

**Технически Университет – София**

**Факултет Приложна Математика и Информатика**

**Катедра Информатика**

**КУРСОВ ПРОЕКТ**

**Тема: РСУБД за избрана предметна област**

**Имена на студента:** Станислав Бисеров Стоянов

**Факултетен номер, група:** 471218066, 76 група

**Съдържание:**

1. Избор предметна област. Формулиране на изисквания и ограничения.
2. Проектиране на концептуален модел. Изясняване на структурни ограничения на обекти и връзки. Съставяне на диаграма на модела.
3. Проектиране на логически модел и схема на БД с избрана РСУБД. Тестване на релационните таблици с нормалните форми, нормализация. Визуализация на резултатната схема на БД.
4. Проектиране на основни операции с базата данни – търсене (няколко примера, влагане на операции), агрегиране, обновяване.
5. Визуализация на резултатите.
6. Използвани източници

1. Избор предметна област. Формулиране на изисквания и ограничения

* Предметна област

Релационна система за управление на база данни, която съдържа информация за повредени компютри (Computer Service), които се изпращат за ремонт на компютърните специалисти. Тази база би могла да се използва успешно със софтуер (настолно или уеб приложение) за обработка на постъпващи данни от фирма, която извършва поддръжка на реален хардуер или предоставя такива услуги на своите потребители.

* Формулиране на изисквания и ограничения

Основното изискване е базата да съдържа релационна схема за т.нар работни казуси (Jobs), части (Parts) и механици (Mechanics), които да обработват постъпващата информация от системата за реален проблем с даден компютър. Всеки компютърен специалист може да работи по много компютри и за всяка работа ще са му необходими части за поправяне на машината. За целта всеки такъв специалист прави поръчка за дадена част от списъка с части (Parts), специфирайки количество, като всяка една част има производител, съдържащ се в релационната схема производители (Vendors).

2. Проектиране на концептуален модел. Изясняване на структурни ограничения на обекти и връзки. Съставяне на диаграма на модела.

* Проектиране на концептуален модел и изясняване на структурни ограничения на обекти и връзки

Един клиент от релационната схема клиенти (Clients) има много работни казуси или за по просто машини за ремонтиране (компютри, лаптопи, друг вид електроника). За всеки клиент се пази информация за имената му и телефонния номер. По една повредена машина се работи от много механици (от релационната схема механици). Всяка машина за поправяне има пореден номер (Id) и име (Name). Като работният казус (машината за поправяне) съдържа информация за статуса (Status), на кой клиент принадлежи, кога е доставена, крайна дата за поправяне и от кой механик се поправя. За поправянето са необходими части, но те първо трябва да бъдат поръчани от механиците в определено количество (Quantity) и затова всеки работен казус има много поръчки за части. Като всяка поръчка съдържа дата за доставяне и дали вече е доставена. Осъществява се връзка много-към-много между поръчките (Orders) и частите (Parts), като тази междинна (mapping) релационна схема съдържа единствено количеството (Quantity), поредните номера на поръчката и частта (OrderId и PartId). От друга страна всеки работен казус (Job) се нуждае от много части и се реализира втора междинна релационна схема между Jobs и Parts, като тя съдържа единствено информация за количеството (Quantity), поредните номера на работния казус (JobId) и частта (PartId). Една част се произвежда от много производители и съответно производителите произвеждат по една част. В релационната схема части (Parts) се съдържа информация за серийния номер на частта (SerialNumber), кратко описание (Description), цена (Price), пореден номер на производителя (VendorId) и стоково количество (StockQty). За всеки производител се съхранява информация за поредния му номер (VendorId) и името му (Name).

* Съставяне на диаграма на модела

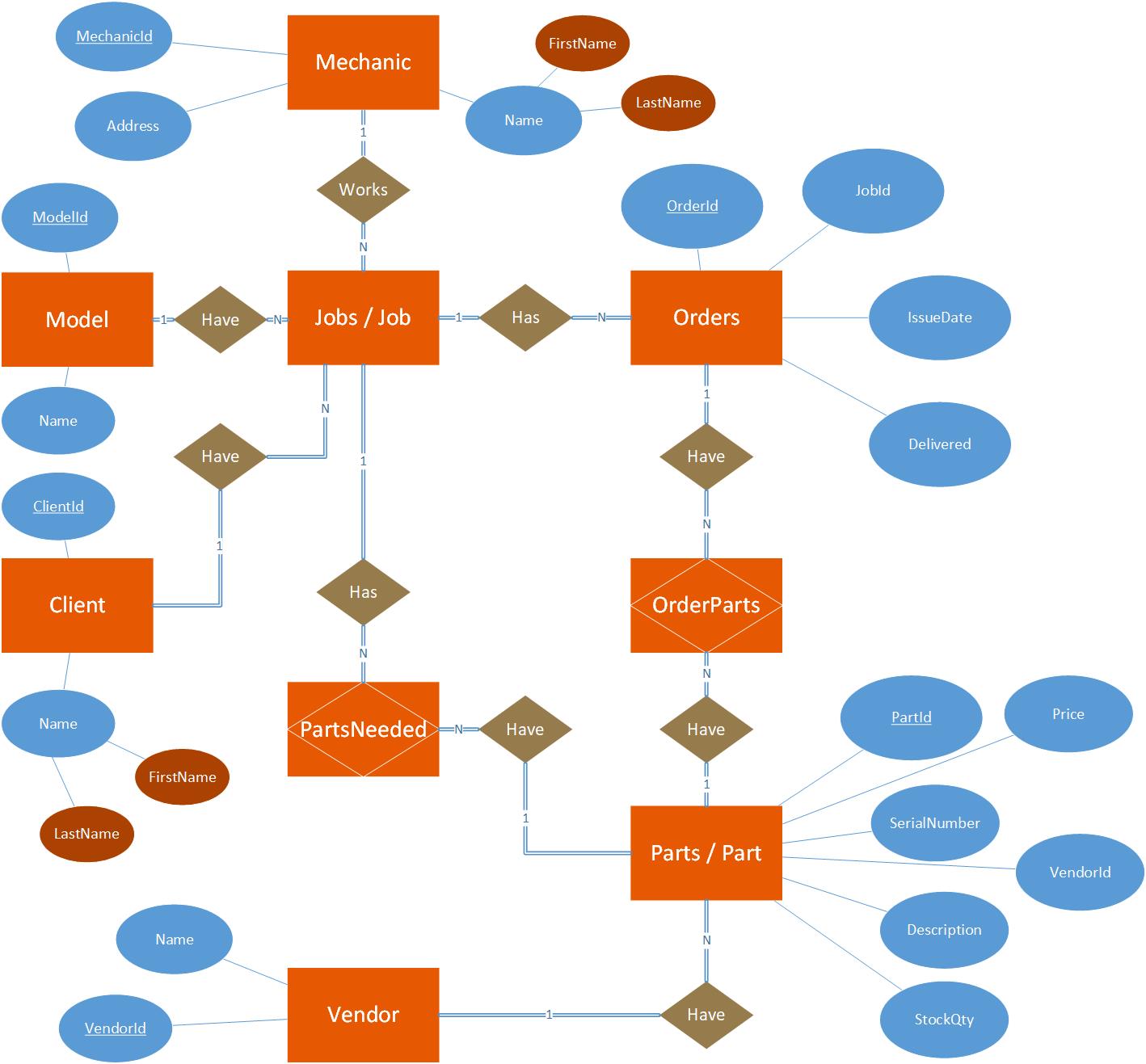


Figure 1 - Концептуален модел - Chen Database Notation

3. Проектиране на логически модел и схема на БД с избрана РСУБД. Тестване на релационните таблици с нормалните форми, нормализация. Визуализация на резултатната схема на БД.

* Проектиране на логически модел и схема на БД с избрана РСУБД

Използвайки концептуалния модел и съгласно структурните ограничения на обектите и връзките в т.2, се изгражда следният логически модел и схема на БД:

