УПРАВЛЕНИЕ НА ПРОЕКТИ

База от данни и Системи за управление на бази от данни (практикуми)

Дата: Януари, 2025
Изготвил: Петя Личева
Факултетен номер: 3MI0700022

Съдържание

Обхват на модела. Дефиниране на задачата

- Базата от данни за управление на проекти ще съхранява информация за задачите, свързани с даден проект, както и за техните изпълнители.
- Базата от данни ще съхранява информация за различни компании. Компанията ще се определя еднозначно от име, адрес и описание на дейността.
- Във всяка компания ще има различни екипи (минимум 1 и максимум 10).
- Всеки екип ще се определя еднозначно от име, описание на дейността на екипа.
- Във всеки екип ще има минимум 1 служител (наследник на същности от потребителите), като всеки екип ще работи по различни проекти, а по всеки проект ще могат да работят много екипи.
- Всеки потребител на системата се определя еднозначно от уникален номер, имейл, потребителско име (до 32 символа) и парола (до 32 символа).
  Потребителите ще са два вида: служители и любители:
  - **Служители:** телефонен номер, адрес, заплата, екип.
  - **Любители:** описание на дейността, с която се занимават любителски.
- Всеки проект се определя еднозначно от име, версия, описание, дата на пускане в употреба.
  Проектът се състои от много задачи, като всяка задача е част от точно един проект.
- Всяка задача се определя еднозначно от уникален номер, начална дата, статус (завършена или блокирана) и текстово описание.
  За някои задачи е дефиниран краен срок (който не може да е преди датата на създаване на задачата).
  Съхранява се информация за това кой потребител и на коя дата е дефинирал задачата.
  Бъговете са специални задачи, за които задължително се съхранява сценарий за възпроизвеждането им, както и номер на версията на приложението, в която са били забелязани за първи път.
  Задачите биват: изпълними задачи или бъгове.
  - **Изпълними задачи:** определени от предполагаема крайна дата.
  - **Бъгове:** определени от сценарий за възпроизвеждането им и номер на версията.
  - **Блокирани задачи:** определени от дата на блокиране и блокираща задача.
    Един потребител може да работи по няколко задачи, а по една задача могат да работят няколко потребителя.
    Дадена задача може да бъде блокирана от други задачи (необходимо е те да бъдат завършени, за да се работи по нея). Една задача може да блокира няколко други задачи.

Множества от същности и техните атрибути

1. Компании: Име на компанията, Адрес на компанията и Описание на дейността
2. Екипи: Име на компанията, Име на екипа и Описание на дейността
3. Потребители: Уникален номер, Имейл, Потребителско име и Парола
4. Служители: Уникален номер, Телефонен номер, Адрес, Заплата и Екип
5. Любители: Уникален номер и Описание на дейността
6. Проекти: Име, Версия, Описание и Дата на пускане
7. Задачи: Уникален номер, Начална дата и Описание
8. Изпълними задачи: Предполагаема крайна дата
9. Бъгове: Сценарий за възпроизвеждането им и Номер на версията

Връзки

- **Потребители – Служители:** ISA връзка.
- **Потребители – Любители:** ISA връзка.
- **Потребители – Задачи:** Един потребител дефинира много задачи; една задача е дефинирана от точно един потребител.
- **Компании – Екипи:** Една компания има между 1 и 10 екипа; един екип участва в точно една компания.
- **Екипи – Служители:** Един екип има много служители; един служител работи в точно един екип.
- **Проекти – Задачи:** Един проект се състои от много задачи; една задача е част от точно един проект.
- **Екипи – Проекти:** Един екип работи по много проекти; по един проект работят много екипи.
- **Любители – Задачи:** Един любител работи по много задачи; една задача се изпълнява от много любители.
- **Задачи – Изпълними задачи:** ISA връзка.
- **Задачи – Бъгове:** ISA връзка.
- **Бъгове – Проекти:** Един бъг е забелязан в точно една версия на проект; една версия съдържа много бъгове.

Ключове

- Компании: (име на компанията, адрес на компанията)
- Екипи: (име на компанията, име на екипа)
- Потребители: уникален номер
- Проекти: (име на проекта, версия на проекта)
- Задачи: уникален номер

Правила и проверки

Компании

- Име на компанията: до 20 символа, не може да е NULL.
- Адрес: до 100 символа, задължително български стандарт, не може да е NULL.
- Описание: до 450 символа, може да е NULL.

