МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных мессенджера с реализацией технологии шифрования»

Выполнил студент Вабищевич Сергей Петрович

(Ф.И.О.)

Руководитель проекта Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[Реферат 3](#_Toc59130586)

[1. Аналитический обзор литературы 4](#_Toc59130587)

[2. Архитектура базы данных 5](#_Toc59130588)

[2.1. Таблицы базы данных 5](#_Toc59130589)

[2.2. Процедуры для решения поставленных задач 6](#_Toc59130590)

[2.3. Функции для решения поставленных задач 8](#_Toc59130591)

[2.4. Индекс базы данных 8](#_Toc59130592)

[3. Реализация задач 9](#_Toc59130593)

[3.1. Регистрация и авторизация пользователей 9](#_Toc59130595)

[3.2. Операции над доской и задачами 9](#_Toc59130596)

[4. Импорт и экспорт данных. Тестирование производительности 13](#_Toc59130597)

[5. Описание технологии 14](#_Toc59130598)

[Заключение 15](#_Toc59130599)

[Список использованных источников 16](#_Toc59130600)

# Реферат

Курсовая работа изложена на 52 страницах, которые включают в себя 7 разделов, 16 подразделов, введение, архитектуру, реализацию, руководство пользователя, тестирование, заключение, используемую литературу и приложения. В работу добавлено 17 изображений.

В курсовой работе разработана модель базы данных, которая состоит из 8 таблиц. В базу данных включено 1 пакет с 24 процедурами.

Используемая СУБД –Microsoft SQL Management Studio 2017.

Изучаемая технология – Reporting services.

Основной целью курсовой работы является изучение и применение технологии создания отчетов.

## Аналитический обзор литературы

**Trello** — облачная программа, которая предназначена для управления проектами небольших групп(рис.1.1). За основу берется парадигма для управления проектами, известная как канбан, метод, который первоначально использовался Tayota в 1980-х годах для управления цепочками поставок.

В июле 2012 года количество посетителей сайта превысило отметку в 500 000. В декабре того же года количество пользователей возросло до 1 000 000.

Trello ограничил поддержку тегов в виде десяти цветных меток, которые можно переименовать. Карточки поддерживают комментарии, вложения, сроки выполнения и контрольные списки.

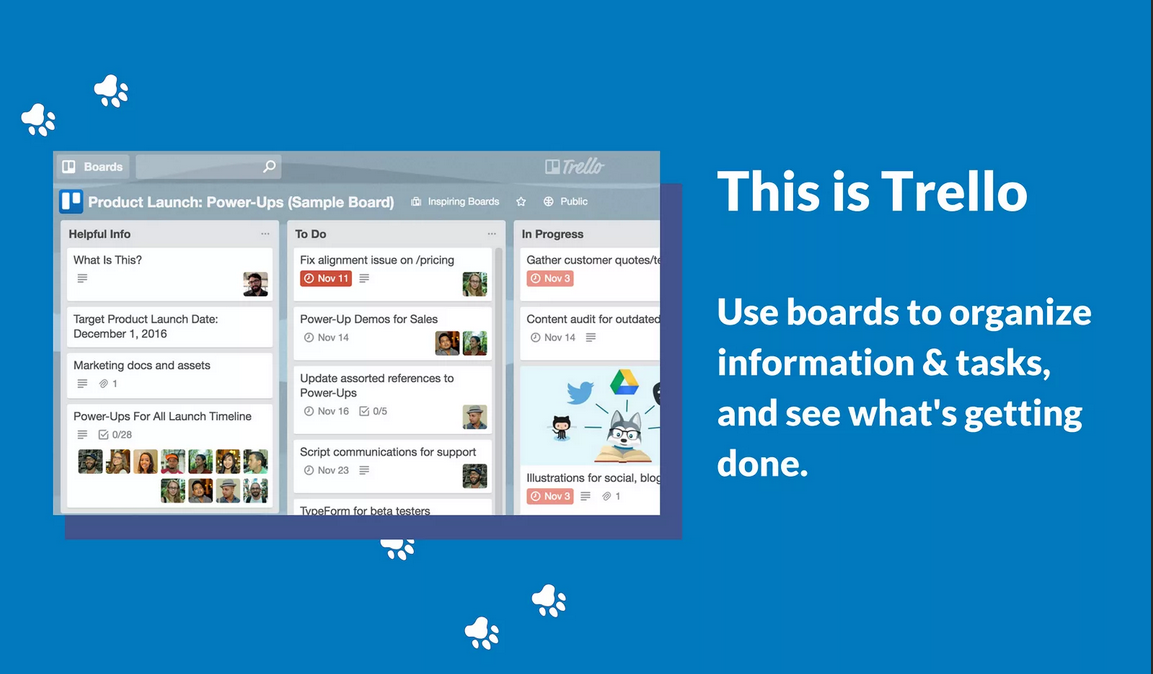


Рисунок 1.1 – Сервис Trello

Wrike – еще одна популярная система управления проектами. Wrike предоставляет много возможностей для управления задачами: копировать и повторять карточки, делать приватными или общедоступными, добавлять в «Избранное» и переносить с проектов – в «Мои задачи». Можно следить за выбранной задачей или предлагать коллегам следить. Добавлять зависимости и связи между задачами.

Одна из фишек Wrike – это отчеты, благодаря которым можно увидеть оценку рабочей нагрузки. В отчетах, доступных в платной версии, можно увидеть общее положение дел по проектам и командам, и они обновляются в реальном времени при каждом новом открытии(рис.1.2).

