

4-3. 집합 연산자 활용 및 기타

홍형경

chariehong@gmail.com

2020.01

집합연산자는 언제 사용할까?

- 어떤 상황에서 집합 연산자를 사용하는지에 대한 규칙은 없음
- 주어진 문제 해결을 위해 쿼리 작성을 하다 보면 자연스럽게 사용
- 명확한 규칙은 없으나 이러저러한 경우에 사용하는 경험 칙 소개

집합연산자는 언제 사용할까?

(1) UNION (ALL)

- UNION 과 UNION ALL은 쓰임새가 같음 (중복 데이터 제거만 차이)
- 구조가 다른 여러 테이블에서 동일한 형태의 데이터를 추출하는 경우
- 컬럼을 로우 형태로 전환해 조회할 경우

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

· 예산 테이블 (budget_table)

```
create table budget_table (  
    yearmon    VARCHAR2(6),  
    budget_amt  NUMBER    );
```

```
INSERT INTO budget_table values('201901', 1000);  
INSERT INTO budget_table values('201902', 2000);  
INSERT INTO budget_table values('201903', 1500);  
INSERT INTO budget_table values('201904', 3000);  
INSERT INTO budget_table values('201905', 1050);
```

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

· 매출 테이블 (sale_table)

```
create table sale_table (  
    yearmon    VARCHAR2(6),  
    sale_amt   NUMBER    );
```

```
INSERT INTO sale_table values('201901', 900);  
INSERT INTO sale_table values('201902', 2000);  
INSERT INTO sale_table values('201903', 1000);  
INSERT INTO sale_table values('201904', 3100);  
INSERT INTO sale_table values('201905', 800);
```

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT *  
FROM buget_table;
```

```
SELECT *  
FROM sale_table;
```

	YEARMON	BUDGET_AMT
1	201901	1000
2	201902	2000
3	201903	1500
4	201904	3000
5	201905	1050

	YEARMON	SALE_AMT
1	201901	900
2	201902	2000
3	201903	1000
4	201904	3100
5	201905	800

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

* 예산 대비 실적은?

년월	예산	실적	달성율
201901	1000	1000	100%
201902	2000	1000	50%
201903	3000	1700	56%

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT yearmon, budget_amt, 0 sale_amt
  FROM budget_table
UNION
SELECT yearmon, 0 budget_amt, sale_amt
  FROM sale_table
ORDER BY 1;
```

	YEARMON	BUDGET_AMT	SALE_AMT
1	201901	0	900
2	201901	1000	0
3	201902	0	2000
4	201902	2000	0
5	201903	0	1000
6	201903	1500	0
7	201904	0	3100
8	201904	3000	0
9	201905	0	800
10	201905	1050	0

1. 집합연산자는 언제 사용할까?

(1.1) UNION (ALL) – 계획 대비 실적

```
SELECT yearmon,  
       SUM(budget_amt) budget,  
       SUM(sale_amt) sale,  
       ROUND(SUM(sale_amt) / SUM(budget_amt),2) * 100 rates  
FROM ( SELECT yearmon, budget_amt, 0 sale_amt  
      FROM budget_table  
      UNION  
      SELECT yearmon, 0 budget_amt, sale_amt  
      FROM sale_table  
      )  
  
GROUP BY yearmon  
ORDER BY 1;
```

	YEARMON	BUDGET	SALE	RATES
1	201901	1000	900	90
2	201902	2000	2000	100
3	201903	1500	1000	67
4	201904	3000	3100	103
5	201905	1050	800	76

1. 집합연산자는 언제 사용할까?

(1.2) UNION (ALL) – 로우를 컬럼으로

```
CREATE TABLE test_score (
```

```
  years  VARCHAR2(4),
```

```
  gubun  VARCHAR2(20),
```

```
  korean NUMBER,
```

```
  english NUMBER,
```

```
  math   NUMBER );
```

```
INSERT INTO test_score VALUES ('2019', '중간고사', 92, 87, 67);
```

```
INSERT INTO test_score VALUES ('2019', '기말고사', 88, 80, 91);
```

```
SELECT *
```

```
FROM test_score;
```

	YEARS	GUBUN	KOREAN	ENGLISH	MATH
1	2019	중간고사	92	87	67
2	2019	기말고사	88	80	91

1. 집합연산자는 언제 사용할까?

(1.2) UNION (ALL) – 로우를 컬럼으로

```
SELECT years, gubun, '국어' subject, korean score  
FROM test_score
```

```
UNION ALL
```

```
SELECT years, gubun, '영어' subject, english score  
FROM test_score
```

```
UNION ALL
```

```
SELECT years, gubun, '수학' subject, math score  
FROM test_score  
ORDER BY 2 desc;
```

	YEARS	GUBUN	SUBJECT	SCORE
1	2019	중간고사	국어	92
2	2019	중간고사	영어	87
3	2019	중간고사	수학	67
4	2019	기말고사	영어	80
5	2019	기말고사	수학	91
6	2019	기말고사	국어	88

1. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

SELECT *

FROM locations;

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA
1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon	CA
2000	40-5-12 Laogianggen	190518	Beijing	(null)	CN
2100	1298 Vileparle (E)	490231	Bombay	Maharashtra	IN
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2300	198 Clementi North	540198	Singapore	(null)	SG
2400	8204 Arthur St	(null)	London	(null)	UK
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK
2600	9702 Chester Road	09629850293	Stretford	Manchester	UK
2700	Schwanthalerstr. 7031	80925	Munich	Bavaria	DE

1. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

```
SELECT state_province dup_loc_name
FROM locations
INTERSECT
SELECT city
FROM locations
ORDER BY 1;
```

	DUP_LOC_NAME
1	Oxford
2	Sao Paulo
3	Utrecht

1. 집합연산자는 언제 사용할까?

(1.3) INTERSECT

```
SELECT state_province, city
FROM locations
WHERE state_province = city
ORDER BY 1;
```

	STATE_PROVINCE	CITY
1	Oxford	Oxford
2	Sao Paulo	Sao Paulo
3	Utrecht	Utrecht

2. 프로시저 (Procedure)

- 오라클에서 제공하는 SubProgram ➔ 사용자 정의 함수, 프로시저
- 함수는 값을 반환하지만, 프로시저는 반환하지 않음
- 프로시저는 자주 사용되는 작업을 구현해 놓고 필요할 때마다 호출
- 프로시저는 주로 원천 테이블을 읽어 대상 테이블에 로직을 적용해
INSERT, UPDATE, DELETE, MERGE 문을 사용해 타겟 테이블에 데이터 적재
- 예) 새로운 부서 입력, 삭제, 수정 로직 구현

2. 프로시저 (Procedure)

- 부서등록 프로시저 생성 → create_dept

```
CREATE OR REPLACE PROCEDURE create_dept_p ( p_dept_id NUMBER,  
                                             p_dept_nm VARCHAR2,  
                                             p_man_id  NUMBER,  
                                             p_loc_id  NUMBER )
```

IS

BEGIN

```
    INSERT INTO departments (department_id, department_name, manager_id, location_id)  
    VALUES ( p_dept_id, p_dept_nm, p_man_id, p_loc_id );  
    COMMIT;
```

EXCEPTION WHEN OTHERS THEN

```
    RAISE_APPLICATION_ERROR (-20205, SQLERRM);  
    ROLLBACK;
```

END;

2. 프로시저 (Procedure)

· 부서등록 프로시저 호출

```
EXEC CREATE_DEPT_P ( 300, 'TEST_DEPT', NULL, 1700);
```

```
SELECT *  
FROM DEPARTMENTS;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	300	TEST_DEPT	(null)	1700
2	10	Administration	200	1700
3	20	Marketing	201	1800
4	30	Purchasing	114	1700
5	40	Human Resources	203	2400
6	50	Shipping	121	1500
7	60	IT	103	1400
8	70	Public Relations	204	2700
9	80	Sales	145	2500
..	--	.	---	----

2. 프로시저 (Procedure)

· 부서등록 프로시저 호출

EXEC CREATE_DEPT_P (10, 'TEST_DEPT', NULL, 1700);

명령의 4 행에서 시작하는 중 오류 발생 -

```
BEGIN CREATE_DEPT_P ( 10, 'TEST_DEPT', NULL, 1700); END;
```

오류 보고 -

ORA-20205: ORA-00001: 무결성 제약 조건 (HR.DEPT_ID_PK) 에 위배됩니다

ORA-06512: "HR.CREATE_DEPT_P", 13행

ORA-06512: 1행

2. 프로시저 (Procedure)

- 부서등록 프로시저 수정 및 실행

Create_dept_p(수정본1).sql 파일 참조 (메모장에서 열어도 됨)

EXEC CREATE_DEPT_P (10, 'TEST_DEPT', NULL, 1700);

명령의 4 행에서 시작하는 중 오류 발생 -

```
BEGIN CREATE_DEPT_P ( 10, 'TEST_DEPT', NULL, 1700); END;
```

오류 보고 -

ORA-20205: ORA-20201: 10번 부서가 이미 존재합니다

ORA-06512: "HR.CREATE_DEPT_P", 25행

ORA-06512: 1행

2. 프로시저 (Procedure)

- 부서등록 프로시저 수정 및 실행

EXEC CREATE_DEPT_P (301, 'TEST_DEPT', NULL, 9000);

명령의 7 행에서 시작하는 중 오류 발생 -

```
BEGIN CREATE_DEPT_P ( 301, 'TEST_DEPT', NULL, 9000); END;
```

오류 보고 -

ORA-20205: ORA-02291: 무결성 제약조건 (HR.DEPT_LOC_FK) 이 위배되었습니다- 부모 키가 없습니다

ORA-06512: "HR.CREATE_DEPT_P", 25행

ORA-06512: 1행

2. 프로시저 (Procedure)

- 부서등록 프로시저 수정 및 실행

Create_dept_p(수정본2).sql 파일 참조 (메모장에서 열어도 됨)

EXEC CREATE_DEPT_P (301, 'TEST_DEPT', NULL, 9000);

명령의 7 행에서 시작하는 중 오류 발생 -

```
BEGIN CREATE_DEPT_P ( 301, 'TEST_DEPT', NULL, 9000); END;
```

오류 보고 -

```
ORA-20205: ORA-20201: 9000가 locations 테이블에 없습니다
```

```
ORA-06512: "HR.CREATE_DEPT_P", 39행
```

```
ORA-06512: 1행
```

2. 프로시저 (Procedure)

- 부서등록 프로시저 수정 및 실행

```
EXEC CREATE_DEPT_P ( 301, 'TEST_DEPT', NULL, 1700);
```

```
SELECT *  
FROM DEPARTMENTS;
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
300	TEST_DEPT	(null)	1700
301	TEST_DEPT	(null)	1700
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700

3. 테이블스페이스(데이터파일) 암호화

- 2018년 11월 매리어트 호텔 예약 데이터베이스 해킹 사건 발생
- 500만개의 여권번호 해킹됨
- 문제는...
암호화가 안되어 있었음

*Marriott Concedes 5 Million
Passport Numbers Lost to
Hackers Were Not Encrypted*



A Marriott hotel in Chicago. The company revealed in late November that its Starwood guest reservation database was hacked. The number of records affected has been revised downward, to about 383 million customers, but is still the biggest breach of customer data in history. Scott Olson/Getty Images

3. 테이블스페이스(데이터파일) 암호화

- 암호화 되지 않은 테이블스페이스(ora_tb)에 테이블 생성 및 데이터 저장

```
create table no_encrypt (  
    id      number,  
    user_name  varchar2(100),  
    emails    varchar2(100),  
    password  varchar2(100)  
);  
  
insert into no_encrypt values (1, '홍길동', 'gildongHong@gmail.coms', '1234');  
insert into no_encrypt values (2, '김유신', 'youshinkim@gmail.coms', 'dielw%$@90');  
insert into no_encrypt values (3, '강감찬', 'kangkamchan@daum.coms', 'dowkfiwe@#%_ew');  
commit;
```


3. 테이블스페이스(데이터파일) 암호화

- 해커가 해당 데이터 파일 획득(ORA_TB.DBF)
- 리눅스 기본 명령어인 grep으로 해당 파일에서 이메일 추출
grep -i "@gmail" ORA_TB.DBF -a

[illegible]

3. 테이블스페이스(데이터파일) 암호화

- 테이블스페이스 암호화(ora_en_tb)

```
CREATE TABLESPACE ora_en_tb datafile  
'D:\app\hghong\product\18.0.0\oradata\XE\XEPDB1\ORA_EN_TB.DBF'  
SIZE 10M AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED  
ENCRYPTION USING 'AES256' DEFAULT STORAGE(ENCRYPT)  
;
```

3. 테이블스페이스(데이터파일) 암호화

- 암호화 된 테이블스페이스(ora_en_tb)에 테이블 생성 및 데이터 저장

```
create table encrypted (
```

```
    id      number,
```

```
    user_name  varchar2(100),
```

```
    emails     varchar2(100),
```

```
    password   varchar2(100)
```

```
) TABLESPACE ORA_EN_TB ;
```

```
insert into no_encrypt values (1, '홍길동', 'gildongHong@gmail.coms', '1234');
```


```
insert into no_encrypt values (2, '김유신', 'youshinkim@gmail.coms', 'dielw%$@90');
```

```
insert into no_encrypt values (3, '강감찬', 'kangkamchan@daum.coms', 'dowkfiwe@#%_ew');
```

```
commit;
```

3. 테이블스페이스(데이터파일) 암호화

- 해커가 해당 데이터 파일 획득(ORA_EN_TB.DBF)
- 리눅스 기본 명령어인 grep으로 해당 파일에서 이메일 추출
`grep -i "@gmail" ORA_EN_TB.DBF -a`

 MINGW64:/d/app/hghong/product/18.0.0/oradata/XE/XEPDB1

```
hghong@DESKTOP-6V9D0MB MINGW64 ~  
$ cd /d/app/hghong/product/18.0.0/oradata/XE/XEPDB1  
  
hghong@DESKTOP-6V9D0MB MINGW64 /d/app/hghong/product/18.0.0/oradata/XE/XEPDB1  
$ grep -i "@gmail" ORA_EN_TB.DBF -a  
  
hghong@DESKTOP-6V9D0MB MINGW64 /d/app/hghong/product/18.0.0/oradata/XE/XEPDB1  
$
```

4. 보너스 문제

- GROUPBYMULTIPLY 테이블 생성

```
create table GROUPBYMULTIPLY (  
    department_name VARCHAR2(100),  
    num_data        NUMBER  
);
```

4. 보너스 문제

· GROUPBYMULTIPLY 테이블 데이터 입력

```
insert into groupbymultiply values ('dept1', 10);  
insert into groupbymultiply values ('dept1', 20);  
insert into groupbymultiply values ('dept1', 30);  
insert into groupbymultiply values ('dept2', 5);  
insert into groupbymultiply values ('dept2', 7);  
insert into groupbymultiply values ('dept2', 40);  
insert into groupbymultiply values ('dept3', 69);  
insert into groupbymultiply values ('dept3', 71);  
insert into groupbymultiply values ('dept3', 12);  
  
commit;
```

4. 보너스 문제

· GROUPBYMULTIPLY 테이블 조회

```
SELECT *  
FROM groupbymultiply;
```

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

4. 보너스 문제

- GROUPBYMULTIPLY 테이블 집계

```
SELECT department_name,  
       SUM(num_data)  
FROM groupbymultiply  
GROUP BY department_name  
ORDER BY 1 ;
```

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

	DEPARTMENT_NAME	SUM(NUM_DATA)
1	dept1	60
2	dept2	52
3	dept3	152

4. 보너스 문제

· GROUPBYMULTIPLY 테이블 문제

Department_name 별로 num_data 컬럼 값을 더하는 것이 아니라 곱한 결과를 조회하는 쿼리를 작성하시오.

	DEPARTMENT_NAME	NUM_DATA
1	dept1	10
2	dept1	20
3	dept1	30
4	dept2	5
5	dept2	7
6	dept2	40
7	dept3	69
8	dept3	71
9	dept3	12

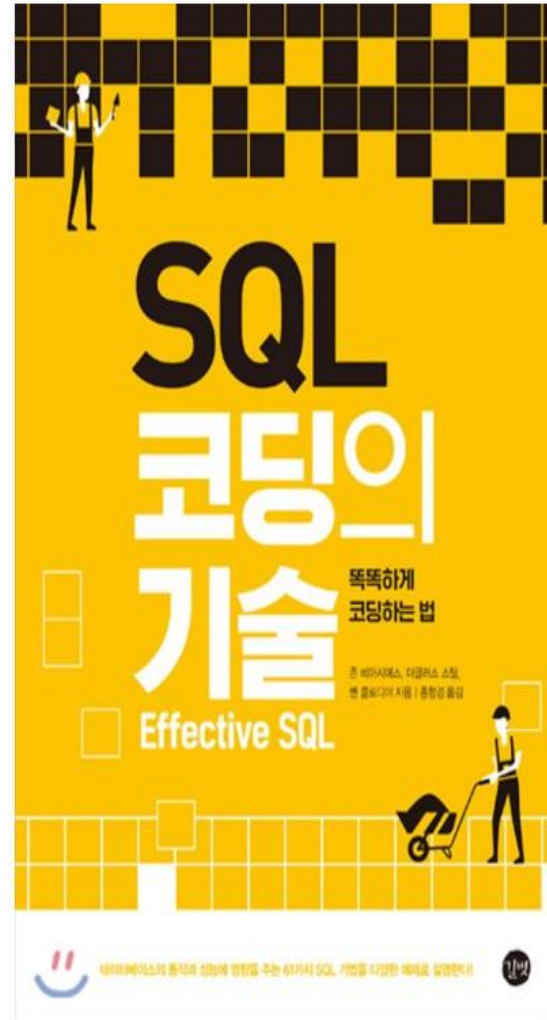
	DEPARTMENT_NAME	MULTIPLY_RESULT
1	dept1	6000
2	dept2	1400
3	dept3	58788

4. 보너스 문제

- GROUPBYMULTIPLY 테이블 문제
- 정답 맞추신 3분 선정해 책 선물
- 정답자가 3명 이상인 경우는 가위,바위,보
- 정답자가 3명 미만인 경우는 제출 쿼리를 보고 제가 판단해 선정
- 선정하기 어려울 경우에도 가위,바위,보

4. 보너스 문제

· GROUPBYMULTIPLY 테이블 문제



소득공제

Effective SQL **SQL 코딩의 기술** 똑똑하게 코딩하는 법

존 비야시예스, 더글러스 스틸, 벤 클로디어 저/홍형경 역 | 길벗 | 2017년 11월 30일

★★★★★ 10.0 회원리뷰(1건) | 판매지수 2322 **베스트** IT 모바일 top20 1주

정가 28,000원

판매가 **25,200원** (10% 할인)

YES포인트 1,400원 (5% 적립)

5만원이상 구매 시 2천원 추가적립

결제혜택 카드/간편결제 혜택을 확인하세요

eBook
22,400원

>

이 상품을 팔기
매입가 4,200원