



Homework2 보고서

SQL & Relational Algebra

과목명	데이터베이스시스템및응용
담당 교수님	차재혁 교수
제출일	2021년 10월 15일(금요일)
소속	한양대학교 공과대학 컴퓨터소프트웨어학부
학번	이름
2019009261	최가온(CHOI GA ON)

I . Tables

sellers.csv, stores.csv, menus.csv 데이터를 입력할 sellers, stores, menus 테이블을 만들기

- 테이블 간의 관계를 고려할 것
- 각 컬럼에 알맞다고 생각하는 데이터 타입과 제약을 사용할 것
 - Foreign Key에 대해 delete cascade 옵션을 꼭 사용할 것
- 참고) 이 예제 데이터들은 “배달의 한양”의 샘플 데이터인데, 이 스키마가 최선인 것은 아니며 나중의 과제는 재가공해서 쓸 필요가 있다.

STEP 1

「homework2」 데이터베이스를 만든다.

유저를 통해 만든 homework2 데이터베이스에 접속한다.

```
CREATE DATABASE homework2;  
\c homework2
```

STEP 2

seller 테이블에서 만들 SEQUENCE 개체를 하나 생성한다.

```
CREATE SEQUENCE seq_seller_id INCREMENT 1 START 0 MINVALUE 0;  
ALTER SEQUENCE seq_seller_id OWNER TO postgres;
```

seller.csv 파일로부터 seller 테이블을 정의한다.

```
CREATE TABLE sellers (  
    seller_id INTEGER NOT NULL DEFAULT nextval (  
        'seq_seller_id'::regclass  
    ),  
    name VARCHAR (20) NOT NULL,  
    phone VARCHAR (20) NOT NULL,  
    local VARCHAR (20) NOT NULL,  
    domain VARCHAR (20) NOT NULL,  
    passwd VARCHAR (20) NOT NULL,  
    PRIMARY KEY (seller_id),  
    CONSTRAINT seller_email UNIQUE (local, domain)  
);
```

이후 seller.csv 파일로부터 sellers 테이블에 데이터를 추가한다.

```
COPY sellers (name, phone, local, domain, passwd)  
FROM '/home/example_db/sellers.csv' DELIMITER ',' CSV HEADER;
```

STEP 3

seller 테이블에서 만들 SEQUENCE 개체를 하나 생성한다.

```
CREATE SEQUENCE seq_store_id INCREMENT 1 START 0 MINVALUE 0;  
ALTER SEQUENCE seq_store_id OWNER TO postgres;
```

stores.csv 파일로부터 stores 테이블을 정의한다.

```
CREATE TABLE stores (  
    store_id INTEGER NOT NULL DEFAULT nextval (  

```

```

    'seq_store_id' :: regclass
),
address VARCHAR (100) NOT NULL,
sname VARCHAR (40) NOT NULL,
latitude FLOAT,
longitude FLOAT,
phone_nums VARCHAR (50) NOT NULL,
seller_id INTEGER,
PRIMARY KEY (store_id),
CONSTRAINT fk_seller FOREIGN KEY (seller_id)
    REFERENCES sellers (seller_id) ON DELETE CASCADE
);

```

이후 store.csv 파일로부터 stores 테이블에 데이터를 추가한다.

```

COPY stores
    (address, sname, latitude, longitude, phone_nums, seller_id)
FROM '/home/example_db/stores.csv' DELIMITER ',' CSV HEADER;

```

STEP 4

menus.csv 파일로부터 menus 테이블을 정의한다.

