ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ

Катедра „Софтуерно Инженерство “

A blue and white logo

Description automatically generated with medium confidence

**Курсова Работа по**

**АБТМУ**

Microsoft технологии за проектиране и администриране на разпределени бази от данни

**Тема:**

Система за следене на задачи

**Разработил:**

Ивайло Пламенов Руменов

Фак. № 23651227

Contents

[Entity диаграма 3](#_Toc169525189)

[Схема на РБД 4](#_Toc169525190)

[SQL команди 5](#_Toc169525191)

[Скрипт за вписване 8](#_Toc169525192)

[Скриптове за изтриване на табливите 10](#_Toc169525193)

[Програмен код на създаване обекти- индекси, изгледи, съхранени процедури функции и тригери 10](#_Toc169525194)

[Тригери 15](#_Toc169525195)

[Справки 16](#_Toc169525196)

# Entity диаграма

# Схема на РБД

# SQL команди

Скрипт за създаване на таблиците.

CREATE DATABASE TASK\_SYNC;

CREATE TABLE Status(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL

);

ALTER TABLE

Status ADD CONSTRAINT status\_id\_primary PRIMARY KEY(id);

CREATE TABLE Position(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL

);

ALTER TABLE

Position ADD CONSTRAINT position\_id\_primary PRIMARY KEY(id);

CREATE TABLE Task(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL,

work\_time BIGINT NOT NULL,

project BIGINT NOT NULL,

due DATE NOT NULL

);

ALTER TABLE

Task ADD CONSTRAINT task\_id\_primary PRIMARY KEY(id);

CREATE TABLE Project(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL,

manager BIGINT NOT NULL

);

ALTER TABLE

Project ADD CONSTRAINT project\_id\_primary PRIMARY KEY(id);

CREATE TABLE Sub\_task(

id\_master\_task BIGINT NULL,

id\_sub\_task BIGINT NOT NULL

);

CREATE TABLE Task\_history(

id BIGINT NOT NULL,

task BIGINT NOT NULL,

status BIGINT NOT NULL,

worker BIGINT NOT NULL,

manager BIGINT NOT NULL,

time\_stamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

history\_status BIGINT NOT NULL

);

ALTER TABLE

Task\_history ADD CONSTRAINT task\_history\_id\_primary PRIMARY KEY(id);

CREATE TABLE Department(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL

);

ALTER TABLE

Department ADD CONSTRAINT department\_id\_primary PRIMARY KEY(id);

CREATE TABLE Worker(

id BIGINT NOT NULL,

name VARCHAR(255) NOT NULL,

department BIGINT NOT NULL,

position BIGINT NOT NULL

);

CREATE TABLE Worker\_projects(

id\_worker BIGINT NULL,

id\_project BIGINT NOT NULL

);

ALTER TABLE

Worker ADD CONSTRAINT worker\_id\_primary PRIMARY KEY(id);

ALTER TABLE

Task\_history ADD CONSTRAINT task\_history\_worker\_foreign FOREIGN KEY(worker) REFERENCES Worker(id);

ALTER TABLE

Task\_history ADD CONSTRAINT task\_history\_manager\_foreign FOREIGN KEY(manager) REFERENCES Worker(id);

ALTER TABLE

Task\_history ADD CONSTRAINT task\_history\_status\_foreign FOREIGN KEY(status) REFERENCES Status(id);

ALTER TABLE

Project ADD CONSTRAINT project\_manager\_foreign FOREIGN KEY(manager) REFERENCES Worker(id);

ALTER TABLE

Sub\_task ADD CONSTRAINT sub\_task\_id\_sub\_task\_foreign FOREIGN KEY(id\_sub\_task) REFERENCES Task(id);

ALTER TABLE

Task ADD CONSTRAINT task\_project\_foreign FOREIGN KEY(project) REFERENCES Project(id);

ALTER TABLE

Worker ADD CONSTRAINT worker\_department\_foreign FOREIGN KEY(department) REFERENCES Department(id);

ALTER TABLE

Worker ADD CONSTRAINT worker\_position\_foreign FOREIGN KEY(position) REFERENCES Position(id);

