Procedimentos

```
CREATE SEQUENCE USERS_SEQ INCREMENT BY 1 MAXVALUE 9999 MINVALUE 1 CACHE 20;

CREATE OR REPLACE PROCEDURE pr_UserAdd

(

 p_USER_NAME  IN VARCHAR2,

 p_email     IN VARCHAR2,

 p_PASSWORD  IN VARCHAR2,

 p_SCORE_POINT_WALLET  IN NUMBER,

 p_AUTHORITY  IN NUMBER,

 p_USER_LEVEL IN NUMBER,

 p_Status    OUT NUMBER

)

AS

BEGIN

 INSERT INTO Users(USER_ID, USER_NAME, EMAIL,
PASSWORD,SCORE_POINT_WALLET,AUTHORITY,USER_LEVEL)

        VALUES(Users_Seq.NextVal, p_USER_NAME, p_email, p_PASSWORD,
p_SCORE_POINT_WALLET,p_AUTHORITY, p_USER_LEVEL);

 p_Status := 1;

END pr_UserAdd;

/

********************************************************************************

CREATE SEQUENCE NEWS_SEQ INCREMENT BY 1 MAXVALUE 9999 MINVALUE 1 CACHE 20;


CREATE OR REPLACE PROCEDURE pr_NEWSAdd

(

 p_TITLE  IN VARCHAR2,

 p_CONTENT     IN VARCHAR2,

 p_NEWDATE  IN DATE,

 p_GAME_NAME  IN VARCHAR2,

 p_Status    OUT NUMBER

)

AS
```

```
BEGIN

 INSERT INTO Users(NEWS_ID, TITLE, CONTENT,
NEWDATE,GAME_NAME,USERS_USER_ID,GAMES_GAME_ID)

        VALUES(NEWS_Seq.NextVal, p_TITLE,  p_CONTENT, p_NEWDATE, p_GAME_NAME);

 p_Status := 1;

END pr_NEWSAdd;

/
```

Functions

```
CREATE OR REPLACE FUNCTION checkOldNews (newsID IN NUMBER)

RETURN VARCHAR2 IS

info VARCHAR2(56) := 'An error occured.';

oldDate DATE;

newDat DATE;

BEGIN

   newDat := TRUNC(SYSDATE);

   SELECT TRUNC(NEWDATE) INTO oldDate FROM NEWS WHERE NEWS_ID = newsID;

   IF newDat-10 > oldDate THEN

      info := 'This new has been deleted.';

      RETURN info;

   ELSE

      info := 'This new is not old.';

      RETURN info;

   END IF;

END;

/

SET serveroutput ON;

DECLARE

message VARCHAR2(56);

BEGIN


  message := checkOldNews(1);

  DBMS_OUTPUT.PUT_LINE( message);
```

```
END;
/
********************************************************************************************
CREATE OR REPLACE FUNCTION countlike(cID NUMBER)
RETURN NUMBER IS likeC NUMBER;
begin
    SELECT COUNT(*) INTO likeC FROM LIKES WHERE COMMENTS_COM_ID = cID;
    IF likeC = NULL THEN
    DBMS_OUTPUT.PUT_LINE('There is no like for this comment.');
    ELSE
    DBMS_OUTPUT.PUT_LINE('There are ' || likeC ||' Likes for this comment.');
    END IF;
end;
/
```

Triggers

```
CREATE OR REPLACE TRIGGER checkingEmail
BEFORE INSERT ON USERS
FOR EACH ROW
ENABLE
DECLARE
flag NUMBER, check_mail VARCHAR(32);
BEGIN
    check_mail := :new.EMAIL;
    FOR j IN 1..len(check_mail);
    LOOP
       IF SUBSTR(check_mail, j, 1) IS '@' THEN
          flag := 1;
          EXIT;
       END IF;
    END LOOP;
    IF flag IS 0 THEN
    RAISE_APPLICATION_ERROR(-20041, 'This is not a correct mail adress.');
```

```
END;
********************************************************************************************
CREATE OR REPLACE TRIGGER checkingUserLevel
BEFORE INSERT ON USERS
FOR EACH ROW
ENABLE
BEGIN
      IF :new.USER_LEVEL > 20 THEN
        RAISE_APPLICATION_ERROR(-20040, 'You can not set users level over 20, it must be less.');
      END IF;
END;
```

Packages

```
CREATE OR REPLACE PACKAGE ADMINN IS
PROCEDURE cursor (P_CHAR varchar2);
END ADMINN;
/
CREATE OR REPLACE PACKAGE BODY ADMINN IS
PROCEDURE cursor (P_CHAR varchar2) IS
CURSOR C_NUM IS
SELECT USER_NAME
FROM USERS
where USER_NAME like '%' ||P_CHAR|| '%'
  order by USER_NAME desc;
BEGIN
FOR V_REG_USER IN C_NUM LOOP
DBMS_OUTPUT.PUT_LINE ('WELICOME  '||V_REG_USER.USER_NAME);
END LOOP;
END cursor;
END ADMINN;
/
set serveroutput on;
exec ADMINN.CURSOR ('A');
```