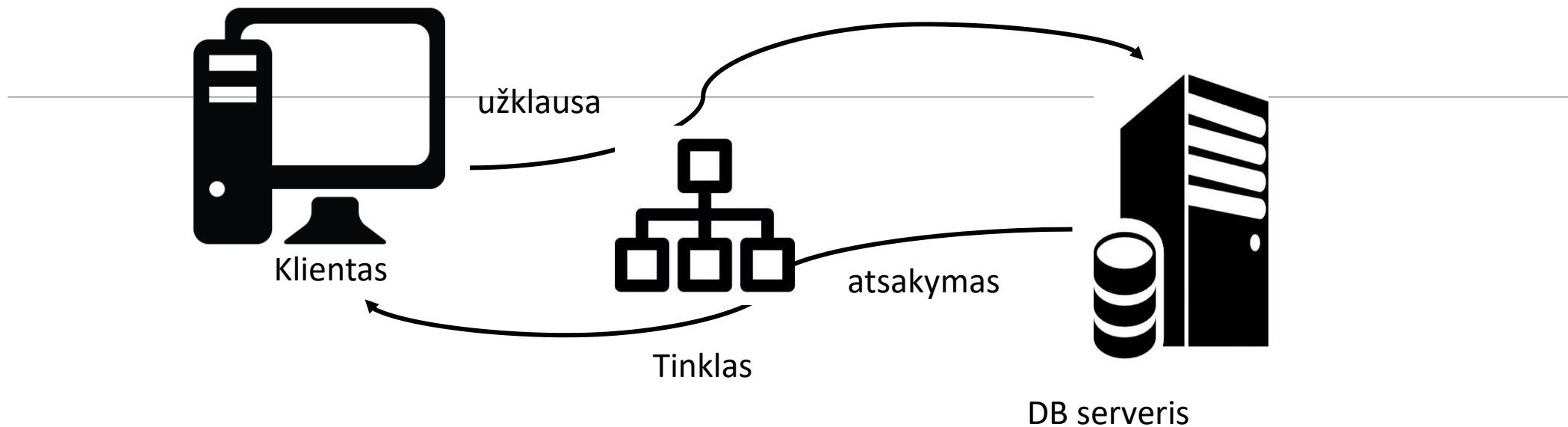
Cullinet Software: IDMS (mainframe); **Oracle Corp.:** Oracle (originaly, mini kompiuteriai); **Ashton-Tate:** dBase (mikrokompiuteriai); **Microrim:** R:BASE ;

Borland: Paradox; **Microsoft:** MS SQL

DBVS evoliucija

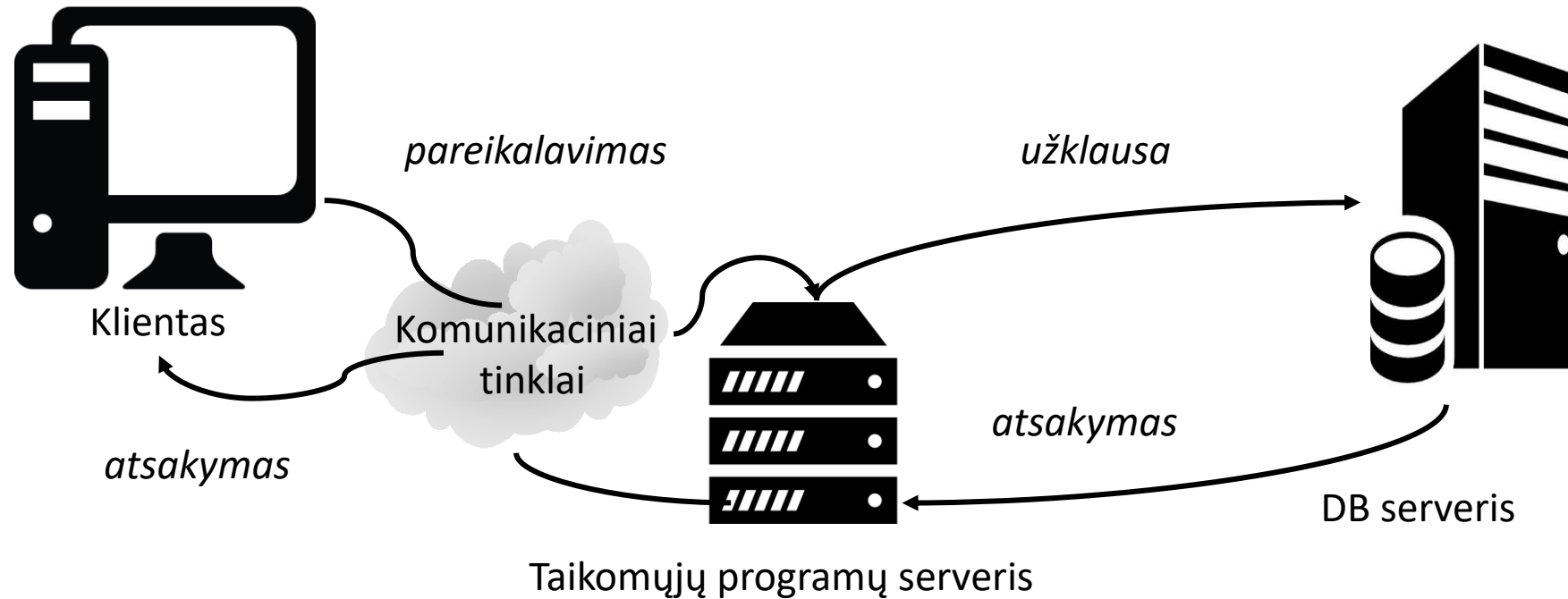


Kliento-serverio architektūra



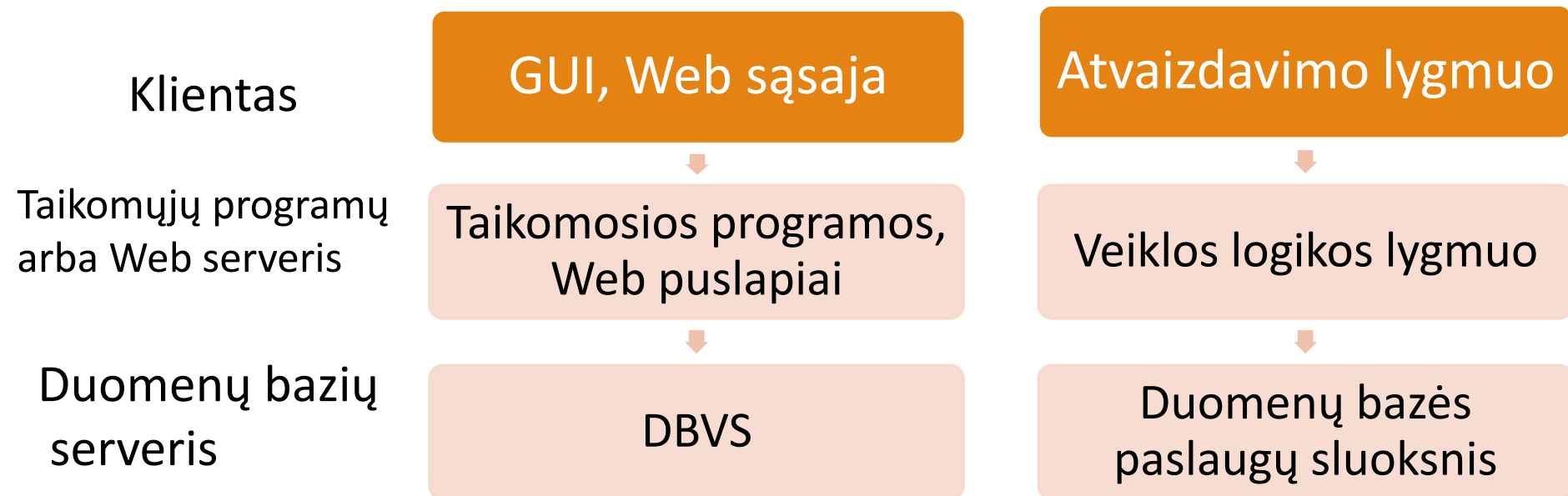
- Sistemos architektūros modelis nedidelėse ir vidutinėse darbo laukio sistemose.
- Pagrindinis duomenų apdorojimo krūvis tenka klientiniam PJ (end-user)