ALTER TABLE

Sub\_task ADD CONSTRAINT sub\_task\_id\_master\_task\_foreign FOREIGN KEY(id\_master\_task) REFERENCES Task(id);

ALTER TABLE

Task\_history ADD CONSTRAINT task\_history\_task\_foreign FOREIGN KEY(task) REFERENCES Task(id);

ALTER TABLE

Worker\_projects ADD CONSTRAINT worker\_id\_projects FOREIGN KEY(id\_worker) REFERENCES Worker(id);

ALTER TABLE

Worker\_projects ADD CONSTRAINT project\_id\_workers FOREIGN KEY(id\_project) REFERENCES Project(id);

## Скрипт за вписване

**Процедура за вписване на нов отдел (Department)**

create procedure

createDepartment(@departmentId as INT, @departmentName as VARCHAR) as

begin

INSERT INTO Department(id, name)

VALUES (@departmentId, @departmentName);

end;

**Процедура за вписване на нова позиция**

create procedure

createPosition(@positionId as INT, @positionName as VARCHAR) as

begin

Insert Into Position(id, name)

VALUES (@positionId, @positionName);

end;

**Процедура за създаване на нов създател**

create procedure

createWorker(@workerId as INT, @workerName as VARCHAR, @department as INT, @position as INT) as

begin

INSERT INTO Worker(id, name, department, position)

VALUES (@workerId, @workerName, @department, @position);

end;

**Процедура за създаване на нов проект**

create procedure

createProject(@projectId as INT, @projectName as VARCHAR, @managerId as INT) as

begin

Insert Into Project(id, name, manager)

VALUES (@projectId, @projectName, @managerId);

end;

**Процедура за вписване на работник към проект**

create procedure

assignWorkerToProject(@workerId as INT, @projectId as INT) as

begin

Insert Into Worker\_projects(id\_worker, id\_project)

values (@workerId, @projectId);

end;

**Процедура за вписване на задача**

create procedure

createTask(@taskId as INT, @taskName as VARCHAR, @projectId as INT,

@taskDue as DATETIME) as

begin

INSERT INTO Task(id, name, work\_time, project, due)

VALUES (@taskId, @taskName, 0, @projectId, @taskDue);

end ;

**Процедура за вписване на историята на дадена задача при промяна**

create procedure

createTaskHistory(@histiD as INT, @taskId as INT, @taskStatus as INT,

@workerId as INT, @managerId as INT) as

begin

Insert Into Task\_history(id, task, status, worker, manager, history\_status)

VALUES (@histiD, @taskId, @taskStatus, @workerId, @managerId, 1);

end;

**Процедура за вписване на подзадача**

create procedure

createSubTask(@taskId as INT, @masterTaskId as INT, @taskName as VARCHAR, @projectId as INT,

@taskDue as DATETIME) as

begin

INSERT INTO Task(id, name, work\_time, project, due)

VALUES (@taskId, @taskName, 0, @projectId, @taskDue);

INSERT INTO Sub\_task(id\_master\_task, id\_sub\_task)

VALUES (@masterTaskId, @taskId);

end;

## Скриптове за изтриване на табливите

drop table Task\_history;

drop table Sub\_task;

drop table Status;

drop table Task;

drop table Worker\_projects;

drop table Project;

drop table Worker;

drop table Department;

drop table Position;

# Програмен код на създаване обекти- индекси, изгледи, съхранени процедури функции и тригери

Създаване на кластеризиран индекс служещ за по лесно изкарвайки задачите на работник по точен проект.

