

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Мареев Георгий Александрович БД-241м

**Практическая работа 2-1. Часть 2. Изучение методов хранения данных на
основе NoSQL. Вариант 13**

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)

Руководитель дисциплины:
Босенко Т.М., доцент департамента
информатики, управления и технологий,
кандидат технических наук

Москва
2025

Введение

Цель: получить практические навыки работы с базой данных Cassandra, изучив основные операции по управлению данными, включая создание и использование ключспейсов, таблиц, выполнение запросов CQL, а также работу с различными инструментами подключения и администрирования.

Основная часть

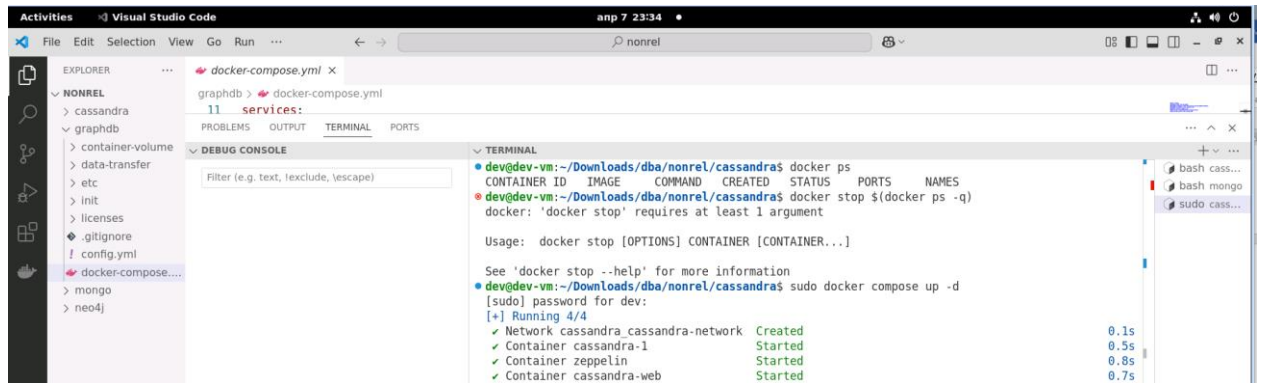
Запуск Cassandra

Проверка запущенных контейнеров:

docker ps

Запуск контейнеров:

sudo docker compose up -d



Подключение к контейнеру

sudo docker exec -ti cassandra-1 cqlsh -u cassandra -p Cassandra

и получение всех существующих в настоящее время пространства ключей

SELECT * FROM system_schema.keyspaces;

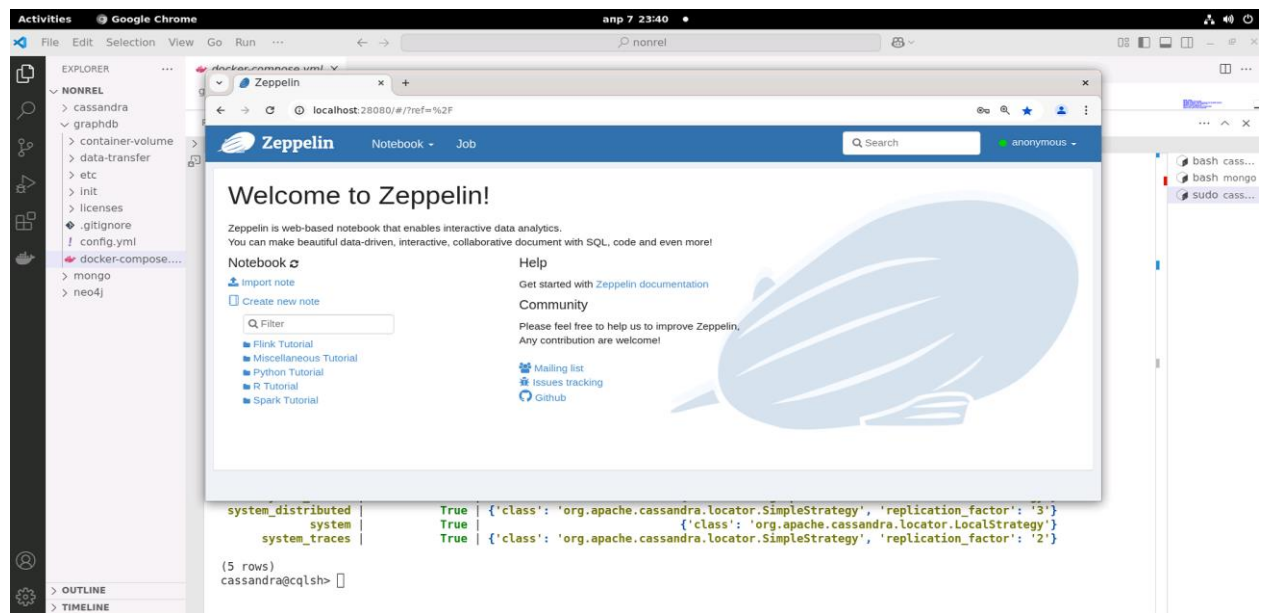
```
dev@dev-vm:~/Downloads/dba/nonrel/cassandra$ sudo docker exec -ti cassandra-1 cqlsh -u cassandra -p cassandra
Connected to CassandraCluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.19 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh> SELECT * FROM system_schema.keyspaces;

keyspace_name | durable_writes | replication
-----
system_auth   | True           | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
system_schema | True           | {'class': 'org.apache.cassandra.locator.LocalStrategy'}
system_distributed | True         | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system        | True           | {'class': 'org.apache.cassandra.locator.LocalStrategy'}
system_traces | True           | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}

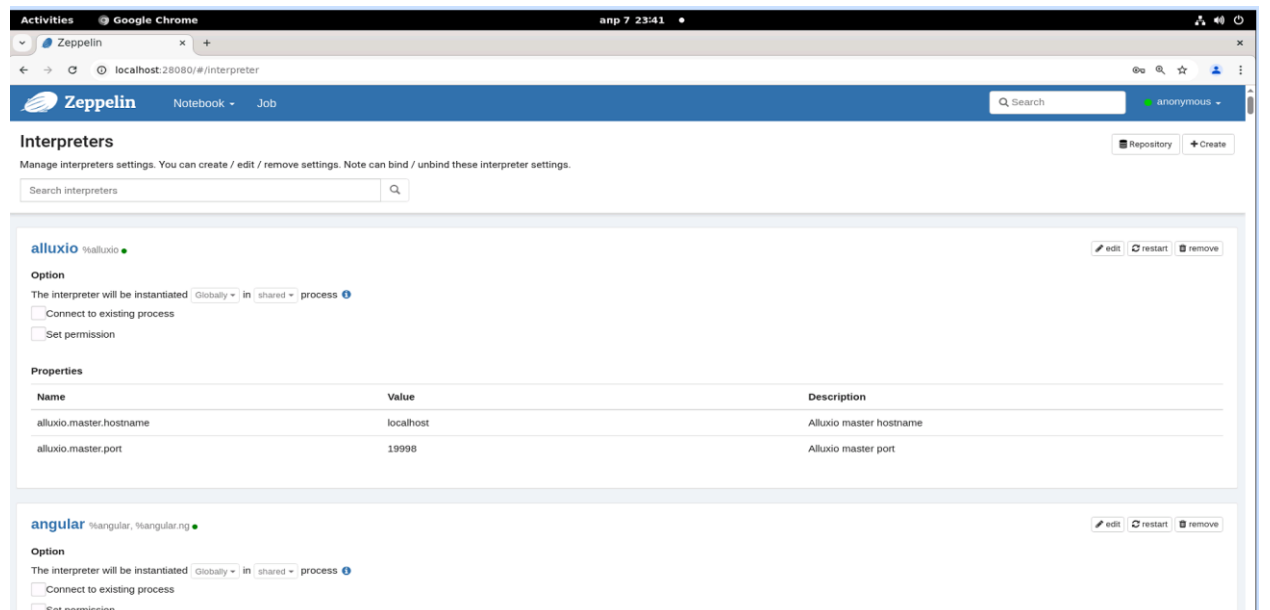
(5 rows)
cassandra@cqlsh>
```