Trijų lygių architektūra

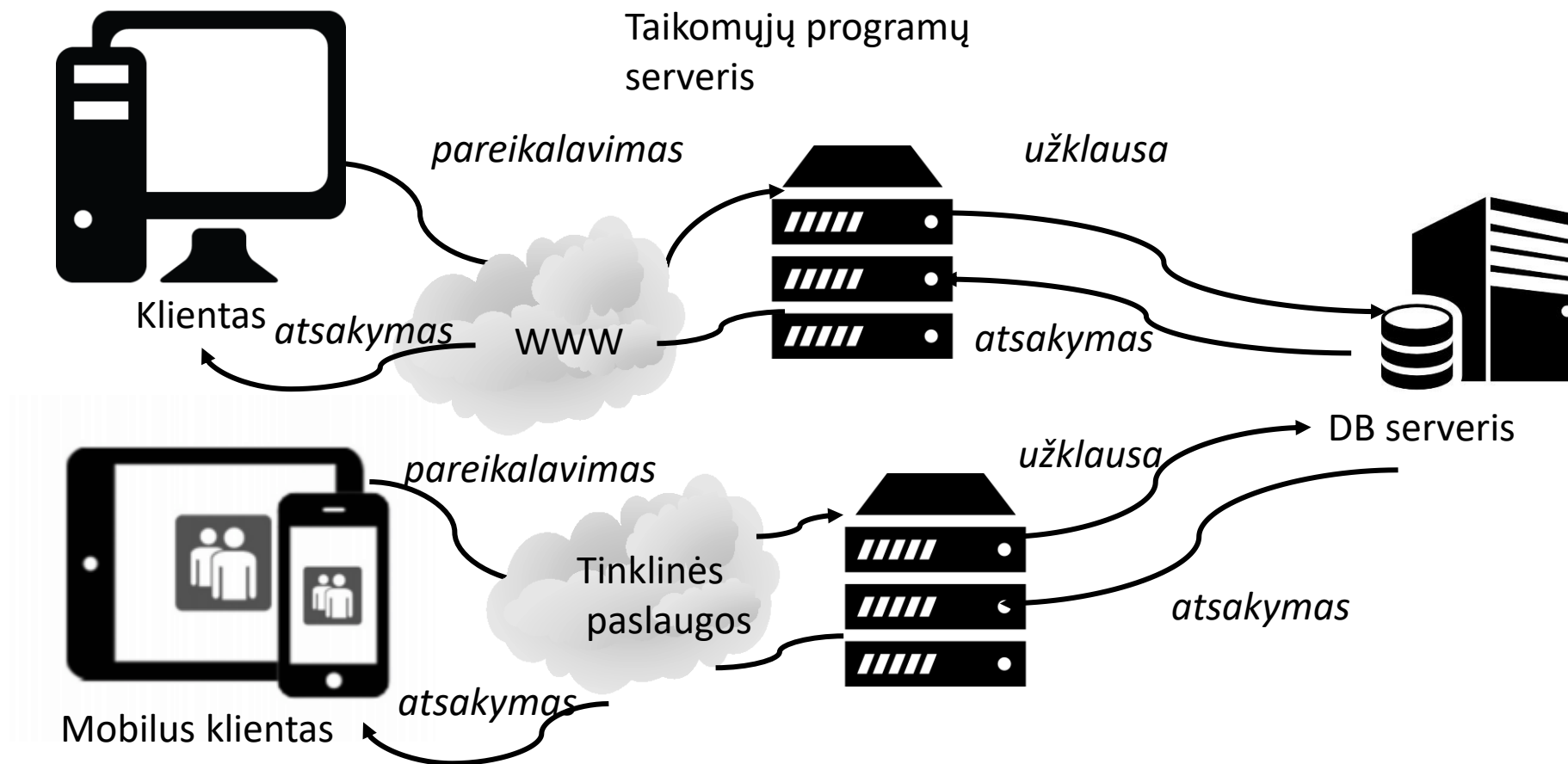


Dažniausia naudojama sistemų architektūra

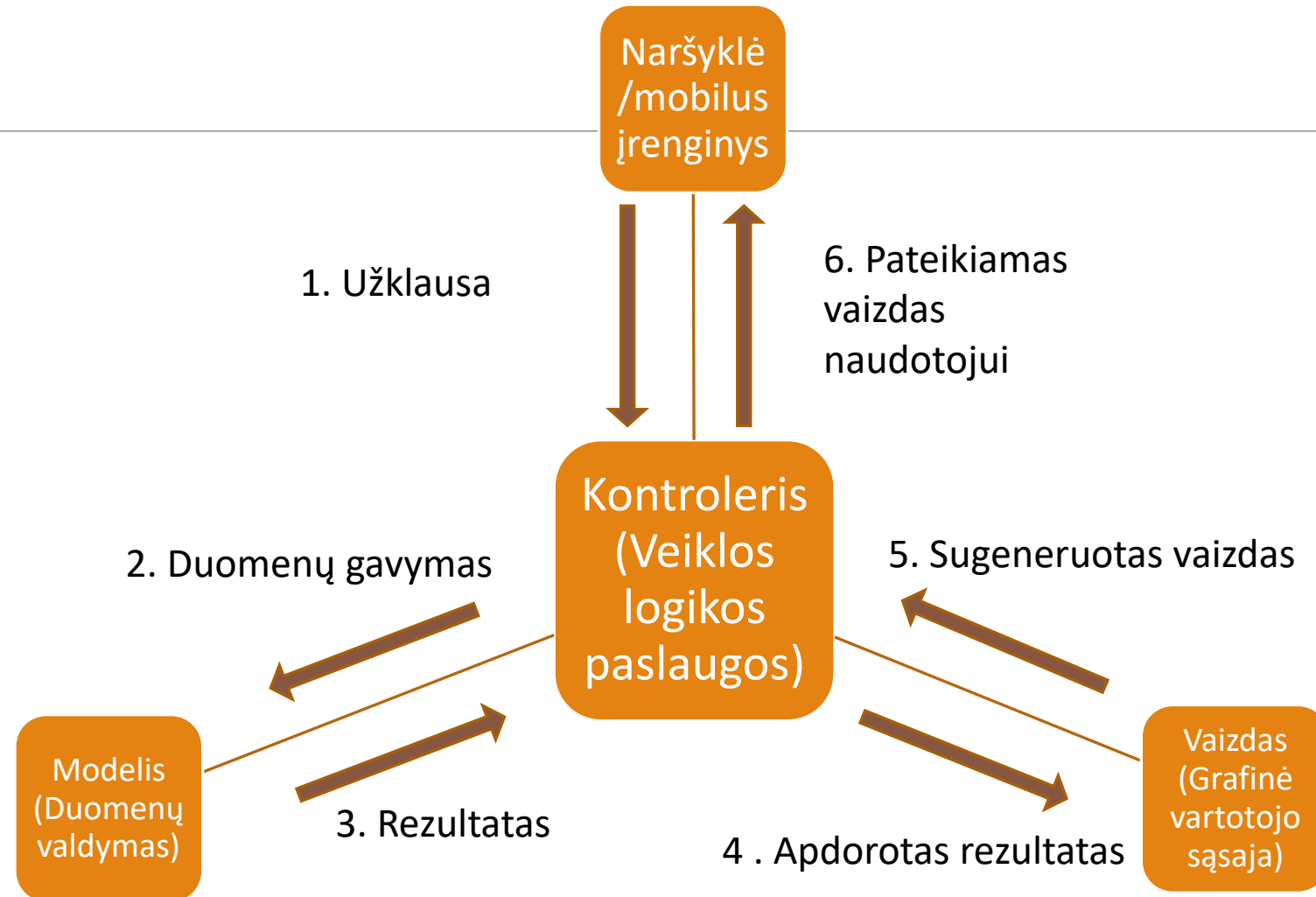
Trijų lygių architektūra









Daugialypė n lygių architektūra



Trijų lygių MVC karkasas



Pavyzdinė IS Autonuoma Php

Katalogas/failas	Paskirties aprašymas
 <i>controls</i>	Talpina valdiklių <i>PHP</i> failus.
 <i>libraries</i>	Talpina dalykinės srities klasių <i>PHP</i> failus, kuriose aprašomos <i>SQL</i> užklausos.
 <i>scripts</i>	Talpina <i>JavaScript</i> failus, skirtus vartotojo sąsajos įvykiams programuoti.
 <i>style</i>	CSS vartotojo sąsajos stilių katalogas.
 <i>templates</i>	Talpina sistemos langų (puslapių) <i>HTML</i> šablonų (t. y. vartotojo grafinės sąsajos) failus.
 <i>utils</i>	Talpina pagalbines klases
<i>index.php</i>	Pagrindinis <i>PHP</i> failas. Šiame faile aprašytą kodą <i>PHP</i> interpretatorius įvykdo kiekvienai IS puslapio naršyklės užklausiai apdoroti.
<i>config.php</i>	Konfigūracijos duomenų failas.
<i>.htaccess</i>	<i>Apache</i> serverio nustatymų failas (neprivalomas).
<i>autonuoma.sql</i>	Duomenų bazės <i>DDL</i> failas importavimui į DBVS.

Pavyzdinė IS Autonuoma C#

📁 .vscode	Visual Studio Code IDE nustatymų failai.
└─ launch.json	Visual Studio Code IDE sutvorto kodo paleidimo (angl. <i>launch</i>) konfigūracijos.
└─ tasks.json	Visual Studio Code IDE mašininio kodo tvėrimo (angl. <i>build</i>) konfigūracijos.
📁 Autonuoma	Pavyzdinės sistemos projekto šakninis katalogas.
└─ 📁 bin	Sutvertas mašininis kodas.
└─ 📁 obj	Darbiniai failai mašininio kodo tvėrimo įrankiams.
└─ 📁 Controllers	Valdiklių kodo katalogas.
└─ 📁 Models	Duomenų esybių ir esybių vaizdų formose modelių katalogas.
└─ 📁 Repositories	Duomenų esybių ir esybių vaizdų modelių susiejimo su duomenų baze kodo katalogas.
└─ 📁 Views	Formų HTML vaizdų kodo katalogas.
└─ 📁 wwwroot	Statinių pavyzdinės sistemos svetainės resursų katalogas.
└─ .gitignore	Failas nurodantis ko netraukti į GIT versijavimo sistemos saugyklą.
└─ appsettings.json	Nustatymai.
└─ Autonuoma.csproj	Pavyzdinės sistemos projekto aprašas.
└─ Config.cs	Nustatymų programinio užkrovimo kodas.
└─ Program.cs	Pavyzdinės sistemos įėjimo taškas.
└─ Sql.cs	Pagalbinės funkcijos darbui su MySQL/MariaDB duomenų baze.
autonuoma.sln	Pavyzdinės sistemos sprendimo projektų jungiantysis failas. Jame užregistruotas pavyzdinės sistemos projektas.

Pavyzdinė IS Autonuoma C#

▼ AUTONUOMA-DNET

- > .vscode
- ▼ Autonuoma
 - > bin
 - ▼ Controllers
 - AutomobilisController.cs
 - DarbuotojasController.cs
 - HomeController.cs
 - KlientasController.cs
 - MarkeController.cs
 - ModelisController.cs
 - PaslaugaController.cs
 - ReportsController.cs
 - SutartisF2Controller.cs
 - SutartisF3Controller.cs
 - > Models
 - > obj
 - > Repositories
 - > Views
 - > wwwroot
