OJT 데이터베이스

11회차 과제물

|  |  |
| --- | --- |
| 지도교수: | 김 양 중 교 수 님 |
| 현장교사: | 류경문부장님 |
| 회 사 명: | 삼정데이타서비스㈜ |
| 학 번: | 2019531001 |
| 이 름: | 권양환 |
| 제 출 일: | 2020-08-06 |

**실습 SQL 작성한 문서 URL**

https://github.com/yhwan0129/kpulms

**실습 SQL문**

**[DDL]**

CRATE TABLE '주문테이블'(

'주문번호' varchar(16) NOT NULL,

'고객번호' varchar(16) NOT NULL,

'주문일' varchar(8) NOT NULL,

'주문가격' decimal(15,2) NOT NULL,

'배송도시' varchar(256),

'배송완료일' varchar(8),

'결제금액' varchar(8),

'할인금액' decimal(15,2) NOT NULL,

'적립포인트' decimal(15,2) NOT NULL,

PRIMARY KEY('주문번호'))

CREATE TABLE NEWTABLE AS SELECT \* FROM '주문테이블';

SELECT \* FROM '주문테이블';

DESC '주문테이블';

ALTER TABLE '주문테이블' CHANGE '배송도시' '배송도시코드' INT;

ALTER TABLE '주문테이블' ADD 'NEWCOLMN' VARCHAR(10);

ALTER TABLE '주문테이블' ADD FOREIGN KEY ('배송도시코드') REFERENCES '도시코드테이블' (code);

**[DML]**

CREATE TABLE employees

(

UID INT NOT NULL,

PRIMARY KEY(UID)

);

CREATE TABLE salaries

(

emp\_no INT NOT NULL,

salary INT NOT NULL,

from\_date DATE NOT NULL,

to\_date DATE,

FOREIGN KEY(emp\_no) REFERENCES employees (UID) ON DELETE CASCADE,

PRIMARY KEY(emp\_no, from\_date));

INSERT INTO employees (UID) VALUES (1002);

INSERT INTO salaries (emp\_no, salary, from\_date) VALUES (1002, 400000, '2017-07-01');

INSERT INTO salaries VALUES (1002, 4000, '2017-08-02');

INSERT INTO salaries (emp\_no, salary, from\_date, to\_date) VALUES

(1002, 2000, '2020-09-09', '2013-03-28'),

(1002, 2000, '2020-09-08', '2013-03-27'),

(1002, 2000, '2020-09-07', '2013-03-26');

INSERT INTO salaries VALUES

(1002, 2000, '2020-10-09', '2013-02-28'),

(1002, 2000, '2020-10-08', '2015-02-27'),

(1002, 2000, '2020-10-07', '2014-02-26');

SELECT \* FROM salaries;

SELECT \* FROM salaries WHERE emp\_no = 1002;

SELECT emp\_no, salary FROM salaries WHERE emp\_no = 1002;

SELECT \* FROM salaries WHERE from\_date = (SELECT MAX(from\_date) FROM salaries);

SELECT \* FROM salaries ORDER BY from\_date DESC LIMIT 1;

UPDATE salaries SET salary = 110000 WHERE emp\_no = 1002 AND from\_date = '2020-10-09';

DELETE FROM salaries WHERE emp\_no = 1002;

**[DCL]**

CREATE USER SUPER@localhost IDENTIFIED BY 'password';

SHOW GRANTS FOR SUPER@localhost;

DROP user super@localhost;

GRAND SELECT, UPDATE ON db.\* TO super;

REVOKE all ON \*.\* FROM super@localhost;

SELECT @@AUTOCOMMIT;

SET AUTOCOMMIT = TRUE; // Auto Commit 설정

SET AUTOCOMMIT = FALSE; // Auto Commit 해지

START TRANSACTION;

savepoint a;

INSERT INTO salaries VALUES(1002, 900, '2014-07-01', '2015-06-30');

SELECT \* FROM salaries;

savepoint b;

UPDATE salaries set salary = 1000 where emp\_no = 1002; // salary 값이 1000으로 변경

SELECT \* FROM salaries;

**[INDEX]**

Create index idx배송도시 ON 주문테이블 (배송도시);

Create index idx주문일포인트 ON 주문테이블 (주문일, 적립포인트);

Create index idx고객번호 ON 주문테이블 (고객번호);

ALTER TABLE 주문테이블 DROP INDEX idx배송도시;

ALTER TABLE 주문테이블 DROP INDEX idx주문일포인트;

ALTER TABLE 주문테이블 DROP INDEX idx고객번호;

**[VIEW]**

CREATE VIEW CHECK\_LOGIN\_VIEW as select UID, PASSWORD from emp;

SELECT UID FROM CHECK\_LOGIN\_VIEW WHERE UID=’user’, PASSWORD=’pw’;

CREATE VIEW CHECK\_WORKPLACE\_VIEW AS select UID, NAME, WORKPLACE from EMP WHERE WORKPLACE IN (‘서울’, ‘부산’);

Create view check\_workday+view (UID, NAME, JOINDAY, WORKDAYS) AS select UID, NAME< JOINDAY, to\_date(NOW() – Date(JOINDAY) ) from EMP;

CREATE VIEW PARTIAL\_VIEW (NAME, POSITION, WORKPLACE) AS select NAME, POSITION, WORKPALCE) FROM EMP;

INSERT INTO PARTIAL\_VIEW VALUES(‘도길동’, ‘대리’,’부산’);

UPDATE PARTIAL\_VIEW SET NAME=’도길동’, POSITION=’대리’, WORKPLACE=’부산’;

(실패내역)