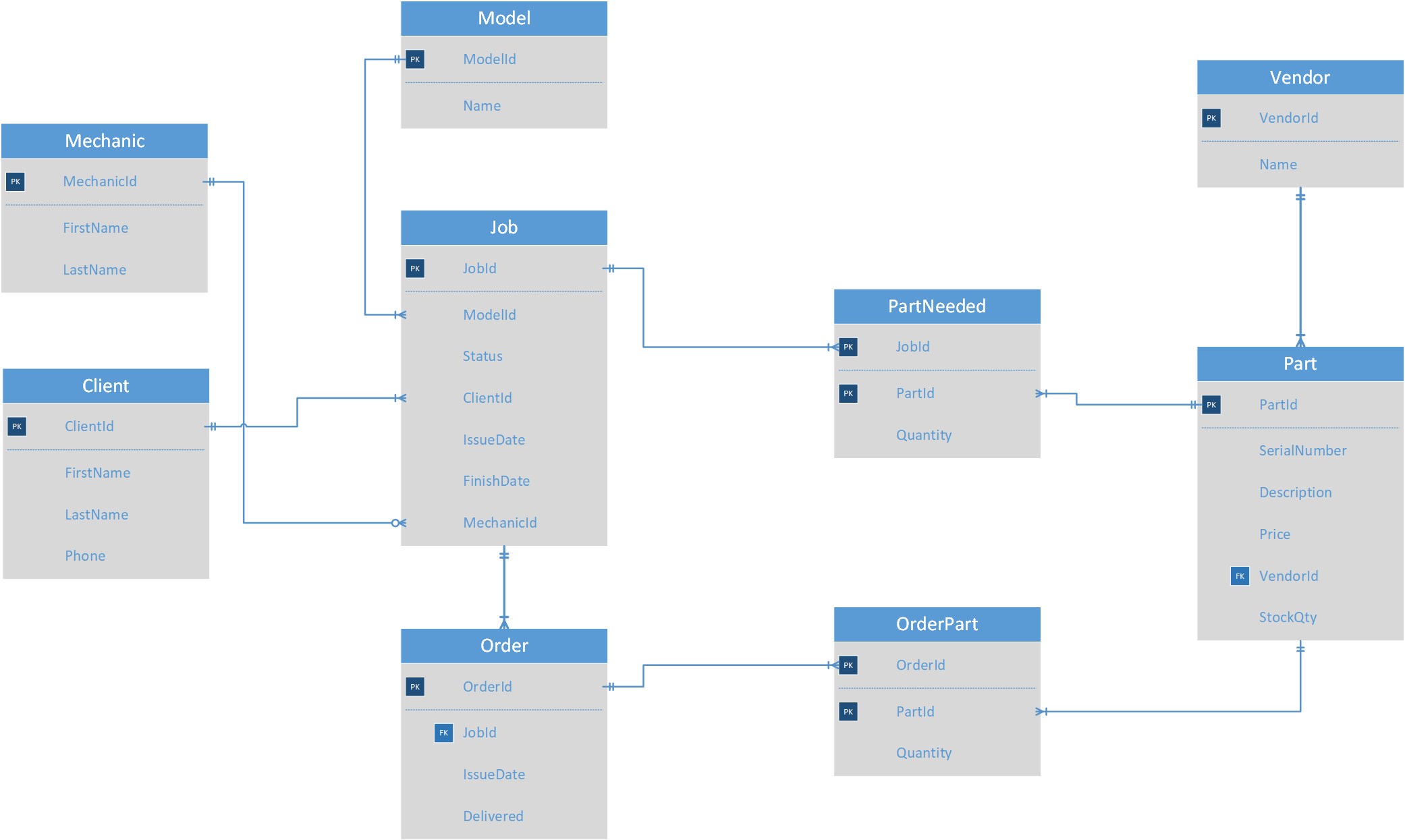


Figure 2 Логически модел - Crow's Foot Database Notation

Резултатната схема на базата данни съдържа следните релационни схеми (таблици), като за всяка една от тях са описани имената и типовете на атрибутите (колоните) и структурните им ограничения (constraints):

***Clients***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| ClientId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| FirstName | Стринг до 50 символа, ASCII |  |
| LastName | Стринг до 50 символа, ASCII |  |
| Phone | Стринг | Дължината на стринга е точно 12 символа |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***Mechanics***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| MechanicId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| FirstName | Стринг до 50 символа, ASCII |  |
| LastName | Стринг до 50 символа, ASCII |  |
| Address | Стринг до 255 символа, ASCII |  |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***Jobs***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| JobId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| ModelId | 32-бита инт (Integer) | Чужд ключ (Foreign Key), Връзка с таблица Models |
| Status | Стринг до 11 символа, ASCII | Позволени стойности: ‘Pending’, ‘In Progress’ и ‘Finished’; Стойност по подразбиране – ‘Pending’ (default value) |
| ClientId | 32-бита инт (Integer) | Чужд ключ (Foreign Key), Връзка с таблица Clients |
| MechanicId | 32-бита инт (Integer) | Чужд ключ (Foreign Key), Връзка с таблица Mechanics; Може да бъде NULL |
| IssueDate | Дата (Date) |  |
| FinishDate | Дата (Date) | Може да бъде NULL |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***Models***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| ModelId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| Name | Стринг до 50 символа, ASCII | Уникална стойност (Unique) |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***Orders***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| OrderId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| JobId | 32-бита инт (Integer) | Чужд ключ (Foreign Key), Връзка с таблица Jobs |
| IssueDate | Дата (Date) | Може да бъде NULL |
| Delivered | Булев тип (Boolean) | Стойност по подразбиране е False (default value) |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***Parts***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| PartId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| SerialNumber | Стринг до 50 символа, ASCII | Уникална стойност (Unique) |
| Description | Стринг до 255 символа, ASCII | Може да бъде NULL |
| Price | Парична стойност до 9999.99 | Не може да бъде <=0 |
| VendorId | 32-бита инт (Integer) | Чужд ключ (Foreign Key), Връзка с таблица Vendors |
| StockQty | 32-бита инт (Integer) | Не може да бъде отрицателна стойност; Стойност по подразбиране е 0 (default value) |

*\*Забежка:* Identity – означава автоинкрементиране на стойността

***OrderParts (мапинг таблица)***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| OrderId | 32-бита инт (Integer) | Първичен ключ (Primary Key); Връзка с таблица Orders |
| PartId | 32-бита инт (Integer) | Първичен ключ (Primary Key); Връзка с таблица Parts |
| Quantity | 32-бита инт (Integer) | Не може да бъде <=0; Стойност по подразбиране е 1 (default value) |

***PartsNeeded (мапинг таблица)***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| JobId | 32-бита инт (Integer) | Първичен ключ (Primary Key); Връзка с таблица Jobs |
| PartId | 32-бита инт (Integer) | Първичен ключ (Primary Key); Връзка с таблица Parts |
| Quantity | 32-бита инт (Integer) | Не може да бъде <= 0; Стойност по подразбиране е 1 (default value) |

***Vendors***

|  |  |  |
| --- | --- | --- |
| **Име на колона** | **Тип на данните** | **Ограничения** |
| VendorId | 32-бита инт (Integer) | Първичен ключ (Primary Key), Identity |
| Name | Стринг до 50 символа, ASCII | Уникална стойност (Unique) |

* Тестване на релационните таблици с нормалните форми, нормализация

Създателят на релационния модел [[1]](#footnote-1)Едгар Код предлага теорията за нормализиране с въвеждането на първа нормална форма (First Normal Form) и той продължава да разширява теорията с втора (Second Normal Form) и трета нормална форма (Third Normal Form). По-късно той се присъединява към Реймънд Ф. Бойс, за да разработи теорията за нормалната форма на Бойс-Код (Boyce-Codd Normal Form).

Теорията за нормализиране на данните в SQL все още се развива. Например, има дискусии дори и за 6-та нормална форма. Въпреки това, в повечето практически приложения, нормализирането постига най-доброто в 3-та нормална форма. Еволюцията на теориите за нормализация е илюстрирана по-долу:

***Първа нормална форма – правила:***

* Всяка клетка на релационната схема трябва да съдържа една стойност
* Всеки запис трябва да бъде уникален (unique)

*Пример:*

За да бъде постигната първа нормална форма, в релационната схема клиенти (Clients) както и в механици (Mechanics) има два атрибута за първо и фамилно име (FirstName и LastName). Лесно може да бъде постигнато нарушение на тази форма и да бъде отделен само един атрибут име (Name). Ако в някой от атрибутите има повече от една стойност, това ще доведе до съхраняване на запис с променлив брой полета, а също така и до записи с променлива дължина. Когато във всяка клетка на таблицата има само по една стойност, първичният ключ не е вече еднозначен, защото се среща повече от веднъж в таблицата. Що се касае до релационната схема работни казуси (Jobs) се образува съставен първичен ключ, който се състои от частичните ключове - “JobId” , “ModelId” и “MechanicId”. Новополучената релация “Jobs” има недостатъци, известни като аномалии при изтриване и промяна на данните.

* Аномалия при изтриване на данните – ако се изтрие работният казус, за когото е отговорен механикът, от таблицата ще бъдат изтрити всички данни за работен казус с пореден номер JobId
* Аномалия при промяна на данните – ако променим ModelId на работния казус само на един ред ще настъпи противоречие в данните

За преодоляване на тези недостатъци се преминава към втора нормална форма.

***Втора нормална форма – правила:***

* Да бъде спазена първа нормална форма
* Всяка колона да има само един първичен ключ (Single Column Primary Key)

*Пример:*

За да бъде спазена втората нормална форма всеки атрибут трябва да е функционално зависим от всички атрибути, съставляващи първичния ключ. Ако в една релация се съдържат атрибути, които са функционално зависими не от целия първичен ключ, а от части от него, то за тях трябва да се състави нова релационна схема (таблица). В базата се създават три отделни таблици за моделите (Models), клиентите (Clients) и механиците (Mechanics). Осъществяват се три връзки 1-към-много (one-to-many) посредством свързването на първичните ключове с чуждите. В резултат, на което таблицата Jobs съдържа три чужди ключа.

* Защо е необходимо използването на чужд ключ (Primary Key)?

Представете си, че потребителят, използващ системата за съхраняване на информацията, се опита да въведе данни за модел на компютър с пореден номер 101 в таблица Jobs, но в същото време не същестувава такъв модел с пореден номер в таблицата Models. Базата хвърля грешка, което сподпомага рефернтната цялост (referential integrity). За преодоляване на този проблем се прилага решението описано по-горе чрез създаване на отделни таблици с чужди ключове.

***Трета нормална форма – правила:***

* Да бъде спазена втора нормална форма
* Да няма функционални зависимости между атрибутите (transitive functional dependencies)

*Пример:*

Нека отново се върнем към горния пример с релационната схема Jobs. Понеже на всеки чужд ключ съотвества първичен в отделна таблица и няма функционални зависимости между атрибутите, следва, че таблицата се намира в трета нормална форма.

