Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет**

по Лабораторной работе № 2

Студент: Пузиков А.А.

ФИТ 3 курс 6 группа

Вариант 10

Преподаватель: Нистюк О.А.

Минск 2024

**Реализация базы данных.**

**Список таблиц с указанием атрибутов для Oracle:**

1. Таблица Users:

* UserID: NUMBER, PRIMARY KEY, идентификатор пользователя
* Name: NVARCHAR2(100), имя пользователя
* Email: NVARCHAR2(100), электронная почта пользователя
* RegistrationDate: DATE, дата регистрации пользователя

1. Таблица Posts:

* PostID: NUMBER, PRIMARY KEY, идентификатор поста
* UserID: NUMBER, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который создал пост
* PostText: NVARCHAR2(400), текст поста
* PublicationDate: DATE, дата публикации поста

1. Таблица Friends:

* UserID: NUMBER, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя
* FriendID: NUMBER, FOREIGN KEY REFERENCES Users(UserID), идентификатор друга пользователя
* PRIMARY KEY (UserID, FriendID), пара UserID и FriendID является первичным ключом

1. Таблица Likes:

* UserID: NUMBER, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который поставил лайк
* PostID: NUMBER, FOREIGN KEY REFERENCES Posts(PostID), идентификатор поста, которому поставлен лайк
* PRIMARY KEY (UserID, PostID), пара UserID и PostID является первичным ключом

1. Таблица Comments:

* CommentID: NUMBER, PRIMARY KEY, идентификатор комментария
* PostID: NUMBER, FOREIGN KEY REFERENCES Posts(PostID), идентификатор поста, к которому добавлен комментарий
* UserID: NUMBER, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который оставил комментарий
* CommentText: NVARCHAR2(400), текст комментария
* PublicationDate: DATE, дата публикации комментария

**Cписок таблиц с указанием атрибутов для SQL Server:**

1. Таблица Users:

* UserID: INT, PRIMARY KEY, идентификатор пользователя
* Name: NVARCHAR(100), имя пользователя
* Email: NVARCHAR(100), электронная почта пользователя
* RegistrationDate: DATETIME, дата регистрации пользователя

1. Таблица Posts:

* PostID: INT, PRIMARY KEY, идентификатор поста
* UserID: INT, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который создал пост
* PostText: NVARCHAR(MAX), текст поста
* PublicationDate: DATETIME, дата публикации поста

1. Таблица Friends:

* UserID: INT, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя
* FriendID: INT, FOREIGN KEY REFERENCES Users(UserID), идентификатор друга пользователя
* PRIMARY KEY (UserID, FriendID), пара UserID и FriendID является первичным ключом

1. Таблица Likes:

* UserID: INT, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который поставил лайк
* PostID: INT, FOREIGN KEY REFERENCES Posts(PostID), идентификатор поста, которому поставлен лайк
* PRIMARY KEY (UserID, PostID), пара UserID и PostID является первичным ключом

1. Таблица Comments:

* CommentID: INT, PRIMARY KEY, идентификатор комментария
* PostID: INT, FOREIGN KEY REFERENCES Posts(PostID), идентификатор поста, к которому добавлен комментарий
* UserID: INT, FOREIGN KEY REFERENCES Users(UserID), идентификатор пользователя, который оставил комментарий
* CommentText: NVARCHAR(MAX), текст комментария
* PublicationDate: DATETIME, дата публикации комментария  
    
  **Cписок дополнительных объектов в базе данных Oracle:**

1. Последовательности:

* seq\_users: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Users.
* seq\_posts: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Posts.
* seq\_comments: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Comments.

1. Триггеры:

* trg\_users: Триггер, который реагирует на событие вставки в таблицу Users, берет значение из последовательности seq\_users и вставляет его вместо поля UserID.
* trg\_posts: Триггер, который реагирует на событие вставки в таблицу Posts, берет значение из последовательности seq\_posts и вставляет его вместо поля PostID.
* trg\_comments: Триггер, который реагирует на событие вставки в таблицу Comments, берет значение из последовательности seq\_comments и вставляет его вместо поля CommentID.

1. Зависимые объекты:

* Триггер trg\_users зависит от таблицы Users и последовательности seq\_users.
* Триггер trg\_posts зависит от таблицы Posts и последовательности seq\_posts.
* Триггер trg\_comments зависит от таблицы Comments и последовательности seq\_comments.