Использование браузерного графического интерфейса

Apache Zeppelin



Interpreter



Cassandra Interpreter

Zeppelin

NotebookJob

Search

anonymous

Interpreters

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

Cassandra

cassandra

%cassandra

editrestartremove

Option

The interpreter will be instantiated

Globallyinprocess

☐ Connect to existing process

☐ Set permission

Properties

Name	Value	Description
cassandra.hosts	localhost	Comma separated Cassandra hosts (DNS name or IP address). Default = localhost. Ex: '192.168.0.12,node2,node3'
cassandra.native.port	9042	Cassandra native port. Default = 9042
cassandra.protocol.version	DEFAULT	Cassandra protocol version. Default = auto-detect
cassandra.cluster	Test Cluster	Cassandra cluster name. Default = 'Test Cluster'
cassandra.keyspace	system	Cassandra keyspace name. Default = 'system'
cassandra.compression.protocol	NONE	Cassandra compression protocol. Available values: NONE, SNAPPY, LZ4. Default = NONE
cassandra.credentials.username	none	Cassandra credentials username. Default = 'none'

Изменения

Zeppelin

NotebookJob

Search

anonymous

Option

The interpreter will be instantiated

Globallyinprocess

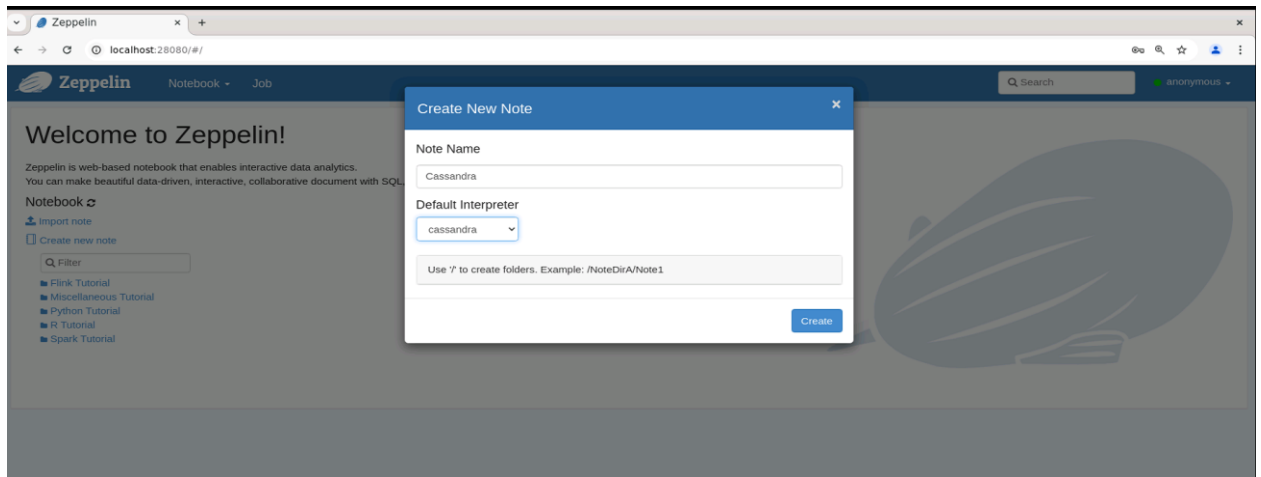
☐ Connect to existing process

☐ Set permission

Properties

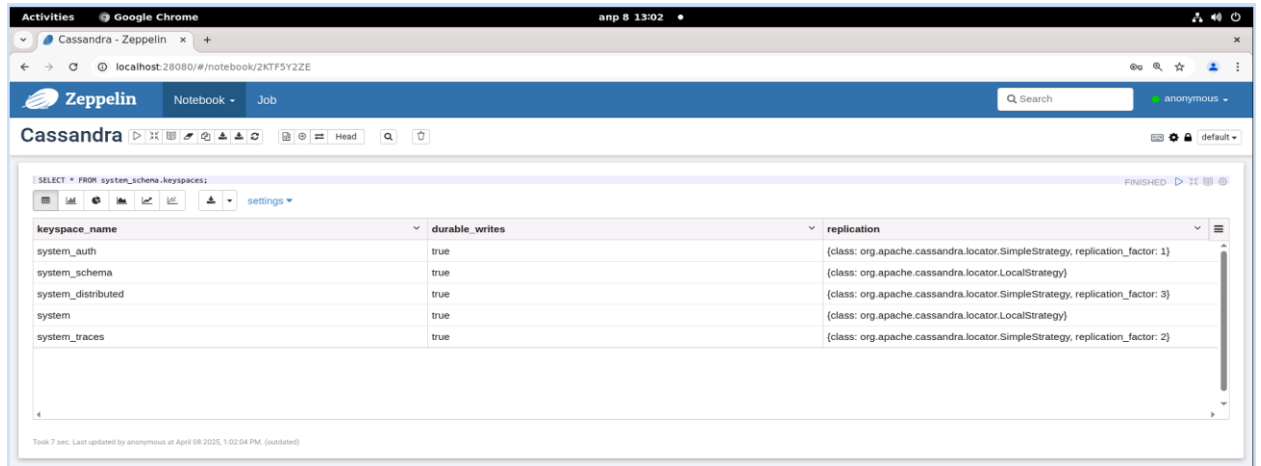
Name	Value	Description	Action
cassandra.hosts	<input type="text" value="localhost:1"/>	Comma separated Cassandra hosts (DNS name or IP address). Default = localhost. Ex: '192.168.0.12,node2,node3'	<input type="button" value="x"/>
cassandra.native.port	<input type="text" value="9042"/>	Cassandra native port. Default = 9042	<input type="button" value="x"/>
cassandra.protocol.version	<input type="text" value="DEFAULT"/>	Cassandra protocol version. Default = auto-detect	<input type="button" value="x"/>
cassandra.cluster	<input type="text" value="Test Cluster"/>	Cassandra cluster name. Default = 'Test Cluster'	<input type="button" value="x"/>
cassandra.keyspace	<input type="text" value="system"/>	Cassandra keyspace name. Default = 'system'	<input type="button" value="x"/>
cassandra.compression.protocol	<input type="text" value="NONE"/>	Cassandra compression protocol. Available values: NONE, SNAPPY, LZ4. Default = NONE	<input type="button" value="x"/>
cassandra.credentials.username	<input type="text" value="cassandra"/>	Cassandra credentials username. Default = 'none'	<input type="button" value="x"/>
cassandra.credentials.password	<input type="password" value="*****"/>	Cassandra credentials password. Default = 'none'	<input type="button" value="x"/>
cassandra.load.balancing.policy	<input type="text" value="DEFAULT"/>	Class name for Load Balancing Policy. Default = DefaultLoadBalancingPolicy	<input type="button" value="x"/>
cassandra.retry.policy	<input type="text" value="DEFAULT"/>	Class name for Retry Policy. Default = DefaultRetryPolicy	<input type="button" value="x"/>
cassandra.reconnection.policy	<input type="text" value="DEFAULT"/>	Class name for Reconnection Policy. Default = ExponentialReconnectionPolicy	<input type="button" value="x"/>

Создание нового блокнота



Существующие на данный момент пространства ключей

```
SELECT * FROM system_schema.keyspaces;
```



Создание Keyspace для примера Movie

```
CREATE KEYSPACE movies WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '1'} AND durable_writes = true;

DESCRIBE KEYSPACE movies;
```

CREATE KEYSPACE movies WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'} AND durable_writes = true;

FINISHED

No Result [Last query execution info](#)

Took 1 sec. Last updated by anonymous at April 08 2025, 1:56:10 PM.

DESCRIBE KEYSPACE movies;