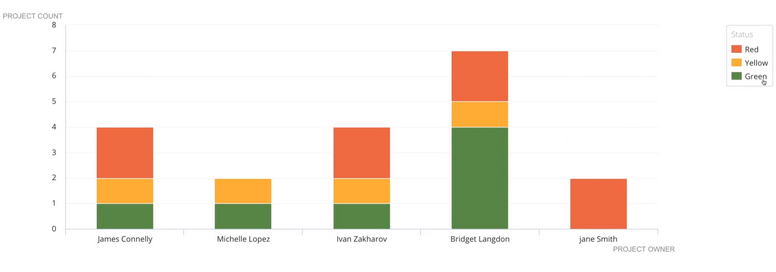


Рисунок 1.2 – Отчеты в Wrike и оценка рабочей нагрузки

## Архитектура базы данных

## Таблицы базы данных

Для хранения и предоставления данных пользователю, была разработана база данных, которая содержит в себе 8 таблиц, все связаны друг с другом внешними ключами. Диаграмма представлена на рисунке 2.1.

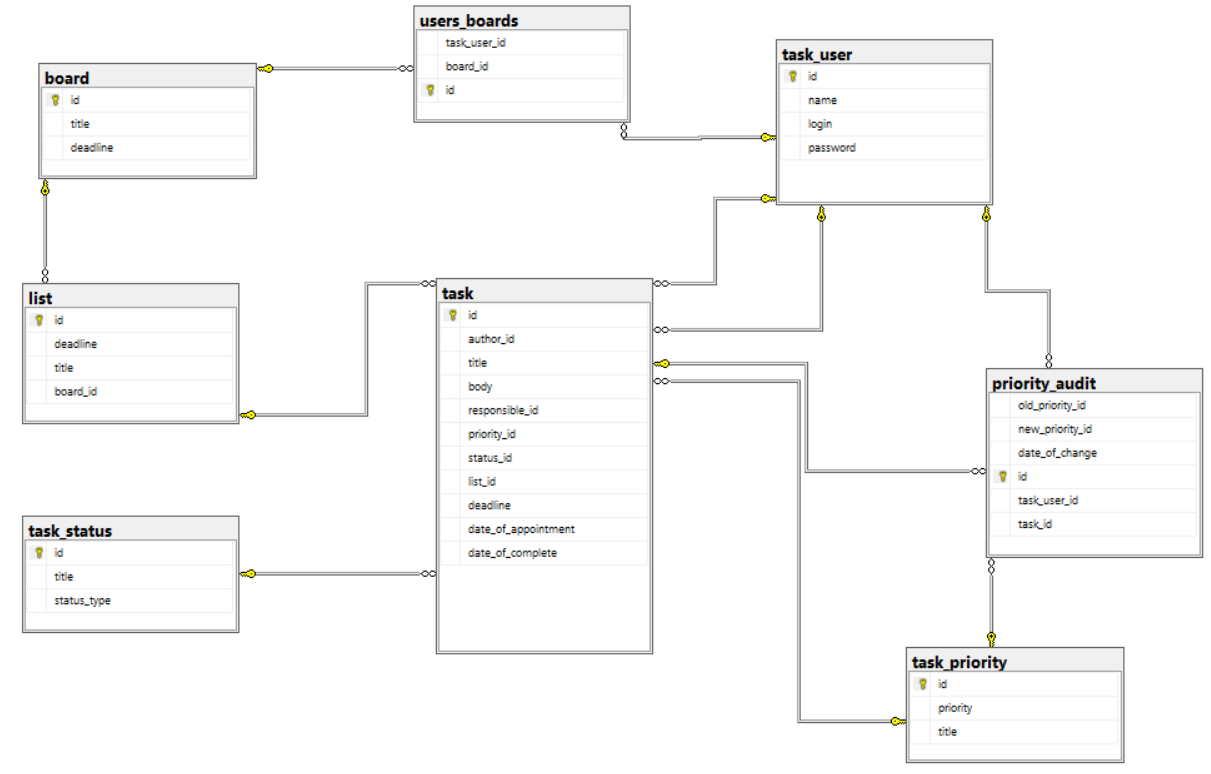


Рисунок 2.1 – Диаграмма базы данных

Таблица task\_user содержит информацию о пользователях для их авторизации, а именно: идентификатор (id, int, pk), логин (login, varchar(30), unique), пароль (password, varchar(30), not null), имя пользователя (varchar(30).

Таблица board содержит информацию о рабочем пространстве для проекта, а именно: идентификатор (id, int, pk), Название (title, varchar(30)), дедлайн(deadline, date, not null).

Таблица users\_boards является связующей между таблицами task\_user и board для реализации связи многие-ко-многим, а именно: идентификатор (id, int, pk), идентификатор пользователя (task\_user\_id, int, fk), идентификатор доски(board\_id, int, fk).

Таблица list содержит информацию о списках для классификации задач, которые хранятся на доске, а именно: идентификатор (id, int, pk), Название (title, varchar(30)), дедлайн(deadline, date, not null), идентификатор доски, к которой относится (id, int, fk).

Таблица task содержит информацию о задачах, а именно: идентификатор (id, int, pk), Название (title, varchar(30)), описание задачи(body, varchar(255)), идентификатор создателя задачи (author\_id, int, fk), идентификатор исполнителя задачи (responsible\_id, int, fk), дедлайн(deadline, date, not null), идентификатор списка, к которой относится (id, int, fk), идентификатор таблицы статусов (status\_id, int, fk), идентификатор таблицы приоритетов(priority\_id, int, fk), дата назначения задачи(date\_of\_appointment, date), дата выполнения задачи(date\_of\_complete, date).

Таблица task\_status содержит информацию о статусе задачи, а именно: идентификатор (id, int, pk), Название статуса(title, varchar(30)), соответствующий код статуса(status\_type, int).

Таблица task\_priority содержит информацию о приоритете задачи, а именно: идентификатор (id, int, pk),Символьное название приоритета(title, varchar(30)), соответствующий код приоритета(status\_type, int).

## Процедуры для решения поставленных задач

Процедуры храняться в нескольких файлах для удобного менеджмента.