Екипи

- Име: до 20 символа, не може да е NULL.
- Описание: до 300 символа, може да е NULL.

Потребители

- Уникален номер: цяло число, не може да е NULL.
- Потребителско име: до 32 символа, задължително започва с буква, не може да е NULL.
- Имейл: валиден имейл адрес до 100 символа, не може да е NULL.
- Парола: до 32 символа, задължително съдържа букви, цифри и специални символи (_ . ! # @), не може да е NULL.

Проекти

- Име: до 30 символа, не може да е NULL.

- Версия: до 20 символа, не може да е NULL.
- Дата на пускане: след 01.01.2023, не може да е NULL.

Задачи

- Уникален номер: цяло число, не може да е NULL.
- Начална дата: след 01.01.2023, може да е NULL.
- Описание: до 300 символа, може да е NULL.

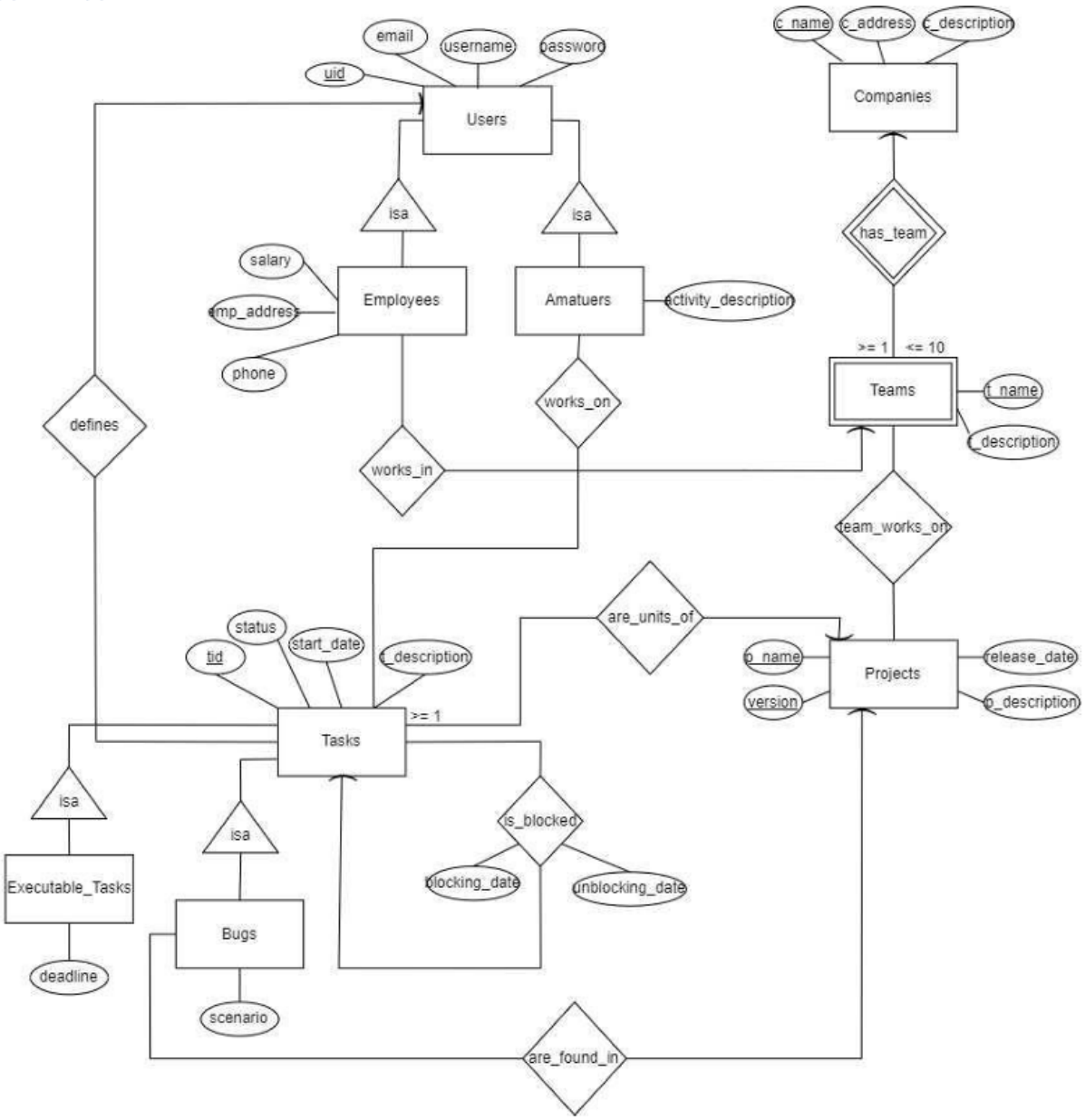
Изпълними задачи

- Предполагаема крайна дата: поне месец след началната дата, може да е NULL.