 - .gitignore
 - appsettings.json
 - Autonuoma.csproj
 - Config.cs
 - Program.cs
 - Sql.cs
 - autonuoma.sln

▼ AUTONUOMA-DNET

- > .vscode
- ▼ Autonuoma
 - > bin
 - > Controllers
 - ▼ Models
 - Aikstele.cs
 - Automobilis.cs
 - ContractsReport.cs
 - Darbuotojas.cs
 - Klientas.cs
 - LateContractsReport.cs
 - Marke.cs
 - Modelis.cs
 - Paslauga.cs
 - PaslaugosKaina.cs
 - ServicesReport.cs
 - SutartisF2.cs
 - SutartisF3.cs
 - > obj
 - > Repositories
 - > Views
 - > wwwroot
 - .gitignore
 - appsettings.json
 - Autonuoma.csproj
 - Config.cs
 - Program.cs
 - Sql.cs
 - autonuoma.sln

▼ AUTONUOMA-DNET

- > .vscode
- ▼ Autonuoma
 - > bin
 - > Controllers
 - > Models
 - > obj
 - ▼ Repositories
 - AiksteleRepo.cs
 - AtaskaitaRepo.cs
 - AutomobilisRepo.cs
 - DarbuotojasRepo.cs
 - KlientasRepo.cs
 - MarkeRepo.cs
 - ModelisRepo.cs
 - PaslaugaRepo.cs
 - PaslaugosKainaRepo.cs
 - SutartisF2Repo.cs
 - SutartisF3Repo.cs
 - > Views
 - > wwwroot
 - .gitignore
 - appsettings.json
 - Autonuoma.csproj
 - Config.cs
 - Program.cs
 - Sql.cs
 - autonuoma.sln

▼ AUTONUOMA-DNET

- > .vscode
- ▼ Autonuoma
 - > bin
 - > Controllers
 - > Models
 - > obj
 - > Repositories
 - ▼ Views
 - > Automobilis
 - > Darbuotojas
 - > Home
 - > Klientas
 - > Marke
 - > Modelis
 - > Paslauga
 - > Reports
 - > SutartisF2
 - > SutartisF3
 - _Layout.cshtml
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - > wwwroot
 - .gitignore
 - appsettings.json
 - Autonuoma.csproj
 - Config.cs
 - Program.cs
 - Sql.cs
 - autonuoma.sln

Pavyzdinė IS Autonuoma

Index.php pagrindinis kontrolieris, kuris priima visas vartotoju užklausas.

Pirmiausia užkraunamas **config.php** , jame pateikiami prisijungimo duomenys,

Taip pat reikia atkreipti dėmesį į prefiksus, visos lentelės turi būti su prefiksais, jei nori kurti savo db,

Reikia pasikeisti prefiksus, kad sistema veiktų korektiškai ir dirbtų su tomis lentelėmis, kurios turi konkretų prefix'ą.

Darbai su DB naudojami **utils/mysql.class.php** Yra parašyti klase darbai su db, kuri naudoja mysqli biblioteka ir turi eilę reikalingų funkcijų.

controls (Controller) kataloge yra kontrolieriai

libraries (Model) yra modelė (duomenų) klasės

templates (View) yra view klasės (atvaizdavimo) klasės.

Darbo logika ir eiga:

Pirmiausia kraunasi index.php kaip pagrindinis kontrolieris ir kviečia pagrindinį šabloną

Jis tada pasikrauna main.ptl.php,

Pavyzdinė IS Autonuoma

Index.php pagrindinis kontrolieris, kuris priima visas vartotoju užklausas.

Pirmiausia užkraunamas **config.php** , jame pateikiami prisijungimo duomenys,

Taip pat reikia atkreipti dėmesį į prefiksus, visos lentelės turi būti su prefiksais, jei nori kurti savo db,

Reikia pasikeisti prefiksus, kad sistema veiktų korektiškai ir dirbtų su tomis lentelėmis, kurios turi konkretų prefix'ą.