Файл RegAuth содержит 2 процедуры:

1.reg\_user:

Назначение: регистрация пользователя

Параметры:

@name,

@login,

@password.

2.auth\_user:

Назначение: авторизация пользователя

Параметры:

@Login,

@password,

@resultId.

Файл CreateBoard содержит 9 процедур:

1.create\_board:

Назначение: создание доски

Параметры:

@user\_id,  
@title,

@deadline.

2.add\_user\_to\_board:

Назначение: добавление пользователя на доску

Параметры:

@user\_id  
@board\_id

3.add\_list\_to\_board:

Назначение: добавление списка на доску

Параметры:

@user\_id  
@board\_id

@title

@deadline

4.add\_task\_to\_list:

Назначение: добавление задачи к листу

Параметры:

@user\_id  
@board\_id

@title

@deadline

5.create\_task\_status:

Назначение: добавление новых статусов

Параметры:

@title  
@status\_type

6.set\_task\_status:

Назначение: установление статуса задаче

Параметры:

@user\_id  
@task\_id

@task\_status\_id

7.create\_task\_priority:

Назначение: добавление новых приоритетов

Параметры:

@title  
@priority

8.set\_task\_priority:

Назначение: установление статуса задаче

Параметры:

@user\_id  
@task\_id

@task\_priority\_id

9.set\_task\_deadline:

Назначение: установка сроков выполнения задаче

Параметры:

@user\_id  
@task\_id

@deadline

## Функции для решения поставленных задач

Функции для решения задач, поставленых при выполнении курсового проекта не требуются.

## Индекс базы данных

Индексы в SQL обеспечивают быстрый доступ к строкам таблиц, сохраняя отсортированные значения указанных столбцов и используя эти отсортированные значения для быстрого нахождения ассоциированных строк таблицы. Индексы позволяют находить строку с определенным значением столбца, просматривая при этом лишь небольшую часть общего объема строк таблицы. Таким образом правильное использование индексов сокращает до минимума количество дорогостоящих операций ввода-вывода.

В данном курсовом проекте индексы для таблиц не требуются.

## Реализация задач



## Регистрация и авторизация пользователей

Ниже приведены процедуры, реализующие регистрацию и авторизацию пользователей в базе данных.

GO

CREATE OR ALTER PROCEDURE Reg\_user

@name VARCHAR(30),

@login Varchar(30),

@password VARCHAR(30)

AS

BEGIN TRY

INSERT INTO task\_user (name,login,password) VALUES(@name,@login,@password)

END TRY

BEGIN CATCH

PRINT 'ЛОГИН УЖЕ ЗАНЯТ';

END CATCH

GO

CREATE OR ALTER PROCEDURE Auth\_user

@login Varchar(30),

@password VARCHAR(30),

@resultId int out

AS

DECLARE

@cou INT,

@pw VARCHAR(30);

BEGIN

SELECT login INTO cou FROM task\_user WHERE login = @login;

IF @cou!= 0

BEGIN

SELECT Password INTO pw FROM task\_user WHERE login = @login;

IF @pw = @password

SET @resultid=0;

Else

SET @resultId=-1;

END;

END;

## Операции над доской и задачами

Ниже приведены процедуры, реализующие операции над доской и задачами, а именно:

- Создание новую доску

- Добавление пользователя на доску

- Создание список на доске

- Создание новой задачи в списке

- Установление приоритета задаче

- Установление срока выполнения задачи

-- Создание новой доски

GO

CREATE OR ALTER PROCEDURE create\_board

@user\_id INT OUT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

INSERT INTO board (title, deadline) VALUES (@title, @deadline)

INSERT INTO users\_boards (board\_id, task\_user\_id) VALUES (SCOPE\_IDENTITY(), @user\_id)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Добавление пользователя на доску

GO

CREATE OR ALTER PROCEDURE add\_user\_to\_board

@user\_id INT OUT,

@board\_id INT OUT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ДОБАВЛЯЕМЫЙ ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM board WHERE id = @board\_id)

THROW 50000, 'ДОСКА НЕ СУЩЕСТВУЕТ', 1;

IF EXISTS ( SELECT task\_user\_id, board\_id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = @board\_id)

THROW 50000, 'ДОБАВЛЯЕМЫЙ ПОЛЬЗОВАТЕЛЬ УЖЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO users\_boards (board\_id, task\_user\_id) VALUES (@board\_id, @user\_id);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Добавление списка к доске

GO

CREATE OR ALTER PROCEDURE add\_list\_to\_board

@user\_id INT OUT,

@board\_id INT OUT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM board WHERE id = @board\_id)

THROW 50000, 'УКАЗАННАЯ ДОСКА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT task\_user\_id, board\_id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = @board\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO list (board\_id, title, deadline) VALUES (@board\_id, @title, @deadline);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Прикрепление задачи к доске

GO

CREATE OR ALTER PROCEDURE add\_task\_to\_list

@user\_id INT OUT,

@list\_id INT OUT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM list WHERE id = @list\_id)

THROW 50000, 'СПИСОК НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = @list\_id))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO task (author\_id, title, list\_id, date\_of\_appointment, deadline) VALUES (@user\_id, @title, @list\_id, GETDATE(), @deadline);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- ДОБАВЛЕНИЕ НОВЫХ СТАТУСОВ

GO

CREATE OR ALTER PROCEDURE create\_task\_status

@title VARCHAR(30),

@status\_type INT

AS

BEGIN TRY

IF EXISTS ( SELECT title, status\_type FROM task\_status WHERE title = @title AND status\_type = @status\_type)

THROW 50000, 'ТАКОЙ СТАТУС УЖЕ СУЩЕСТВУЕТ', 1;

INSERT INTO task\_status (title, status\_type) VALUES (@title, @status\_type)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА СТАТУСА ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_status

