ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА
ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ
Катедра „Софтуерно Инженерство “

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

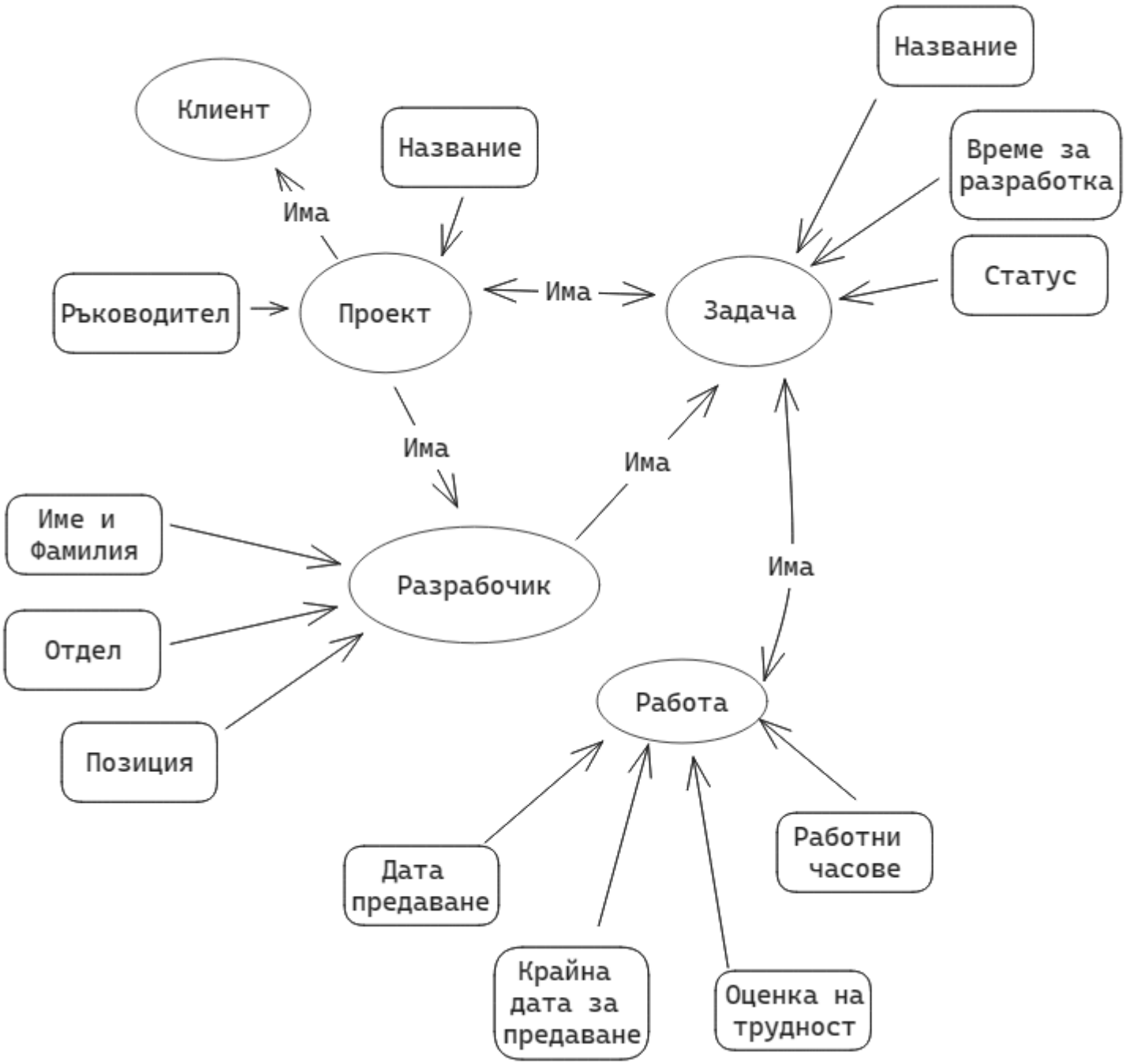
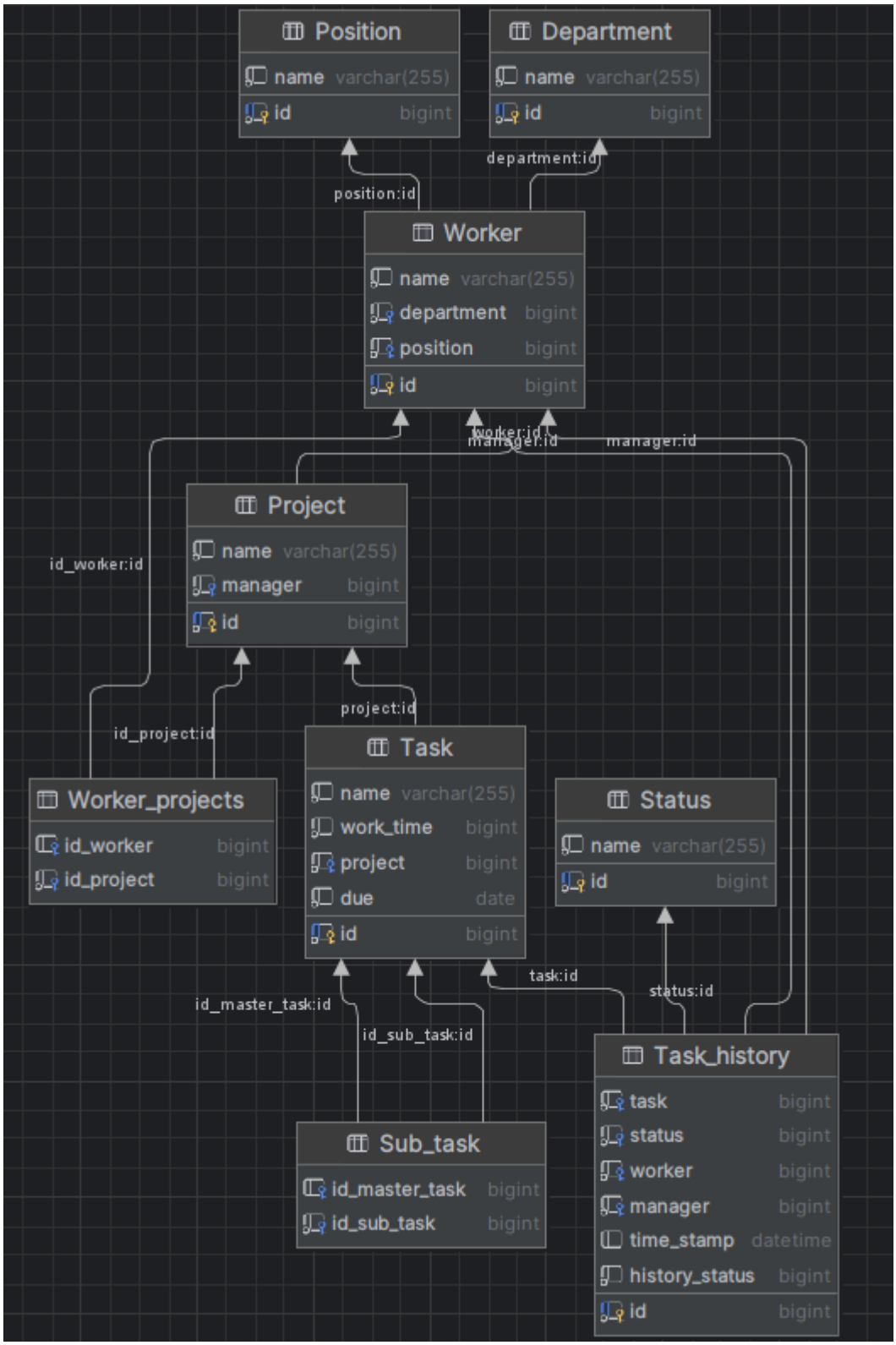


Схема на РБД


Position
name varchar(255)
id bigint
Department
name varchar(255)
id bigint
position:id
department:id
Worker
name varchar(255)
department bigint
position bigint
id bigint
worker:id
manager:id
manager:id
Project
name varchar(255)
manager bigint
id bigint
id_worker:id
id_project:id
project:id
Worker_projects
id_worker bigint
id_project bigint
Task
name varchar(255)
work_time bigint
project bigint
due date
id bigint
Status
name varchar(255)
id bigint
task:id
status:id
id_master_task:id
id_sub_task:id
Sub_task
id_master_task bigint
id_sub_task bigint
Task_history
task bigint
status bigint
worker bigint
manager bigint
time_stamp datetime
history_status bigint
id bigint

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
```

	t.name	work_time	due	w.name
1	Creating database	0	2024-03-25	Jordan
2	Creating creation scripts	0	2024-03-12	Jordan
3	Creating insertion script	0	2024-03-13	Jhon
4	Creating drop scripts	0	2024-03-14	Jhon
5	Creating Triggers	0	2024-03-15	Jordan
6	Creating Procedures	0	2024-03-16	Jhon
7	Test database	0	2024-03-24	Olivia
8	Creating log in Endpoint	0	2024-03-25	Jordan
9	Creating register in Endpoint	0	2024-03-12	Qna
10	Creating web UI	0	2024-03-13	Qna

Фиг.1: Резултати след извикване на извикване на изглед който съдържа 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;
```

	taskCreationDate
1	2024-03-14

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
    inner join Project pj on pj.id = wp.id_project
where pj.id = projectId
```

	workerName	departmentName	possitionName	projectName
1	Ivan	Development	Project Manager	Bulgerian Trains
2	Iliq	Development	Manager	Bulgerian Trains
3	Koko	Development	Manager	Bulgerian Trains
4	Plamen	Development	Manager	Bulgerian Trains
5	Qna	Development	Programmer	Bulgerian Trains
6	Jordan	Development	Programmer	Bulgerian Trains
7	Jhon	Development	Programmer	Bulgerian Trains
8	Jhoana	Development	QA	Bulgerian Trains
9	Emma	Development	QA	Bulgerian Trains
10	Luna	Development	Tester	Bulgerian Trains
11	Olivia	Development	Tester	Bulgerian Trains

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
end;
```

	unfinishedTasks
1	6

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;
end;
```

	taskName	taskDueDate	workerName	taskStatus
1	Creating creation scripts	2024-03-12	Jordan	Undefined
2	Creating insertion script	2024-03-13	Jhon	Undefined
3	Creating drop scripts	2024-03-14	Jhon	Undefined
4	Creating Triggers	2024-03-15	Jordan	Undefined
5	Creating Procedures	2024-03-16	Jhon	Undefined
6	Test database	2024-03-24	Olivia	Undefined