Бъгове

- Сценарий: до 600 символа, не може да е NULL.
- Номер на версия: до 20 символа, не може да е NULL.

E/R модел на данни



Таблици

ISA йерархия: Потребители

Релационен подход

- **Users** (uid: int (> 0), email: string (320), username: string (10), password: string (64))
- **Employees** (uid: int (> 0), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13))
- **Amateurs** (uid: int (> 0), activity_description: string (350))

ООП подход

- **Users** (uid: int (> 0), email: string (320), username: string (10), password: string (64))
  - PK: uid (unique, not null)
- **Users_Employees** (uid: int (> 0), email: string (320), username: string (10), password: string (64), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13), team_id: int (> 0))
  - PK: uid (unique, not null)
  - FK: Users(team_id) -> Teams(tid)
- **Users_Amateurs** (uid: int (> 0), email: string (320), username: string (10), password: string (64), activity_description: string (350))
  - PK: uid (unique, not null)

NULL подход

- **Users** (uid: int (> 0), email: string (320), username: string (10), password: string (64), team_id: int (> 0), salary: double (> 0.0), emp_address: string (100), phone: string (>= 10 && <= 13), activity_description: string (350))

ISA йерархия: Задачи

Релационен подход

- **Tasks** (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), uid: int (> 0), project_id: int (> 0), btid: int (> 0))
  - PK: tid (unique, not null)
  - FK: Tasks(uid) -> Users(uid)
  - FK: Tasks(project_id) -> Projects(pid)
  - FK: Tasks(btid) -> Tasks(tid)

- **Executable_Tasks** (tid: int (> 0), deadline: date (>= 01.01.2023))
  - PK: tid (unique, not null)
- **Bugs_Tasks** (tid: int (> 0), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))
  - PK: tid (unique, not null)
  - FK: Bugs_Tasks(project_id) -> Projects(pid)

ООП подход

- **Tasks** (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350))
- **Executables** (tid: int (> 0), deadline: date (>= 01.01.2023))
- **Bugs** (tid: int (> 0), scenario: string (600), project_id: int (> 0))
- **Executable_Tasks** (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), deadline: date (>= start_date))
- **Bugs_Tasks** (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

NULL подход

- **Tasks** (tid: int (> 0), status: string (10 – unfinished), start_date: date (>= 01.01.2023), tdescription: string (350), deadline: date (>= start_date), scenario: string (600), project_id: int (> 0), blocking_date: date (>= 01.01.2023), unblocking_date: date (>= blocking_date))

