<https://github.com/LuckyForce/DBI-PLSQL-Project-Winter-22-23>

Functions

-- Check if Admin of Club  
CREATE OR REPLACE FUNCTION *isUserAdminOfClub*(p\_customer\_id INTEGER, p\_club\_id INTEGER)  
RETURN BOOLEAN  
AS  
 v\_is\_admin BOOLEAN;  
 v\_club\_id INTEGER;  
 CURSOR v\_clubs IS SELECT Id FROM Club WHERE AdminId = p\_customer\_id AND Id = p\_club\_id;  
BEGIN  
 OPEN v\_clubs;  
 FETCH v\_clubs INTO v\_club\_id;  
 IF v\_clubs%FOUND THEN  
 v\_is\_admin := TRUE;  
 ELSE  
 v\_is\_admin := FALSE;  
 END IF;  
 CLOSE v\_clubs;  
 RETURN v\_is\_admin;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- ClubPaid  
CREATE OR REPLACE FUNCTION *isClubPaid*(p\_club\_id INTEGER)  
RETURN BOOLEAN  
AS  
 v\_is\_paid BOOLEAN;  
 v\_paid\_date DATE;  
 CURSOR v\_club IS SELECT PAIDTILL FROM Club WHERE Id = p\_club\_id;  
BEGIN  
 OPEN v\_club;  
 FETCH v\_club INTO v\_paid\_date;  
 IF v\_club%NOTFOUND THEN  
 v\_is\_paid := FALSE;  
 ELSE  
 IF v\_paid\_date > *SYSDATE* THEN  
 v\_is\_paid := TRUE;  
 ELSE  
 v\_is\_paid := FALSE;  
 END IF;  
 END IF;  
 CLOSE v\_club;  
 RETURN v\_is\_paid;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- User Verified  
CREATE OR REPLACE FUNCTION *isVerified*(p\_customer\_id INTEGER)  
RETURN BOOLEAN  
AS  
 v\_is\_verified BOOLEAN;  
 v\_costumer\_id INTEGER;  
 CURSOR c\_user IS SELECT ID FROM Customer WHERE Id = p\_customer\_id AND Verified = 1;  
BEGIN  
 OPEN c\_user;  
 FETCH c\_user INTO v\_costumer\_id;  
 IF c\_user%NOTFOUND THEN  
 v\_is\_verified := FALSE;  
 ELSE  
 v\_is\_verified := TRUE;  
 END IF;  
 CLOSE c\_user;  
 RETURN v\_is\_verified;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Court Type Valid  
CREATE OR REPLACE FUNCTION *isCourtTypeValid*(p\_court\_type\_id INTEGER)  
RETURN BOOLEAN  
AS  
 v\_is\_valid BOOLEAN;  
 v\_court\_type\_id INTEGER;  
 CURSOR v\_court\_type IS SELECT ID FROM CourtType WHERE Id = p\_court\_type\_id;  
BEGIN  
 OPEN v\_court\_type;  
 FETCH v\_court\_type INTO v\_court\_type\_id;  
 IF v\_court\_type%NOTFOUND THEN  
 v\_is\_valid := FALSE;  
 ELSE  
 v\_is\_valid := TRUE;  
 END IF;  
 CLOSE v\_court\_type;  
 RETURN v\_is\_valid;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Procedures