@user\_id INT OUT,

@task\_id INT OUT,

@task\_status\_id INT OUT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task\_status WHERE id = @task\_status\_id)

THROW 50000, 'СТАТУС НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET status\_id = @task\_status\_id WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- ДОБАВЛЕНИЕ НОВЫХ ПРИОРИТЕТОВ

GO

CREATE OR ALTER PROCEDURE create\_task\_priority

@title VARCHAR(30),

@priority INT

AS

BEGIN TRY

IF EXISTS ( SELECT priority, title FROM task\_priority WHERE priority = @priority AND title = @title)

THROW 50000, 'ТАКОЙ ПРИОРИТЕТ УЖЕ СУЩЕСТВУЕТ', 1;

INSERT INTO task\_priority(priority, title) VALUES (@priority, @title)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА ПРИОРИТЕТА ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_priority

@user\_id INT OUT,

@task\_id INT OUT,

@task\_priority\_id INT OUT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task\_priority WHERE id = @task\_priority\_id)

THROW 50000, 'ПРИОРИТЕТ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET priority\_id = @task\_priority\_id WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА СРОКОВ ВЫПОЛНЕНИЯ ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_deadline

@task\_id INT OUT,

@deadline DATE

AS

UPDATE task

SET deadline = @deadline

WHERE id = @task\_id

## Импорт и экспорт данных. Тестирование производительности

Для реализации процедур импорта и экспорта потребовалось создать объект директории, которая бы указывала путь для xml файла, в который будет производиться импорт и экспорт. В нашем случае имя директории – «bcp», путь – «C:\bcp\».

Процедура экспорта данных:

GO

CREATE OR ALTER PROCEDURE ExportTable

@name VARCHAR(30)

AS

BEGIN TRY

IF OBJECT\_ID(@name, 'U') IS NULL

THROW 50000, 'ТАБЛИЦА НЕ СУЩЕСТВУЕТ', 1;

DECLARE @path VARCHAR(30) = 'C:\BCP\'

--'bcp "SELECT \* FROM dbo.task\_user FOR XML PATH(''object''),ROOT(''objects'');" queryout C:\dvp\my.xml -c -T -S';

DECLARE @cmd VARCHAR(2000) = 'bcp "SELECT \* FROM dbo.' + @name +' FOR XML PATH(''ProductDescription''),ROOT(''Root''); " queryout "' + @path + @name + '.xml" -c -Cutf8 -T';

--PRINT @cmd

EXEC master..xp\_cmdshell @cmd;

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

Процедура импорта из xml файла:

CREATE OR ALTER PROCEDURE ImportTable\_task\_user

AS

BEGIN TRY

Select \* FROM task\_user -- показать, что сейчас

INSERT INTO task\_user (name, login, password)

SELECT

IMPORT\_XML.XmlCol.query('name').value('.', 'VARCHAR(30)'),

IMPORT\_XML.XmlCol.query('login').value('.', 'VARCHAR(30)'),

IMPORT\_XML.XmlCol.query('password').value('.', 'VARCHAR(30)')

FROM (SELECT CAST(IMPORT\_XML AS xml)

FROM OPENROWSET(BULK 'C:\BCP\task\_user.xml', SINGLE\_BLOB) AS T(IMPORT\_XML)) AS T(IMPORT\_XML)

CROSS APPLY IMPORT\_XML.nodes('Root/ProductDescription') AS IMPORT\_XML (XmlCol);

Select \* FROM task\_user -- показать, что экспортировалось

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

Тестирование производительности:

DECLARE @obj INT = OBJECT\_ID('TEST\_CRS\_PRJ.task\_user')

, @sql NVARCHAR(MAX)

, @cnt INT = 100000

;WITH

E1(N) AS (

SELECT \* FROM (

VALUES

(1),(1),(1),(1),(1),

(1),(1),(1),(1),(1)

) t(N)

),

E2(N) AS (SELECT 1 FROM E1 a, E1 b),

E4(N) AS (SELECT 1 FROM E2 a, E2 b),

E8(N) AS (SELECT 1 FROM E4 a, E4 b)

SELECT @sql = '

DELETE FROM ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(@obj))

+ '.' + QUOTENAME(OBJECT\_NAME(@obj)) + '

;WITH

E1(N) AS (

SELECT \* FROM (

VALUES

(1),(1),(1),(1),(1),

(1),(1),(1),(1),(1)

) t(N)

),

E2(N) AS (SELECT 1 FROM E1 a, E1 b),

E4(N) AS (SELECT 1 FROM E2 a, E2 b),

E8(N) AS (SELECT 1 FROM E4 a, E4 b)

INSERT INTO ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(@obj))

+ '.' + QUOTENAME(OBJECT\_NAME(@obj)) + '(' +

STUFF((

SELECT ', ' + QUOTENAME(name)

FROM sys.columns c

WHERE c.[object\_id] = @obj

AND c.is\_identity = 0

AND c.is\_computed = 0

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 2, '')

+ ')

SELECT TOP(' + CAST(@cnt AS VARCHAR(10)) + ') ' +

STUFF((

SELECT '

, ' + QUOTENAME(name) + ' = ' +

CASE

WHEN TYPE\_NAME(c.system\_type\_id) IN (

'varchar', 'char', 'nvarchar',

'nchar', 'ntext', 'text'

)

THEN (

STUFF((

SELECT TOP(

CASE WHEN max\_length = -1

THEN CAST(RAND() \* 10000 AS INT)

ELSE max\_length

END

/

CASE WHEN TYPE\_NAME(c.system\_type\_id) IN ('nvarchar', 'nchar', 'ntext')

THEN 2

ELSE 1

END

) '+SUBSTRING(x, (ABS(CHECKSUM(NEWID())) % 80) + 1, 1)'

FROM E8

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 1, '')

)