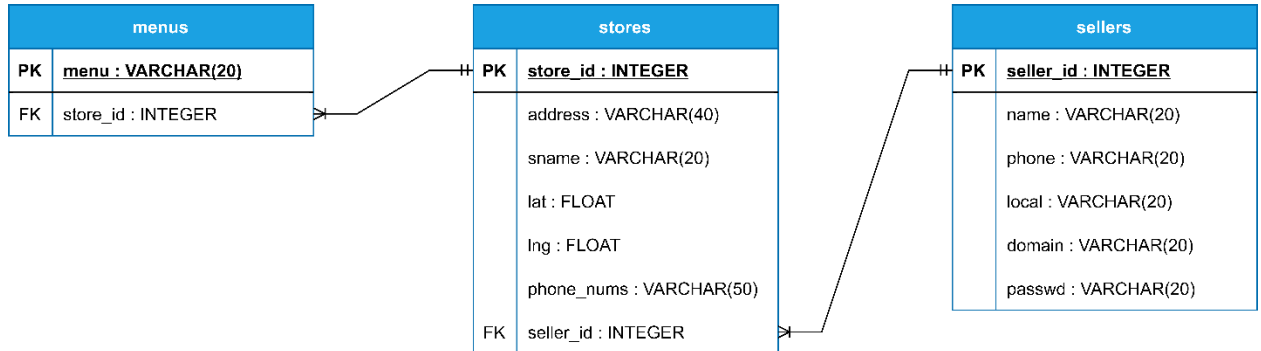
```

CREATE TABLE menus (
    menu VARCHAR (20) NOT NULL,
    store_id INTEGER NOT NULL,
    CONSTRAINT fk_store
        FOREIGN KEY (store_id) REFERENCES stores (store_id)
        ON DELETE CASCADE
);

```

STEP 5

아래의 그림은 sellers, stores, menus 테이블에 대한 E-R 다이어그램이다.



만약, 위의 설치를 제거하고 재설치할 경우 아래의 코드를 입력한다.

```
DROP TABLE menus;  
DROP TABLE stores;  
DROP TABLE num_store;  
DROP TABLE sellers;  
DROP SEQUENCE seq_seller_id;  
DROP SEQUENCE seq_store_id;
```

II. SQL & Relational Algebra

위 테이블을 이용해 다음 조건들을 SQL과 Relational Algebra로 나타내기

단, Relational Algebra는 밑줄 친 부분에 대해서만 나타내면 된다.

1. sellers 테이블의 이름과 전화번호를, 이름을 기준으로 내림차순으로 출력할 것

SQL Sentence

```
SELECT s.name, s.phone
FROM sellers s
ORDER BY s.name DESC;
```

Data Output

Explain

Messages

Notifications

	<div>name</div> <div>character varying (20)</div>	<div>phone</div> <div>character varying (20)</div>
1	행S응고르	01220306107
2	로버트JR	01220803622
3	로버트킬JR	01229519664
4	존F케네디	01288051408
5	존F	01272464629

✓

Successfully run. Total query runtime: 2 secs 752 msec. 50001 rows affected.

Relational Algebra

$$\tau_{s.name \downarrow} \left(\pi_{s.name, s.phone} (\rho_s(sellers)) \right)$$

2. sellers 테이블과 stores 테이블을 연결해 점주 이름이 '신동화'인 가게명을 모두 출력할 것

SQL Sentence

```
SELECT s2.sname
FROM sellers s1, stores s2
```

```
WHERE s2.seller_id = s1.seller_id AND s1.name = '신동화';
```

Data Output	Explain	Messages	Notifications
<div><div>sname</div><div>character varying (40)</div><div>1우정식당</div></div>			
<div>✓ Successfully run. Total query runtime: 104 msec. 1 rows affected.</div>			

Relational Algebra

$$\pi_{s2.sname} \left(\sigma_{s2.seller_id = s1.seller_id \text{ AND } s1.seller = \text{"신동화"}} (\rho_{s1}(\text{ sellers }) \times \rho_{s2}(\text{ stores })) \right)$$

3. stores 테이블의 주소에서 특별광역시도와 시군구를 분리해 서울특별시 중구에 있는 가게명을 출력할 것

SQL Sentence

```
SELECT s.sname
FROM stores s
WHERE SPLIT_PART (s.address, ' ', 1) = '서울특별시'
      AND SPLIT_PART (s.address, ' ', 2) = '중구';
```

Data Output		Explain	Messages	Notifications
	<div>sname</div> <div>character varying (40)</div>			
1	유림식당			
2	하동관			
3	제일			
4	한일			
5	프리모			

✓ Successfully run. Total query runtime: 131 msec. 7786 rows affected.







Relational Algebra

$$\pi_{s.sname} (\sigma_{\text{split_part}(s.address, " ", 1) = \text{"서울특별시"}, \text{split_part}(s.address, " ", 2) = \text{"중구"}} (\rho_s(\text{ stores })))$$

4. sellers 테이블에 본인 정보 넣기(사실일 필요 없음)

SQL Sentence

```
INSERT INTO sellers (name, phone, local, domain, passwd)
VALUES ('최가온', '01036969592', 'choigaon1028',
'hanyang.ac.kr', 'dbdbdeep');
```

Data Output		Explain	Messages	Notifications	
 seller_id [PK] integer	 name character varying (20)	 phone character varying (20)	 local character varying (20)	 domain character varying (20)	 passwd character varying (20)
1	50000 최가온	01036969592	choigaon1028	hanyang.ac.kr	dbdbdeep

✔ Successfully run. Total query runtime: 152 msec. 1 rows affected.

Relational Algebra

$$sellers \leftarrow sellers \cup \{ '최가온', '01036969592', 'choigaon1028', 'hanyang.ac.kr', 'dbdbdeep' \}$$

5. 판매자의 비밀번호 길이가 5 이하인 행을 찾아서 비밀번호를 6자리 이상의 임의의 영문자로 수정하기

SQL Sentence

STEP 1

비밀번호의 길이가 5 이하인 행을 sellers 테이블에서 찾는다.

```
SELECT * FROM sellers
WHERE length( passwd ) <= 5;
```

Data Output	Explain	Messages	Notifications			
<div><div><div><div></div><div>seller_id</div><div>[PK] integer</div></div></div></div>	<div><div><div><div></div><div>name</div><div>character varying (20)</div></div></div></div>	<div><div><div><div></div><div>phone</div><div>character varying (20)</div></div></div></div>	<div><div><div><div></div><div>local</div><div>character varying (20)</div></div></div></div>	<div><div><div><div></div><div>domain</div><div>character varying (20)</div></div></div></div>	<div><div><div><div></div><div>passwd</div><div>character varying (20)</div></div></div></div>	
<div><div><div><div></div><div>1</div></div></div></div>	<div><div><div><div></div><div>383</div></div></div></div>	<div><div><div><div></div><div>구맹희</div></div></div></div>	<div><div><div><div></div><div>01271816350</div></div></div></div>	<div><div><div><div></div><div>atodi</div></div></div></div>	<div><div><div><div></div><div>xing.com</div></div></div></div>	<div><div><div><div></div><div>asdf</div></div></div></div>

STEP 2








비밀번호의 길이가 5 이하인 행에 대해 영어 소문자/대문자 조합의 비밀번호로 passwd 내용을 바꾼다. 이때 비밀번호의 길이는 6 이상 10 이하의 임의의 길이를 갖는다.

```
UPDATE sellers
SET passwd = (
  SELECT array_to_string (ARRAY (
    SELECT chr ((
      65 + 32*round(random()) + round(random()*25)
    )::INTEGER)
    FROM generate_series (
      1, CAST(6 + floor(random()*5) AS INT)
    )), ''
  )
)
WHERE LENGTH(passwd) <= 5;
```

STEP 3

위의 STEP 2를 실행하기 전 passwd의 길이가 5 이하였던 행을 재검색하여 passwd를 확인한다. 비밀번호가 6자리 이상으로 새롭게 변경된 것을 확인할 수 있다.

```
SELECT * FROM sellers
WHERE name = '구맹회';
```

Data Output		Explain	Messages	Notifications		
 seller_id	 name	 phone	 local	 domain	 passwd	
[PK] integer	character varying (20)	character varying (20)	character varying (20)	character varying (20)	character varying (20)	
1	383 구맹회	01271816350	atodi	xing.com	gPOFJPC	

Relational Algebra

$$s1 \leftarrow \pi_{\text{seller_id, name, phone, local, domain, passwd}} \left(\sigma_{\text{length(passwd)} > 5}(\text{sellors}) \right)$$

$$s2 \leftarrow \pi_{\text{seller_id, name, phone, local, domain, passwd}} \left(\sigma_{\text{length(passwd)} \leq 5}(\text{sellors}) \right)$$

$$\text{sellors} \leftarrow s1 \cup s2$$

6. stores 테이블에 'rating'이라는 column을 추가하고, 0 이상 5 이하의 랜덤한 정수로 채워 넣기

SQL Sentence

```
ALTER TABLE stores
ADD COLUMN ratings INT
DEFAULT floor( random () * 6 );
```

Data Output Explain Messages Notifications

ALTER TABLE

Query returned successfully in 150 msec.

✓ Query returned successfully in 150 msec.

Data Output Explain Messages Notifications

	store_id [PK] integer	address character varying (100)	sname character varying (40)	latitude double precision	longitude double precision	phone_nums character varying (50)	seller_id integer	ratings integer
1	0	제주특별자치도 제주시 제원4...	거부	33.4863	126.489	"\0285434668\","021221...	39200	5
2	1	제주특별자치도 제주시 일주...	고기왕	33.4934	126.43	"\0285434668\","	36815	1
3	2	제주특별자치도 제주시 노연...	고담2015신제주점	33.4862	126.485	"\0285434668\","021221...	2481	4
4	3	제주특별자치도 제주시 서광...	고려회관	33.5013	126.527	"\0285434668\","021221...	24938	3
5	4	제주특별자치도 제주시 임항...	고집돌우럭제주공항점	33.5163	126.528	"\0285434668\","021221...	42558	5
6	5	제주특별자치도 제주시 비자...	꽃자왈	33.435	126.675	"\0285434668\","	40801	5

7. 필요한 테이블들을 연결해 판매자 이름, 판매자 id, 소유한 가게 수를 내림차순으로 출력하는 select query를 만들고, 그 결과를 새로운 테이블(num_store) 만들기

SQL Sentence

```
CREATE TABLE num_store AS (
  SELECT s1.name, s1.seller_id, COUNT (*) AS store_num
  FROM sellers s1, stores s2
  WHERE s1.seller_id = s2.seller_id
  GROUP BY s1.seller_id
  ORDER BY store_num DESC
);
```

Data Output	Explain	Messages	Notifications																																				
	<table> <thead> <tr> <th></th><th>name</th><th>seller_id</th><th>store_num</th></tr> <tr> <th></th><th>character varying (20)</th><th>integer</th><th>bigint</th></tr> </thead> <tbody> <tr><td>1</td><td>강명섭</td><td>13707</td><td>6</td></tr> <tr><td>2</td><td>권처런</td><td>20306</td><td>6</td></tr> <tr><td>3</td><td>김성혁</td><td>4994</td><td>6</td></tr> <tr><td>4</td><td>장욱경</td><td>25805</td><td>6</td></tr> <tr><td>5</td><td>함영원</td><td>33117</td><td>5</td></tr> <tr><td>6</td><td>스타이론</td><td>41845</td><td>5</td></tr> <tr><td>7</td><td>송이내</td><td>24606</td><td>5</td></tr> </tbody> </table>		name	seller_id	store_num		character varying (20)	integer	bigint	1	강명섭	13707	6	2	권처런	20306	6	3	김성혁	4994	6	4	장욱경	25805	6	5	함영원	33117	5	6	스타이론	41845	5	7	송이내	24606	5		
	name	seller_id	store_num																																				
	character varying (20)	integer	bigint																																				
1	강명섭	13707	6																																				
2	권처런	20306	6																																				
3	김성혁	4994	6																																				
4	장욱경	25805	6																																				
5	함영원	33117	5																																				
6	스타이론	41845	5																																				
7	송이내	24606	5																																				

Successfully run. Total query runtime: 231 msec. 26335 rows affected.

Relational Algebra

$$\tau_{\text{store_num} \downarrow} ($$

$$\pi_{s1.name, seller_id, COUNT(*) \rightarrow store_num} ($$

$$\gamma_{seller_id} ($$

$$\sigma_{s1.seller_id = s2.seller_id} (\rho_{s1}(\text{sellers}) \times \rho_{s2}(\text{stores}))$$

$$)))$$

8. num_store 테이블에서 가장 상위에 있는 판매자 아이디를 참고해, 판매자 테이블에서 해당 판매자를 삭제하기. 그를 참조하는 다른 테이블까지 cascading 되어야 한다.

SQL Sentence

STEP 1

num_store 테이블에서 store_num(소유 가게 수)의 최댓값을 찾는다.

```
SELECT MAX (store_num) FROM num_store;
```

STEP 2

num_store 테이블에서 소유 가게수가 최댓값에 해당하는 점주의 seller_id를 추출한다.

```
SELECT seller_id FROM num_store
WHERE store_num = (
    SELECT MAX(store_num)
    FROM num_store
);
```

Data Output

Explain

Messages

Notifications

	seller_id	
	integer	
1	13707	
2	20306	
3	4994	
4	25805	

✓ Successfully run. Total query runtime: 91 msec. 4 rows affected.

STEP 3

STEP 2에서 추출한 seller_id에 해당하는 점주에 대한 모든 column을 sellers 테이블에서 가져온다.

```
SELECT * FROM sellers s
```

```

WHERE s.seller_id IN (
    SELECT seller_id
    FROM num_store
    WHERE store_num = (
        SELECT MAX (store_num)
        FROM num_store
    )
);

```

Data Output		Explain	Messages	Notifications		
	seller_id [PK] integer	name character varying (20)	phone character varying (20)	local character varying (20)	domain character varying (20)	passwd character varying (20)
1	4994	김성혁	01287139579	gdumbbf	netscape.com	z2z01x82
2	13707	강명섭	01284158539	mferrd	wufoo.com	55mv0t7ww
3	20306	권처런	01235291990	mwrih	over-blog.com	e3d6vad1x
4	25805	장욱경	01279896728	nolivsa	webeden.co.uk	phsm00f7
✔ Successfully run. Total query runtime: 217 msec. 4 rows affected.						

STEP 4

(최종정답코드)

```

DELETE FROM sellers
WHERE seller_id IN (
    SELECT seller_id
    FROM num_store
    WHERE store_num = (
        SELECT MAX (store_num)
        FROM num_store
    )
);

```

Data Output	Explain	Messages	Notifications
DELETE 4			
Query returned successfully in 689 msec.			<div> Query returned successfully in 689 msec. </div>

STEP 5

이제 제대로 **sellers** 테이블에서 해당 점주의 정보가 지워졌는지 확인한다.

```
SELECT *  
FROM sellers  
WHERE seller_id IN ( 4994, 13707, 20306, 25805 );
```

Data Output	Explain	Messages	Notifications
seller_id [PK] integer	name character varying (20)	phone character varying (20)	local character varying (20)
			domain character varying (20)
			passwd character varying (20)

✓ Successfully run. Total query runtime: 94 msec. 0 rows affected.

STEP 6

해당 점주가 소요한 가게의 정보가 **stores** 테이블에서 지워졌는지 확인한다.

```
SELECT * FROM stores  
WHERE seller_id IN (4994, 13707, 20306, 25805);
```

Data Output	Explain	Messages	Notifications
store_id [PK] integer	address character varying (100)	sname character varying (40)	latitude double precision
			longitude double precision
			phone_nums character varying (50)
			seller_id integer

✓ Successfully run. Total query runtime: 346 msec. 0 rows affected.

STEP 7

해당 점주가 소요한 가게에서 제공하는 메뉴에 대한 정보가 모두 지워졌는지 확인한다.

```
SELECT * FROM menus
WHERE store_id IN (3458, 4560, 5465, 6147, 8353, 10476,
14510, 15883, 17661, 19675, 22279, 22313, 23759, 25692,
27816, 29117, 29551, 30565, 31989, 33218, 34209, 34770,
35714, 37308
);
```

Data Output	Explain	Messages	Notifications
menu character varying (20)	store_id integer		

Relational Algebra

위의 SQL문은 nested query 형태를 갖는다.

The most-inner query is:

$$\pi_{\text{MAX}(\text{store_num})}(\gamma_{\text{MAX}(\text{store_num})}(\text{num_store}))$$

The second query is:

$$\pi_{\text{seller_id}}(\sigma_{\text{store_num}=(\pi_{\text{MAX}(\text{store_num})}(\gamma_{\text{MAX}(\text{store_num})}(\text{num_store})))}(\text{num_store}))$$

The third query is:

$$\sigma_{\text{seller_id} \in \pi_{\text{seller_id}}(\sigma_{\text{store_num}=(\pi_{\text{MAX}(\text{store_num})}(\gamma_{\text{MAX}(\text{store_num})}(\text{num_store})))}(\text{num_store}))}(\rho_s(\text{seller}))$$

The final query is(최종정답):

(1)

$$\rho_{R1}(\pi_{\text{seller_id}}(\sigma_{\text{store_num}=(\pi_{\text{MAX}(\text{store_num})}(\gamma_{\text{MAX}(\text{store_num})}(\text{num_store})))}(\text{num_store})))$$

(2)

$$\text{ sellers } \leftarrow \text{ sellers } - \sigma_{\text{seller_id} \in R1}(\rho_s(\text{seller}))$$