CREATE

UNIQUE

CLUSTERED INDEX IX\_MyIndexedView ON WrokerOnProjectWhitTasks (worker\_name,due);

-- Check if the view is schema-bound

SELECT OBJECTPROPERTY(OBJECT\_ID('dbo.WrokerOnProjectWhitTasks'), 'IsSchemaBound') AS IsSchemaBound;

create view WrokerOnProjectWhitTasks

as

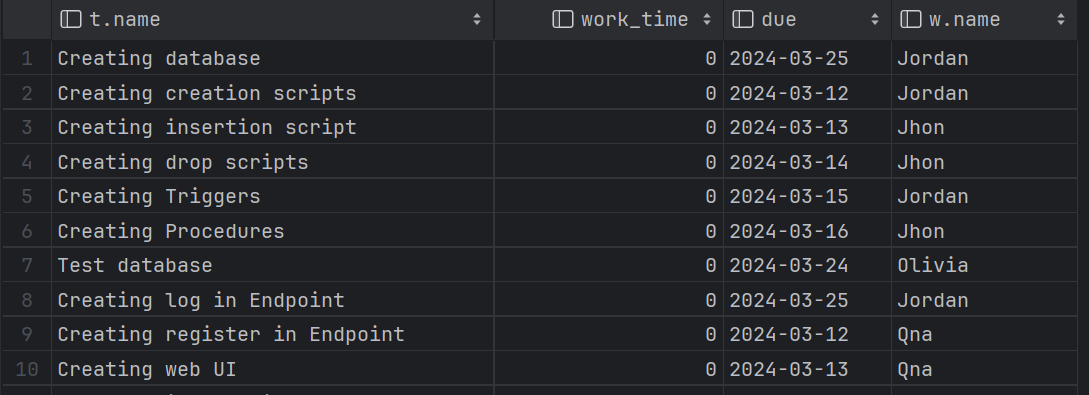
select t.name as tickate\_name, t.work\_time, t.due, w.name as worker\_name

from Task t

join Task\_history th on t.id = th.task

join Worker w on th.worker = w.id

join Worker\_projects wp on wp.id\_worker = w.id

Фиг.1: Резултати след извикване на извикване на изглед който съдуржа индексираните променливи.

Създаване на процедура за избиране на всички работници по идентификационен номер на проекта.

create procedure allWorkerInProject(@projectId as INT) as

begin

Select w.name as workerName, d.name as departmentName, p.name as possitionName, pj.name as projectName

from worker w

inner join Department d on d.id = w.department

inner join Position p on p.id = w.position

inner join Worker\_projects wp on wp.id\_worker = w.id

inner join Project pj on pj.id = wp.id\_project

where pj.id = @projectId

end;

Извикване на процедурата.

EXEC allWorkerInProject 1;

Скрипт за създаване на процедура за вземане на всички под задачи по идентификационен номер на главната задача.

create procedure getAllUnfinishedSubTasks(@masterTaskId as INT) as

begin

Select t.name as taskName, t.due as taskDueDate, w.name as workerName, s.name as taskStatus

from Sub\_task st

inner join Task t on st.id\_sub\_task = t.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where st.id\_master\_task = @masterTaskId

and th.status != 4

end;

Exec getAllUnfinishedSubTasks 1;

Създаване на процедура за проверка дали дадена задачи има не затворени подзадачи. Ако дадената задача има не завършени транзакцият.

create procedure hasUninishedSubTasks(@masterTaskId as INT) as

begin

Select count(t.id) as unfinishedTasks

from Sub\_task st

inner join Task t on st.id\_sub\_task = t.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where st.id\_master\_task = @masterTaskId

and th.status != 4

end;

exec hasUninishedSubTasks 2

create procedure getTaskStastDate(@taskId as INT)

as

begin

Select top 1 CONVERT(date, th.time\_stamp)

as taskCreationDate

from Task\_history th

where th.task = @taskId

order by th.time\_stamp asc;

end;

exec getTaskStastDate 1

Създаване на процедура за вземане на всички задачи по даден идентификационен номер на проект.

create procedure getAllTasksInaProject(@projectId as INT) as

begin

Select t.name as taskName, t.due as taskDueDate, w.name as workerName, s.name as taskStatus

from Project pj

inner join Task t on t.project = pj.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where pj.id = @projectId

and th.status != 4

end;

exec getAllTasksInaProject 1;

Процедура за връщане на транзакция при незавършени под задачи.