1. Процедуры:

* add\_user: Процедура, которая добавляет нового пользователя в таблицу Users. Она принимает три параметра: имя пользователя (p\_name), электронную почту пользователя (p\_email) и дату регистрации (p\_date). Зависимый объект - таблица Users.
* add\_post: Процедура, которая добавляет новый пост в таблицу Posts. Она принимает три параметра: идентификатор пользователя (p\_userID), текст поста (p\_postText) и дату публикации (p\_date). Зависимые объекты - таблицы Users и Posts.
* add\_comment: Процедура, которая добавляет новый комментарий в таблицу Comments. Она принимает пять параметров: идентификатор поста (p\_postID), идентификатор пользователя (p\_userID), текст комментария (p\_commentText) и дату публикации (p\_date). Зависимые объекты - таблицы Users, Posts и Comments.

1. Функции

* add\_friend: Функция, которая добавляет новую запись в таблицу Friends. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор друга (@p\_friendID). Зависимый объект - таблица Friends.
* add\_like: Функция, которая добавляет новую запись в таблицу Likes. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор поста (@p\_postID). Зависимые объекты - таблицы Users, Posts и Likes.
* remove\_like: Функция, которая удаляет запись из таблицы Likes. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор поста (@p\_postID). Зависимые объекты - таблицы Users, Posts и Likes.

**Cписок дополнительных объектов в базе данных SQL Server:**

1. Последовательности:

* seq\_users: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Users.
* seq\_posts: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Posts.
* seq\_comments: Последовательность, используемая для генерации уникальных идентификаторов для таблицы Comments.

1. Триггеры:

* trg\_users: Триггер, который реагирует на событие вставки в таблицу Users, берет значение из последовательности seq\_users и вставляет его вместо поля UserID.
* trg\_posts: Триггер, который реагирует на событие вставки в таблицу Posts, берет значение из последовательности seq\_posts и вставляет его вместо поля PostID.
* trg\_comments: Триггер, который реагирует на событие вставки в таблицу Comments, берет значение из последовательности seq\_comments и вставляет его вместо поля CommentID.

1. Зависимые объекты:

* Триггер trg\_users зависит от таблицы Users и последовательности seq\_users.
* Триггер trg\_posts зависит от таблицы Posts и последовательности seq\_posts.
* Триггер trg\_comments зависит от таблицы Comments и последовательности seq\_comments.

1. Процедуры:

* , add\_user: Процедура, которая добавляет нового пользователя в таблицу Users. Она принимает три параметра: имя пользователя (@p\_name), электронную почту пользователя (@p\_email) и дату регистрации (@p\_date). Зависимый объект - таблица Users.
* add\_post: Процедура, которая добавляет новый пост в таблицу Posts. Она принимает три параметра: идентификатор пользователя (@p\_userID), текст поста (@p\_postText) и дату публикации (@p\_date). Зависимые объекты - таблицы Users и Posts.
* add\_comment: Процедура, которая добавляет новый комментарий в таблицу Comments. Она принимает пять параметров: идентификатор поста (@p\_postID), идентификатор пользователя (@p\_userID), текст комментария (@p\_commentText) и дату публикации (@p\_date). Зависимые объекты - таблицы Users, Posts и Comments.

1. Функции

* add\_friend: Функция, которая добавляет новую запись в таблицу Friends. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор друга (@p\_friendID). Зависимый объект - таблица Friends.
* add\_like: Функция, которая добавляет новую запись в таблицу Likes. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор поста (@p\_postID). Зависимые объекты - таблицы Users, Posts и Likes.
* remove\_like: Функция, которая удаляет запись из таблицы Likes. Она принимает два параметра: идентификатор пользователя (@p\_userID) и идентификатор поста (@p\_postID). Зависимые объекты - таблицы Users, Posts и Likes.