FINISHED

movies

 DESCRIBE KEYSPACE movies;

Legend

Replication	Durable Writes
{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}	true

As CQL statement

Took 0 sec. Last updated by anonymous at April 08 2025, 1:57:51 PM.

```
SELECT * FROM system_schema.keyspaces;
```

SELECT * FROM system_schema.keyspaces;

FINISHED

settings

keyspace_name	durable_writes	replication
system_auth	true	{class: org.apache.cassandra.locator.SimpleStrategy, replication_factor: 1}
system_schema	true	{class: org.apache.cassandra.locator.LocalStrategy}
movies	true	{class: org.apache.cassandra.locator.SimpleStrategy, replication_factor: 1}
system_distributed	true	{class: org.apache.cassandra.locator.SimpleStrategy, replication_factor: 3}
system	true	{class: org.apache.cassandra.locator.LocalStrategy}
system_traces	true	{class: org.apache.cassandra.locator.SimpleStrategy, replication_factor: 2}

Took 0 sec. Last updated by anonymous at April 08 2025, 1:56:25 PM.

Создание таблицы Movie и Actor

Создание таблицы Movie

```
DROP TABLE IF EXISTS movies.movie;
```

```
CREATE TABLE movies.movie (movie_id int,
                             title text,
                             release_year int,
                             running_time int,
                             languages set<text>,
                             genres set<text>,
                             plot_outline text,
```

```
cover_url text,          // cover url
top250_rank int,         // top 250 rank
PRIMARY KEY (movie_id)
);
```

```
DROP TABLE IF EXISTS movies.movie;
CREATE TABLE movies.movie (movie_id int,
    title text,          // title
    release_year int,    // year
    running_time int,    // runtimes
    languages set<text>, // language codes
    genres set<text>,    // genres
    plot_outline text,   // plot outline
    cover_url text,      // cover url
    top250_rank int,     // top 250 rank
    PRIMARY KEY (movie_id)
);
```

No Result [Last query execution info](#)

Took 1 sec. Last updated by anonymous at April 08 2025, 2:01:23 PM.

Actor:

```
DROP TABLE IF EXISTS movies.actor;
CREATE TABLE movies.actor (actor_id int,
    name text,          // name
    headshot_url text,  // headshot
    mini_biography text, // mini biography
    birth_date text,    // birth date
    trade_mark list<text>, // trade mark
    PRIMARY KEY (actor_id)
);
```



```
DROP TABLE IF EXISTS movies.actor;
CREATE TABLE movies.actor (actor_id int,
                             name text,           // name
                             headshot_url text,    // headshot
                             mini_biography text,  // mini biography
                             birth_date text,      // birth date
                             trade_mark list<text>, // trade mark
                             PRIMARY KEY (actor_id)
);
```

No Result [Last query execution info](#)

Took 2 sec. Last updated by anonymous at April 08 2025, 2:02:44 PM.

Просмотре метаданных:

DESCRIBE TABLE movie;

DESCRIBE TABLE movie; FINISHED ▶ ⌵ ⌶ ⌵

DESCRIBE TABLE movie; Legend ▾

movie

Column Type	Column Name	Data Type
❖	movie_id	int
	cover_url	text
	genres	set<text>
	languages	set<text>
	plot_outline	text
	release_year	int
	running_time	int
	title	text
	top250_rank	int

As CQL statement ▾

Took 0 sec. Last updated by anonymous at April 08 2025, 2:03:38 PM.

DESCRIBE TABLE actor;

DESCRIBE TABLE actor; FINISHED ▶ ⌵ ⌶ ⌵

DESCRIBE TABLE actor; Legend ▾

actor

Column Type	Column Name	Data Type
❖	actor_id	int
	birth_date	text
	headshot_url	text
	mini_biography	text
	name	text
	trade_mark	list<text>

As CQL statement ▾

Took 0 sec. Last updated by anonymous at April 08 2025, 2:03:44 PM.

