Wireless Software Laboratory

PostgreSQL SQL Fundamental 11

wisoft

차례

Aggregation & Grouping

Subqueries

Views

Join Method

Set Theory

http://me12.tt.co.kr/~fabis70/wp-content/uploads/2014/02/lapalette-package-2.jpg

Aggregation & Grouping

Aggregation

- Abstract

Function	Description
AVG	Calculates the average value in a column
COUNT	Determines the number of rows in a table
MAX	Determines the maximum value in a column
MIN	Determines the minimum value in a column
SUM	Calculates a total of the values in a column

- Calculate the total, minimum, maximum, count, and average of the ctime.

```
SELECT SUM(CTIME), MIN(CTIME), MAX(CTIME), COUNT(CTIME), AVG(CTIME)
  FROM COURSE;
```

Grouping

- Creates groups of rows that share common characteristics
- Calculations in the SELECT command are performed for the entire group

  - 학과별 교수 인원을 조회하라.

```
SELECT PDEPT, COUNT(*)
  FROM PROFESSOR
 GROUP BY PDEPT;
```

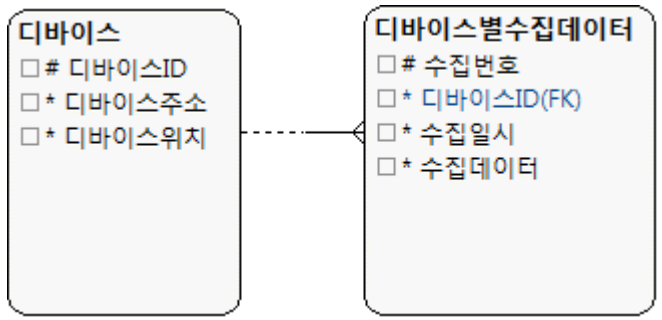
Processing

Build Environment

- Run Scripts

```
SQL> \i c:/sql/devices.sql;
```

- Query




```
SQL> \d DEVICE;

SQL> \d DEVICE_DATA;

SQL> SELECT * FROM DEVICE;

SQL> SELECT * FROM DEVICE_DATA;
```

연습문제 (1/3)

- DEVICE ID별로 수집된 데이터의 개수, 총합, 평균을 조회하라.

질의 결과

DD_DEVICE_ID	개수	총합	평균
11111	11	149	13.55
22222	11	273	24.82
33333	11	387	35.18
44444	11	501	45.55
55555	11	605	55

연습문제 (2/3)

- C501, 503에서 수집된 데이터를 장치 아이디 별로 합계와 평균을 조회하시오.

질의 결과

DD_DEVICE_ID	개수	총합	평균
11111	11	149	13.55
33333	11	387	35.18

연습문제 (3/3)

- 장치 아이디 별로 수집한 데이터의 총합이 280 이상인 디바이스ID, 디바이스 위치, 데이터의 총합과 평균을 조회하시오.

질의 결과

DEVICE_ID	DEVICE_LOC	총합	평균
33333	C503	387	35.18
44444	C504	501	45.55
55555	C505	605	55

Distinct

Overview

- Remove duplicate rows
- count() function to determine the number of members in each group
    - Quiz - 몇 개의 디바이스에서 데이터가 수집되었는가?

NULL

Overview

- NULL
  - missing information and inapplicable information
  - 데이터 값이 존재하지 않는다는 것을 지시하는데 사용하는 특별한 표시어

- Query Expression

```
-- NULL OPERATION
SELECT NULL = NULL, NULL <> NULL, 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;


-- True이면 ‘t’, False이면 ‘f’를 RETURN
SELECT NULL IS NULL, 1 IS NULL, 1 IS NOT NULL;
```

http://www.bigfootcmms.com/wp-content/uploads/SQL_query.jpg

Sub Queries

Subqueries

are a powerful tool that you can use in all four SQL data statements

Overview

- Subquery
  - is a query contained within another SQL statement.
  - is always enclosed within parentheses.
  - is usually executed prior to the containing statement.

- 운영체제의 시수와 같거나 큰 과목을 조회하라.

```
SELECT C1.CNAME
  FROM COURSE C1, COURSE C2
 WHERE C1.CTIME >= C2.CTIME
   AND C2.CNAME = 'OS';
```

```
SELECT CNAME
  FROM COURSE
 WHERE CTIME >= (SELECT CTIME
                   FROM COURSE
                  WHERE CNAME = 'OS');
```

Operators

- Single Row
  - ‘=’, ‘>’, ‘>=’, ‘<’, ‘<=’, ‘<>’, ‘!=’
- Multiple Row
  - IN, NOT IN, ANY, ALL, EXISTS

Types (1/2)

- Single-Row / Multiple-Row Subquery
  - Single-Row: SELECT 문장에서 오직 하나의 행을 검색하는 질의
  - Multiple-Row: SELECT 문장에서 하나 이상의 행을 검색하는 질의