Таблици на останалите множества от същности и взаимоотношенията им

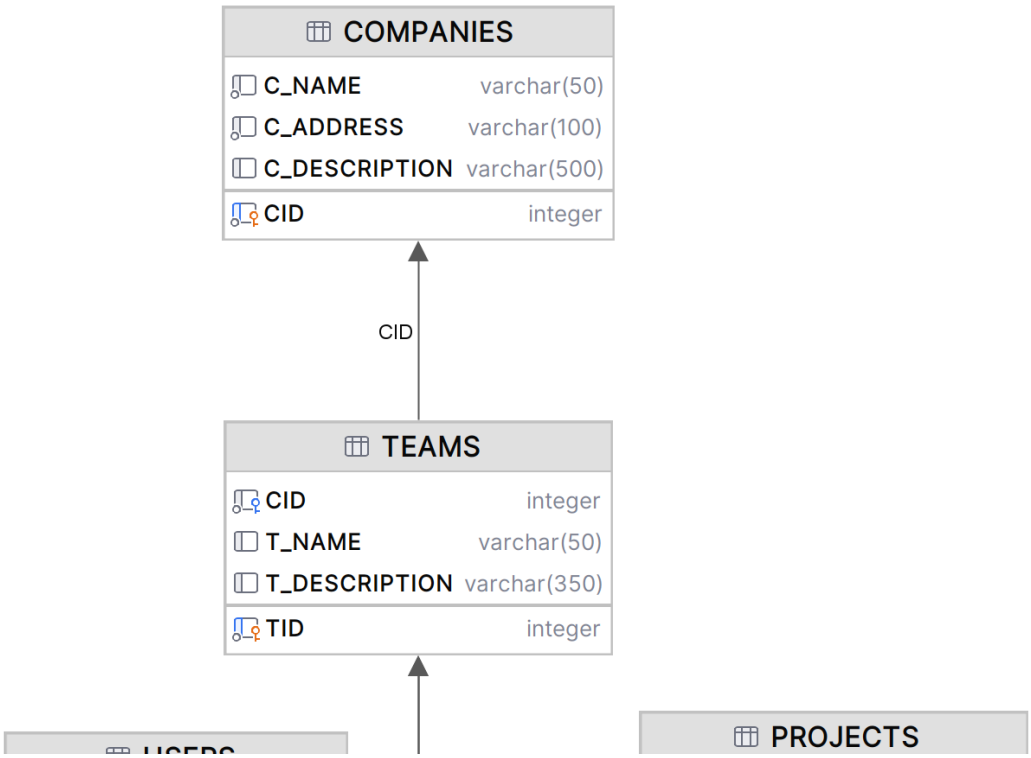
- **Companies** (cid: int (> 0), cname: string (50), caddress: string (100), cdescription: string (500))
  - PK: cid (unique, not null)
- **Teams** (team_id: int (> 0), cid: int (> 0), tname: string (50), tdescription: string (350))
  - PK: team_id (unique, not null)
  - FK: Teams(cid) -> Companies(cid)
- **Projects** (project_id: int (> 0), pname: string (50), version: string (20), release_date: date (>= 01.01.2023), pdescription: string (350))
  - PK: project_id (unique, not null)
- **WorksOn** (uid: int (> 0), tid: int (> 0))
  - PK: uid (unique, not null), tid (unique, not null)
  - FK: WorksOn(uid) -> Users(uid)
  - FK: WorksOn(tid) -> Tasks(tid)
- **TeamsWorkOn** (team_id: int (> 0), project_id: int (> 0))
  - PK: team_id (not null), project_id (unique, not null)
  - FK: TeamsWorkOn(team_id) -> Teams(team_id)

- FK: `TeamsWorkOn(project_id) -> Projects(project_id)`

Ограничения

- **Users**
  - Email: `_@_.`
  - Username: `_%`
  - Password: `_%`
- **Users_Employees**
  - Email: `_@_.`
  - Username: `_%`
  - Password: `_%`
  - Salary: `> 0.0`
- **Users_Amateurs**
  - Email: `_@_.`
  - Username: `_%`
  - Password: `_%`
- **Tasks**
  - Status: `finished/unfinished`
  - Start date: `>= 01.01.2023`
- **Executable_Tasks**
  - Deadline: `>= start_date`
- **Bugs**
  - Blocking date: `>= 01.01.2023`
  - Unblocking date: `>= blocking_date`
- **Projects**
  - Release date: `>= 01.01.2023`

DB - Schema:




USERS
EMAIL varchar(320)
USERNAME varchar(10)
PASSWORD character(64)
UID integer
P_NAME varchar(50)
P_VERSION varchar(20)
RELEASE_DATE date
P_DESCRIPTION varchar(350)
PID integer
DEFINED_BY_UID:UID
PROJECT_ID:PID
TEAM_ID:TID
UID
UID
UID
TEAM_ID:TID
P_ID:PID
USERS_EMPLOYEES
EMAIL varchar(320)
USERNAME varchar(10)
PASSWORD character(64)
SALARY double
EMP_ADDRESS varchar(100)
PHONE character(13)
TEAM_ID integer
UID integer
TEAMSWORKON
TEAM_ID integer
P_ID integer
USERS_AMATEURS
EMAIL varchar(320)
USERNAME varchar(10)
PASSWORD character(64)
ACTIVITY_DESCRIPTION varchar(350)
UID integer
TASKS
STATUS character(10)
START_DATE date
T_DESCRIPTION varchar(350)
DEFINED_BY_UID integer
PROJECT_ID integer
TID integer
TID
TID
TID
EXECUTABLES
DEADLINE date
TID integer
DELETED_PROJECTS_LOG
P_NAME varchar(50)
P_VERSION varchar(20)
RELEASE_DATE date
P_DESCRIPTION varchar(350)
PID integer
BUGS
SCENARIO varchar(600)
PROJECT_ID integer
BLOCKING_DATE date
UNBLOCKING_DATE date
TID integer
WORKSON
UID integer
TID integer