Darbai su DB naudojami **utils/mysql.class.php** Yra parašyti klase darbai su db, kuri naudoja mysqli biblioteka ir turi eilę reikalingų funkcijų.

controls (Controller) kataloge yra kontrolieriai

libraries (Model) yra modelė (duomenų) klasės

templates (View) yra view klasės (atvaizdavimo) klasės.

Darbo logika ir eiga:

Pirmiausia kraunasi index.php kaip pagrindinis kontrolieris ir kviečia pagrindinį šabloną

Jis tada pasikrauna main.ptl.php,

Pavyzdinė IS Autonuoma

O jau pasirinkus konkretu meniu yra pakraunamas (iškviečiamas) konkretaus modulio kontroleris

```
<?php
```

```
// sukuriame sutarčių klasės objektą, užkraunamos duomenų klasės
```

```
include 'libraries/contracts.class.php';
```

```
$contractsObj = new contracts();
```

```
// suskaičiuojame bendrą įrašų kiekį
```

```
$elementCount = $contractsObj->getContractListCount();
```

```
// sukuriame puslapiavimo klasės objektą
```

```
include 'utils/paging.class.php';
```

```
$paging = new paging(config::NUMBER_OF_ROWS_IN_PAGE);
```

```
// suformuojame sąrašo puslapius
```

```
$paging->process($elementCount, $pageId);
```

```
// išrenkame nurodyto puslapio sutartis
```

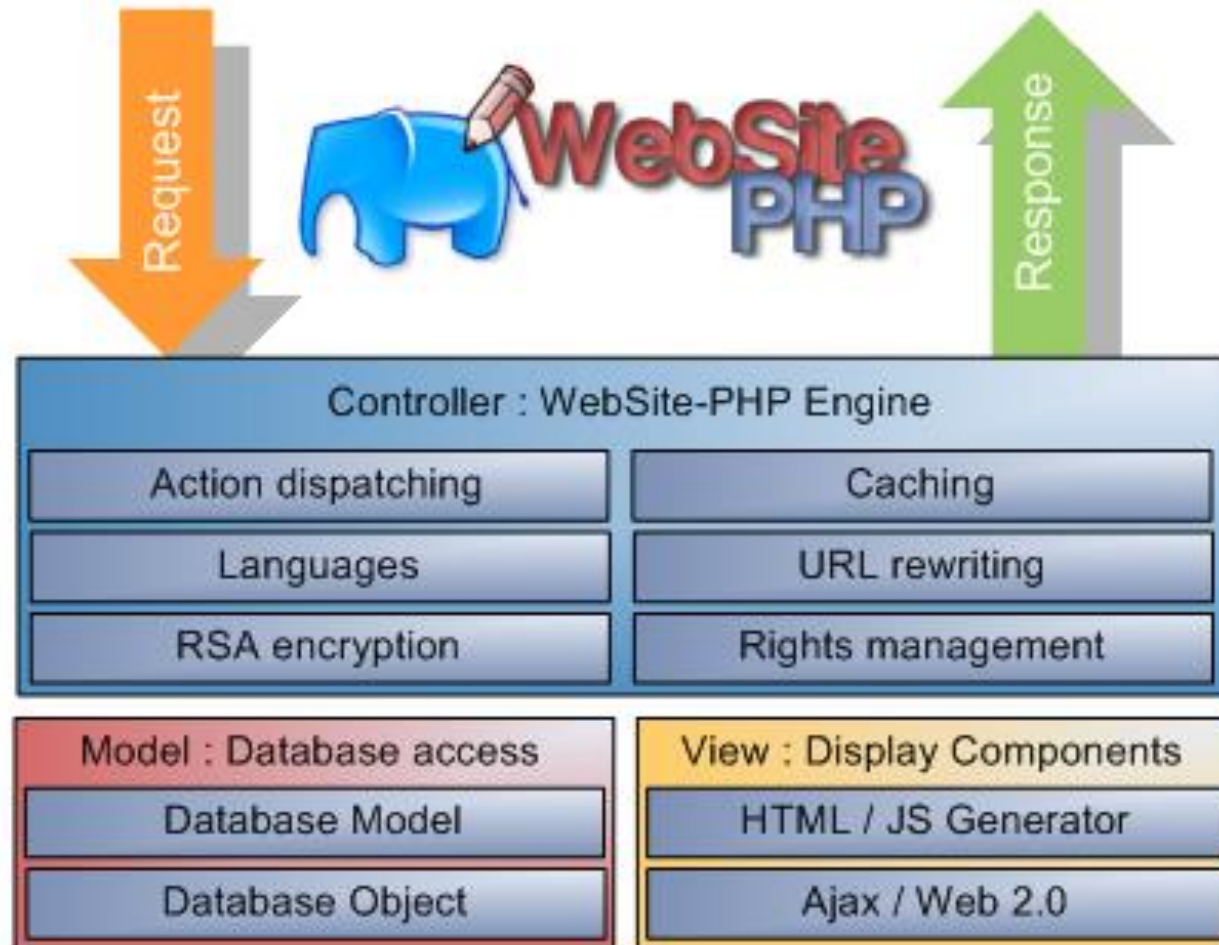
```
$data = $contractsObj->getContractList($paging->size, $paging->first);
```

```
// įtraukiame šabloną
```

```
include 'templates/contract_list.tpl.php';
```

```
?>
```

Triukšmingas MVC keršelis (PHP)



MY SQL architektūra

Taikomųjų programų lygmuo

- Užtikrina bendrąjį naudotojų autentifikavimą ir jų teisių pajungimą;
- Pj klientinės dalies komunikacija su DBVS.