***Заключение:***

Представената базата не е толкова комплексна, за да предполага такова високо ниво на нормализация. Всъщност, реализацията на релационните схеми спазва първите три нормални форми. За да се премине към останалите, самата база трябва да бъде комплексна и в по-голям мащаб.

* Визуализация на резултатната схема на БД

Като основен иструмент за реализация на базата се използва услугата на Microsoft и по – точно [[2]](#footnote-2)SQL Server Management Studio (SSMS). Езикът на всички заявки по-долу е MSSQL. Следвайки изготвения логически модел, проектирането на базата се разделя на 4 части – относто структурата на базата - DDL (Data Definition Language), относно операциите с базата - DML (Data Manipulation Language), основни заявки и програмна логика (Programmability). DML, основните заявки и програмната логика ще бъдат засегнати в точка 4.

***Резултатна схема на БД:***

4. Проектиране на основни операции с базата данни – търсене (няколко примера, влагане на операции), агрегиране, обновяване

* Data Definition Language (DDL) – основна заявка за създаване на базата и релационните схеми (намира се в т.5 Визуализация на резултатите)
* Data Manipulation Language (DML) – заявки за Insert, Update и Delete

Преди реализацията на заявките се импортира готов набор с данни за базата (DDL\_Dataset.sql), който е приложен към проекта.

* Insert (Въвеждане на данни)

***Clients (Клиенти)***

|  |  |  |
| --- | --- | --- |
| **Първо име** | **Фамилия** | **Телефонен номер** |
| Teri | Ennaco | 570-889-5187 |
| Merlyn | Lawler | 201-588-7810 |
| Georgene | Montezuma | 925-615-5185 |
| Jettie | Mconnell | 908-802-3564 |
| Lemuel | Latzke | 631-748-6479 |
| Melodie | Knipp | 805-690-1682 |
| Candida | Corbley | 908-275-8357 |

*SQL заявка:*

INSERT INTO Clients (FirstName, LastName, Phone) VALUES

('Teri', 'Ennaco', '570-889-5187')

('Merlyn', 'Lawler', '201-588-7810')

('Georgene', 'Montezuma', '925-615-5185')

('Jettie', 'Mconnell', '908-802-3564')

('Lemuel', 'Latzke', '631-748-6479')

('Melodie', 'Knipp', '805-690-1682')

('Candida', 'Corbley', '908-275-8357')

***Parts (Части)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Сериен номер** | **Описание** | **Цена** | **Име на производител** |
| WP8182119 | Intel Core i5-9600K | 497 | Intel |
| W10780048 | Intel Core i9-9900K | 1118.42 | Intel |
| W10841140 | AMD Ryzen Threadripper 2950X | 1830 | AMD |
| WPY055980 | AMD RYZEN 7 1700X | 507.46 | AMD |

*SQL заявка:*

INSERT INTO Parts (SerialNumber, [Description], Price, VendorId) VALUES

('WP8182119', 'Intel Core i5-9600K', 497, 2),

('W10780048', 'Intel Core i9-9900K', 1118.42, 2),

('W10841140', 'AMD Ryzen Threadripper 2950X', 1830, 1),

('WPY055980', 'AMD RYZEN 7 1700X', 507.46, 1)

* Update (Обновяване на данни)

*Задача 1:*

*Да се обнови стойност ‘Pending’ на статуса (Status) в таблица Jobs на механик с име Ryan Harnos и да се обнови този статус на ‘In Progress’.*

*SQL заявка:*

UPDATE Jobs

SET MechanicId = 3, [Status] = 'In Progress'

WHERE [Status] = 'Pending'

* Delete (Изтриване на данни)

*Задача 2:*

*Да се отмени поръчка с пореден номер 19 – изтриване на поръчката от базата и всички свързани връзки в мапинг таблицата (OrderParts).*

*SQL заявка:*

DELETE FROM OrderParts

WHERE OrderId = 19

DELETE FROM Orders

WHERE OrderId = 19

* Заявки (Querying)

*Задача 3 – Клиенти по име:*

*Да се селектират всички клиенти сортирани възходящо по фамилия и след това по пореден номер отново възходящо.*

*SQL заявка:*

SELECT FirstName, LastName, Phone

FROM Clients

ORDER BY LastName, ClientId

*Задача 4 – Статус на работен казус (Job Status)*

*Да се намерят всички работни казуси, които са активни и в процес на обработване (не са завършени) и да се селектира статусът и датата (IssueDate). Да се сортират възходящо датата и поредният номер.*

*SQL заявка:*

SELECT [Status], IssueDate

FROM Jobs

WHERE [Status] = 'In Progress' OR [Status] = 'Pending'

ORDER BY IssueDate, JobId

*Задача 5 – Механици и работни казуси*

*Да се селектират пълното име на механика, статусът и датата на работните му казуси (Jobs). Да се сортират възходящо поредният номер на механика, датата и поредният номер на работния казус.*

*SQL заявка:*

SELECT m.FirstName + ' ' + m.LastName AS [Mechanic],

j.[Status],

j.IssueDate

FROM Mechanics AS m

LEFT JOIN Jobs AS j ON j.MechanicId = m.MechanicId

ORDER BY m.MechanicId, j.IssueDate, j.JobId

*Задача 6 – Производителност на механиците*

*Да се селектират първото и фамилно име на всички механици и средното време в дни, което им отнема, за да завършват своите работни казуси (Jobs). Средното време да бъде целочислено число (Integer). Да се сортира възходящо по поредните номера на механиците.*

*SQL заявка:*

SELECT CONCAT(m.FirstName, ' ', m.LastName) AS [Mechanic],

AVG(DATEDIFF(DAY, j.IssueDate, j.FinishDate)) AS [Average Days]

FROM Mechanics AS m

LEFT JOIN Jobs AS j ON m.MechanicId = j.MechanicId

GROUP BY m.MechanicId,

m.FirstName,

m.LastName

ORDER BY m.MechanicId

*Задача 7 – Механици с най-много работа*

*Да се селектират първото име, фамилията и броят работни казуси (Jobs) на първите трима механици, които имат повече от 1 активен работен казус (Job), т.е незавършен. Да се сортират низходящо по броя работни казуси и след това възходящо по поредния номер на механика.*

*SQL заявка:*

SELECT TOP (3) CONCAT(m.FirstName, ' ', m.LastName) AS Mechanic,

COUNT(j.JobId) AS Jobs

FROM Mechanics AS m

LEFT JOIN Jobs AS j ON m.MechanicId = j.MechanicId

GROUP BY m.FirstName,

m.LastName,

m.MechanicId,

j.[Status]

HAVING COUNT(j.JobId) > 1

AND j.[Status] <> 'Finished'

ORDER BY Jobs DESC,

m.MechanicId

*Задача 8 – Цена на части*

*Да се селектира общата цена на всички поръчани части през последните 3 седмици. Нека начална дата бъде 24 април 2017г.*

*SQL заявка:*

SELECT SUM(p.Price \* op.Quantity) AS [Parts Total]

FROM Parts AS p

LEFT JOIN OrderParts AS op ON p.PartId = op.PartId

LEFT JOIN Orders AS o ON op.OrderId = o.OrderId

WHERE DATEDIFF(WEEK, o.IssueDate, '2017-04-24') <= 3

*Задача 9 - Разходи за части*

*Да се селектират всички завършени работни казуси (Jobs) и общата цена на всички поръчани части за тях. Да се сортира низходящо общата цена на частите и възходящо по пореден номер на работния казус (Job Id).*

*SQL заявка:*

SELECT j.JobId AS [Job Id],

ISNULL((SUM(p.Price \* op.Quantity)), 0) AS [Total Parts Cost]

FROM Jobs AS j

LEFT JOIN Orders AS o ON j.JobId = o.JobId

LEFT JOIN OrderParts AS op ON o.OrderId = op.OrderId

LEFT JOIN Parts AS p ON op.PartId = p.PartId

WHERE j.[Status] = 'Finished'

GROUP BY j.JobId

ORDER BY [Total Parts Cost] DESC,

j.JobId

*Задача 10 – Компютър, който се поврежда най-често*

*Да се селектира компютърът, който се поврежда най-често (има най-много работни казуси свързани с него (Jobs)) както и общият брой на обслужвани пъти (Times Serviced). Да се селектира също така общата цена на всички поръчани части за този компютър. Ако има повече от един такъв компютър, да се селектират всички.*

*SQL заявка:*

SELECT TOP (1) WITH TIES m.[Name] AS [Model],

COUNT(j.JobId) AS [Times Serviced],

(

SELECT ISNULL(SUM(p.Price \* op.Quantity), 0)

FROM Orders AS o

LEFT JOIN Jobs AS j ON j.JobId = o.JobId

LEFT JOIN OrderParts AS op ON op.OrderId = o.OrderId

LEFT JOIN Parts AS p ON p.PartId = op.PartId

WHERE j.ModelId = m.ModelId

) AS [Parts Total]

FROM Models AS m

LEFT JOIN Jobs AS j ON j.ModelId = m.ModelId

GROUP BY m.ModelId,

m.[Name]

ORDER BY [Times Serviced] DESC

* Програмна логика (Programmability)

*Задача 11 – Да се напише потребителски дефинирана функция (user defined function), която получава като входен параметър пореден номер на работен казус (Job Id) и за този номер връща като резултат общата цена на всички поръчани части. Ако няма такива поръчки, да връща 0.*

*SQL заявка:*

CREATE FUNCTION udf\_GetCost(@jobId INT)

RETURNS DECIMAL(8, 2)

AS

BEGIN

DECLARE @totalCost DECIMAL(8, 2);