|  |
| --- |
| CREATE SEQUENCE seq\_users; CREATE TRIGGER trg\_users BEFORE INSERT ON Users FOR EACH ROW BEGIN  SELECT seq\_users.nextval INTO :new.UserID FROM dual; END; /  CREATE SEQUENCE seq\_posts; CREATE TRIGGER trg\_posts BEFORE INSERT ON Posts FOR EACH ROW BEGIN  SELECT seq\_posts.nextval INTO :new.PostID FROM dual; END; /  CREATE SEQUENCE seq\_comments; CREATE TRIGGER trg\_comments BEFORE INSERT ON Comments FOR EACH ROW BEGIN  SELECT seq\_comments.nextval INTO :new.CommentID FROM dual; END; /  CREATE OR REPLACE PROCEDURE add\_user(p\_name NVARCHAR2, p\_email NVARCHAR2, p\_date DATE) AS BEGIN  INSERT INTO Users (Name, Email, RegistrationDate) VALUES (p\_name, p\_email, p\_date); END; /  CREATE OR REPLACE PROCEDURE add\_post(p\_userID NUMBER, p\_postText NVARCHAR2, p\_date DATE) AS BEGIN  INSERT INTO Posts (UserID, PostText, PublicationDate) VALUES (p\_userID, p\_postText, p\_date); END; /  CREATE OR REPLACE PROCEDURE add\_comment(p\_postID NUMBER, p\_userID NUMBER, p\_commentText NVARCHAR2, p\_date DATE) AS BEGIN  INSERT INTO Comments (PostID, UserID, CommentText, PublicationDate) VALUES (p\_postID, p\_userID, p\_commentText, p\_date); END; /  CREATE OR REPLACE PROCEDURE add\_friend(  p\_userID IN NUMBER,  p\_friendID IN NUMBER ) AS BEGIN  INSERT INTO Friends(UserID, FriendID)  VALUES (p\_userID, p\_friendID);  COMMIT; EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE; END add\_friend;   CREATE OR REPLACE PROCEDURE add\_like(  p\_userID IN NUMBER,  p\_postID IN NUMBER ) AS BEGIN  INSERT INTO Likes(UserID, PostID)  VALUES (p\_userID, p\_postID);  COMMIT; EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE; END add\_like;    CREATE OR REPLACE PROCEDURE remove\_like(  p\_userID IN NUMBER,  p\_postID IN NUMBER ) AS BEGIN  DELETE FROM Likes  WHERE UserID = p\_userID AND PostID = p\_postID;  COMMIT; EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE; END remove\_like; |

**SQL Server**

|  |
| --- |
| CREATE SEQUENCE seq\_users AS INT START WITH 1 INCREMENT BY 1; GO CREATE TRIGGER trg\_users ON Users AFTER INSERT AS BEGIN  UPDATE Users  SET UserID = NEXT VALUE FOR seq\_users  FROM inserted  WHERE Users.UserID = inserted.UserID; END; GO  CREATE SEQUENCE seq\_posts AS INT START WITH 1 INCREMENT BY 1; GO CREATE TRIGGER trg\_posts ON Posts AFTER INSERT AS BEGIN  UPDATE Posts  SET PostID = NEXT VALUE FOR seq\_posts  FROM inserted  WHERE Posts.PostID = inserted.PostID; END; GO  CREATE SEQUENCE seq\_comments AS INT START WITH 1 INCREMENT BY 1; GO CREATE TRIGGER trg\_comments ON Comments AFTER INSERT AS BEGIN  UPDATE Comments  SET CommentID = NEXT VALUE FOR seq\_comments  FROM inserted  WHERE Comments.CommentID = inserted.CommentID; END; GO  CREATE PROCEDURE *add\_user* @p\_name NVARCHAR(100), @p\_email NVARCHAR(100), @p\_date DATETIME AS BEGIN  INSERT INTO Users (Name, Email, RegistrationDate) VALUES (@p\_name, @p\_email, @p\_date); END; GO  CREATE PROCEDURE *add\_post* @p\_userID INT, @p\_postText NVARCHAR(MAX), @p\_date DATETIME AS BEGIN  INSERT INTO Posts (UserID, PostText, PublicationDate) VALUES (@p\_userID, @p\_postText, @p\_date); END; GO   CREATE PROCEDURE *add\_comment* @p\_postID INT, @p\_userID INT, @p\_commentText NVARCHAR(MAX), @p\_date DATETIME AS BEGIN  INSERT INTO Comments (PostID, UserID, CommentText, PublicationDate) VALUES (@p\_postID, @p\_userID, @p\_commentText, @p\_date); END; GO   CREATE PROCEDURE *add\_friend* @p\_userID INT,  @p\_friendID INT AS BEGIN  INSERT INTO Friends(UserID, FriendID)  VALUES (@p\_userID, @p\_friendID); END;  CREATE PROCEDURE *add\_like* @p\_userID INT,  @p\_postID INT AS BEGIN  INSERT INTO Likes(UserID, PostID)  VALUES (@p\_userID, @p\_postID); END;  CREATE PROCEDURE *remove\_like* @p\_userID INT,  @p\_postID INT AS BEGIN  DELETE FROM Likes  WHERE UserID = @p\_userID AND PostID = @p\_postID; END; |

**Oracle**