WHEN TYPE\_NAME(c.system\_type\_id) = 'tinyint'

THEN '50 + CRYPT\_GEN\_RANDOM(10) % 50'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('int', 'bigint', 'smallint')

THEN 'CRYPT\_GEN\_RANDOM(10) % 25000'

WHEN TYPE\_NAME(c.system\_type\_id) = 'uniqueidentifier'

THEN 'NEWID()'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('decimal', 'float', 'money', 'smallmoney')

THEN 'ABS(CAST(NEWID() AS BINARY(6)) % 1000) \* RAND()'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('datetime', 'smalldatetime', 'datetime2')

THEN 'DATEADD(MINUTE, RAND(CHECKSUM(NEWID()))

\*

(1 + DATEDIFF(MINUTE, ''20000101'', GETDATE())), ''20000101'')'

WHEN TYPE\_NAME(c.system\_type\_id) = 'bit'

THEN 'ABS(CHECKSUM(NEWID())) % 2'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('varbinary', 'image', 'binary')

THEN 'CRYPT\_GEN\_RANDOM(5)'

ELSE 'NULL'

END

FROM sys.columns c

WHERE c.[object\_id] = @obj

AND c.is\_identity = 0

AND c.is\_computed = 0

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 8, '

')

+ '

FROM E8

CROSS APPLY (

SELECT x = ''0123456789-ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz''

) t'

EXEC sys.sp\_executesql @sql

SELECT \* FROM task\_user

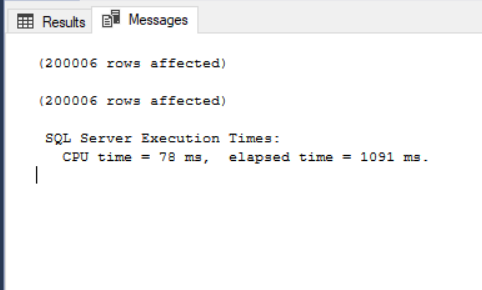
-- ЗАМЕР СКОРОСТИ ВЫПОЛНЕНИЯ ПРОЦЕДУРЫ

set statistics time on

SELECT \* FROM task\_user

set statistics time off

Результат замера скорости выполнения на 200 000 строк:



## Описание технологии

**SQL Server Reporting Services** — программная серверная система создания отчетов, разработанная корпорацией Microsoft. Она может быть использована для подготовки множества интерактивных и печатных отчетов.

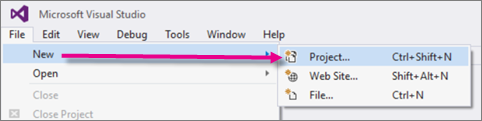
В SSRS отчеты описываются при помощи Report Definition Language (RDL) на языке разметки XML. Отчеты могут проектироваться при помощи последних версий Microsoft Visual Studio (включая Visual Studio.NET 2003 и Visual Studio 2005) с входящим в них дополнением Business Intelligence Projects или при помощи входящего в комплект Report Builder — упрощенного инструмента, не предлагающего полного функционала Visual Studio. Отчеты, определенные при помощи RDL, могут создаваться во множестве различных форматов [1], включая Excel, PDF, CSV, XML, TIFF (и других графических форматах[2]), а также HTML Web Archive. SQL Server 2008 SSRS также может подготавливать отчеты в формате Microsoft Word (DOC).

Пользователи могут работать с веб-службой Report Server напрямую или использовать Report Manager — веб-приложение, взаимодействующее с веб-службой Report Server. При помощи Report Manager могут просматривать и управлять отчетами, также как и управлять и оперировать источниками данных и настройками безопасности. Отчеты могут рассылаться по электронной почте или записываться на файловую систему как обычный файл. Защита выполняется на основе ролей и может накладываться на отдельные элементы, как например, отчет или источник данных, каталог элементов или сайт вообще. Роли безопасности и права являются наследуемыми и могут быть переопределены.

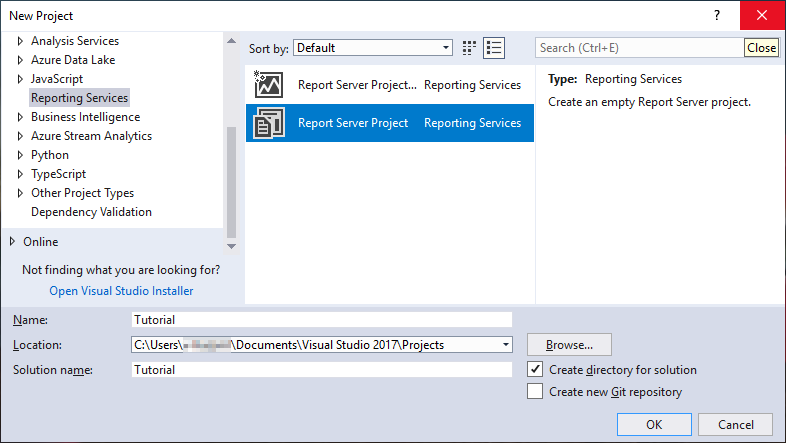
SQL reporting services также поддерживает обработку произвольных отчетов: разработчик создает схему отчета и развертывает ее на сервере отчетности, где пользователь может выбирать нужные поля/данные и создавать отчеты в соответствии со своими нуждами. Потом пользователи могут загружать отчеты локально.

## Создание проекта сервера отчетов

1. В меню **Файл** выберите пункт **Создать** > **Проект**.



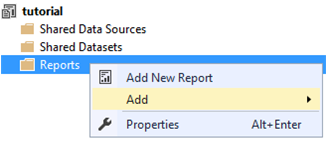
1. В крайнем левом столбце в разделе **Установленные** выберите **Службы Reporting Services**. В некоторых случаях этот пункт может оказаться в группе **Бизнес-аналитика**.