Вставка данных в Movie и Actor

Фильмы “The Matrix”, “Pulp Fiction”, “Speed”

```
// insert "The Matrix" - 0133093
```

```
INSERT INTO movies.movie (movie_id, title, release_year, running_time,  
languages,
```

```
genres, plot_outline, cover_url, top250_rank)
```

```
VALUES (0133093,
```

```
'The Matrix',
```

```
1999,
```

```
136,
```

```
{'en'},
```

```
{'Action', 'Sci-Fi'},
```

\$\$Thomas A. Anderson is a man living two lives. By day he is an average computer programmer and by night a hacker known as Neo. Neo has always questioned his reality, but the truth is far beyond his imagination. Neo finds himself targeted by the police when he is contacted by Morpheus, a legendary computer hacker branded a terrorist by the government. Morpheus awakens Neo to the real world, a ravaged wasteland where most of humanity have been captured by a race of machines that live off of the humans' body heat and electrochemical energy and who imprison their minds within an artificial reality known as the Matrix. As a rebel against the machines, Neo must return to the Matrix and confront the agents: super-powerful computer programs devoted to snuffing out Neo and the entire human rebellion\$\$,

```
'https://m.media-  
amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDll  
ZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_SX101  
_CR0,0,101,150_.jpg',
```

```
19);
```

```
// insert "Pulp Fiction" - 0110912
```

```
INSERT INTO movies.movie (movie_id, title, release_year, running_time,  
languages,
```

```
genres, plot_outline, cover_url, top250_rank)
```

VALUES (0110912,

'Pulp Fiction',

1994,

154,

{ 'en', 'es', 'fr' },

{ 'Crime', 'Drama' },

\$\$Jules Winnfield (Samuel L. Jackson) and Vincent Vega (John Travolta) are two hit men who are out to retrieve a suitcase stolen from their employer, mob boss Marsellus Wallace (Ving Rhames). Wallace has also asked Vincent to take his wife Mia (Uma Thurman) out a few days later when Wallace himself will be out of town. Butch Coolidge (Bruce Willis) is an aging boxer who is paid by Wallace to lose his fight. The lives of these seemingly unrelated people are woven together comprising of a series of funny, bizarre and uncalled-for incidents.\$\$

'https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBiZi00MTRiLWFjM2ItYzViMjE3YzI5MjljXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_SY150_CR1,0,101,150_.jpg',

8);

// insert "Speed" - 0111257

INSERT INTO movies.movie (movie_id, title, release_year, running_time, languages,

genres, plot_outline, cover_url, top250_rank)

VALUES (0111257,

'Speed',

1994,

116,

{ 'en' },

{ 'Action', 'Adventure', 'Crime', 'Thriller' },

\$\$Bomber extortionist's elevator plan backfires, so he rigs a bomb to a LA city bus. The stipulation is: once armed, the bus must stay above 50 mph to keep from exploding. Also if LAPD Officer tries to unload any passengers off,

```

        'https://m.media-
amazon.com/images/M/MV5BYjc0MjYyN2EtZGRhMy00NzJiLWI2Y2QtYzhiY
TU3NzAxNzg4XkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_SY150_CR0,0,10
1,150_.jpg',

        null);

```

Добавление актеров, играющих в этих фильмах ("Bruce Willis", "John Travolta", "Sandra Bullock", "Samuel L. Jackson", "Uma Thurman" & "Quentin Tarantino"):

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,
trade_mark)
```

'Bruce Willis'.

amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_CR8,0,67,98_AL_.jpg',

['Frequently plays a man who suffered a tragedy, had lost something or had a crisis of confidence or conscience.',

'Headlines action-adventures, often playing a policeman, hitman or someone in the military',

control',
'Often plays men who get caught up in situations far beyond their
'Sardonic one-liners',
'Shaven head',
'Distinctive, gravelly voice',
'Smirky grin.',
'Known for playing cynical anti-heroes with unhappy personal
lives']]);

// insert "John Travolta" - 0000237

INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,
trade_mark)

VALUES (0000237,

'John Travolta',

'https://m.media-
amazon.com/images/M/MV5BMTUwNjQ0ODkxN15BMl5BanBnXkFtZTcwMDc
5NjQwNw@@._V1_UY98_CR3,0,67,98_AL_.jpg',

'1954-02-18',

['Cleft chin and razor-sharp cheekbones',

'Often works some sort of dance into his roles',

'New Jersey accent',

'Black hair and blue eyes']]);

// insert "Sandra Bullock" - 0000113

INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,
trade_mark)

VALUES (0000113,

'Sandra Bullock',

```
'https://m.media-  
amazon.com/images/M/MV5BMTI5NDY5NjU3NF5BMl5BanBnXkFtZTcwMzQ  
0MTMyMw@@._V1_UX67_CR0,0,67,98_AL_.jpg',
```

```
'1964-07-26',
```

```
null);
```

```
// insert "Samuel L. Jackson" - 0000168
```

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,  
trade_mark)
```

```
VALUES (0000168,
```

```
'Samuel L. Jackson',
```

```
'https://m.media-  
amazon.com/images/M/MV5BMTQ1NTQwMTYxNl5BMl5BanBnXkFtZTYwMj  
A1MzY1._V1_UX67_CR0,0,67,98_AL_.jpg',
```

```
'1948-12-21',
```

```
['Deep authoritative voice',
```

```
'Rebellious characters who are disliked or considered strange by  
others in the story',
```

```
'Often plays police officers or government officials. Both prone to  
intimidation or violence',
```

```
'Often plays very wise and intelligent characters with great capacities  
for violence',
```

```
'Frequently plays tough characters who swear a lot',
```

```
'Frequent swearing',
```

```
'Often sports a moustache or goatee in his films',
```

```
'Shaven head',
```

```
'Kangol hats',
```

```
'Often plays hotheaded characters with a fiery temper',
```

```
'Often shouts the word "motherf*****" at some point in a film',
```

```
'Frequently cast by Quentin Tarantino']);
```

```
// insert "Uma Thurman" - 0000235
```

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,  
trade_mark)
```

```
VALUES (0000235,
```

```
        'Uma Thurman',
```

```
        'https://m.media-  
amazon.com/images/M/MV5BMjMxNzk1MTQyMl5BMl5BanBnXkFtZTgwMDI  
zMDEyMTE@._V1_UX67_CR0,0,67,98_AL_.jpg',
```

```
        '1970-04-29',
```

```
        ['Long blond hair and blue eyes', 'Statuesque, model-like figure']);
```