-- Procedure register(email, password)  
CREATE OR REPLACE PROCEDURE *register*(p\_email IN VARCHAR2, p\_password IN VARCHAR2)  
IS  
 v\_user\_id NUMBER;  
 CURSOR c\_customer IS  
 SELECT ID  
 FROM customer  
 WHERE email = p\_email;  
BEGIN  
 OPEN c\_customer;  
 FETCH c\_customer INTO v\_user\_id;  
 IF c\_customer%FOUND THEN  
 CLOSE c\_customer;  
 *RAISE\_APPLICATION\_ERROR*(-20001, 'Email already exists');  
 END IF;  
 CLOSE c\_customer;  
 INSERT INTO customer (uuid, email, password, verificationcode, verified, resetcode, firstname, lastname, gender, welcomed)  
 VALUES (*sys\_guid*(), p\_email, p\_password, '233213', 0, '132321', 'Some Name', 'Some Name', 1, 0);  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure verify(email, verificationcode)  
CREATE OR REPLACE PROCEDURE *verify*(p\_email IN VARCHAR2, p\_verificationcode IN VARCHAR2)  
IS  
 v\_user\_id NUMBER;  
 CURSOR c\_customer IS  
 SELECT ID  
 FROM customer  
 WHERE email = p\_email AND verificationcode = p\_verificationcode;  
BEGIN  
 OPEN c\_customer;  
 FETCH c\_customer INTO v\_user\_id;  
 IF NOT c\_customer%FOUND THEN  
 CLOSE c\_customer;  
 *RAISE\_APPLICATION\_ERROR*(-20002, 'Customer does not exist');  
 END IF;  
 CLOSE c\_customer;  
 UPDATE customer  
 SET verified = 1  
 WHERE email = p\_email AND verificationcode = p\_verificationcode;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure set\_personal\_data(FirstName, LastName, Gender, BirthDate, user\_id)  
CREATE OR REPLACE PROCEDURE *set\_personal\_data*(p\_costumer\_id NUMBER, p\_firstname IN VARCHAR2, p\_lastname IN VARCHAR2)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_costumer\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 UPDATE customer  
 SET firstname = p\_firstname, lastname = p\_lastname  
 WHERE id = p\_costumer\_id;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure create\_club(name, link, user)  
CREATE OR REPLACE PROCEDURE *create\_club*(p\_name IN VARCHAR2, p\_link IN VARCHAR2, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 INSERT INTO CLUB (LINK, ADMINID, IBAN, PAIDTILL, FREETRIALTILL, NAME, INFO, ADDRESS, ZIPCODE, IMAGEPATH)  
 VALUES (p\_link, p\_user\_id, 'NL00RABO0000000000', *SYSDATE*, *SYSDATE* + 30, p\_name, '', '', '', 'default.png');  
END;

Ein Bild, das Text, Bildschirm, Screenshot, schließen enthält.

Automatisch generierte Beschreibung



-- Procedure pay\_club(club\_id, user\_id)  
CREATE OR REPLACE PROCEDURE *pay\_club*(p\_club\_id NUMBER, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_club\_id, p\_user\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'User is not admin of club');  
 END IF;  
  
 UPDATE CLUB  
 SET PAIDTILL = PAIDTILL + 30  
 WHERE ID = p\_club\_id;  
END;



Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure create\_event(name, description, date, time, location, club\_id, user\_id)  
CREATE OR REPLACE PROCEDURE *create\_event*(p\_title IN VARCHAR2, p\_info IN VARCHAR2, p\_club\_id NUMBER, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_user\_id, p\_club\_id);  
 v\_is\_paid BOOLEAN := *ISCLUBPAID*(p\_club\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Customer is not admin of club');  
 END IF;  
  
 IF NOT v\_is\_paid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Club is not paid');  
 END IF;  
  
 INSERT INTO CLUBEVENT (TITLE, INFO, CLUBNAVIGATIONID)  
 VALUES (p\_title, p\_info, p\_club\_id);  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure create\_news(title, description, date, club\_id, user\_id)  
CREATE OR REPLACE PROCEDURE *create\_news*(p\_title IN VARCHAR2, p\_info IN VARCHAR2, p\_club\_id NUMBER, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_user\_id, p\_club\_id);  
 v\_is\_paid BOOLEAN := *ISCLUBPAID*(p\_club\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Customer is not admin of club');  
 END IF;  
  
 IF NOT v\_is\_paid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Club is not paid');  
 END IF;  
  
 INSERT INTO CLUBNEWS (TITLE, INFO, CLUBNAVIGATIONID)  
 VALUES (p\_title, p\_info, p\_club\_id);  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure create\_court(name, location, club\_id)  
CREATE OR REPLACE PROCEDURE *create\_court*(p\_name IN VARCHAR2, p\_type NUMBER, p\_club\_id NUMBER, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_user\_id, p\_club\_id);  
 v\_is\_paid BOOLEAN := *ISCLUBPAID*(p\_club\_id);  
 v\_is\_court\_type\_valid BOOLEAN := *ISCOURTTYPEVALID*(p\_type);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Customer is not admin of club');  
 END IF;  
  
 IF NOT v\_is\_paid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Club is not paid');  
 END IF;  
  
 IF NOT v\_is\_court\_type\_valid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20006, 'Court type is not valid');  
 END IF;  
  
 INSERT INTO COURT (NAME, BOOKABLE, TYPE, APRICE, BPRICE, ATIMEFROM, ATIMETILL, AWEEKENDTIMETILL, CLUBNAVIGATIONID)  
 VALUES (p\_name, 1, 1, 10, 10, 10, 10, 10, p\_club\_id);  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