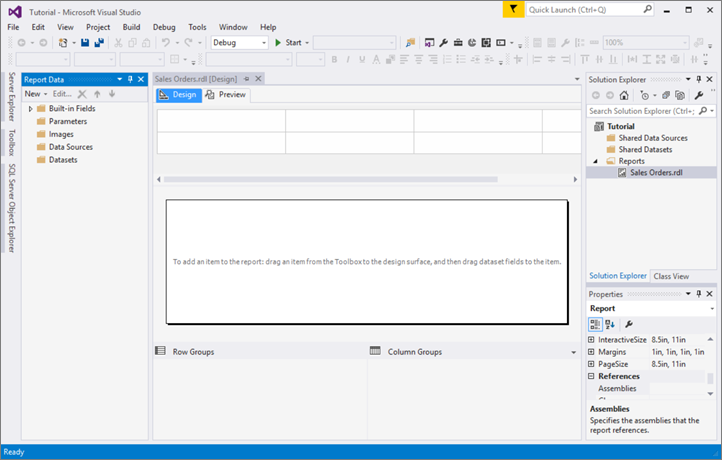
1. Выберите значок **Проект сервера отчетов**   ssrs_ssdt_report_server_project   в центральном столбце диалогового окна **Новый проект**.
2. В текстовом поле **Имя** введите имя проекта "Tutorial" (Обучение). По умолчанию текстовое поле **Расположение** отображает путь к папке Documents\Visual Studio 20xx\Projects". Конструктор отчетов создаст в этой папке вложенную папку Tutorial и разместит в ней проект Tutorial. Если этот проект не принадлежит решению VS, то VS дополнительно создаст файл решения (.sln).
3. Чтобы создать проект, щелкните **ОК**. Проект отобразится в области **обозревателя решений** справа.

## Создание файла определения отчета (RDL)

1. В области **обозревателя решений** щелкните правой кнопкой мыши папку **Reports** (Отчеты). Если область **обозревателя решений** не отображается, в меню **Вид** выберите пункт **Обозреватель решений**.
2. Щелкните **Добавить** > **Новый элемент**.



1. В окне **Добавление нового элемента** щелкните значок **Отчет**.
2. Введите в текстовое поле **Имя**.
3. Нажмите кнопку **Добавить** справа внизу в диалоговом окне **Добавление нового элемента** , чтобы завершить процесс. Конструктор отчетов откроет и отобразит в представлении конструктора новый файл отчета.



## Заключение

В ходе курсовой работы основные задачи были выполнены: была спроектирована база данных для системы управления задачами на базе СУБД Microsoft SQL Management Studio 17. Основные требования, которые были поставлены к работе, были выполнены:

* Был реализован ряд процедур и функций, благодаря которому в итоговой базе данных поддерживается весь необходимый функционал;
* Возможность авторизации/регистрации пользователей;
* Создание новой доски для работы
* Добавление пользователя на доску
* Прикрепление задач к доске
* Установление приоритета и сроков выполнения для задач
* Создание отчетов
* Разобрана технология для создания отчетов с помощью Microsoft reporting services

# Список использованных источников

[1] METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com Дата доступа: 10.12.2020

[2] stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com Дата доступа: 12.12.2020

[3] docs.microsoft.com [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ru-ru/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports?view=sql-server-ver15 Дата доступа: 12.12.2020

[4] habr.com [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/310328/ Дата доступа: 15.12.2020

# ПРИЛОЖЕНИЕ A

Создание базы данных, добавление внешних ключей

*-----------------------------------------------------------*DROP TABLE IF EXISTS priority\_audit;

DROP TABLE IF EXISTS task;

DROP TABLE IF EXISTS task\_priority;

DROP TABLE IF EXISTS task\_status;

DROP TABLE IF EXISTS list;

DROP TABLE IF EXISTS users\_boards;

DROP TABLE IF EXISTS board;

DROP TABLE IF EXISTS task\_user;

CREATE TABLE task\_user (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

name VARCHAR(30) NOT NULL,

login VARCHAR(30) UNIQUE NOT NULL,

password VARCHAR(30) NOT NULL);

CREATE TABLE board (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

title VARCHAR(30) NOT NULL,

deadline DATE);

CREATE TABLE users\_boards (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

board\_id INT NOT NULL,

task\_user\_id INT NOT NULL);

CREATE TABLE task (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

author\_id INT NOT NULL,

title VARCHAR(30) NOT NULL,

body VARCHAR(30),

responsible\_id INT,

priority\_id INT,

status\_id INT,

list\_id INT NOT NULL,

deadline DATE,

date\_of\_appointment DATE NOT NULL,

date\_of\_complete DATE);

CREATE TABLE priority\_audit (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

task\_id INT,

old\_priority\_id INT,

new\_priority\_id INT,

date\_of\_change DATE,

task\_user\_id INT,);

CREATE TABLE task\_status (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

title VARCHAR(30) NOT NULL,

status\_type INT NOT NULL);

CREATE TABLE task\_priority (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

priority INT NOT NULL,

title VARCHAR(30) NOT NULL);

CREATE TABLE list (

id INT PRIMARY KEY IDENTITY(1,1) NOT NULL,

board\_id INT NOT NULL,

title VARCHAR(30) NOT NULL,

deadline DATE);

ALTER TABLE users\_boards ADD CONSTRAINT users\_boards\_task\_user\_id FOREIGN KEY (task\_user\_id) REFERENCES task\_user(id);

ALTER TABLE users\_boards ADD CONSTRAINT users\_boards\_board\_id FOREIGN KEY (board\_id) REFERENCES board(id);

ALTER TABLE task ADD CONSTRAINT task\_author\_id FOREIGN KEY (author\_id) REFERENCES task\_user(id);

ALTER TABLE task ADD CONSTRAINT task\_responsible\_id FOREIGN KEY (responsible\_id) REFERENCES task\_user(id);