- Single-Column / Multiple-Column Subquery
  - Single-Column: SELECT 문장에서 오직 하나의 컬럼을 검색하는 질의
  - Multiple-Column: SELECT 문장에서 하나 이상의 컬럼을 검색하는 질의

Types (2/2)

- INLINE VIEW
  - FROM 절에서 사용하는 Subquery
  - VIEW 형태로 동작하여 INLINE VIEW라고 함
- Noncorrelated Subqueries
  - 일반적인 Subquery
- Correlated Subqueries
  - 외부(밖) 쿼리의 컬럼 중 하나가 내부(Subquery)의 조건에 활용하는 방식

http://www.bizwatch.co.kr/files/2014/06/20/ce16d45b5b1baadcca66602a1f416e3017 4405.jpg

Views

Overview

- About View

  - Virtual Table

    is simply a mechanism for querying data

  - Unlike tables, views do not involve data storage

    you won't need to worry about views filling up your disk space.

View의 장점

- Advantages
  - 보안

    사용자에게 필요 없는 정보를 숨길 수 있음

  - 복잡한 질의를 간단한 명령으로 단순하게 사용할 수 있음

  - 데이터 모델(설계)이 변경되어도 기존 질의의 변경을 최소화할 수 있음

View

- Syntax

```
CREATE [OR REPLACE] VIEW view_name [(column_list)]
    AS select_statement
```

  - viewname

    생성할 뷰(View) 명을 입력

  - OR REPLACE

    기존에 동일한 이름의 뷰 명이 있으면 다시 생성함

  - column-list

    뷰를 구성하는 컬럼 목록을 지정

  - Whitepaper

    https://www.postgresql.org/docs/9.6/static/sql-createview.html

연습문제

- 박현주 교수님께서 강의하는 과목명, 강의 시수, 그리고 강의실을 조회하라.

Join Method

JOIN

KEY Element of the RDBMS

About Join

- About JOIN
  - 관계형 데이터베이스(Relational Database, RDB)의 꽃
  - 조인을 명확하게 이해하기 위해서는 관계형 데이터베이스에 대한 이해가 필요

- 조인이 왜 필요할까요?
  - 관계형 데이터베이스의 구조적 특징으로 말미암아 의미 있는 데이터의 집합으로 테이블이 구성되고, 각 테이블끼리는 관계(Relationship)를 가짐

    관계형 데이터베이스는 저장 공간의 효율성과 확장성이 향상
  - 서로 관계 있는 데이터가 여러 테이블로 나뉘어 저장되므로, 각 테이블에 저장된 데이터를 효과적으로 검색하기 위해 방법이 필요

    각 테이블 간 의미 있는 데이터(행)를 연결하는 데 활용되는 메커니즘

Why is the Join Method so important?

Key Point

- SQL 문장의 의미를 제대로 파악하라.
  - 잘못 사용하면 커다란 재앙이 뒤따른다.
- 조인 조건을 명확하게 제공해야 한다.
  - CROSS JOIN(Cartesian Product)이 발생할 가능성이 있음
- 조인을 적용한 후 반드시 테스트를 수행하여 검증하라.

Join Types

Join Types

- Category
  - CARTITION PRODUCT
  - INNER JOIN
  - OUTER JOIN
    - LEFT OUTER JOIN
    - RIGHT OUTER JOIN
    - FULL OUTER JOIN
  - …

Cartesian Product (1/2)

- 조인(JOIN)에 참여한 테이블들의 모든 데이터가 RETURN

  - 발생 이유

    JOIN 조건이 잘못된 경우

    JOIN 조건을 정의하지 않았을 때

    첫 번째 테이블의 모든 행과 두 번째 테이블의 모든 행이 조인되는 경우

  - 해결 방법

    테이블 개수가 N개라면, 적어도 **N-1개의 JOIN 조건을 질의 안에 포함**해야 함

Cartesian Product (2/2)

- CARTESIAN PRODUCT EXAMPLE

COL1	COL2
A	1
B	2
C	3
D	4

×

COL3
1
2

=

COL1	COL2	COL3
A	1	1
B	2	1
C	3	1
D	4	1
A	1	2
B	2	2
C	3	2
D	4	2

내부 조인

Inner Join

EQUI JOIN

- 가장 일반적인 조인 형태
- 둘 이상의 테이블에 존재하는 공통 컬럼의 값이 같은 것을 결과로 추출

```
SELECT *
  FROM A INNER JOIN B
    ON A.ID = B.ID;
```

```
SELECT *
  FROM A, B
 WHERE A.ID = B.ID;
```

Processing

Build Environment

- Run Scripts

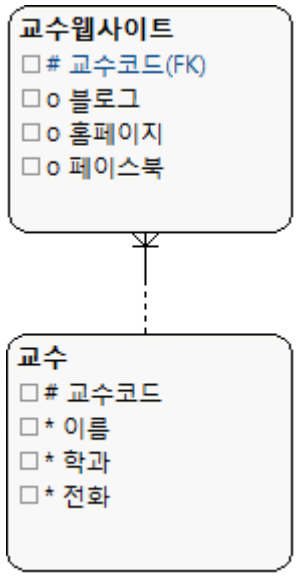
```
SQL> \i c:/sql/professor_website.sql;
```

- Query

```
SQL> \d PROF_WEBSITE;

SQL> SELECT * FROM PROF_WEBSITE;
```

연습문제



- 교수코드 P004의 교수이름, 블로그, 홈페이지, 페이스북을 조회하라.

```
SELECT   PROFESSOR.PNAME, PROF_WEBSITE.BLOG,
         PROF_WEBSITE.HOMEPAGE, PROF_WEBSITE.FACEBOOK
  FROM   PROFESSOR, PROF_WEBSITE
 WHERE   PROFESSOR.PCODE = PROF_WEBSITE.PWCODE
   AND   PROFESSOR.PCODE = 'P004';
```

```
SELECT   P.PNAME, PW.BLOG, PW.HOMEPAGE, PW.FACEBOOK
  FROM   PROFESSOR P, PROF_WEBSITE PW
 WHERE   P.PCODE = PW.PWCODE
   AND   P.PCODE = 'P004';
```

Outer Join (1/4)

- OUTER JOIN
  - LEFT OUTER JOIN (⟕)

    오른쪽 테이블(예제: B)에 조인할 컬럼의 값이 없는 경우에 사용함
  - RIGHT OUTER JOIN (⟖)

    왼쪽 테이블(예제: B에 조인할 컬럼의 값이 없는 경우에 사용함
  - FULL OUTER JOIN (⟗)

    양쪽 테이블 모두 OUTER JOIN이 필요할 때 사용함

Outer Join (2/4)

- LEFT OUTER JOIN

COL1	COL2
A	1
B	2
D	3
F	4
E	5

⋈

COL3	COL4
A	1
C	2
D	3
E	4

=

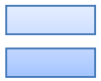
COL1	COL2	COL3	COL4
A	1	A	1
D	3	D	3
E	5	E	4
B	2		
F	4		

Outer Join (3/4)

- RIGHT OUTER JOIN

COL1	COL2
A	1
B	2
D	3
F	4
E	5

COL3	COL4
A	1
C	2
D	3
E	4



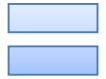
COL1	COL2	COL3	COL4
A	1	A	1
D	3	D	3
E	5	E	4
		C	2

Outer Join (4/4)