SELECT @totalCost = SUM(p.Price \* op.Quantity)

FROM Jobs AS j

LEFT JOIN Orders AS o ON o.JobId = j.JobId

LEFT JOIN OrderParts AS op ON op.OrderId = o.OrderId

LEFT JOIN Parts AS p ON p.PartId = op.PartId

WHERE j.JobId = @jobId

IF(@totalCost IS NULL)

BEGIN

SET @totalCost = 0;

END;

RETURN @totalCost;

END

SELECT dbo.udf\_GetCost(1) AS [Result]

*Задача 12 – Да се напише потребителски дефинирана процедура (user defined procedure), която получава като входен параметър пореден номер на работен казус (Job Id), сериен номер на част и нейното количество. Да се създаде (insert) поръчка със зададените параметри при условие, че началната дата е липсваща стойност NULL. Ако вече съществува такава поръчка, да се увеличи количеството й чрез входната стойност на процедурата. Когато се добави нова поръчка да се зададе NULL за начална дата.*

*Ограничения:*

* *Не може да се направи поръчка за работен казус, който е завършен (Finished). Съобщение на грешката – “Работният казус не е активен!”, с пореден номер 50011*
* *Количеството не може да бъде <= 0. Съобщение на грешката – “Количеството на частта не може да бъде по-малко или равно на нула!” , с пореден номер 50012*
* *Поредният номер на работния казус трябва да съществува в базата. Съобщение на грешката – “Не е намерен работен казус в базата!”, с пореден номер 50013.*
* *Частта с дадения сериен номер трябва да съществува в базата. Съобщение на грешката – “Не е намерена част в базата!”, с пореден номер 50014*

*Ако някое от описаните ограничения не се спази, да се хвърли грешка с подходящото съобщение и код 1 (state).*

*SQL заявка:*

CREATE PROCEDURE usp\_PlaceOrder

(

@jobId INT,

@partSerialNumber VARCHAR(50),

@quantity INT

)

AS

BEGIN

-- Намираме дадената поръчка и част.

DECLARE @partId INT = (SELECT PartId FROM Parts WHERE SerialNumber = @partSerialNumber);

DECLARE @orderId INT = (SELECT TOP (1) OrderId FROM Orders WHERE JobId = @jobId AND IssueDate IS NULL);

-- Проверяваме дали количеството е в допустимите стойности.

IF(@quantity <= 0)

BEGIN

;THROW 50012, 'Количеството на частта не може да бъде по-малко или равно на нула!', 1

END

-- Проверяваме дали има такъв работен казус със статус 'Завършен'

IF((SELECT JobId FROM Jobs WHERE JobId = @jobId AND [Status] = 'Finished') IS NOT NULL)

BEGIN

;THROW 50011, 'Работният казус не е активен!', 1

END

-- Проверяваме дали съществува такъв работен казус в базата

IF((SELECT JobId FROM Jobs WHERE JobId = @jobId) IS NULL)

BEGIN

;THROW 50013, 'Не е намерен работен казус в базата!', 1

END

-- Проверяваме дали съществува такава част в базата

IF(@partId IS NULL)

BEGIN

;THROW 50014, 'Не е намерена част в базата!', 1

END

-- Ако няма такава поръчка, я създаваме. След това я добавяме в Orders и OrderParts.

IF(@orderId IS NULL)

BEGIN

-- Delivered има стойност по подразбиране. За начална дата задаваме NULL стойност.

INSERT INTO Orders(JobId, IssueDate) VALUES

(@jobId, NULL)

-- Взимаме поредния номер на вече добавената поръчка

DECLARE @id INT = (SELECT TOP 1 OrderId FROM Orders WHERE JobId = @jobId);

INSERT INTO OrderParts(OrderId, PartId, Quantity) VALUES

(@id, @partId, @quantity)

END

ELSE

BEGIN

-- Ако съществува такава поръчка за дадения работен казус, т.е имаме @orderId и @partId в OrderParts, трябва да ъпдейтнем нейното количество.

IF((SELECT OrderId FROM OrderParts WHERE OrderId = @orderId AND PartId = @partId) IS NOT NULL)

BEGIN

UPDATE OrderParts

SET Quantity += @quantity

WHERE OrderId = @orderId AND PartId = @partId

END

ELSE

-- Ако съществува такъв @orderId в базата, но не съществува такава част, съответно добавяме поръчката отново в OrderParts.

BEGIN

INSERT INTO OrderParts(OrderId, PartId, Quantity) VALUES

(@orderId, @partId, @quantity)

END

END

END

-- Тестване на процедурата

DECLARE @err\_msg AS NVARCHAR(MAX);

BEGIN TRY

-- Изпълнение на процедурата

EXEC usp\_PlaceOrder 1, 'ZeroQuantity', 0

END TRY

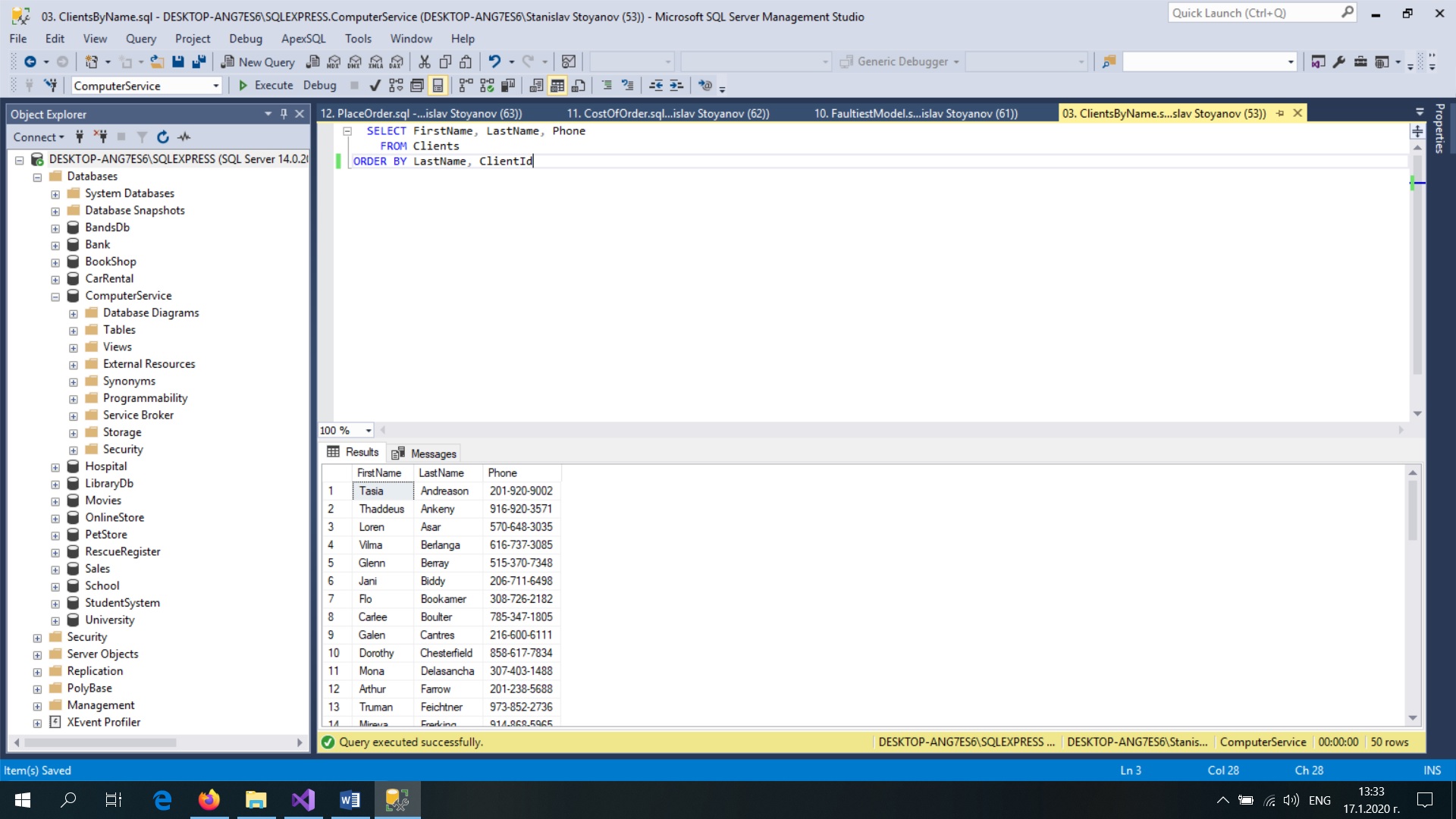
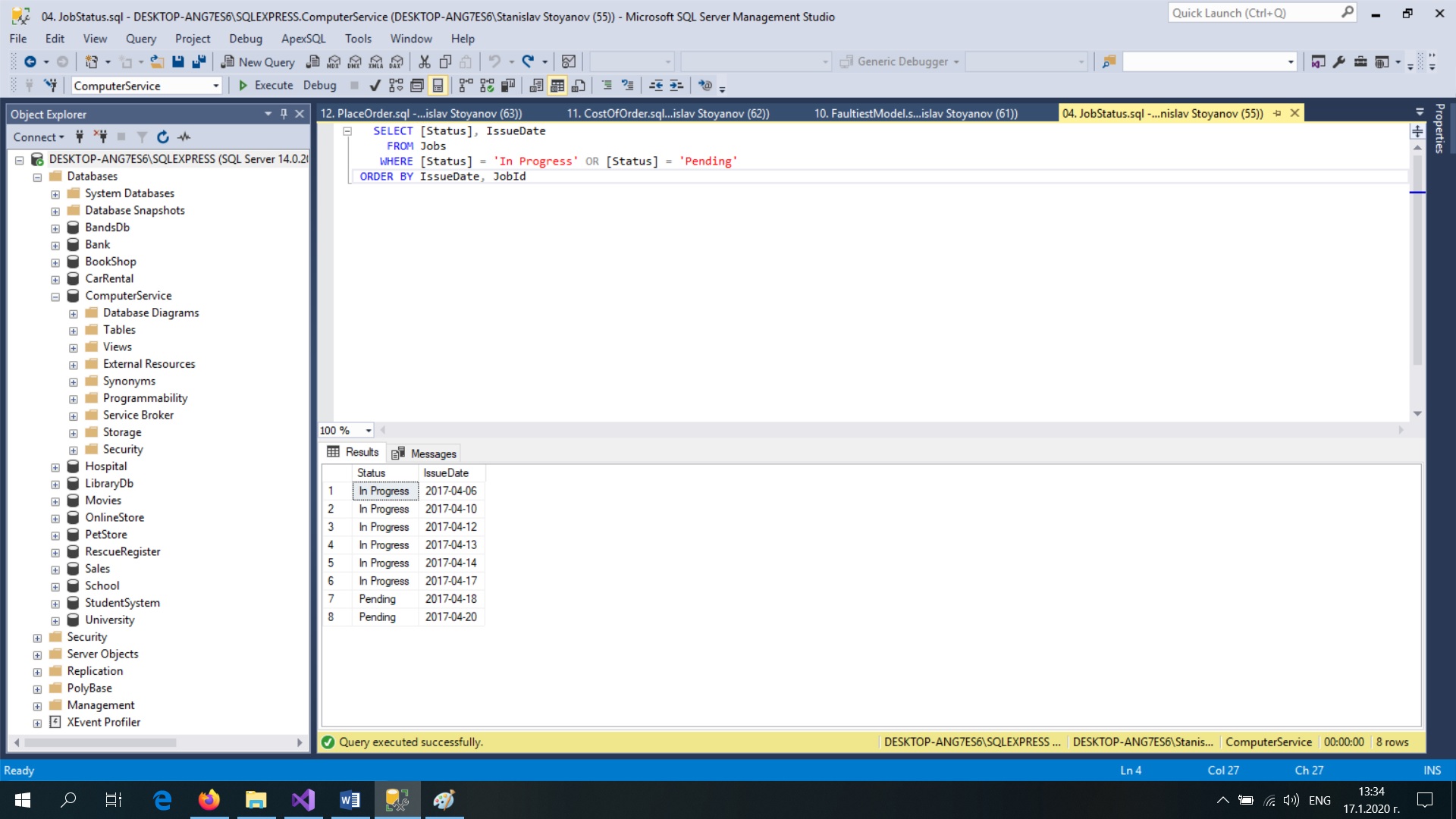
BEGIN CATCH