ALTER TABLE task ADD CONSTRAINT task\_priority\_id FOREIGN KEY (priority\_id) REFERENCES task\_priority(id);

ALTER TABLE task ADD CONSTRAINT task\_status\_id FOREIGN KEY (status\_id) REFERENCES task\_status(id);

ALTER TABLE task ADD CONSTRAINT task\_list\_id FOREIGN KEY (list\_id) REFERENCES list(id);

ALTER TABLE priority\_audit ADD CONSTRAINT priority\_audit\_old\_priority\_id FOREIGN KEY (old\_priority\_id) REFERENCES task\_priority(id);

ALTER TABLE priority\_audit ADD CONSTRAINT priority\_audit\_new\_priority\_id FOREIGN KEY (new\_priority\_id) REFERENCES task\_priority(id);

ALTER TABLE priority\_audit ADD CONSTRAINT priority\_audit\_task\_user\_id FOREIGN KEY (task\_user\_id) REFERENCES task\_user(id);

ALTER TABLE priority\_audit ADD CONSTRAINT priority\_audit\_task\_id FOREIGN KEY (task\_id) REFERENCES task(id);

ALTER TABLE list ADD CONSTRAINT list\_board\_id FOREIGN KEY (board\_id) REFERENCES board(id);

*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ B

Операции связаные с доской и задачами

*-----------------------------------------------------------*-- Создание новой доски

GO

CREATE OR ALTER PROCEDURE create\_board

@user\_id INT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

INSERT INTO board (title, deadline) VALUES (@title, @deadline)

INSERT INTO users\_boards (board\_id, task\_user\_id) VALUES (SCOPE\_IDENTITY(), @user\_id)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Удалить доску

/\*

GO

CREATE OR ALTER PROCEDURE delete\_board

@user\_id INT,

@board\_id INT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM board WHERE id = @board\_id)

THROW 50000, 'ДОСКА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT task\_user\_id, board\_id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = @board\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ЯВЛЯЕТСЯ СОЗДАТЕЛЕМ ДОСКИ', 1;

DELETE FROM users\_boards VALUES (@user\_id, @board\_id);

DELETE FROM users\_boards VALUES (@user\_id, @board\_id);

DELETE FROM users\_boards VALUES (@user\_id, @board\_id);

DELETE FROM users\_boards VALUES (@user\_id, @board\_id);

DELETE FROM users\_boards VALUES (@user\_id, @board\_id);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

\*/

--------------------------------------------------------------

-- Добавление пользователя на доску

---- Добавить на доску может только тот, кто уже привязан к доске

---- Пользователи должны существовать в базе данных

---- Доска должна существовать в базе данных

---- Нельзя добавить того, кто уже добавлен

GO

CREATE OR ALTER PROCEDURE add\_user\_to\_board

@user\_id INT,

@board\_id INT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ДОБАВЛЯЕМЫЙ ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM board WHERE id = @board\_id)

THROW 50000, 'ДОСКА НЕ СУЩЕСТВУЕТ', 1;

IF EXISTS ( SELECT task\_user\_id, board\_id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = @board\_id)

THROW 50000, 'ДОБАВЛЯЕМЫЙ ПОЛЬЗОВАТЕЛЬ УЖЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO users\_boards (board\_id, task\_user\_id) VALUES (@board\_id, @user\_id);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Добавление списка к доске

GO

CREATE OR ALTER PROCEDURE add\_list\_to\_board

@user\_id INT,

@board\_id INT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM board WHERE id = @board\_id)

THROW 50000, 'УКАЗАННАЯ ДОСКА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT task\_user\_id, board\_id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = @board\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO list (board\_id, title, deadline) VALUES (@board\_id, @title, @deadline);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- Прикрепление задачи к доске

GO

CREATE OR ALTER PROCEDURE add\_task\_to\_list

@user\_id INT,

@list\_id INT,

@title VARCHAR(30),

@deadline DATE = NULL

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM list WHERE id = @list\_id)

THROW 50000, 'СПИСОК НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = @list\_id))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

INSERT INTO task (author\_id, title, list\_id, date\_of\_appointment, deadline) VALUES (@user\_id, @title, @list\_id, GETDATE(), @deadline);

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- ДОБАВЛЕНИЕ НОВЫХ СТАТУСОВ

GO

CREATE OR ALTER PROCEDURE create\_task\_status

@title VARCHAR(30),

@status\_type INT

AS

BEGIN TRY

IF EXISTS ( SELECT title, status\_type FROM task\_status WHERE title = @title AND status\_type = @status\_type)

THROW 50000, 'ТАКОЙ СТАТУС УЖЕ СУЩЕСТВУЕТ', 1;

INSERT INTO task\_status (title, status\_type) VALUES (@title, @status\_type)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА СТАТУСА ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_status

@user\_id INT,

@task\_id INT,

@task\_status\_id INT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task\_status WHERE id = @task\_status\_id)

THROW 50000, 'СТАТУС НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET status\_id = @task\_status\_id WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- ДОБАВЛЕНИЕ НОВЫХ ПРИОРИТЕТОВ

GO

CREATE OR ALTER PROCEDURE create\_task\_priority

@title VARCHAR(30),

@priority INT

AS

BEGIN TRY

IF EXISTS ( SELECT priority, title FROM task\_priority WHERE priority = @priority AND title = @title)

THROW 50000, 'ТАКОЙ ПРИОРИТЕТ УЖЕ СУЩЕСТВУЕТ', 1;

INSERT INTO task\_priority(priority, title) VALUES (@priority, @title)

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА ПРИОРИТЕТА ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_priority

@user\_id INT,

@task\_id INT,

@task\_priority\_id INT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task\_priority WHERE id = @task\_priority\_id)

THROW 50000, 'ПРИОРИТЕТ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET priority\_id = @task\_priority\_id WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

--------------------------------------------------------------

-- УСТАНОВКА СРОКОВ ВЫПОЛНЕНИЯ ЗАДАЧЕ

GO

CREATE OR ALTER PROCEDURE set\_task\_deadline

@task\_id INT,

@deadline DATE

AS

UPDATE task SET deadline = @deadline WHERE id = @task\_id

--------------------------------------------------------------

GO

CREATE OR ALTER PROCEDURE set\_date\_of\_complete

@user\_id INT,

@task\_id INT,

@date\_of\_complete DATE

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET date\_of\_complete = @date\_of\_complete WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

GO

CREATE OR ALTER PROCEDURE set\_body\_in\_task

@user\_id INT,

@task\_id INT,

@body varchar(30)

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET body = @body WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH

----------------------------------------------------------------

GO

CREATE OR ALTER PROCEDURE set\_responsible

@user\_id INT,

@task\_id INT

AS

BEGIN TRY

IF NOT EXISTS (SELECT id FROM task\_user WHERE id = @user\_id)

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS (SELECT id FROM task WHERE id = @task\_id)

THROW 50000, 'ЗАДАЧА НЕ СУЩЕСТВУЕТ', 1;

IF NOT EXISTS ( SELECT id FROM users\_boards WHERE task\_user\_id = @user\_id AND board\_id = (SELECT id FROM list WHERE id = (SELECT list\_id FROM task WHERE id = @task\_id)))

THROW 50000, 'ПОЛЬЗОВАТЕЛЬ НЕ ПРИВЯЗАН К ДОСКЕ', 1;

UPDATE task SET responsible\_id = @user\_id WHERE id = @task\_id

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ C

Заполнение таблицы 100 000 записями,тестирование производительности

*-----------------------------------------------------------*DECLARE @obj INT = OBJECT\_ID('TEST\_CRS\_PRJ.task\_user')

, @sql NVARCHAR(MAX)

, @cnt INT = 100000

;WITH

E1(N) AS (

SELECT \* FROM (

VALUES

(1),(1),(1),(1),(1),

(1),(1),(1),(1),(1)

) t(N)

),

E2(N) AS (SELECT 1 FROM E1 a, E1 b),

E4(N) AS (SELECT 1 FROM E2 a, E2 b),

E8(N) AS (SELECT 1 FROM E4 a, E4 b)

SELECT @sql = '

DELETE FROM ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(@obj))

+ '.' + QUOTENAME(OBJECT\_NAME(@obj)) + '

;WITH

E1(N) AS (

SELECT \* FROM (

VALUES

(1),(1),(1),(1),(1),

(1),(1),(1),(1),(1)

) t(N)

),

E2(N) AS (SELECT 1 FROM E1 a, E1 b),

E4(N) AS (SELECT 1 FROM E2 a, E2 b),

E8(N) AS (SELECT 1 FROM E4 a, E4 b)

INSERT INTO ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(@obj))

+ '.' + QUOTENAME(OBJECT\_NAME(@obj)) + '(' +

STUFF((

SELECT ', ' + QUOTENAME(name)

FROM sys.columns c

WHERE c.[object\_id] = @obj

AND c.is\_identity = 0

AND c.is\_computed = 0

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 2, '')

+ ')

SELECT TOP(' + CAST(@cnt AS VARCHAR(10)) + ') ' +

STUFF((

SELECT '

, ' + QUOTENAME(name) + ' = ' +

CASE

WHEN TYPE\_NAME(c.system\_type\_id) IN (

'varchar', 'char', 'nvarchar',

'nchar', 'ntext', 'text'

)

THEN (

STUFF((

SELECT TOP(

CASE WHEN max\_length = -1

THEN CAST(RAND() \* 10000 AS INT)

ELSE max\_length

END

/

CASE WHEN TYPE\_NAME(c.system\_type\_id) IN ('nvarchar', 'nchar', 'ntext')

THEN 2

ELSE 1

END

) '+SUBSTRING(x, (ABS(CHECKSUM(NEWID())) % 80) + 1, 1)'

FROM E8

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 1, '')

)

WHEN TYPE\_NAME(c.system\_type\_id) = 'tinyint'

THEN '50 + CRYPT\_GEN\_RANDOM(10) % 50'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('int', 'bigint', 'smallint')

THEN 'CRYPT\_GEN\_RANDOM(10) % 25000'

WHEN TYPE\_NAME(c.system\_type\_id) = 'uniqueidentifier'

THEN 'NEWID()'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('decimal', 'float', 'money', 'smallmoney')

THEN 'ABS(CAST(NEWID() AS BINARY(6)) % 1000) \* RAND()'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('datetime', 'smalldatetime', 'datetime2')

THEN 'DATEADD(MINUTE, RAND(CHECKSUM(NEWID()))

\*

(1 + DATEDIFF(MINUTE, ''20000101'', GETDATE())), ''20000101'')'

WHEN TYPE\_NAME(c.system\_type\_id) = 'bit'

THEN 'ABS(CHECKSUM(NEWID())) % 2'

WHEN TYPE\_NAME(c.system\_type\_id) IN ('varbinary', 'image', 'binary')

THEN 'CRYPT\_GEN\_RANDOM(5)'

ELSE 'NULL'

END

FROM sys.columns c

WHERE c.[object\_id] = @obj

AND c.is\_identity = 0

AND c.is\_computed = 0

FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 8, '

')

+ '

FROM E8

CROSS APPLY (

SELECT x = ''0123456789-ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz''

) t'

EXEC sys.sp\_executesql @sql

SELECT \* FROM task\_user

-- ЗАМЕР СКОРОСТИ ВЫПОЛНЕНИЯ ПРОЦЕДУРЫ

set statistics time on

SELECT \* FROM task\_user

set statistics time off