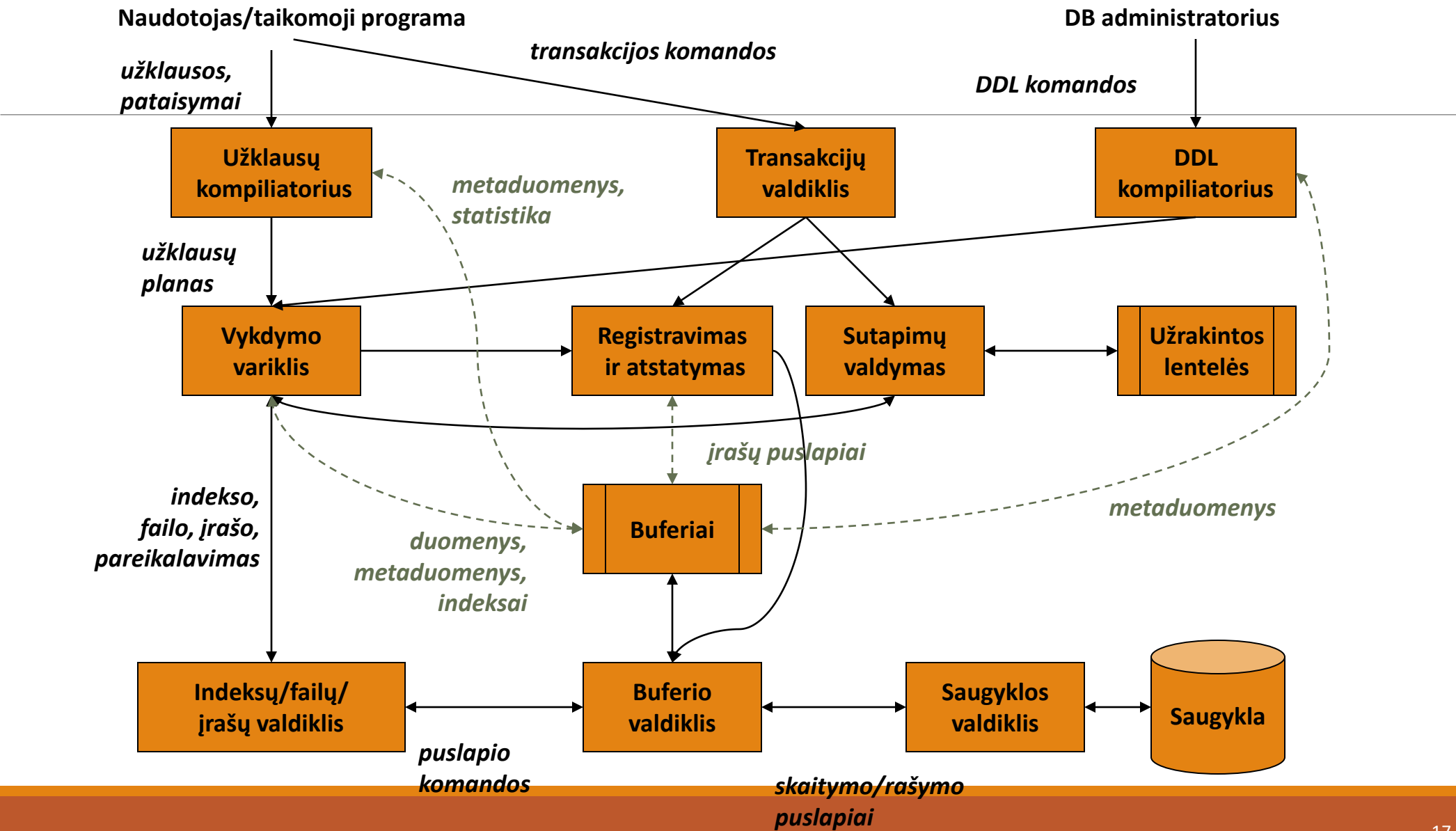
Loginis lygmuo

- Užklausų apdorojimas, analizė, kešavimas ir t.t.
- Papildomų funkcijų palaikymas;

Fizinis lygmuo

- Atsakingas už duomenų saugojimą ir gavimą iš MySQL;
- Šis lygmuo užtikrina DBVS variklio realizavimą.

DBVS komponentai



Instrukcijų (kalbų) grupės

DDL (angl. *Data-Definition Language*) – duomenų apibrėžimo instrukcijos arba kalba. Pavyzdžiui, leidžia sukurti DB lenteles.

DML (angl. *Data-Manipulation Language*) – duomenų apdorojimo kalba. Formuluojamos užklausos duomenims rasti, atnaujinti, papildyti ir šalinti.

DBVS komponentai detaliau (1)

DDL kompiliatorius – gramatiškai nagrinėja DDL komandas ir perduoda jas vykdymo varikliui.

Užklausų kompiliatorius – gramatiškai nagrinėja ir optimizuoja užklausas, sudaro užklausos planą ir perduodą jį vykdymo varikliui.

Vykdymo variklis – vykdo užklausų planą bei DDL komandas. Šiam tikslui jis gali pareikalausti įrašų iš lentelių, duomenų arba indeksų failų.

DBVS komponentai detaliau (2)

Transakcija – tai grupė užklausų arba kitų DML aprašomų veiksmų, kurie atliekami atomiškai (nedalomai) ir izoliuotai nuo kitų veiksmų.

Transakcijų valdiklis priima transakcijų komandas iš taikomųjų programų.

Sutapimų valdiklis – atsakingas už transakcijų atomiškumą ir izoliaciją.

Registravimo ir atstatymo valdiklis – atsakingas už transakcijos patvarumą.

MySQL db varikliai

- InnoDB
- MyISAM
- Memory
- NDB Cluster
- Merge
- Archive
- Federated
- CSV
- Blackhole
- Example

MySQL db lentelių tipai

- InnoDB – pilnai palaiko transakcijas ir ACID (Atomicity, Consistency, Isolation, Durability) modelį, dydis iki 64TB;
- MyISAM – optimizuotos talpumui (suspaudus lentelę, ji tampa tik skaitoma (anlg, read only) ir greitaveikai, dydis iki 256TB.

Tiek InnoDB, tiek MyISAM atveju paleidžiant DB variklį patikrinama struktūrą ir jei randama klaidų, pakoreguoja, abiem atvejais lentelės lengvai pernešamos tarp platformų ir OS.

MySQL db lentelių tipai

- Memory – MyISAM lentelės saugomos operatyvinėje atmintyje, pasiekiam didesnė sparta;
- Merge – virtuali lentelė, kuri apjungia keletą MyISAM lentelių su panašia struktūra, neturi savo indeksų, galimos operacijos select, delete, update, insert. Šalinant lentelę, šaltinių lentelės neliečiamos. Tinka naudoti, kai reikia nuolatos dirbti su keliomis apjunktomis lentelėmis.

MySQL db lentelių tipai

- Archive – leidžia saugoti didelius kiekius archyvinių duomenų, duomenys suspaudžiami, galimos operacijos tik insert ir select, nepalaiko indeksų;
- Federated – virtuali lentelė, kuri skirta darbui su nutolusiu mysql serveriu, kai nenaudojamas klasteris arba replikavimo technologija;
- CSV – saugo kableliais atskirtus duomenų laukus, naudinga, kai reikia tokiame formate duomenis exportuoti.

MySQL db lentelių tipai

- CSV – saugo kableliais atskirtus duomenų laukus, naudinga, kai reikia tokiame formate duomenis exportuoti.

```
mysql> CREATE TABLE test (i INT NOT NULL, c CHAR(10) NOT NULL)
-> ENGINE = CSV;
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO test VALUES(1,'record one'),(2,'record two');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM test;
+-----+-----+
| i     | c           |
+-----+-----+
|      1 | record one  |
|      2 | record two  |
+-----+-----+
2 rows in set (0.00 sec)
```

MySQL db varikliai

Savybės	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	Ne	384EB
Transactions	Ne	Ne	TAIP	Ne	TAIP
Locking granularity	Lentelė	Lentelė	Eilutė	Eilutė	Eilutė
MVCC	Ne	Ne	TAIP	Ne	Ne
Geospatial data type support	TAIP	Ne	TAIP	TAIP	TAIP
Geospatial indexing support	TAIP	Ne	TAIP	Ne	Ne
B-tree indexes	TAIP	TAIP	TAIP	Ne	Ne
T-tree indexes	Ne	Ne	Ne	Ne	TAIP
Full-text search indexes	TAIP	Ne	TAIP	Ne	Ne
Clustered indexes	Ne	Ne	TAIP	Ne	Ne
Data caches	Ne	Netaikoma	TAIP	Ne	TAIP
Index caches	TAIP	Netaikoma	TAIP	Ne	TAIP
Compressed data	TAIP	Ne	TAIP	TAIP	Ne
Encrypted data	TAIP	TAIP	TAIP	TAIP	TAIP
Cluster database support	Ne	Ne	Ne	Ne	TAIP
Replication support	TAIP	TAIP	TAIP	TAIP	TAIP
Foreign key support	Ne	Ne	TAIP	Ne	Ne
Backup / point-in-time recovery	TAIP	TAIP	TAIP	TAIP	TAIP
Query cache support	TAIP	TAIP	TAIP	TAIP	TAIP
Update statistics for data dictionary	TAIP	TAIP	TAIP	TAIP	TAIP

MY SQL transakcijų mechanizmas (ACID)

Atomiškumas (angl. Atomicity)

- Visos transakcijos operacijos turi būti įvykdytos sėkmingai ir tik tada transakcijos gali būti uždaroma patvirtinant visus rezultatus;

Stabilumas (angl. Consistency)

- Tik teisingi duomenys yra įrašomi į duomenų bazę ir atliktų pakeitimų rezultatas yra tvarkinga duomenų bazė;

Izoliacija (angl. Isolation)