```
// insert "Keanu Reeves" - 0000206
```

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,  
trade_mark)
```

```
VALUES (0000206,
```

```
        'Keanu Reeves',
```

```
        'https://m.media-  
amazon.com/images/M/MV5BNjUxNDcwMTg4Ml5BMl5BanBnXkFtZTcwMjU4  
NDYyOA@@._V1_UY98_CR4,0,67,98_AL_.jpg',
```

```
        '1964-09-02',
```

```
        ['Intense contemplative gaze',
```

```
        'Deep husky voice',
```

```
        'Known for playing stoic reserved characters']);
```

```
// insert "Quentin Tarantino" - 0000233
```

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date,  
trade_mark)
```

```
VALUES (0000233,
```

```
        'Quentin Tarantino',
```

'https://m.media-amazon.com/images/M/MV5BMTgyMjI3ODAzNl5BMl5BanBnXkFtZTcwNzY2MDYxOQ@@._V1_UX67_CR0,0,67,98_AL_.jpg',

'1963-03-27',

['Lead characters usually drive General Motors vehicles, particularly Chevrolet and Cadillac, such as Jules 1974 Nova and Vincents 1960s Malibu.',

'Briefcases and suitcases play an important role in Pulp Fiction (1994), Reservoir Dogs (1992), Jackie Brown (1997), True Romance (1993) and Kill Bill: Vol. 2 (2004).',

'Makes references to cult movies and television',

'Frequently works with Harvey Keitel, Tim Roth, Michael Madsen, Uma Thurman, Michael Bowen, Samuel L. Jackson, Michael Parks and Christoph Waltz.',

'His films usually have a shot from inside an automobile trunk',

'He always has a Dutch element in his films: The opening tune, Little Green Bag, in Reservoir Dogs (1992) was performed by George Baker Selection and written by Jan Gerbrand Visser and Benjamino Bouwens who are all Dutch. The character Freddy Newandyke, played by Tim Roth is a direct translation to a typical Dutch last name, Nieuwendijk. The code name of Tim Roth is Mr. Orange, the royal color of Holland and the last name of the royal family. The Amsterdam conversation in Pulp Fiction (1994), Vincent Vega smokes from a Dutch tobacco shag (Drum), the mentioning of Rutger Hauer in Jackie Brown (1997), the brides name is Beatrix, the name of the Royal Dutch Queen.',

'[The Mexican Standoff] All his movies (including True Romance (1993), which he only wrote and did not direct) feature a scene in which three or more characters are pointing guns at each other at the same time.',

'Often uses an unconventional storytelling device in his films, such as retrospect (Reservoir Dogs (1992)), non-linear (Pulp Fiction (1994)), or "chapter" format (Kill Bill: Vol. 1 (2003)).',

'His films will often include one long, unbroken take where a character is followed around somewhere.']);

Cassandra - Zeppelin

localhost:28080/#notebook/2KTF5Y2ZE

```
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date, trade_mark)
VALUES (0000113,
        'Sandra Bullock',
        'https://m.media-amazon.com/images/M/MV5BMTI5NDY5NjU3NF5BM15BanBnXkFtZTcwMTQwMTYwMjA1MzY1_V1_UX67_CR0,0,67,98_AL_.jpg',
        '1964-07-26',
        null);

// insert "Samuel L. Jackson" - 0000168
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date, trade_mark)
VALUES (0000168,
        'Samuel L. Jackson',
        'https://m.media-amazon.com/images/M/MV5BMTQ1NTQwMTYxN15BM15BanBnXkFtZTcwMTzY2YwMjA1MzY1_V1_UX67_CR0,0,67,98_AL_.jpg',
        '1948-12-21',
        ['Deep authoritative voice',
        'Rebellious characters who are disliked or considered strange by others in the story',
        'Often plays police officers or government officials. Both prone to intimidation or violence',
        'Often plays very wise and intelligent characters with great capacities for violence',
        'Frequently plays tough characters who swear a lot',
        'Frequent swearing',
        'Often sports a moustache or goatee in his films',
        'Shaven head',
        'Kangol hats',
        'Often plays hotheaded characters with a fiery temper',
        'Often shouts the word "motherf*****" at some point in a film',
        'Frequently cast by Quentin Tarantino']);

// insert "Uma Thurman" - 0000235
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date, trade_mark)
VALUES (0000235,
        'Uma Thurman',
        'https://m.media-amazon.com/images/M/MV5BMjMxNDY5NjU3NF5BM15BanBnXkFtZTcwMTzY2YwMjA1MzY1_V1_UX67_CR0,0,67,98_AL_.jpg',
        '1970-04-29',
        ['Long blond hair and blue eyes', 'Statuesque, model-like figure']);

// insert "Keanu Reeves" - 0000206
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date, trade_mark)
VALUES (0000206,
        'Keanu Reeves',
        'https://m.media-amazon.com/images/M/MV5BNjUxNDcwMTg4M15BM15BanBnXkFtZTcwMTU4NDYyOAAw_V1_UY98_CR4,0,67,98_AL_.jpg',
        '1964-09-02',
        ['Intense contemplative gaze',
        'Deep husky voice',
        'Known for playing stoic reserved characters']);

// insert "Quentin Tarantino" - 0000233
INSERT INTO movies.actor (actor_id, name, headshot_url, birth_date, trade_mark)
VALUES (0000233,
        'Quentin Tarantino',
        'https://m.media-amazon.com/images/M/MV5BMTgyMjI3ODAxN15BM15BanBnXkFtZTcwMTzY2YwMjA1MzY1_V1_UX67_CR0,0,67,98_AL_.jpg',
        '1963-03-27',
        ['Lead characters usually drive General Motors vehicles, particularly Chevrolet and Cadillac, such as Jules 1974 Nova and Vincents 1968s Malibu.',
        'Briefcases and suitcases play an important role in Pulp Fiction (1994), Reservoir Dogs (1992), Jackie Brown (1997), True Romance (1993) and Kill Bill: Vol. 2 (2004).',
        'Makes references to cult movies and television',
        'Frequently works with Harvey Keitel, Tim Roth, Michael Madsen, Uma Thurman, Michael Bowen, Samuel L. Jackson, Michael Parks and Christoph Waltz.',
        'His films usually have a shot from inside an automobile trunk',
        'He always has a Dutch element in his films: The opening tune, Little Green Bag, in Reservoir Dogs (1992) was performed by George Baker Selection and written by Jan Gerbrand Visser and Benjamin Bouwens who are all Dutch. The character Freddy Newandylke, played by Tim Roth is a direct translation to a typical Dutch last name, Nieuwendijk. The code name of Tim Roth is Mr. Orange, the royal color of Holland and the last name of the royal family. The Amsterdam conversation in Pulp Fiction (1994), Vincent Vega smokes from a Dutch tobacco shag (Grun), the mentioning of Rutger Hauer in Jackie Brown (1997), the brides name is Beatrix, the name of the Royal Dutch Queen.',
        'The Mexican Standoff All his movies (including True Romance (1993), which he only wrote and did not direct) feature a scene in which three or more characters are pointing guns at each other at the same time.',
        'Often uses an unconventional storytelling device in his films, such as retrospect (Reservoir Dogs (1992)), non-linear (Pulp Fiction (1994)), or "chapter" format (Kill Bill: Vol. 1 (2003)).',
        'His films will often include one long, unbroken take where a character is followed around somewhere.']);
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 2:07:15 PM.