Функции

Първа функция:

```
SET SCHEMA FN3MI0700022;

-- A function that returns all bugs in a project full data as a table.
-- This function can be used to make an audit of the bugs in a definite
-- project by its ID.
CREATE OR REPLACE FUNCTION F_GET_ALL_BUGS_IN_PROJECT(PID INT)
    RETURNS TABLE
            (
                TID             INT,
                SCENARIO        VARCHAR(600),
                PROJECT_ID      INT,
                BLOCKING_DATE   DATE,
                UNBLOCKING_DATE DATE
            )
    LANGUAGE SQL
    RETURN SELECT B.TID, B.SCENARIO, B.PROJECT_ID, B.BLOCKING_DATE,
B.UNBLOCKING_DATE
            FROM FN3MI0700022.BUGS B
            WHERE B.PROJECT_ID = PID;

-- execution of the function
GRANT EXECUTE ON FUNCTION FN3MI0700022.F_GET_ALL_BUGS_IN_PROJECT TO PUBLIC;
SELECT * FROM TABLE(FN3MI0700022.F_GET_ALL_BUGS_IN_PROJECT(10));
```

- Първата функция, която съм имплементирала, връща пълна информацияза бъговете в даден проекткато таблица.

Втора функция:

```
-- A function that returns all teams in a definite company tasks. This function
can be
-- used for audit of all tasks that a definite company works on.
CREATE OR REPLACE FUNCTION F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS(P_CID INT)
    RETURNS TABLE
            (
                TEAM_ID          INT,
                TEAM_NAME        VARCHAR(50),
                TASK_STATUS      CHAR(10),
                TASK_START_DATE  DATE,
                TASK_DESCRIPTION VARCHAR(350),
                DEFINED_BY_USER  VARCHAR(10),
                PROJECT_NAME     VARCHAR(50),
                PROJECT_VERSION  VARCHAR(20)
            )
    RETURN
        SELECT T.TID,
               T.T_NAME,
               TSK.STATUS,
```

```
                TSK.START_DATE,
                TSK.T_DESCRIPTION,
                U.USERNAME,
                P.P_NAME,
                P.P_VERSION
        FROM FN3MI0700022.TEAMS T
                 LEFT JOIN FN3MI0700022.TEAMSWORKON TW
                           ON T.TID = TW.TEAM_ID
                 LEFT JOIN FN3MI0700022.PROJECTS P
                           ON TW.P_ID = P.PID
                 LEFT JOIN FN3MI0700022.TASKS TSK
                           ON TSK.PROJECT_ID = P.PID
                 LEFT JOIN FN3MI0700022.USERS U
                           ON TSK.DEFINED_BY_UID = U.UID
        WHERE TW.P_ID = P_CID;

-- execution of the function
GRANT EXECUTE ON FUNCTION FN3MI0700022.F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS TO
PUBLIC;
SELECT * FROM TABLE(FN3MI0700022.F_GET_ALL_TEAMS_IN_A_COMPANY_TASKS(10));
```

- Втората функция, която съм имплментирала, връща пълна информация за задачите, покоито работят екипите в дадена компаниякато таблица.

Тригери

Първи тригер:

```
SET SCHEMA FN3MI0700022;

-- A trigger that validates the user data before insert.
CREATE OR REPLACE TRIGGER TRIG_BEFORE_USER_INSERT
    BEFORE INSERT ON FN3MI0700022.USERS
    REFERENCING NEW AS N
    FOR EACH ROW
BEGIN
    DECLARE V_IS_VALID_DATA BOOLEAN;

    -- Call the validation procedure
    CALL FN3MI0700022.USER_MOD.P_VALIDATE_USER_DATA(N.EMAIL, N.USERNAME,
                                                    V_IS_VALID_DATA);

    -- Signal an error if data is invalid
    IF V_IS_VALID_DATA = FALSE THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
            'Validation failed: Email or Username is not unique.';
    END IF;
END;

-- trigger execution - signal error that the username or email is not unique
INSERT INTO FN3MI0700022.USERS(EMAIL, USERNAME, PASSWORD)
```

```
VALUES ('dummyaddress@outlook.com', 'speedy423', '123456');

-- trigger execution - inserts successfully a user
INSERT INTO FN3MI0700022.USERS(EMAIL, USERNAME, PASSWORD)
VALUES ('test@outlook.com', 'test123', '123456');
```

- Първият тригер, който съм имплементирала, валидира данните на потребител, който се опитва да се регистрира преди реално да го направи.