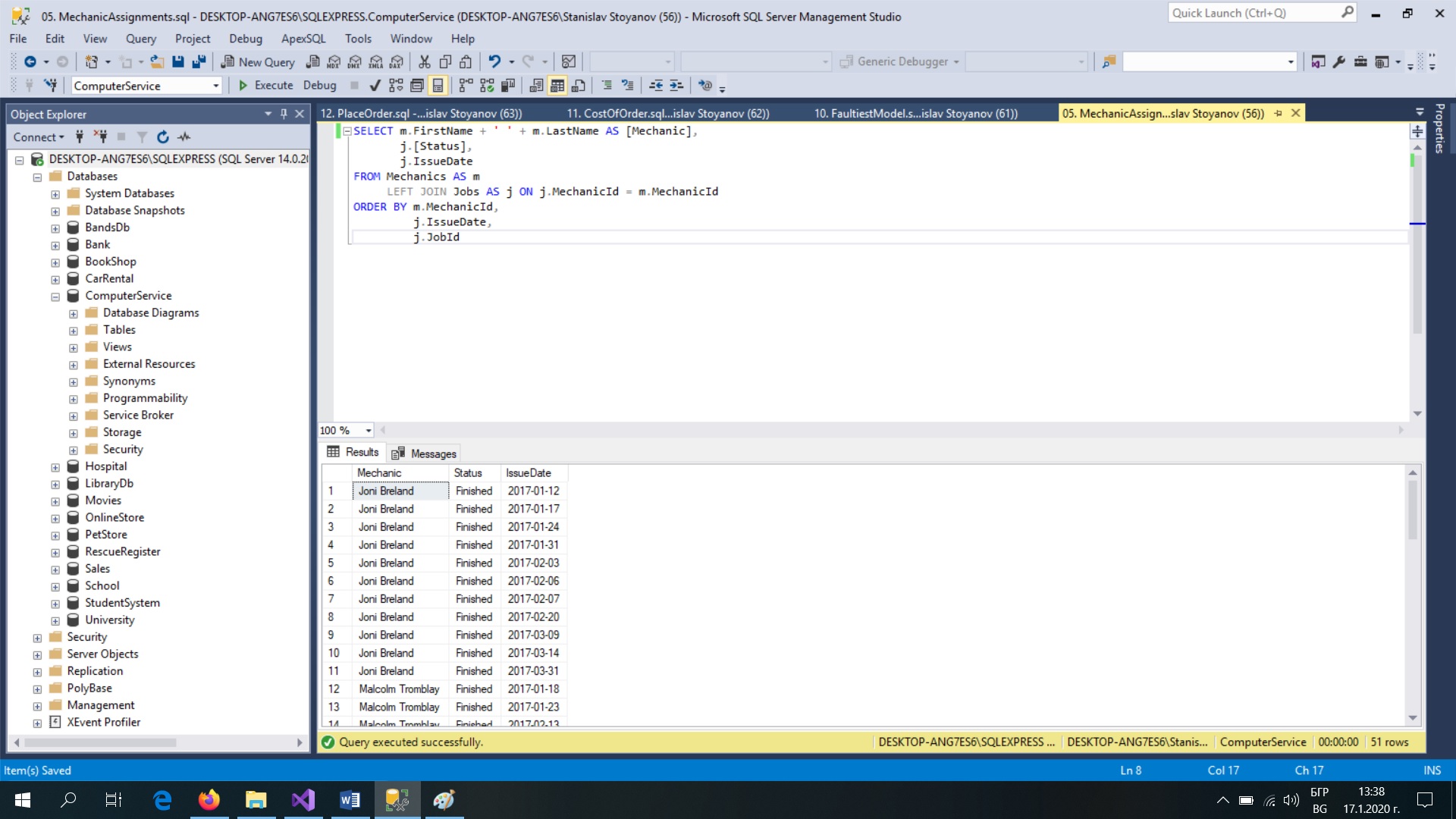
SET @err\_msg = ERROR\_MESSAGE();

SELECT @err\_msg AS [Result]

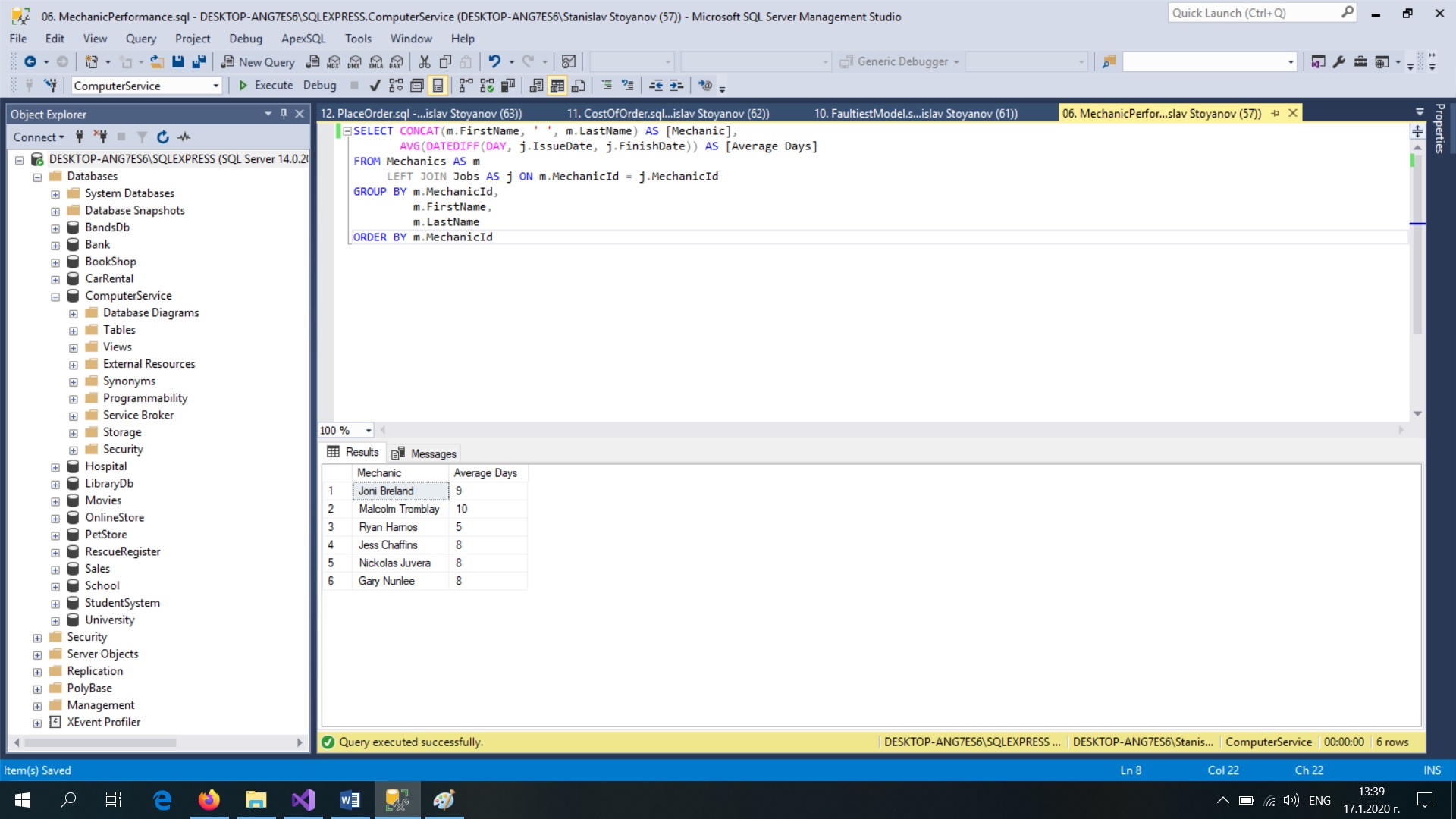
END CATCH

5. Визуализация на резултатите.