Отображение данных:

Movie

SELECT movie_id, title

FROM movies.movie;

SELECT movie_id, title
FROM movies.movie;

FINISHED

settings

movie_id	title
111257	Speed
133093	The Matrix
110912	Pulp Fiction

Took 0 sec. Last updated by anonymous at April 08 2025, 2:08:41 PM.

Actors

SELECT *
FROM movies.actor;

SELECT *
FROM movies.actor;

FINISHED

settings

actor_id	birth_date	headshot_url	mini_biography	name	trade_mark
237	1954-02-18	https://m.media-amazon.com/images/M/MV5BMTUwNjQ0ODkxN15BMi5BanBnXkFtZTcwMDc5NjQwNw@@._V1_UY98_CR3,0,67,98_AL_.jpg	null	John Travolta	[Cleft chin and razor-sharp cheekbones, Often works some sort of dance into his roles, New Jersey accent, Black hair and blue eyes]
113	1964-07-26	https://m.media-amazon.com/images/M/MV5BMTI5NDY5NjU3NF5BMi5BanBnXkFtZTcwMzQ0MTMyMw@@._V1_UX67_CR0,0,67,98_AL_.jpg	null	Sandra Bullock	null
233	1963-03-27	https://m.media-	null	Quentin Tarantino	[Lead characters usually drive

Took 0 sec. Last updated by anonymous at April 08 2025, 2:10:46 PM.

Обновление данных

Bruce Willis

UPDATE movies.actor
SET name = 'Bruce Walter Willis'
WHERE actor_id = 0000246;

```
UPDATE movies.actor  
SET name = 'Bruce Walter Willis'  
WHERE actor_id = 0000246;
```

No Result

Last query execution info

Took 0 sec. Last updated by anonymous at April 08 2025, 2:12:14 PM.

SELECT *

FROM movies.actor

WHERE actor_id = 0000246;

SELECT *

FROM movies.actor

WHERE actor_id = 0000246;

FINISHED

⌕

⌕

🏠

📊

📈

📉

📊

📉

👤

⚙️

settings

actor_id	birth_date	headshot_url	mini_biography	name	trade_mark
246	1955-03-19	https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMi5BanBnXkFtZTcwMzQ2ODE3Mw@@_V1_UY98_CR8,0,67,98_AL_.jpg	null	Bruce Walter Willis	[Frequently plays a man who suffered a tragedy, had lost something or had a crisis of confidence or conscience., Frequently plays likeable wisecracking heroes with a moral centre, Headlines action-adventures, often playing a policeman, hitman or someone in the military, Often plays men who get caught up in situations far beyond their control, Sardonic

Использование “Dynamic” таблиц (wide row)

Создание таблицы «Фильмы по актерам» и «Актёры по фильмам»

```
DROP TABLE IF EXISTS movies.movies_by_actor;
```

```
CREATE TABLE movies.movies_by_actor (actor_id int,
```

movie_id int,

title text,

PRIMARY KEY (actor_id, movie_id)

);

[illegible]

No Result

Last query execution info

Took 1 sec. Last updated by anonymous at April 08 2025, 2:16:06 PM.

```
DROP TABLE IF EXISTS movies.actors_by_movie;
CREATE TABLE movies.actors_by_movie (movie_id int,
                                     title text STATIC,
                                     actor_id int,
                                     name text,
                                     PRIMARY KEY (movie_id, actor_id)
);
```

```
DROP TABLE IF EXISTS movies.actors_by_movie;
CREATE TABLE movies.actors_by_movie (movie_id int,
                                     title text STATIC,
                                     actor_id int,
                                     name text,
                                     PRIMARY KEY (movie_id, actor_id)
);
```

No Result

[Last query execution info](#)

Took 1 sec. Last updated by anonymous at April 08 2025, 2:16:26 PM.

Вставка данных в две таблицы

// Фильмы для актера "Bruce Willis"

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0110912, 'Pulp Fiction');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 1606378, 'A Good Day to Die Hard');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0217869, 'Unbreakable');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0377917, 'The Fifth Element');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0112864, 'Die Hard: With a Vengeance');
```

```
// Фильмы для актера "Keanu Reeves"
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0133093, 'The Matrix');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0234215, 'The Matrix Reloaded');
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0111257, 'Speed');
```

```
// Фильмы для актера "Sandra Bullock"
```

```
INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000113, 0111257, 'Speed');
```

```
// Фильмы для актера "Bruce Willis"

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0110912, 'Pulp Fiction');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 1606378, 'A Good Day to Die Hard');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0217869, 'Unbreakable');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0377917, 'The Fifth Element');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000246, 0112864, 'Die Hard: With a Vengeance');

// Фильмы для актера "Keanu Reeves"

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0133093, 'The Matrix');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0234215, 'The Matrix Reloaded');

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000206, 0111257, 'Speed');

// Фильмы для актера "Sandra Bullock"

INSERT INTO movies.movies_by_actor (actor_id, movie_id, title)
VALUES (0000113, 0111257, 'Speed');
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 2:17:34 PM.

// Актеры для фильма "The Matrix"

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0000206, 'Keanu Reeves');
```

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0000401, 'Laurence Fishburne');
```

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0005251, 'Carrie-Anne Moss');
```

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0915989, 'Hugo Weaving');
```

// Актеры для фильма "Pulp Fiction"

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000237, 'John Travolta');
```

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000168, 'Samuel L. Jackson');
```

```
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000246, 'Bruce Willis');
```

```
// Актеры для фильма "The Matrix"
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0000206, 'Keanu Reeves');

INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0000401, 'Laurence Fishburne');

INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0005251, 'Carrie-Anne Moss');

INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0133093, 'The Matrix', 0915989, 'Hugo Weaving');

// Актеры для фильма "Pulp Fiction"
INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000237, 'John Travolta');

INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000168, 'Samuel L. Jackson');

INSERT INTO movies.actors_by_movie (movie_id, title, actor_id, name)
VALUES (0110912, 'Pulp Fiction', 0000246, 'Bruce Willis');
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 2:17:57 PM.

Все фильмы, в которых играл "Bruce Willis"

```
SELECT title
FROM movies.movies_by_actor
WHERE actor_id = 0000246;
```

SELECT title
FROM movies.movies_by_actor
WHERE actor_id = 0000246;

FINISHED ▶ ⌂ ⚙

table icon bar chart icon pie chart icon line chart icon settings icon

settings ▼

title
Pulp Fiction
Die Hard: With a Vengeance
Unbreakable
The Fifth Element
A Good Day to Die Hard

Took 0 sec. Last updated by anonymous at April 08 2025, 2:18:41 PM. (outdated)

Актеры, сыгравшие в фильме «Криминальное чтиво»

```
SELECT name, title
FROM movies.actors_by_movie
WHERE movie_id = 0110912;
```

SELECT name, title
FROM movies.actors_by_movie
WHERE movie_id = 0110912;

FINISHED ▶ ⌂ ⚙

table icon bar chart icon pie chart icon line chart icon settings icon

settings ▼

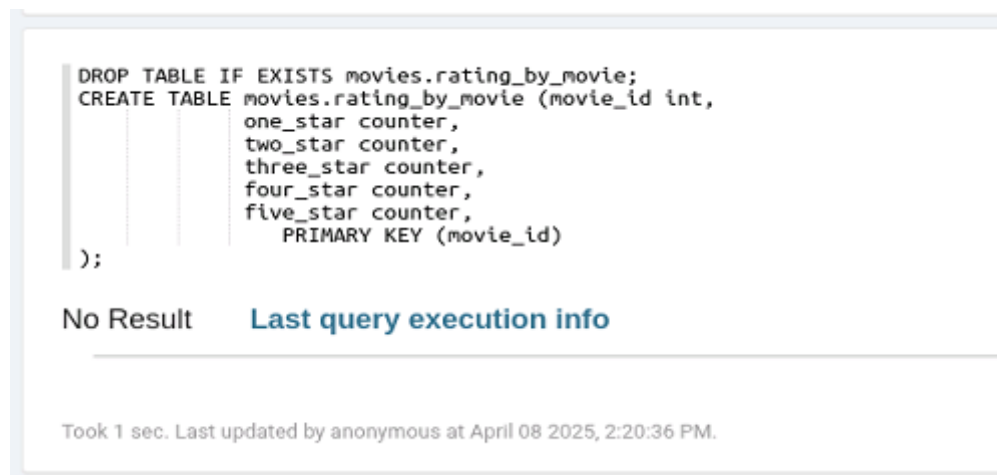
name	title
Samuel L. Jackson	Pulp Fiction
John Travolta	Pulp Fiction
Bruce Willis	Pulp Fiction

Took 0 sec. Last updated by anonymous at April 08 2025, 2:19:39 PM.

Использование столбцов-счетчиков

Создание новой таблицы

```
DROP TABLE IF EXISTS movies.rating_by_movie;  
CREATE TABLE movies.rating_by_movie (movie_id int,  
    one_star counter,  
    two_star counter,  
    three_star counter,  
    four_star counter,  
    five_star counter,  
    PRIMARY KEY (movie_id)  
);
```



Добавление рейтинга

```
UPDATE movies.rating_by_movie  
SET five_star = five_star + 1  
WHERE movie_id = 0110912;
```

```
UPDATE movies.rating_by_movie  
SET four_star = four_star + 1  
WHERE movie_id = 0110912;
```

```
UPDATE movies.rating_by_movie
```

```
SET five_star = five_star + 1  
WHERE movie_id = 0110912;
```

```
UPDATE movies.rating_by_movie  
SET five_star = five_star + 1  
WHERE movie_id = 0110912;
```

```
UPDATE movies.rating_by_movie  
SET two_star = two_star + 1  
WHERE movie_id = 0110912;
```

```
UPDATE movies.rating_by_movie  
SET five_star = five_star + 1  
WHERE movie_id = 0110912;  
  
UPDATE movies.rating_by_movie  
SET four_star = four_star + 1  
WHERE movie_id = 0110912;  
  
UPDATE movies.rating_by_movie  
SET five_star = five_star + 1  
WHERE movie_id = 0110912;  
  
UPDATE movies.rating_by_movie  
SET five_star = five_star + 1  
WHERE movie_id = 0110912;  
  
UPDATE movies.rating_by_movie  
SET two_star = two_star + 1  
WHERE movie_id = 0110912;
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 2:24:33 PM.

Проверка текущего рейтинга фильма «Криминальное чтиво»

SELECT * FROM movies.rating_by_movie WHERE movie_id = 0110912;



The screenshot shows a database query result for the query: `SELECT * FROM movies.rating_by_movie WHERE movie_id = 0110912;`. The result is displayed in a table with 6 columns: `movie_id`, `five_star`, `four_star`, `one_star`, `three_star`, and `two_star`. The data row shows `movie_id` as 110912, `five_star` as 3, `four_star` as 1, `one_star` as null, `three_star` as null, and `two_star` as 1. The interface includes a toolbar with icons for table, text, insert, update, delete, and settings. The status bar at the bottom indicates "Took 0 sec. Last updated by anonymous at April 08 2025, 2:24:45 PM. (outdated)".

movie_id	five_star	four_star	one_star	three_star	two_star
110912	3	1	null	null	1

Таблица для подсчёта количества просмотров каждого фильма, разделенных на зрителей-мужчин и женщин, и по месяцам

DROP TABLE IF EXISTS movies.movie_viewed_by_time;

CREATE TABLE movies.movie_viewed_by_time (movie_id int,

year int,

month int,

male counter,

female counter,

PRIMARY KEY (movie_id, year, month)

) WITH CLUSTERING ORDER BY (year DESC, month DESC);

```
DROP TABLE IF EXISTS movies.movie_viewed_by_time;
CREATE TABLE movies.movie_viewed_by_time (movie_id int,
      year int,
      month int,
      male counter,
      female counter,
      PRIMARY KEY (movie_id, year, month)
) WITH CLUSTERING ORDER BY (year DESC, month DESC);
```

No Result [Last query execution info](#)

Took 2 sec. Last updated by anonymous at April 08 2025, 2:25:14 PM.

Добавление нескольких примеров значений

// Pulp Fiction Views 2019/03

UPDATE movies.movie_viewed_by_time

SET male = male + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 03;

UPDATE movies.movie_viewed_by_time

SET male = male + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 03;

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 03;

// Pulp Fiction Views 2019/04

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time

SET male = male + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 04;

// Pulp Fiction Views 2019/05

UPDATE movies.movie_viewed_by_time

SET male = male + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time

SET male = male + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time

SET female = female + 1

WHERE movie_id = 0110912 AND year = 2019 and month = 05;

```
// Pulp Fiction Views 2019/03
UPDATE movies.movie_viewed_by_time
SET male = male + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 03;

UPDATE movies.movie_viewed_by_time
SET male = male + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 03;

UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 03;

// Pulp Fiction Views 2019/04
UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time
SET male = male + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 04;

UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 04;

// Pulp Fiction Views 2019/05
UPDATE movies.movie_viewed_by_time
SET male = male + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time
SET male = male + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 05;

UPDATE movies.movie_viewed_by_time
SET female = female + 1
WHERE movie_id = 0110912 AND year = 2019 and month = 05;
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 2:25:38 PM.

Просмотры фильма "Криминальное чтиво" за все время

SELECT *

FROM movies.movie_viewed_by_time

WHERE movie_id = 0110912;

SELECT *
FROM movies.movie_viewed_by_time
WHERE movie_id = 0110912;

FINISHED

movie_id	year	month	female	male
110912	2019	5	2	2
110912	2019	4	3	1
110912	2019	3	1	2

Took 0 sec. Last updated by anonymous at April 08 2025, 2:26:04 PM. (outdated)

Просмотры фильма "Криминальное чтиво" за один месяц

```
SELECT *  
  
FROM movies.movie_viewed_by_time  
  
WHERE movie_id = 0110912 AND year = 2019 AND month = 05;
```

SELECT *
FROM movies.movie_viewed_by_time
WHERE movie_id = 0110912 AND year = 2019 AND month = 05;

FINISHED

settings

movie_id	year	month	female	male
110912	2019	5	2	2

Took 0 sec. Last updated by anonymous at April 08 2025, 2:26:19 PM. (outdated)

Просмотры фильма "Криминальное чтиво" за месяц с января по май

```
SELECT *  
  
FROM movies.movie_viewed_by_time  
  
WHERE movie_id = 0110912 AND year = 2019 AND month >= 01 AND month  
<= 5;
```

SELECT *
FROM movies.movie_viewed_by_time
WHERE movie_id = 0110912 AND year = 2019 AND month >= 01 AND month <= 5;

FINISHED

settings

movie_id	year	month	female	male
110912	2019	5	2	2
110912	2019	4	3	1
110912	2019	3	1	2

Took 0 sec. Last updated by anonymous at April 08 2025, 2:26:38 PM.

Индивидуальное задание. Вариант 13

Задание:

1. Создайте ключспейс library с репликацией SimpleStrategy и коэффициентом репликации 1.

```
CREATE KEYSPACE library WITH replication =  
  {'class':'SimpleStrategy','replication_factor':1};  
DESCRIBE KEYSPACE library;
```

```
CREATE KEYSPACE library WITH replication =  
{'class':'SimpleStrategy','replication_factor':1};  
DESCRIBE KEYSPACE library;
```

library

DESCRIBE KEYSPACE library;

Legend

Replication	Durable Writes
{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}	true

As CQL statement

Took 1 sec. Last updated by anonymous at April 08 2025, 2:30:05 PM.

2. Создайте таблицу books в ключспейсе library с полями book_id (int), title (text), author (text), year_published (int), genre (text), isbn (text) и первичным ключом book_id.

```
DROP TABLE IF EXISTS library.books;  
CREATE TABLE library.books (  
  book_id int,  
  title text,  
  author text,  
  year_published int,  
  genre text,  
  isbn text,  
  PRIMARY KEY (book_id)  
);
```

```
DROP TABLE IF EXISTS library.books;  
CREATE TABLE library.books (  
  book_id int,  
  title text,  
  author text,  
  year_published int,  
  genre text,  
  isbn text,  
  PRIMARY KEY (book_id)  
);
```

No Result [Last query execution info](#)

Took 1 sec. Last updated by anonymous at April 08 2025, 2:59:39 PM.

3. Вставьте четыре книги в таблицу books.

-- Классика

```
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Classic', '978-0061120084');
```

-- Фантастика

```
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (2, 'Dune', 'Frank Herbert', 1965, 'Science Fiction', '978-0441172719');
```

-- Программирование

```
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (3, 'Clean Code', 'Robert C. Martin', 2008, 'Programming', '978-0132350884');
```

-- Детектив

```
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (4, 'The Girl with the Dragon Tattoo', 'Stieg Larsson', 2005, 'Mystery', '978-0307269751');
```

```
-- Классика
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Classic', '978-0061120084');

-- Фантастика
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (2, 'Dune', 'Frank Herbert', 1965, 'Science Fiction', '978-0441172719');

-- Программирование
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (3, 'Clean Code', 'Robert C. Martin', 2008, 'Programming', '978-0132350884');

-- Детектив
INSERT INTO library.books (book_id, title, author, year_published, genre, isbn)
VALUES (4, 'The Girl with the Dragon Tattoo', 'Stieg Larsson', 2005, 'Mystery', '978-0307269751');
```

No Result [Last query execution info](#)

Took 0 sec. Last updated by anonymous at April 08 2025, 3:01:02 PM.

4. Выберите все книги из таблицы books.

```
SELECT * FROM library.books;
```

SELECT * FROM library.books; FINISHED

book_id	author	genre	isbn	title	year_published
1	Harper Lee	Classic	978-0061120084	To Kill a Mockingbird	1960
2	Frank Herbert	Science Fiction	978-0441172719	Dune	1965
4	Stieg Larsson	Mystery	978-0307269751	The Girl with the Dragon Tattoo	2005
3	Robert C. Martin	Programming	978-0132350884	Clean Code	2008

Took 0 sec. Last updated by anonymous at April 08 2025, 3:01:35 PM.

5. Обновите поле year_published книги с book_id = 2 на 1950.

```
UPDATE library.books
```

```
SET year_published = 1950
```

```
WHERE book_id = 2;
```

Проверка:

```
SELECT title, author, year_published
```

```
FROM library.books
```

```
WHERE book_id = 2;
```

The screenshot displays a database management tool interface. The top section shows the execution of an UPDATE query: `UPDATE library.books SET year_published = 1950 WHERE book_id = 2;`. The status is 'FINISHED' and the result is 'No Result'. Below this, a SELECT query is shown: `SELECT title, author, year_published FROM library.books WHERE book_id = 2;`. The status is also 'FINISHED'. The results are displayed in a table with columns 'title', 'author', and 'year_published'. The table contains one row: 'Dune', 'Frank Herbert', and '1950'.

title	author	year_published
Dune	Frank Herbert	1950

Заключение

В результате выполнения данной работы, были получены практические и теоретические навыки по работе с Cassandra: создание и настройка keyspace, проектирование таблиц с различными типами данных, выполнение основных операций CRUD (INSERT, SELECT, UPDATE, DELETE) с использованием языка CQL, а также анализ производительности запросов.