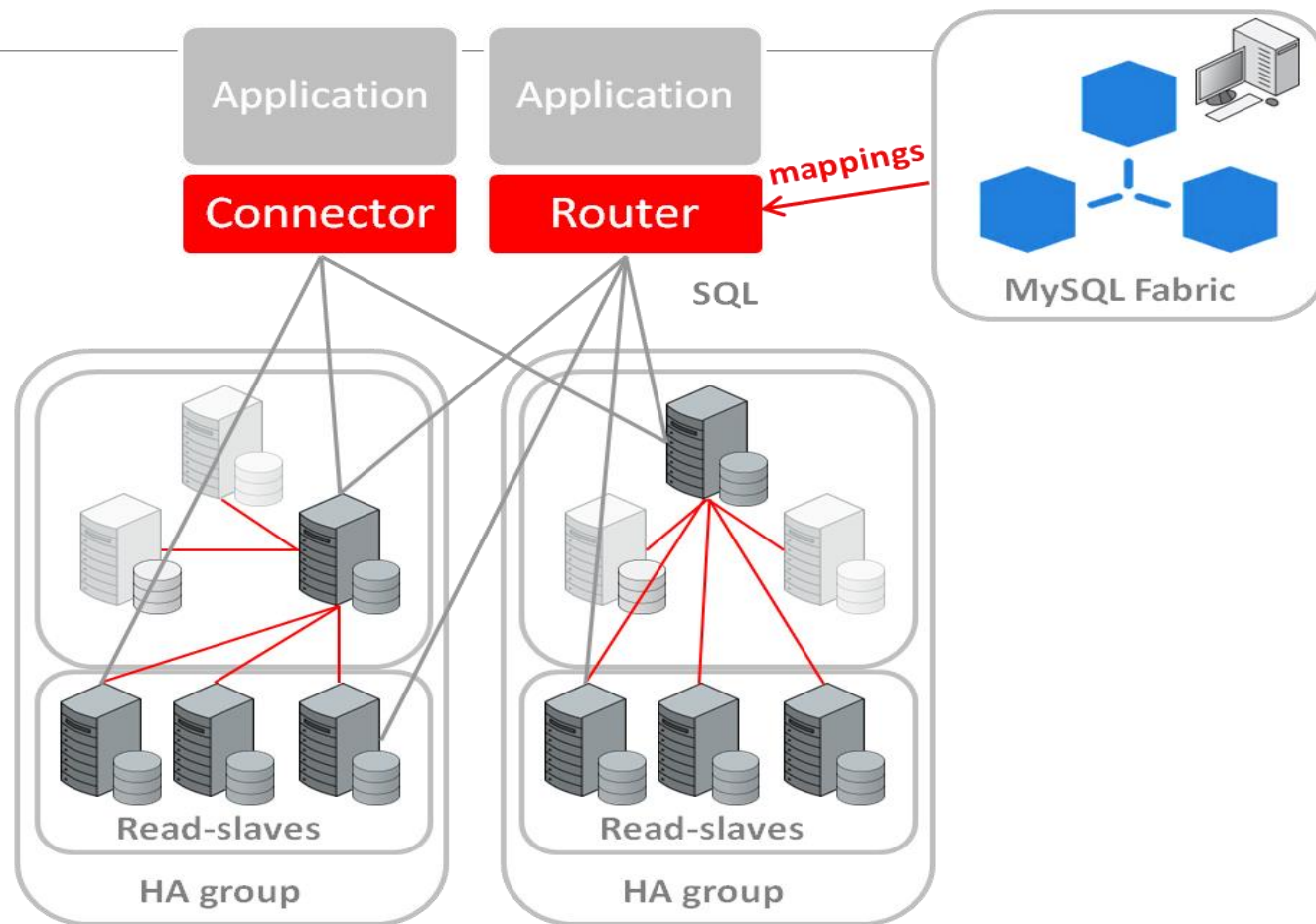
- Kol manipuliavimas duomenimis vyksta transakcijos viduje, kitos transakcijos pakeitimų nemato;

Ilgalaikiškumas (angl. Durability)

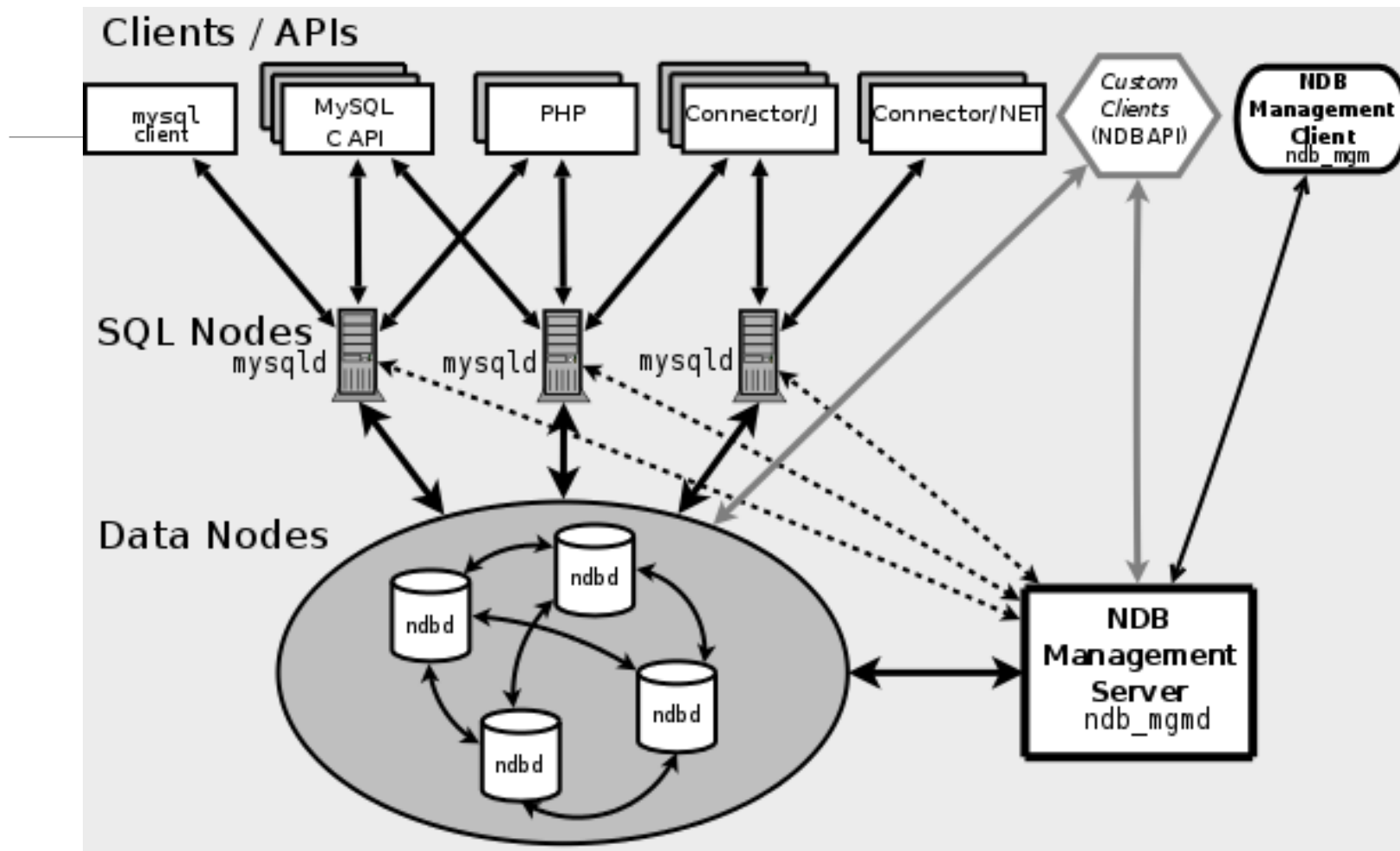
- Jei transakcija buvo patvirtinta, jos rezultatas duomenų bazėje yra nekintantis.

MySQL infrastruktūros architektūra

Tai yra išskirstytųjų duomenų bazių sprendimas NDB variklio pagrindu.