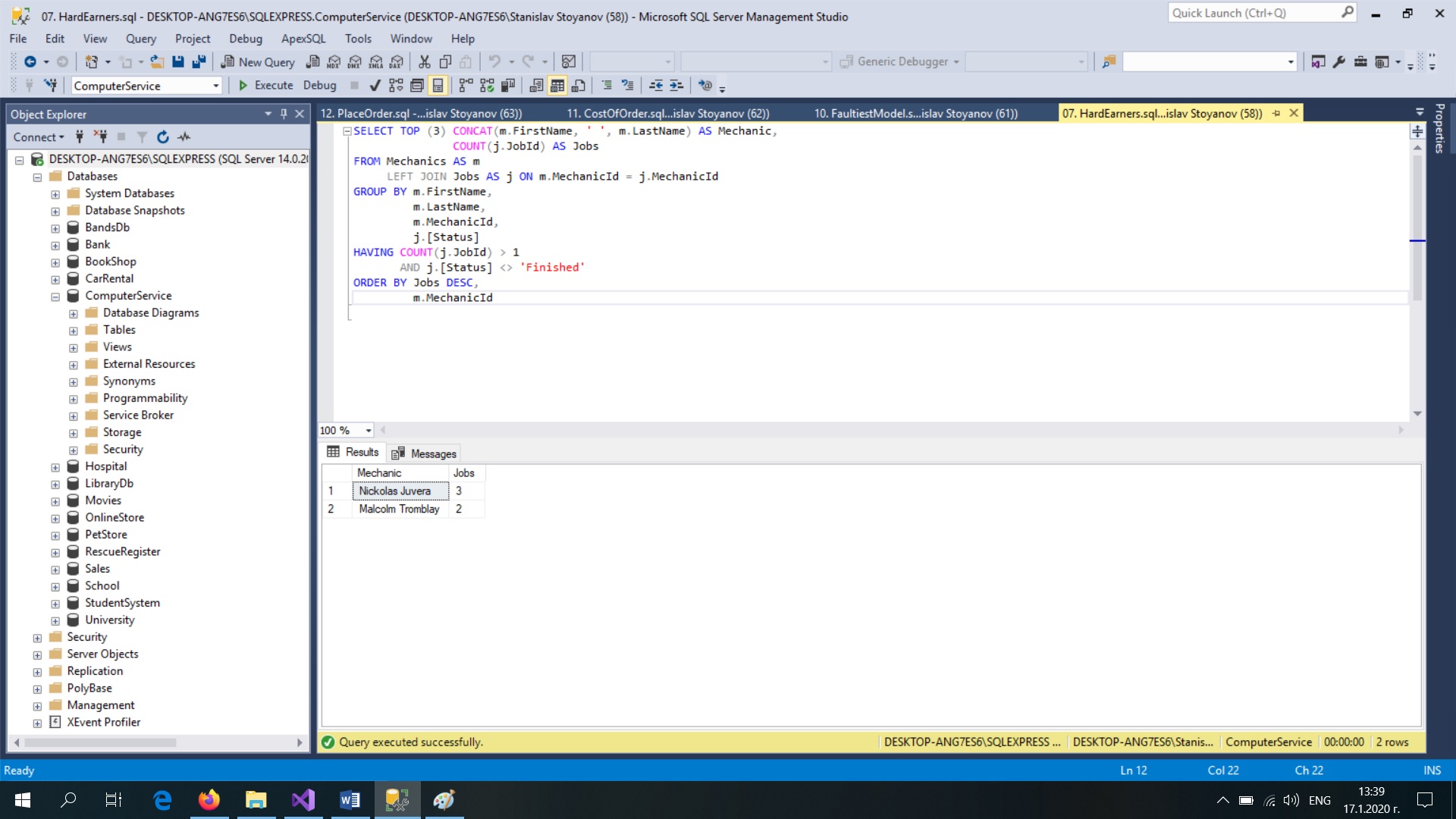
* Заявки (Querying)
* *Задача 3 – Клиенти по име*
* *Задача 4 – Статус на работен казус (Job Status)*
* *Задача 5 – Механици и работни казуси*

**

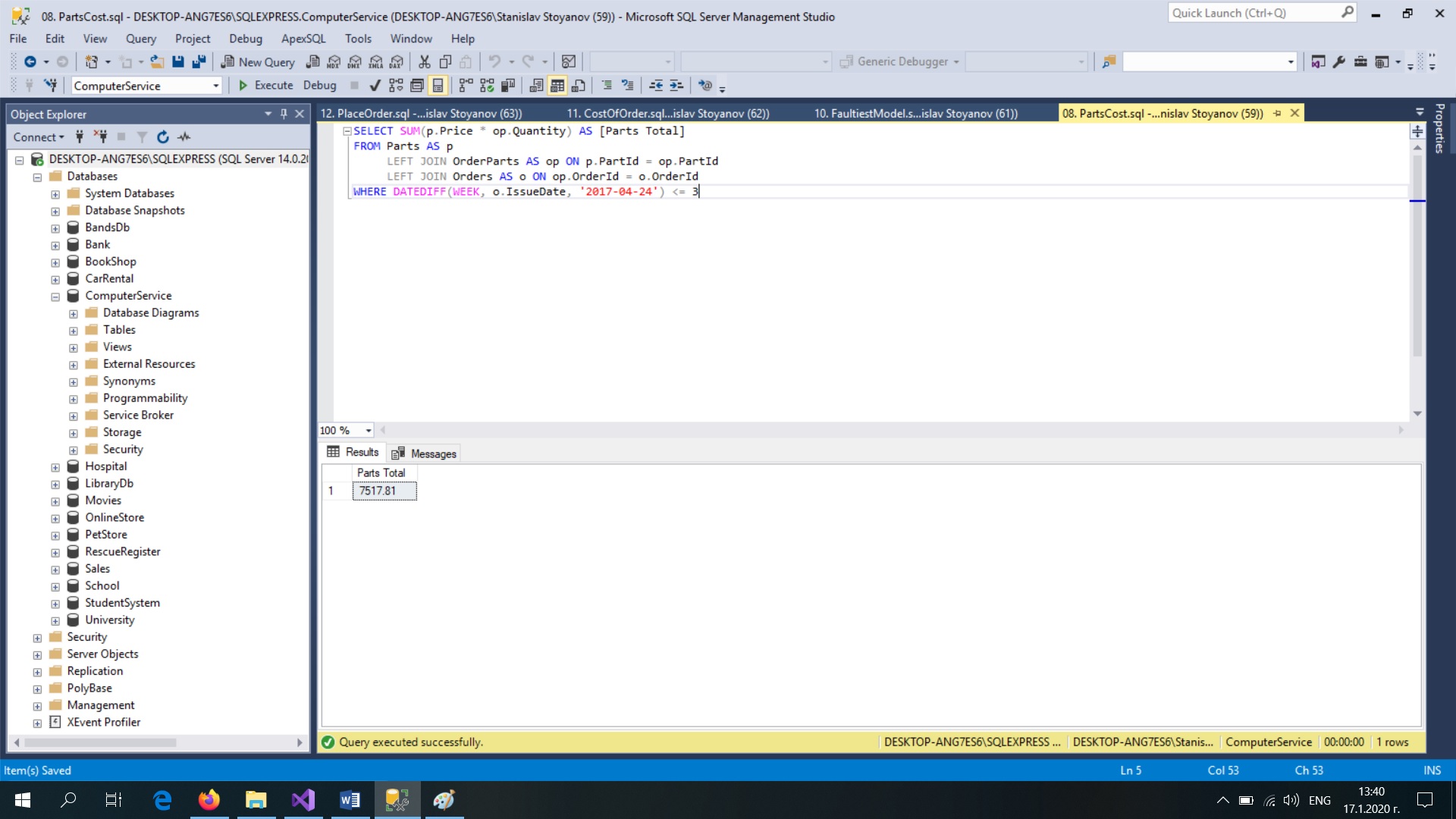
* *Задача 6 – Производителност на механиците*

**

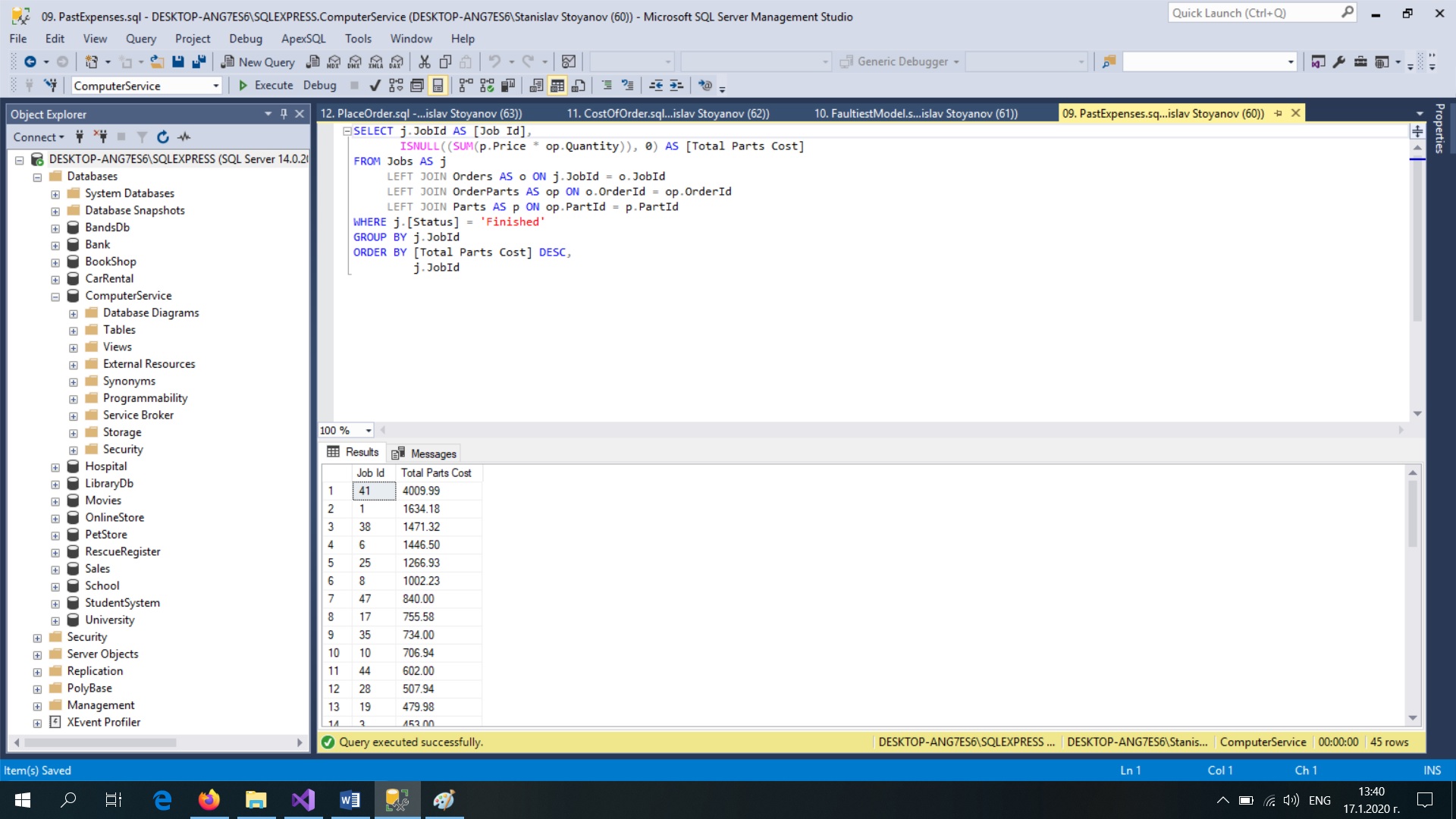
* *Задача 7 – Механици с най-много работа*