UPDATE CHECK\_WORKPLACE\_VIEW SEY NAME=’도길동’, WORKPLACE=’부산’ WHERE UID = 1090;

(성공 내역)

DROP VIEW CHECK\_LOGIN\_VIEW;

DROP VIEW CHECK\_WORKPLACE\_VIEW;

DROP VIEW CHECK\_WORKDAY\_VIEW;

**[JOIN]**

SELECT productCode, productName, textDescription

FROM product t1

INNER JOIN product t2 ON t1.productLine = t2.productLine;

SELECT productCode, productName, textDescription

FROM products t1

INNER JOIN productlines t2 USING (productLine);

SELECT productCode, productName, textDescription

FROM product t1

NATURAL JOIN productLines t2;

SELECT productCode, productName, textDescription

FROM products t1 CROSS JOIN productlines t2;

SELECT productCode, productName, texstDescription

FROM products t1

INNER JOIN productlines t2;

SELECT productCOde, productName, textDescription

FROM products t1

LEFT OUTER JOIN productlines t2 ON t1.priductLine = t2.productlLine;

SELECT productname, buyprice

FROM products p1

WHERE buyprice > (SELECT AVG(buyprice) FROM products WHERE productline = p1.productline);

SELECT productcode, productname

FROM products

UNION

SELECT productLine, textDescription

FROM productlines;

SELECT productcode id, productname name

FROM products

UNION

SELECT productLine id, textDescription name

FROM productlines;

ELECT \* FROM tb\_member;

--tb\_member테이블에 테이터 추가(insert)시 "멤버가 추가되었습니다.." 라는 메시지를 출력하는 트리거 생성.

SET serveroutput ON

CREATE OR REPLACE TRIGGER tr\_member\_insert

after INSERT

ON tb\_member

FOR each ROW

BEGIN

IF inserting THEN

DBMS\_OUTPUT.PUT\_LINE('멤버가 추가되었습니다..');

END IF;

END;

/

-- 데이터 추가

INSERT INTO TB\_MEMBER

(IDX,USERID,USERNAME,USERPWD,EMAIL,SSN1,SSN2,HP1,HP2,HP3,GENDER,HOBBY,REGISTDATE,M\_BANKCODE)

VALUES

(13,'kkk','케이이','1234','kkk@naver.com','123456','1233333','010','111','212','남','컴퓨터',sysdate,'01');

-- 트리거 사용사례

-- 회원이 가입되면서 insert해야할 테이블이 여러개일경우

-- 회원가입 -> tb\_member

-- -> tb\_point

-- -> tb\_profile

-- -> tb\_login\_history

-- 트리거 예제.

-- 테이블 생성.

CREATE TABLE tb\_point\_history(

ph\_idx NUMBER PRIMARY KEY,

ph\_userid varchar2(20) CONSTRAINT fk\_tb\_point\_histor REFERENCES tb\_member2(m\_userid),

ph\_input NUMBER,

ph\_output NUMBER,

ph\_point NUMBER,

ph\_registdate DATE

);

-- 시퀀스 생성. tb\_point\_history에 사용할 시퀀스

CREATE SEQUENCE seq\_tb\_point\_history

INCREMENT BY 1

START WITH 1;

-- 시퀀스 제거.

DROP SEQUENCE seq\_tb\_point\_history;

-- TB\_MEMBER2 테이블 데이터 추가.

INSERT INTO TB\_MEMBER2

VALUES (6,'ffff','에프에','1234','fff@naver.com','1234561263363','010','222','212','남자','자전거',0,sysdate,'01',NULL);

-- tb\_point\_history 테이블 데이터 추가.

INSERT INTO tb\_point\_history VALUES (seq\_tb\_point\_history.NEXTVAL, 'ffff', 0,0,0, sysdate);

SET serveroutput ON

-- 트리거 생성. : tb\_member2 테이블에 데이터가 추가되면 tb\_point\_history테이블에 데이터를 자동으로 추가

CREATE OR REPLACE TRIGGER tr\_member2\_insert

after INSERT

ON tb\_member2

FOR each ROW

BEGIN

IF inserting THEN

INSERT INTO tb\_point\_history VALUES (seq\_tb\_point\_history.NEXTVAL, :NEW.m\_userid, :NEW.M\_point,0,:NEW.M\_point, sysdate); --변경전의 ' :OLD ' , 변경후의 ' :NEW '

DBMS\_OUTPUT.PUT\_LINE('멤버가 추가되었습니다..');

END IF;

END;

/

INSERT INTO TB\_MEMBER2

VALUES (6,'ffff','에프에','1234','fff@naver.com','1234561263363','010','222','212','남자','자전거',0,sysdate,'01',NULL);

INSERT INTO TB\_MEMBER2

VALUES (7,'gggg','지지지','1234','gggg@naver.com','1234561264263','010','222','338','남자','자전거',NULL,sysdate,'01',NULL);

-- 결과확인

SELECT \* FROM tb\_point\_history;

SELECT \* FROM tb\_member2;

-- tb\_member2 테이블에서 회원 데이터가 제거될때 tb\_point\_history테이블에 해당 회원 정보를 제거하는 트리거 생성

CREATE OR REPLACE TRIGGER tr\_member\_delete

after DELETE

ON tb\_member2

FOR each ROW

BEGIN

DELETE FROM tb\_point\_history WHERE ph\_userid=:OLD.m\_userid; -- 지워진 다음에 처리

DBMS\_OUTPUT.PUT\_LINE('회원 정보가 삭제되었습니다.');

END;

/

DELETE FROM tb\_member2 WHERE m\_userid='ffff';

SELECT \* FROM tb\_point\_history;

SELECT \* FROM tb\_member2;