MySQL klasteris



Grindžiama replikavimu ir globaliu transakcijų identifikatoriumi (angl. global transaction identifier -**GTID**)

Prisijungimo prie DBVS variantai

Jungimasis per dedikuotas tvarkykles (angl. Vendor Specific Database Extensions)

Jungimasis naudojant apibendrintas tvarkykles (angl. Database abstraction layers)

Objektų-realiacinio modelio karkasas (angl. ORM -Object Relational Mapping)

<http://php.net/manual/en/refs.database.php>

Prisijungimo prie DBVS variantai

1. Dedikuota tvarkyklė skirta jungimuisi prie konkrečios duomenų bazių valdymo sistemos, darbas su duomenimis vyks per SQL užklausas arba dedikuotas tvarkyklės komandas, programinis kodas bus specializuotas konkrečiai dbvs pagal tvarkyklę, todėl pernešimas darbui su kita DBVS pareikalaus programinio kodo modifikacijos.
2. Apibendrintos tvarkyklės leidžia jungtis prie skirtingų DBVS ir jų komandų rinkinys darbui su DBVS nereikalauja specializuoto programinio kodo, darbas su duomenimis vyks per SQL užklausas arba dedikuotas tvarkyklės komandas.
3. Šiuo atveju jungtis galima prie keleto DBVS, darbas su DB duomenimis vyks per objektinį duomenų modelį.

MY SQL dedikuotos tvarkyklės

Connector/ODBC – standartizuoti dbvs driveriai Windows, Linux, Mac OS X, ir Unix platformoms.

Connector/Net

Connector/J

Connector/Python

Connector/C++

Connector/C (libmysqlclient)

MySQL native driver for PHP – mysqlnd

Jungimasis per dedikuotas tvarkykles PHP (angl. [Vendor Specific Database Extensions](#))

CUBRID

DB++

dBase

filePro

Firebird/InterBase

FrontBase

**IBM DB2 — IBM DB2, Cloudscape
and Apache Derby**

Informix

**Ingres — Ingres DBMS, EDBC, and
Enterprise Access Gateways**

MaxDB

Mongo — MongoDB driver (legacy)

MongoDB — MongoDB driver

mSQL

Mssql — Microsoft SQL Server

MySQL — MySQL Drivers and Plugins

OCI8 — Oracle OCI8

Paradox — Paradox File Access

PostgreSQL

SQLite

SQLite3

SQLSRV — Microsoft SQL Server

Driver for PHP

Sybase

tokyo_tyrant

Jungimasis naudojant apibendrintas tvarkykles (angl. Database abstraction layers)

- DBA — Database (dbm-style) Abstraction
- ODBC — ODBC (Unified)
- PDO — PHP Data Objects

ODBC

ODBC (angl. akronimas *Open Database Connectivity*) yra standartizuota taikomosios [programinės įrangos](#) (dalykinių programų) (aplikacijų) programavimo sąsaja ([API](#)) prisijungimui prie [duomenų bazių](#) ([RDBMS](#)).

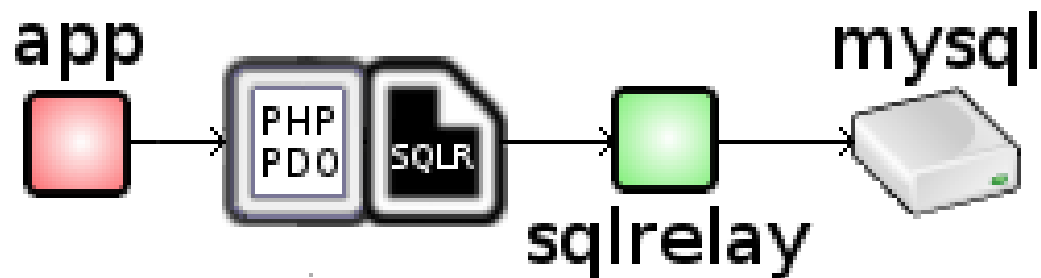
Ši [API](#) nėra skirta konkrečiai programavimo kalbai ar [duomenų bazių](#) valdymo sistemai, bei nėra apribota kuria nors [operacine sistema](#). Ji remiasi ([angl. Call Level Interface \(CLI\)](#)) specifikacijomis iš [SQL](#), [X/Open](#) (dabar dalis [The Open Group](#)), ir [ISO/IEC](#). ODBC buvo sukurta [SQL Access Group](#) ir pirmą kartą paskelbta rugsėjo mėn. [1992](#).

Pirmoji ODBC realizavo ir pateikė savo [Windows](#) operacinėje sistemoje [Microsoft](#) kompanija, bet dabar ODBC palaiko ir UNIX, OS/2 bei iOS sistemas.

Naudojant ODBC galima kurti programas, kurios nekeičiant kodo ar keičiant tik nežymiai leidžia naudotis skirtingomis duomenų bazių valdymo sistemomis.

Norint per ODBC sąsają naudotis tam tikra duomenų baze, šiaip duomenų bazei turi būti parašyta ODBC tvarkyklė. Pavyzdžiui, ODBC sąsaja [MyODBC](#), leidžia pasiekti [MySQL](#) duomenis bet kuria kalba, neturinčia specialios bibliotekos, tačiau palaikančia ODBC komunikavimo mechanizmą.

Jungimasis naudojant apibendrintas tvarkykles



<http://sqlrelay.sourceforge.net/documentation.html>

Jungimasis naudojant apibendrintas tvarkykles

Java – **Java Object Oriented Querying (JOOQ)**

<http://www.jooq.org/>

PHP - **Zend/BD**

Jungimasis naudojant ORM(angl. Object Relational Mapping)

Frameworks yra programavimo karkasai, kurių pagalba programos kuriamos daug greičiau nei jas kuriant vien su tos programavimo kalbos API.

ORM (Object Relational Mapping), smarkiai supaprastina darbą su duomenų bazėmis ir programavimo kalbos objektuose paslepia tiesiogines duomenų bazės užklausas, kurie tuo pasirūpina už programuoją

- 1.Prisijungtų prie duomenų bazės
- 2.Pradėtų tranzakciją
- 3.Sugeneruotų duomenų bazės užklausą
- 4.Ją įvykdytų
- 5.Užbaigtų tranzakciją

Tuo tarpu su ORM tai atrodo maždaug taip:

```
post = Post.new  
post.title = "Programavimo kalbų pasirinkimas"  
post.body = "Mano rašinio tekstas"  
post.save
```

Tai ne tik supaprastina kodo skaitomumą, bet kartu ir smarkiai palengvina jo priežiūrą ateityje.

ORM karkasai

JAVA

Hibernate

.NET

Entity Framework

NHibernate

PHP

Doctrine

Propel

Jungimasis naudojant apibendrintas tvarkykles VS Dedikuotos tvarkyklės

	ext/mysqli	PDO_MySQL	ext/mysql
PHP version introduced	5.0	5.1	2.0
Included with PHP 5.x	Yes	Yes	Yes
Included with PHP 7.x	Yes	Yes	No
Development status	Active	Active	Maintenance only in 5.x; removed in 7.x
Lifecycle	Active	Active	Deprecated in 5.x; removed in 7.x
Recommended for new projects	Yes	Yes	No
OOP Interface	Yes	Yes	No
Procedural Interface	Yes	No	Yes
API supports non-blocking, asynchronous queries with mysqlnd	Yes	No	No
Persistent Connections	Yes	Yes	Yes
API supports Charsets	Yes	Yes	Yes
API supports server-side Prepared Statements	Yes	Yes	No
API supports client-side Prepared Statements	No	Yes	No
API supports Stored Procedures	Yes	Yes	No
API supports Multiple Statements	Yes	Most	No
API supports Transactions	Yes	Yes	No
Transactions can be controlled with SQL	Yes	Yes	Yes
Supports all MySQL 5.1+ functionality	Yes	Most	No

PDO VS MySQLi

	PDO	MySQLi
DBVS palaikymas	12 DBVS	MySQL
API (aplikacijų programavimo sąsaja)	OOP	OOP + procedural
Connection	Lengvas	Lengvas
Named parameters	Taip	Ne
Object mapping	Taip	Taip
Performance	Greitas	Greitesnis
Stored procedures	Taip	Taip

Jungimasia naudojant apibendrintas tvarkykles VS Dedikuotos tvarkyklės

```
<?php
// mysqli
$mysqli = new mysqli("example.com", "user", "password", "database");
$result = $mysqli->query("SELECT 'Hello, dear MySQL user!' AS _message FROM DUAL");
$row = $result->fetch_assoc();
echo htmlentities($row['_message']);

// PDO
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user', 'password');
$stmt = $pdo->query("SELECT 'Hello, dear MySQL user!' AS _message FROM DUAL");
$row = $stmt->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['_message']);

// mysql
$c = mysql_connect("example.com", "user", "password");
mysql_select_db("database");
$result = mysql_query("SELECT 'Hello, dear MySQL user!' AS _message FROM DUAL");
$row = mysql_fetch_assoc($result);
echo htmlentities($row['_message']);
?>
```

Paruoštos užklausos (angl. prepared statements)

Java JDBC[edit]

This example uses Java and the JDBC API:

```
java.sql.PreparedStatement stmt = connection.prepareStatement(  
    "SELECT * FROM users WHERE USERNAME = ? AND ROOM = ?");  
stmt.setString(1, username);  
stmt.setInt(2, roomNumber);  
stmt.executeQuery();
```

Paruoštos užklausos (angl. prepared statements)

C# ADO.NET[edit]

This example uses C# and ADO.NET:

```
using (SqlCommand command = connection.CreateCommand())
{
    command.CommandText = "SELECT * FROM users WHERE USERNAME = @username
AND ROOM = @room";

    command.Parameters.AddWithValue("@username", username);
    command.Parameters.AddWithValue("@room", room);

    using (SqlDataReader dataReader = command.ExecuteReader())
    {
        // ...
    }
}
```

Paruoštos užklausos (angl. prepared statements)

PHP PDO

This example uses PHP and PHP Data Objects (PDO):

```
$stmt = $dbh->prepare("SELECT * FROM users WHERE USERNAME = ? AND  
PASSWORD = ?");
```

```
$stmt->execute(array($username, $pass));
```

Alternately:

```
$stmt = $dbh->prepare("SELECT * FROM users WHERE USERNAME=:username  
AND PASSWORD=:pass");
```

```
$stmt->execute(array('username' => $username, 'pass' => $pass));
```

Paruoštos užklausos (angl. prepared statements)

Naudojant paruoštas užklausas

```
$stmt = $mysqli->prepare("SELECT pavarde FROM users WHERE vardas=?") $stmt->bind_param("s",$vardas);
```

Nereikia pakartotinai rengti ir padavinėti užklausos

```
$vardas = "vardenis"; $stmt->execute();
```

Kitas zmogus:

```
$vardas = "jonas"; $stmt->execute();
```

Vardiniai kintamieji (angl. Named Parameters)

```
$params = array(':username' => 'test', ':email' => $mail, ':last_login' => time() - 3600);
```

```
$pdo->prepare('
    SELECT * FROM users
    WHERE username = :username
    AND email = :email
    AND last_login > :last_login');
```

```
$pdo->execute($params);
```

```
$query = $mysqli->prepare('
    SELECT * FROM users
    WHERE username = ?
    AND email = ?
    AND last_login > ?');
```

```
$query->bind_param('sss', 'test', $mail, time() - 3600);
$query->execute();
```

Įterptinės užklausos (angl. SQL injection)

Naudojant paruoštas užklausas ir MySQLi, nereikia naudoti `mysqli_real_escape_string` ir t.t. Už mus viską tai padaro PHP.

Object oriented style

```
string mysqli::real_escape_string ( string $escapestr )
```

Procedural style

```
string mysqli_real_escape_string ( mysqli $link , string $escapestr )
```


Buferizuota užklausa

Pagal nutylėjimą visų įvykdytų užklausų rezultatai nedelsiant yra grąžinami į klientinę pusę (serverį, galutinį klientą), t.y. užklausos rezultatas yra buferizuojamas. Naudojant tokį metodą, vyksta greitesnė komunikacija klientinėje pusėje, galima atlikti papildomas apdorojimo operacijas. Taip pat esant buferizuotom užklausoms galima vykdyti kitas užklausas. Jei užklausa grąžina labai didelį kiekį duomenų, tai buferizavimo mechanizmas gali pareikalaus didesnio kiekio atminties, tai reikia įvertinti.

Nebuferizuotų užklausų atvejų užklausos rezultatas yra saugomas DBVS pusėje ir reikalingais kiekiais susigrąžinamas iš DBVS. Toks apdorojimas reikalauja mažiau atminties kliento pusėje, bet atitinkamai laiko užimtą sesiją ir padidiną serverio apkrautumą, be to neleidžia vykdyti kitų užklausų kol visas rezultatas nebus susigrąžintas.

Taigi, buferizuotas užklausas rekomenduojam naudoti, kai gražinate pakankamai ribotą duomenų, kai reikia sužinoti grąžintų įrašų kiekį anksčiau, nei jį nuskaitėte. Nebuferizuotos užklauso turėtų būti naudojamas, kai jūs tikėtės iš užklausos gauti labai didelius kiekius duomenų.

Nebuferizuota užklausa pvz. `mysqli`

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");
$result = $mysqli->query("SELECT Name FROM City",
MYSQLI_USE_RESULT);
if ($result)
{
while ($row = $result->fetch_assoc())
    { echo $row['Name'] . PHP_EOL; }
}
$result->close();
?>
```

Nebuferizuota užklausa pvz. PDO_mysql

```
<?php
    $pdo = new PDO("mysql:host=localhost;dbname=world", 'my_user', 'my_pass');
    $pdo->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, false);
    $result = $pdo->query("SELECT Name FROM City");
    if ($result)
    {
        while ($row = $result->fetch(PDO::FETCH_ASSOC))
            { echo $row['Name'] . PHP_EOL; }
    }
?>
```

PDO VS MySQLi

- Ir PDO and MySQLi yra greiti duomenų apdorojimo sprendimai, MySQLi veikia – ~2.5% greičiau testuose, kai naudojamos neparuoštos užklausos ir ~6.5% greičiau, kai naudojamos paruoštos užklausos. O MySQL tvarkyklė veikia ženkliai greičiau už PDO ir MySQLi, jei trūksta našumo galima mėginti jį išspausti per MySQL tvarkyklę, bet turite įvertinti našumą.

PDO VS MySQLi

- MySQLi veikia 3-4 kartus lėčiau nei MySQL, kai dirbama su mažesne nei 500 įrašų aibe;
- MySQLi veikia 2-4 kartus greičiau nei MySQL, kai dirbama su didesne nei 500 įrašų aibe;
- PDO veikia 2-5 kartus lėčiau nei MySQL/MySQLi;
- Nebuferizuotos užklausos 15-40 procentų greitesnės nei buferizuotos naudojant MySQLi;
- Nebuferizuotos užklausos 10-25 procentų greitesnės nei buferizuotos naudojant MySQL, kai įrašų skaičius neviršija 10000;
- Nebuferizuotos užklausos 3-7 procentų lėtesnės nei buferizuotos naudojant MySQL, kai apdorojama ženkliai daugiau 10000 įrašų;
- Nebuferizuotos užklausos 0-5 procentais greitesnės nei buferizuotos, kai naudojama PDO;

Pateiktos įžvalgos yra asmeninio pobūdžio, todėl kiekvienu atveju situaciją reikia vertinti atskirai.

<http://we-love-php.blogspot.lt/2012/07/mysql-or-mysqli-or-pdo.html>

Ačiū už dėmesį