**

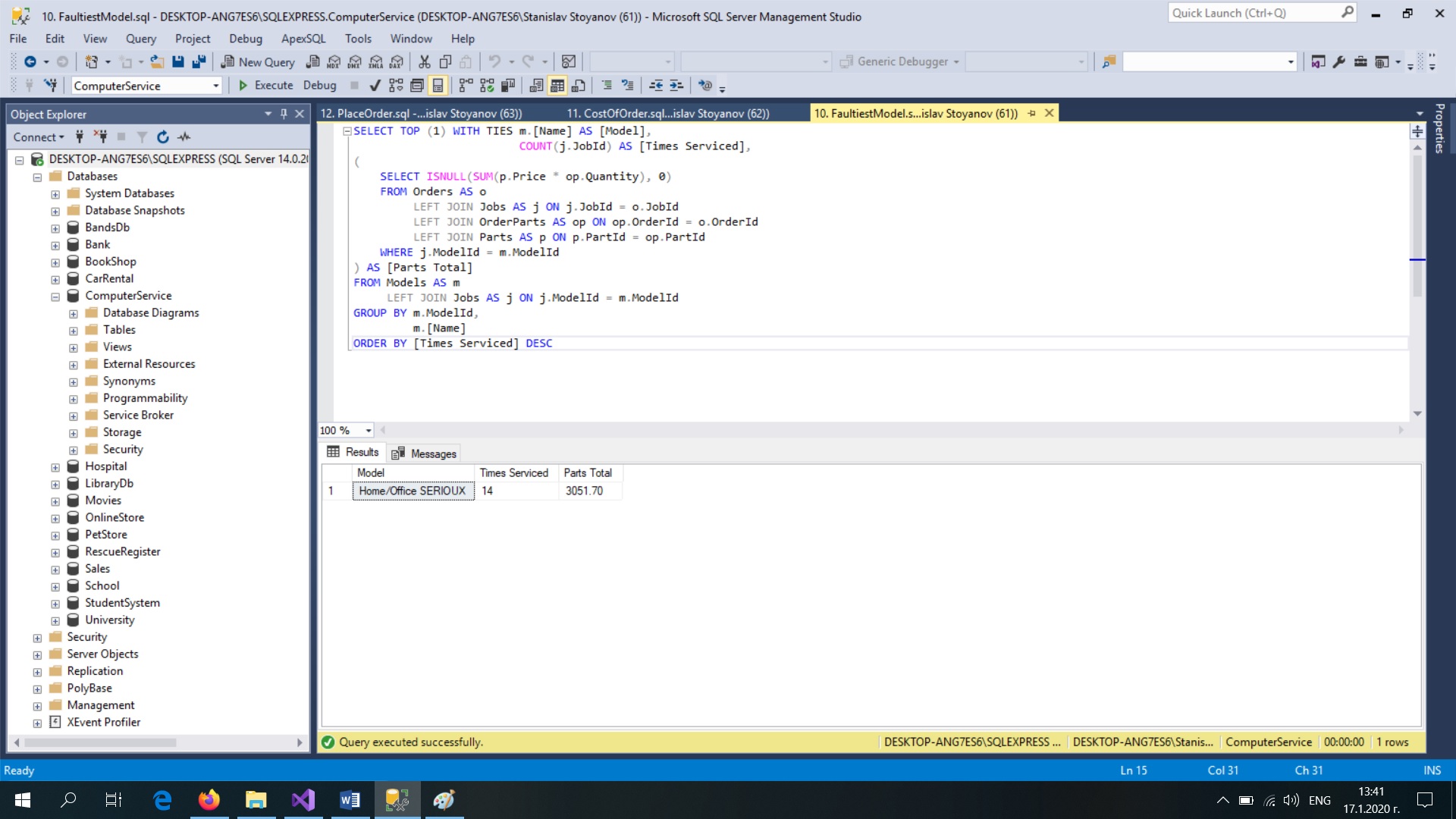
* *Задача 8 – Цена на части*

**

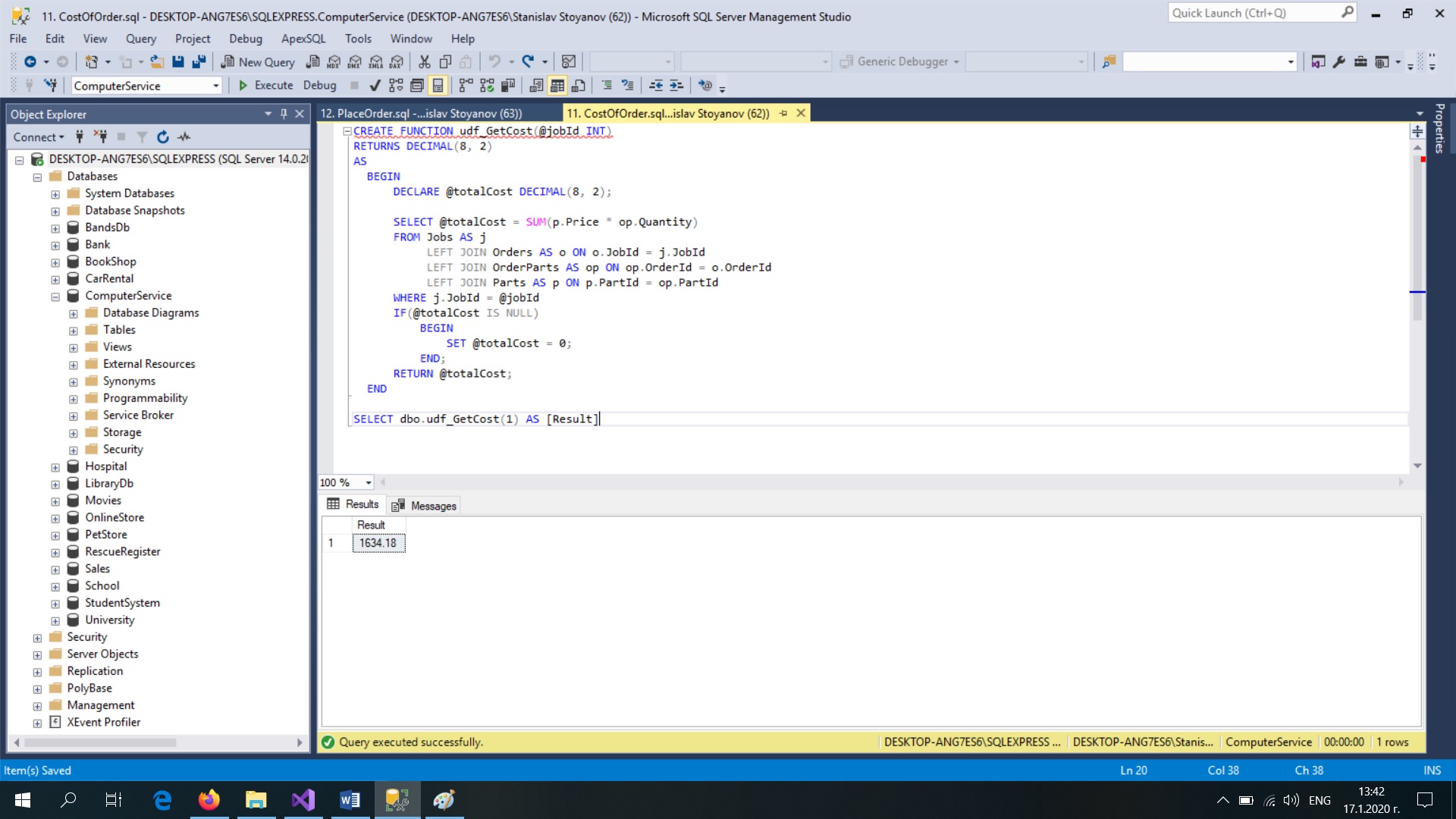
* *Задача 9 - Разходи за части*

**

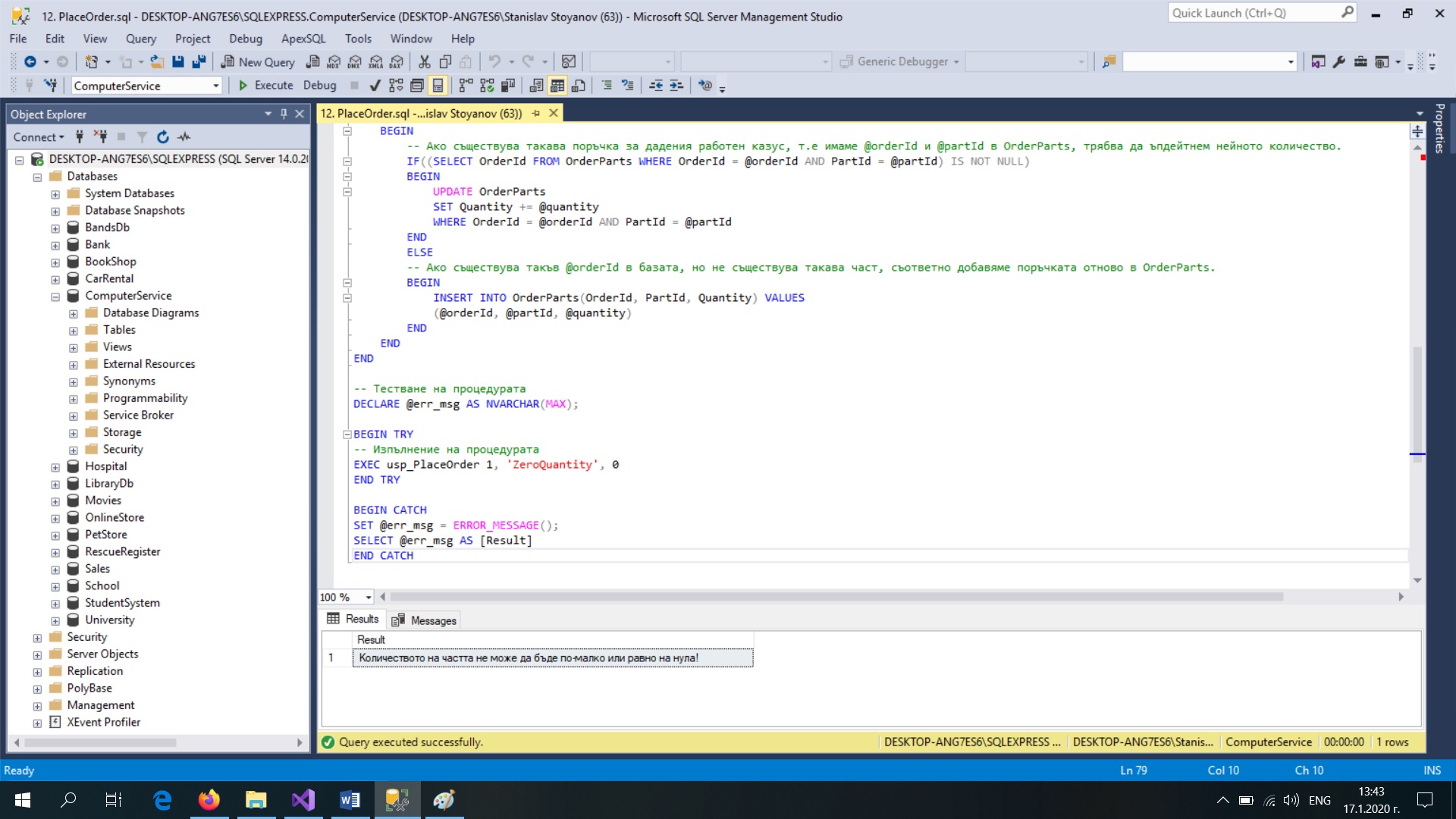
* *Задача 10 – Компютър, който се поврежда най-често*

**

* *Задача 11 – потребителски дефинирана функция*

**

* *Задача 12 – потребителски дефинирана процедура*

**

* Data Definition Language (DDL)

CREATE DATABASE ComputerService

USE ComputerService

CREATE TABLE Clients (

ClientId INT PRIMARY KEY IDENTITY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

Phone CHAR(12) NOT NULL

)

CREATE TABLE Mechanics (

MechanicId INT PRIMARY KEY IDENTITY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

[Address] NVARCHAR(255) NOT NULL

)

CREATE TABLE Models (

ModelId INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(50) NOT NULL UNIQUE

)

CREATE TABLE Jobs (

JobId INT PRIMARY KEY IDENTITY,

ModelId INT NOT NULL,

[Status] NVARCHAR(11) DEFAULT 'Pending' CHECK ([Status] IN ('Pending', 'In Progress', 'Finished')),

ClientId INT NOT NULL,

MechanicId INT,

IssueDate DATE NOT NULL,

FinishDate DATE,

CONSTRAINT FK\_Jobs\_Models

FOREIGN KEY (ModelId) REFERENCES Models(ModelId),

CONSTRAINT FK\_Jobs\_Clients

FOREIGN KEY (ClientId) REFERENCES Clients(ClientId),

CONSTRAINT FK\_Jobs\_Mechanics

FOREIGN KEY (MechanicId) REFERENCES Mechanics(MechanicId)

)

CREATE TABLE Orders (

OrderId INT PRIMARY KEY IDENTITY,

JobId INT NOT NULL FOREIGN KEY REFERENCES Jobs(JobId),

IssueDate DATE,

Delivered BIT DEFAULT 0

)

CREATE TABLE Vendors (

VendorId INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(50) NOT NULL UNIQUE

)

CREATE TABLE Parts (

PartId INT PRIMARY KEY IDENTITY,

SerialNumber NVARCHAR(50) UNIQUE,

[Description] NVARCHAR(255),

Price DECIMAL(6, 2) NOT NULL CHECK(Price > 0),

VendorId INT NOT NULL,

StockQty INT NOT NULL DEFAULT 0 CHECK(StockQty >= 0),

CONSTRAINT FK\_Parts\_Vendors

FOREIGN KEY(VendorId) REFERENCES Vendors(VendorId)

)

CREATE TABLE OrderParts (

OrderId INT FOREIGN KEY REFERENCES Orders(OrderId),

PartId INT FOREIGN KEY REFERENCES Parts(PartId),

Quantity INT DEFAULT 1 CHECK(Quantity >= 1),

CONSTRAINT PK\_OrderParts PRIMARY KEY(OrderId, PartId))

CREATE TABLE PartsNeeded (

JobId INT FOREIGN KEY REFERENCES Jobs(JobId),

PartId INT FOREIGN KEY REFERENCES Parts(PartId),

Quantity INT DEFAULT 1 CHECK(Quantity >= 1),

CONSTRAINT PK\_PartsNeeded PRIMARY KEY(JobId, PartId)

)

6. Използвани източници

* Уикипедия - <https://bg.wikipedia.org>
* Какво представлява нормализацията? Примери за нормализация. – <https://www.guru99.com/database-normalization.html>

1. **Едгар Франк Код** (на английски: Edgar Frank Codd) (1923 – 2003) е британски информатик [↑](#footnote-ref-1)
2. SQL Server Management Studio (SSMS) – (Студио за управление на SQL Server) е софтуерно приложение, стартирано за първи път с Microsoft SQL Server 2005, което се използва за конфигуриране, управление и администриране на всички компоненти в SQL Server. [↑](#footnote-ref-2)