-- Procedure edit\_club\_link(club\_id, link)  
CREATE OR REPLACE PROCEDURE *edit\_club\_link*(p\_club\_id NUMBER, p\_link IN VARCHAR2, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_user\_id, p\_club\_id);  
 v\_is\_paid BOOLEAN := *ISCLUBPAID*(p\_club\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Customer is not admin of club');  
 END IF;  
  
 IF NOT v\_is\_paid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Club is not paid');  
 END IF;  
  
 UPDATE CLUB  
 SET LINK = p\_link  
 WHERE ID = p\_club\_id;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Procedure edit\_club\_name(club\_id, name)  
CREATE OR REPLACE PROCEDURE *edit\_club\_name*(p\_club\_id NUMBER, p\_name IN VARCHAR2, p\_user\_id NUMBER)  
IS  
 v\_verified BOOLEAN := *isverified*(p\_user\_id);  
 v\_is\_admin BOOLEAN := *ISUSERADMINOFCLUB*(p\_user\_id, p\_club\_id);  
 v\_is\_paid BOOLEAN := *ISCLUBPAID*(p\_club\_id);  
BEGIN  
 IF v\_verified THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Customer is not verified');  
 END IF;  
  
 IF NOT v\_is\_admin THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Customer is not admin of club');  
 END IF;  
  
 IF NOT v\_is\_paid THEN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Club is not paid');  
 END IF;  
  
 UPDATE CLUB  
 SET NAME = p\_name  
 WHERE ID = p\_club\_id;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Get all clubs  
CREATE OR REPLACE PROCEDURE *get\_clubs*(p\_search IN VARCHAR2)  
IS  
 CURSOR c\_clubs IS  
 SELECT ID, NAME, LINK  
 FROM CLUB  
 WHERE NAME LIKE '%' || p\_search || '%';  
 v\_clubs c\_clubs%ROWTYPE;  
BEGIN  
 IF p\_search IS NULL THEN  
 *RAISE\_APPLICATION\_ERROR*(-20001, 'Search is null');  
 END IF;  
 FOR v\_clubs IN c\_clubs LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_clubs.ID || ' ' || v\_clubs.NAME || ' ' || v\_clubs.LINK);  
 END LOOP;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Get all courts  
CREATE OR REPLACE PROCEDURE *get\_courts*(p\_club\_id NUMBER)  
IS  
 CURSOR c\_courts IS  
 SELECT \*  
 FROM COURT  
 WHERE CLUBNAVIGATIONID = p\_club\_id;  
 v\_courts c\_courts%ROWTYPE;  
BEGIN  
 FOR v\_courts IN c\_courts LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_courts.ID || ' ' || v\_courts.NAME || ' ' || v\_courts.BOOKABLE || ' ' || v\_courts.TYPE || ' ' || v\_courts.APRICE || ' ' || v\_courts.BPRICE || ' ' || v\_courts.ATIMEFROM || ' ' || v\_courts.ATIMETILL || ' ' || v\_courts.AWEEKENDTIMETILL || ' ' || v\_courts.CLUBNAVIGATIONID);  
 END LOOP;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Get all events  
CREATE OR REPLACE PROCEDURE *get\_events*(p\_club\_id NUMBER)  
IS  
 CURSOR c\_events IS  
 SELECT \*  
 FROM CLUBEVENT  
 WHERE CLUBNAVIGATIONID = p\_club\_id;  
 v\_events c\_events%ROWTYPE;  
BEGIN  
 FOR v\_events IN c\_events LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_events.ID || ' ' || v\_events.TITLE || ' ' || v\_events.INFO || ' ' || v\_events.CLUBNAVIGATIONID);  
 END LOOP;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

-- Get all news  
CREATE OR REPLACE PROCEDURE *get\_news*(p\_club\_id NUMBER)  
IS  
 CURSOR c\_news IS  
 SELECT \*  
 FROM CLUBNEWS  
 WHERE CLUBNAVIGATIONID = p\_club\_id;  
 v\_news c\_news%ROWTYPE;  
BEGIN  
 FOR v\_news IN c\_news  
 LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_news.ID || ' ' || v\_news.TITLE || ' ' || v\_news.INFO || ' ' || v\_news.CLUBNAVIGATIONID);  
 END LOOP;  
END;

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung