데이터베이스 프로그래밍

SUBQUERY

학습 목표

- Subquery 로 다중 테이블 검색 가능
- Single Row Subquery 작성
- Multi Row Subquery 작성
- Multi Column Subquery를 작성하고 PAIRWISE 비교 가능
- Scalar, Correlated(상호연관) Subquery 활용 가능

학습할 내용

- Subquery 의 정의와 유형
- Single Row Subquery 및 Multi Row Subquery
- Multi Column Subquery
- Scalar Subquery
- Correlated Subquery

Subquery란?

- 하나의 SQL 문 안에 포함되어 있는 또 다른 SQL문
- 알려지지 않은 기준을 이용한 검색을 위해 사용
- 일반적으로 Subquery가 먼저 실행되고, 그 결과를 사용하여 Mainquery에 대한 질의 조건 완성

Subquery란?

- 형식

```
SELECT column_list FROM table
WHERE expr operator (SELECT column_list FROM table
                     WHERE 조건식)
```

- 설명
  - 조인은 조인에 참여하는 모든 테이블이 대등한 관계에 있기 때문에 조인에 참여하는 모든 테이블의 칼럼을 어느 위치에서라도 자유롭게 사용 가능
  - subsquery는 mainquery의 column 모두 사용 가능하지만 mainquery는 subquery의 column 사용 불가능

SUBQUERY의 종류

종류	설 명
단일 행 서브쿼리 (Single Row Subquery)	· SELECT 문장으로부터 오직 하나의 행만을 검색하는 질의어
다중 행 서브쿼리 (Multi Row Subquery)	· SELECT 문장으로부터 하나 이상의 행을 검색하는 질의어
다중 열 서브쿼리 (Multi Column Subquery)	· SELECT 문장으로부터 하나 이상의 컬럼을 검색하는 질의어
상관관계 서브쿼리 (Correlated Subquery)	· 바깥쪽 쿼리의 컬럼 값이 안쪽 서브쿼리 수행 시 이용되어 수행하는 방식 · LOOP처럼 사용됨 · main query의 개수와 관련
INLINE VIEW	· FROM 절 상에 오는 서브쿼리로 VIEW 처럼 작용

Subquery의 사용

- 포함 구문
  - SELECT 절, FROM 절, WHERE 절, HAVING 절, ORDER BY 절
  - UADATE 문의 SET 절
  - INSERT 문의 INTO절
- 주의사항
  - subquery를 괄호로 감싸서 사용
  - 가독성을 위해 비교 조건의 오른쪽에 subquery 사용
  - 일반적인 subquery에서는 ORDER BY 절을 사용하지 않음
  - subquery는 단일 행(Single Row) 또는 복수 행(Multi Row) 비교연산자와 함께 사용 가능
  - 단일 행 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하이어야 하고, 복수 행 비교 연산자는 서브쿼리의 결과 건수와 상관없음

Subquery의 사용 예

- **employee_id가 120인 사원의 job_id와 동일한 job_id를 가진 사원들의 이름과 job_id 조회**

```
SELECT job_id
FROM employees
WHERE employee_id=120;
```

ST_MAN

```
SELECT first_name, job_id
FROM employees
WHERE job_id='ST_MAN';
```

SQL | 인출된 모든 행: 5

	FIRST_NAME	JOB_ID
1	Matthew	ST_MAN
2	Adam	ST_MAN
3	Payam	ST_MAN
4	Shanta	ST_MAN
5	Kevin	ST_MAN

단일행(Single_Row) 서브쿼리

서브쿼리에서 단 하나의 행을 메인쿼리로 반환

```
SELECT first_name, job_id FROM employees
WHERE job_id = ( SELECT job_id FROM employees
                 WHERE employee_id=120 );
```

다중 행 서브쿼리

- 서브쿼리가 여러 행을 반환
- 여러 행을 비교하기 위해 다음 연산자 이용

연산자	설 명
IN	· 목록에 있는 임의의 값과 동일하면 참
ANY	· 서브쿼리에서 반환된 각각의 값과 비교하여 하나라도 참이면 참 · < ANY는 최대값보다 작음 < MAX() · > ANY는 최소값보다 큼 > MIN() · = ANY는 IN과 동일
ALL	· 서브쿼리에서 반환된 모든 값과 비교하여 모두 참이어야 참 · < ALL은 최소값보다 작음 < MIN() · > ALL은 최대값보다 큼 > MAX()

다중 행 서브쿼리 예

```
SELECT employee_id, first_name, salary, department_id
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees GROUP BY department_id)
ORDER BY department_id;
```

ERROR

- 이유

```
SELECT MAX(salary) FROM employees GROUP BY department_id
ORDER BY department_id;
```

	MAX(SALARY)
1	4400
2	13000
3	11000
4	6500
5	8200
6	9000
7	10000
8	14000
9	24000
10	12008
11	12008
12	7000

- 해결

```
SELECT employee_id, first_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MAX(salary) FROM employees GROUP BY department_id)
ORDER BY department_id;
```

다중 열 서브쿼리 (MULTI COLUMN SUBQUERY)

- 서브쿼리의 결과값이 두 개 이상의 컬럼을 반환
- 비교 방식
  - NON-PAIRWISE 비교
  - PAIRWISE 비교
    - WHERE 절에서 비교하는 컬럼이 하나가 아니라, 여러 개의 컬럼을 동시에 비교하는 경우

Diana, Adam과 관리자 및 부서가 같은 사원의 이름, 관리자번호, 부서번호 조회

```
SELECT first_name, manager_id, department_id
FROM employees
WHERE first_name NOT IN ('Diana', 'Adam')          PAIRWISE subquery
    AND (manager_id, department_id) IN (SELECT manager_id, department_id
                                        FROM employees
                                        WHERE first_name IN ('Diana', 'Adam'));
```

SQL | 인출된 모든 행: 7(0.031초)

	FIRST_NAME	MANAGER_ID	DEPARTMENT_ID
1	Matthew	100	50
2	Payam	100	50
3	Shanta	100	50
4	Kevin	100	50
5	Bruce	103	60
6	David	103	60
7	Valli	103	60

상호 연관 서브쿼리 (CORRELATED SUBQUERY)

- 상위 질의에 있는 테이블의 컬럼을 참조

```
SELECT column1, column2,...
FROM table1 main
WHERE column1 operator (SELECT column1 FROM table2 WHERE expr1=main.expr2);
```

- 메인 쿼리의 한 ROW에 대해서 서브쿼리가 한 번씩 실행
- 테이블에서 행을 읽어서 각 행의 값을 관련된 데이터와 비교하는 방법 중 하나
- 기본 질의에서 고려된 각 후보 행에 대해 서브쿼리가 다른 결과를 반환해야 하는 경우에 사용
- 각 행의 값에 따라 응답이 달라지는 다중 질의에 응답할 때 사용
- 서브 쿼리에서 메인 쿼리의 컬럼명을 사용할 수 있으나, 메인 쿼리에서는 서브쿼리의 컬럼명을 사용할 수 없다.

상호 연관 서브쿼리 (CORRELATED SUBQUERY)

- 실행
  - 후보 행을 가져온다(메인쿼리에서 추출)
  - 후보 행의 값을 사용하여 서브쿼리를 실행
  - 서브쿼리의 결과값을 사용하여 후보 행의 조건 확인
  - 후보 행이 남지 않을 때까지 반복

FROM절 서브쿼리

- FROM 절에 기술한 서브쿼리는 마치 뷰와 같은 역할 ➔ Inline View
- 인라인 뷰(inline view) : 동적 뷰(Dynamic View)
  - SQL이 실행될 때만 임시적으로 생성 → 동적 뷰
  - 데이터베이스에 해당 정보 저장되지 않음
  - 테이블 명이 올 수 있는 곳에 사용
  - 서브쿼리의 컬럼은 메인 쿼리에서 사용할 수 없으나, 인라인 뷰의 컬럼은 SQL 문에서 자유롭게 사용 가능

일반적인 뷰
→ 정적 뷰(Static View))

→ 동적으로 생성된 테이블로 조인방식과 유사하므로

```
SELECT column1, column2,...
FROM (SELECT column1, column2,... FROM table1 WHERE column1=... )
WHERE 조건식
```

TOP-N 서브쿼리 1

- employees 테이블 중 급여가 가장 작은 5명의 이름, 급여 조회

```
SELECT rownum, first_name, salary   --3
FROM employees                      --1
WHERE rownum < 6                    --2
ORDER BY salary;                    --4
```

SQL | 인출된 모든 행: 5(0.001초)

ROWNUM	FIRST_NAME	SALARY
1	An%%dy	2400
2	M%ary	3400
5	Neena	17000
4	Steven	24000
3	Mickey	(null)

SELECT * FROM employees ORDER BY salary;

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
132	TJ	Olson	TJOLSON	650.124.8234	07/04/10	ST_CLERK	2100	(null)	121	50
128	Steven	Markle	SMARKLE	650.124.1434	08/03/08	ST_CLERK	2200	(null)	120	50
136	Hazel	Philtanker	HPHILTAN	650.127.1634	08/02/06	ST_CLERK	2200	(null)	122	50
208	An%%dy	Chaplin	ChaplinMAIL	515.135.9876	12/03/27	FI_ACCOUNT	2400	(null)	100	30
127	James	Landry	JLANDRY	650.124.1334	07/01/14	ST_CLERK	2400	(null)	120	50
135	Ki	Gee	KGEE	650.127.1734	07/12/12	ST_CLERK	2400	(null)	122	50
119	Karen	Colmenares	KCOLMENA	515.127.4566	07/08/10	PU_CLERK	2500	(null)	114	30
131	James	Marlow	JAMRLOW	650.124.7234	05/02/16	ST_CLERK	2500	(null)	121	50
140	Joshua	Patel	JPATEL	650.121.1834	06/04/06	ST_CLERK	2500	(null)	123	50
144	Peter	Vargas	PVARGAS	650.121.2004	06/07/09	ST_CLERK	2500	(null)	124	50

  - 5명을 추출하여
  - 추출된 5명을 키 순으로 정렬

TOP-N 서브쿼리 2

- **employees 테이블 중 급여가 가장 작은 5명의 이름, 급여 조회**
  - **키 순으로 정렬 → inline view로 작성**
  - **위부터 5개의 data 추출**

```
SELECT rownum, first_name, salary
FROM (SELECT first_name, salary FROM employees ORDER BY salary) emp
WHERE rownum < 6;
```

SCALAR 서브쿼리

- 하나의 행에서 하나의 컬럼 값만 반환하는 서브쿼리
- SCALAR 서브쿼리의 값은 서브쿼리의 SELECT 목록에 있는 항목 값
- 서브쿼리가 0개의 행을 반환하면 SCALAR 서브쿼리의 값은 NULL
- 서브쿼리가 2개 이상의 행을 반환하는 경우 오류 반환
- 사용하는 경우
  - SELECT 문
  - INSERT 문의 VALUES 목록
  - DECODE 및 CASE의 조건 및 표현식 부분
  - GROUP BY를 제외한 SELECT 의 모든 절
  - UPDATE 문의 SET 절 및 WHERE 절에서 연산자 목록

사원번호, 이름, IT구분(지역번호가 1400인 부서와 같은 부서번호를 가진 경우 'IT' 그렇지 않은 경우 'NON_IT' 출력)

```
SELECT employee_id, first_name, (CASE WHEN department_id=
                                    (SELECT department_id
                                     FROM departments
CASE 표현식에 스칼라 서브쿼리 사용          WHERE location_id=1400) "IT구분"
                                  THEN 'IT' ELSE 'NON_IT' end)
FROM employees;
```

SQL | 인출된 모든 행: 111(0.009초)

EMPLOYEE_ID	FIRST_NAME	IT구분
100	Steven	NON_IT
101	Neena	NON_IT
102	Lex	NON_IT
103	Alexander	IT
104	Bruce	IT
105	David	IT
106	Valli	IT
107	Diana	IT
108	Nancy	NON_IT
109	Daniel	NON_IT
110	John	NON_IT
111	Ismael	NON_IT
112	Jose Manuel	NON_IT
113	Luis	NON_IT
114	Den	NON_IT
203	Susan	NON_IT
204	Hermann	NON_IT
205	Shelley	NON_IT
206	William	NON_IT
207	JOHN	NON_IT
208	An%%dy	NON_IT
209	M%ary	NON_IT
210	Mickey	NON_IT

EXISTS 연산자

- 부분 범위 체크 처리
- 외부 질의에서 검색된 값이 서브쿼리의 결과집합에 존재하는지 여부를 검사하기 위한 상호 연관 서브쿼리에 자주 사용됨
- 서브쿼리가 한 행이상 반환하면 true를 반환하고, 해당 값이 없으면 false 반환
  - 서브쿼리 행의 값이 발견되는 경우
    - 서브쿼리에서 더 이상 검색하지 않음
    - 조건 flag가 true가 됨
- NOT EXISTS는 외부 질의에서 검색된 값이 내부 질의결과에 존재하지 않으면 true

소속 사원이 존재하는 부서의 부서번호, 부서명 조회

```
SELECT department_id, department_name FROM departments dept
WHERE EXISTS (SELECT 'A' FROM employees
                WHERE department_id=dept.department_id);
```

SQL | 인출된 모든 행: 11(0.005초)

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
110	Accounting

WITH 구문

- 질의 블록을 정의하여 질의에서 사용 가능
- 복합 질의에서 여러 번 발생하는 같은 징의 블록을 SELECT 문에서 재사용 가능
- 같은 질의 블록을 여러 번 참조하거나 조인 및 집계를 해야 하는 경우 매우 유용
- 오라클 서버는 질의 블록의 결과를 실행하여 사용자의 임시 테이블 스페이스에 저장(반복 참조 시 성능 향상)
- 이점
  - 질의를 읽기 쉽게 만듦
  - 해당 절이 질의에서 여러 번 사용될지라도 한 번만 실행하므로 성능 향상