create procedure rollbackIfSubtaskIsNotFnished(@masterTaskId as INT)

as

begin

DECLARE

@subtasks DECIMAL

Select @subtasks = count(t.id)

from Sub\_task st

inner join Task t on st.id\_sub\_task = t.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where st.id\_master\_task = @masterTaskId

and th.status != 4

IF @subtasks > 0

BEGIN

RAISERROR

('All sub tasks must be finished before closing ticket.', 16, 1)

ROLLBACK TRANSACTION

END;

Създаване на процедура която връща всички задачи на всички работници в даден проект по идентификационен номер на проекта.

create procedure selectAllWorkerWhitTasks(@projectId as INT)

as

begin

select \*

from Task t

join Task\_history th on t.id = th.task

join Worker w on th.worker = w.id

join Worker\_projects wp on wp.id\_worker = w.id

where wp.id\_project = @projectId

end;

Функция която избира и връща специална таблица състояща се от работници по конкретен проект зададен индентифационен номер

CREATE FUNCTION WorkersOnProject(@workerName as VARCHAR(100))

RETURNS table

AS

return

(

select \*

from WrokerOnProjectWhitTasks wp

where wp.worker\_name = @workerName

)

drop function WorkersOnProject

select \*

from WorkersOnProject('Jhon')

select \*

from WrokerOnProjectWhitTasks wp

where wp.worker\_name = 'Jhon'

## Тригери

Тригер за актуализиране на наличното количество при добавяне на продажба

CREATE TRIGGER trg\_UpdateStockOnSale

ON Sold\_Products

AFTER INSERT

AS

BEGIN

DECLARE @stock\_id INT, @quantity INT;

SELECT @stock\_id = stock\_id, @quantity = quantity FROM inserted;

UPDATE Stocks

SET available\_quantity = available\_quantity - @quantity

WHERE stock\_id = @stock\_id;

END;

Тригер за проверка при изтриване на продажба

CREATE TRIGGER trg\_CheckStockBeforeDelete

ON Sold\_Products

BEFORE DELETE

AS

BEGIN

DECLARE @stock\_id INT, @quantity INT;

SELECT @stock\_id = stock\_id, @quantity = quantity FROM deleted;

UPDATE Stocks

SET available\_quantity = available\_quantity + @quantity

WHERE stock\_id = @stock\_id;

END;

# Справки

Създаване на процедура, която връща справка на съдържаща дана на създаване на задачата.

create procedure selectTaskComplitionData(@tastId as INT) as

begin

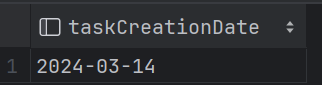
Select top 1 CONVERT(date, th.time\_stamp)

as taskCreationDate

from Task\_history th

where th.task = @tastId

order by th.time\_stamp asc;

end;

Създаване на процедура, която връща всички работници по проект.

create procedure getWorkersInProject(@projectId as INT) as

begin

Select w.name as workerName, d.name as departmentName, p.name as possitionName, pj.name as projectName

from worker w

inner join Department d on d.id = w.department

inner join Position p on p.id = w.position

inner join Worker\_projects wp on wp.id\_worker = w.id

A screenshot of a computer

Description automatically generated inner join Project pj on pj.id = wp.id\_project

where pj.id = projectId

Процедура която връща бройката на всички незавършени под задачи по зададена задача.

create procedure get unfinishedTasks (@taskId as INT) as

begin

Select count(t.id) as unfinishedTasks

from Sub\_task st

inner join Task t on st.id\_sub\_task = t.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where st.id\_master\_task = @taskId

and th.status != 4

A black background with white text

Description automatically generatedend;

Създаване на процедура която връза детейли за всички на завършени под задачи на дадена задача.

create procedure getAllUnfinishedTasks (@taskId as INT) as

begin

Select t.name as taskName, t.due as taskDueDate, w.name as workerName, s.name as taskStatus

from Sub\_task st

inner join Task t on st.id\_sub\_task = t.id

inner join Task\_history th on th.task = t.id

left join Worker w on w.id = th.worker

inner join Status s on s.id = th.status

where st.id\_master\_task = @taskId

and th.status != 4;

A screenshot of a computer

Description automatically generatedend;