Втори тригер:

```
-- A trigger that validates the user date before update.
CREATE OR REPLACE TRIGGER TRIG_BEFORE_USER_UPDATE
    BEFORE UPDATE ON FN3MI0700022.USERS
    REFERENCING OLD AS O NEW AS N
    FOR EACH ROW
BEGIN
    DECLARE V_IS_VALID_DATA BOOLEAN;

    SET V_IS_VALID_DATA = O.USERNAME != N.USERNAME OR O.EMAIL != N.EMAIL;

    -- Signal an error if data is invalid
    IF V_IS_VALID_DATA = FALSE THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
            'Validation failed: Email or Username is not unique.';
    END IF;

    -- Optionally, check for specific fields being updated
    IF O.EMAIL = N.EMAIL AND O.USERNAME = N.USERNAME AND O.PASSWORD = N.PASSWORD
THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No changes detected in the
update operation.';
    END IF;
END;

-- trigger execution - successfully updates the user with a definite username
UPDATE FN3MI0700022.USERS U SET U.EMAIL = 'newtest@gmail.com' WHERE U.USERNAME =
'test123';
```

- Вторият тригер, който съм имплментирала, валидира данните на потребител, който се опитва да си обнови профила преди реално да го направи.

Трети тригер:

```
-- A trigger that logs deletions from Projects table
-- (after a definite project is deleted).
CREATE TABLE DELETED_PROJECTS_LOG (
    PID INTEGER GENERATED ALWAYS AS IDENTITY
    CONSTRAINT PK_DEL_PROJECTS
```

```
    PRIMARY KEY,
    P_NAME VARCHAR(50),
    P_VERSION VARCHAR(20),
    RELEASE_DATE DATE DEFAULT CURRENT_DATE,
    P_DESCRIPTION VARCHAR(350),
    CHECK (P_VERSION LIKE '%_'),
    CHECK (RELEASE_DATE >= DATE('2023-01-01'))
);

-- A trigger that logs data for the deleted project after the project deletion.
CREATE OR REPLACE TRIGGER TRIG_LOG_AFTER_PROJECT_DELETION
    AFTER DELETE ON FN3MI0700022.PROJECTS
    REFERENCING OLD AS O
    FOR EACH ROW
BEGIN
    INSERT INTO FN3MI0700022.DELETED_PROJECTS_LOG(P_NAME, P_VERSION,
P_DESCRIPTION)
        VALUES (O.P_NAME, O.P_VERSION, O.P_DESCRIPTION);
END;

-- deletes all connections to the project before executing the next line
DELETE FROM FN3MI0700022.EXECUTABLES E WHERE E.TID = 7;
DELETE FROM FN3MI0700022.BUGS B WHERE B.PROJECT_ID = 1;
DELETE FROM FN3MI0700022.WORKSON W WHERE W.TID = 7;
DELETE FROM FN3MI0700022.TASKS T WHERE T.TID = 7;
DELETE FROM FN3MI0700022.TEAMSWORKON TW WHERE TW.P_ID = 1;

-- then executes the project deletion line and after the deletion in the
-- FN3MI0700022.DELETED_PROJECTS_LOG table will appear a log for this
-- deleted project
DELETE FROM FN3MI0700022.PROJECTS P WHERE P.PID = 1;
```

- Третият тригер, който съм имплементирала, записва текущо изтрит проект в таблицата DELETED_PROJECTS_LOG с цел одитна вече изтритипроекти.

Изгледи

Employees view

```
-- set my schema as default schema
SET SCHEMA FN3MI0700022;

-- Employees view: shows information about the employees in dependence of
-- in which team and in which company he or she works.
CREATE VIEW "Employees" AS
SELECT FN3MI0700022.USERS_EMPLOYEES.EMAIL,
       FN3MI0700022.USERS_EMPLOYEES.USERNAME,
       FN3MI0700022.USERS_EMPLOYEES.EMP_ADDRESS,
       FN3MI0700022.USERS_EMPLOYEES.PHONE,
       FN3MI0700022.USERS_EMPLOYEES.SALARY,
       FN3MI0700022.TEAMS.T_NAME,
```

```
       FN3MI0700022.COMPANIES.C_NAME
FROM FN3MI0700022.USERS_EMPLOYEES, FN3MI0700022.TEAMS, FN3MI0700022.COMPANIES
WHERE FN3MI0700022.USERS_EMPLOYEES.TEAM_ID = FN3MI0700022.TEAMS.TID AND
      FN3MI0700022.TEAMS.CID = FN3MI0700022.COMPANIES.CID;

-- executes the Employees view query
SELECT * FROM "Employees" WHERE C_NAME = 'Nexus Dynamics';
```

Tasksview

```
-- Tasks view: Shows information for all tasks that deadline is after current date
-- in other word shows information about all unfinished tasks.
CREATE VIEW "Tasks" AS
SELECT * FROM FN3MI0700022.EXECUTABLES
WHERE DEADLINE > CURRENT_DATE;

-- executes the Tasks view query
SELECT * FROM "Tasks";
```

Bugs view

```
-- Bugs view: Shows information about all bugs that still blocks any task.
CREATE VIEW "Bugs" AS
SELECT * FROM "Bugs"
WHERE FN3MI0700022."Bugs".UNBLOCKING_DATE > FN3MI0700022."Bugs".BLOCKING_DATE;

-- executes the Bugs view query
SELECT * FROM "Bugs";

-- inserts new record in Bugs table
INSERT INTO Bugs(TID, SCENARIO, PROJECT_ID, BLOCKING_DATE, UNBLOCKING_DATE)
VALUES(7, 'new bug scenario 2025', 10,
       '2024-04-22', '2025-01-05');

-- re-executes the Bugs view query
SELECT * FROM "Bugs";
```

Процедури

Първа процедура (с прихващане на грешки):

```
SET SCHEMA FN3MI0700022;

CREATE OR REPLACE MODULE USER_MOD;

-- A procedure that validates user data. This procedure can be used in the
```

```
triggers
-- that will validate the user data before insert and before update.
ALTER MODULE USER_MOD PUBLISH PROCEDURE P_VALIDATE_USER_DATA(IN U_EMAIL
VARCHAR(320),
                                                             IN U_USERNAME
VARCHAR(10),
                                                             OUT V_IS_VALID
BOOLEAN);
ALTER MODULE USER_MOD ADD PROCEDURE P_VALIDATE_USER_DATA(IN U_EMAIL VARCHAR(320),
                                                         IN U_USERNAME
VARCHAR(10),
                                                         OUT V_IS_VALID BOOLEAN)
BEGIN
    DECLARE V_UID INT;
    DECLARE V_IS_VALID BOOLEAN;
    SET V_UID = (SELECT U.UID FROM FN3MI0700022.USERS U
                              WHERE U.USERNAME = U_USERNAME OR U.EMAIL = U_EMAIL);

    IF V_UID IS NOT NULL
        THEN
            SET V_IS_VALID = FALSE;
            SIGNAL SQLSTATE '70001'
            SET MESSAGE_TEXT = 'Username or email is not unique!';
    ELSE
        SET V_IS_VALID = TRUE;
    END IF;
END;

-- this procedure is tested in the before insert trigger for the
FN3MI0700022.USERS table
```

- Първата процедура, която съм имплементирала, валидира данните на даден потребител.

Втора процедура (с курсор и while цикъл):

```
CREATE OR REPLACE MODULE COMPANIES_MOD;

-- A procedure that will return information about all companies in which amateurs
-- and experienced employees work, the count of amateurs and the count of
experienced
-- employees. This information can be used for statistical researches.
ALTER MODULE COMPANIES_MOD PUBLISH PROCEDURE
P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO();
ALTER MODULE COMPANIES_MOD ADD PROCEDURE
P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO()
BEGIN
    DECLARE V_C_ID INT;
    DECLARE V_C_NAME VARCHAR(50);
    DECLARE V_C_ADDRESS VARCHAR(100);
    DECLARE V_C_DESCRIPTION VARCHAR(500);
    DECLARE V_AMATEURS_EMPLOYEES_CNT INT;
```

```
    DECLARE V_EXPERIENCED_EMPLOYEES_CNT INT;

    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';

    DECLARE CURSOR_COMPANIES CURSOR FOR (
        SELECT C.*,
               AMATEURS.AMATEURS_EMPLOYEES_COUNT,
               EXPERIENCED.EXPERIENCED_EMPLOYEES_COUNT
        FROM FN3MI0700022.COMPANIES C
                 LEFT JOIN FN3MI0700022.TEAMS T ON C.CID = T.CID
                 LEFT JOIN (
            SELECT T2.TID, COUNT(*) AS AMATEURS_EMPLOYEES_COUNT
            FROM FN3MI0700022.USERS_AMATEURS UA
                     LEFT JOIN FN3MI0700022.WORKSON ET ON UA.UID = ET.UID
                     LEFT JOIN FN3MI0700022.TEAMS T2 ON ET.TID = T2.TID
            GROUP BY T2.TID
        ) AS AMATEURS ON T.TID = AMATEURS.TID
                 LEFT JOIN (
            SELECT UE.TEAM_ID, COUNT(*) AS EXPERIENCED_EMPLOYEES_COUNT
            FROM FN3MI0700022.USERS_EMPLOYEES UE
            GROUP BY UE.TEAM_ID
        ) AS EXPERIENCED ON T.TID = EXPERIENCED.TEAM_ID
    );

    CALL DBMS_OUTPUT.PUT_LINE('CID, C_NAME, C_ADDRESS, C_DESCRIPTION,
AMATEURS_EMPLOYEES_COUNT, ' ||
                              'EXPERIENCED_EMPLOYEES_COUNT');

    OPEN CURSOR_COMPANIES;
    FETCH CURSOR_COMPANIES INTO V_C_ID, V_C_NAME, V_C_ADDRESS, V_C_DESCRIPTION,
        V_AMATEURS_EMPLOYEES_CNT, V_EXPERIENCED_EMPLOYEES_CNT;
    WHILE SQLSTATE = '00000'
        DO
            CALL DBMS_OUTPUT.PUT_LINE(V_C_ID || ', ' || V_C_NAME || ', ' ||
V_C_ADDRESS || ', '
                || V_C_DESCRIPTION || ', ' || V_AMATEURS_EMPLOYEES_CNT || ', '
                                           || V_EXPERIENCED_EMPLOYEES_CNT);
            FETCH CURSOR_COMPANIES INTO V_C_ID, V_C_NAME, V_C_ADDRESS,
V_C_DESCRIPTION,
                V_AMATEURS_EMPLOYEES_CNT, V_EXPERIENCED_EMPLOYEES_CNT;
        END WHILE;
    CLOSE CURSOR_COMPANIES;
END;

-- executes the procedure
CALL
FN3MI0700022.COMPANIES_MOD.P_GET_ALL_COMPANIES_IN_WHICH_AMATEURS_WORKS_INFO();
```

- Втората процедура, която съм имплементирала, връща таблица, в която се съдържа информацияза всички компании, в които работят както начинаещи, така и служителисъс дългогодишен опит, както и броят на едната група и броят на другата група служители.

Трета процедура (с курсор, входни и изходни данни):

```
-- A procedure that returns the history of a definite employee on the base of his
/ her
-- username. This procedure can be used for employees audit.
ALTER MODULE USER_MOD PUBLISH PROCEDURE
P_GET_EMPLOYEE_HISTORY_BY_EMPLOYEE_USERNAME(
    IN P_UNM VARCHAR(10), OUT P_CURSOR_EMPLOYEE_HISTORY CURSOR);
ALTER MODULE USER_MOD ADD PROCEDURE P_GET_EMPLOYEE_HISTORY_BY_EMPLOYEE_USERNAME(
    IN P_UNM VARCHAR(10), OUT P_CURSOR_EMPLOYEE_HISTORY CURSOR)
BEGIN
    DECLARE P_CURSOR_EMPLOYEE_HISTORY CURSOR WITH RETURN FOR
        SELECT * FROM FN3MI0700022.USERS_EMPLOYEES E WHERE E.USERNAME = P_UNM;

    OPEN P_CURSOR_EMPLOYEE_HISTORY;
END;
```

- Третата процедура, която съм имплементирала, връща таблица, съдържаща информация за историята нададен служител по неговия псевдоним (уникално поле в таблиците USERS, USERS_AMATEURS и EMPLOYEES_USERS).

Приложение за достъп до базата от данни

main() method

- стартовата точка наприложението

printMenu() method

- принтира менюто и въвежда потребителя в базата данни като му дава опции, от които да си избере каква заявка иска да изпълни върху таблицата Employees

execute()

- извиква и изпълнява конкретен метод,който се грижи за даден тип заявки

Конзола с резултатните множества от заявките

SELECT

- показва всички записи в таблицата Employees с всичките им атрибути и съответните им стойности

INSERT INTO

- добавя запис в таблицата Employees като потребителят трябва ръчно да въведе стойност в конзолата на приложението за всяко необходимо поле

DELETE

- изтрива запис за служител от таблицата Employees по подадено име на служителя,ако такъв съществува

EXIT

- излиза от приложението и терминира работата на конзолата