- FULL OUTER JOIN

COL1	COL2
A	1
B	2
D	3
F	4
E	5

COL3	COL4
A	1
C	2
D	3
E	4

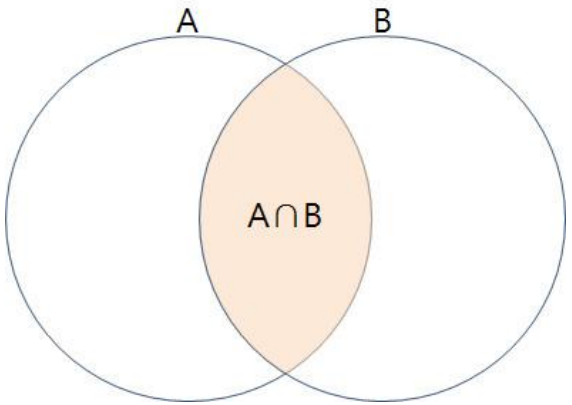


COL1	COL2	COL3	COL4
A	1	A	1
D	3	D	3
E	5	E	4
B	2		
F	4		
		C	2

Set Theory

INTERSECTION

- 두 집합 A와 B가 있을 때, A와 B에 모두 속하는 원소의 집합

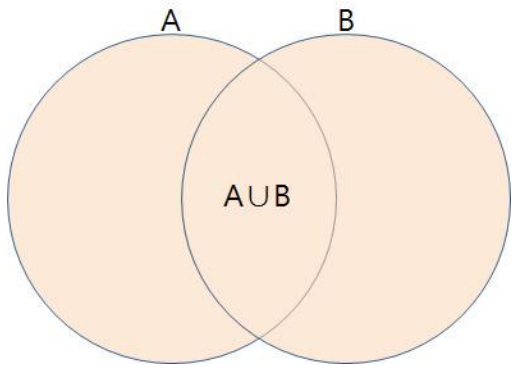


- Query Expression

```
SELECT *
  FROM A INNER JOIN B
    ON A.ID = B.ID
;
```

UNION

- 두 집합 A와 B가 있을 때, A에 속하거나 B에 속하는 원소의 집합
  - 두 집합의 중복값을 제거함

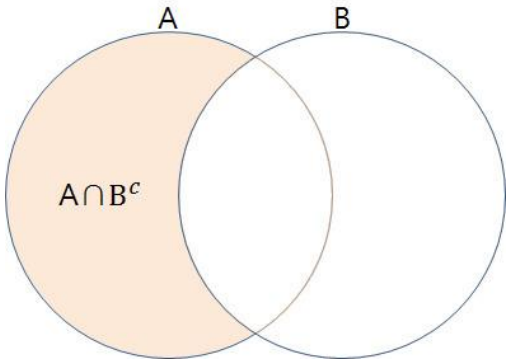


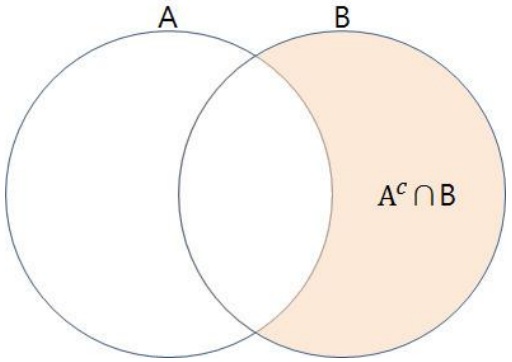
- Query Expression

```
SELECT *
  FROM A FULL OUTER JOIN B
    ON A.ID = B.ID
;
```

Relative Complement

- 집합에 포함되면서 다른 집합에 포함되지 않는 원소들의 집합
  - 어떤 집합의 원소에서 다른 집합의 원소를 뺀 집합





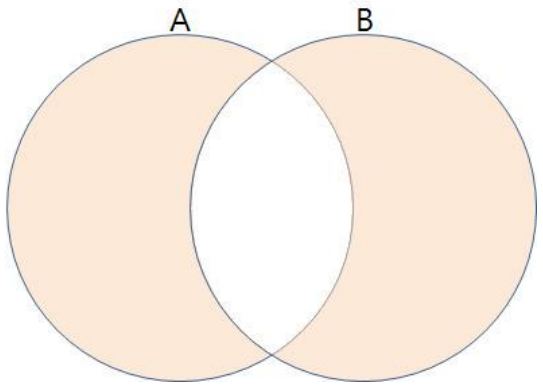
- Query Expression

```
SELECT *
  FROM A LEFT OUTER JOIN B
    ON A.ID = B.ID
 WHERE B.ID IS NULL;
```

```
SELECT *
  FROM A RIGHT OUTER JOIN B
    ON A.ID = B.ID
 WHERE A.ID IS NULL;
```

Symmetric difference

- 두 집합 A와 B 둘 중 하나에 포함되지만, 두 집합 모두에 포함되지 않는 원소들의 모임으로 간단하게, 두 차집합의 합집합이라고 볼 수 있음



- Query Expression

```
SELECT *
  FROM A FULL OUTER JOIN B
    ON A.ID = B.ID
 WHERE A.ID IS NULL OR B.ID IS NULL;
```

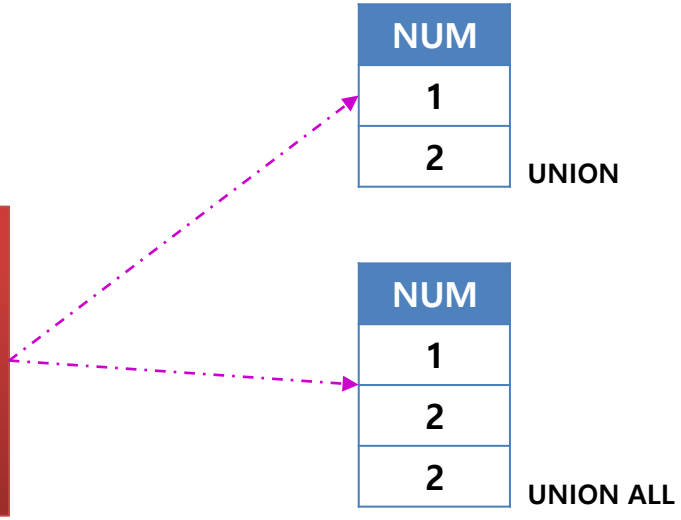
Set Operation

UNION

- UNION
  - 집합의 합집합을 추출
  - 중복을 제거함

- UNION ALL
  - 집합의 합집합을 추출
  - 중복을 제거하지 않음

```
SELECT 1 NUM UNION [ALL]
SELECT 2 UNION [ALL]
SELECT 2;
```



INTERSECT, EXCEPT

- INTERSECT
  - 집합의 교집합을 추출

```
SELECT 1 NUM INTERSECT SELECT 1;
```

- EXCEPT
  - 집합의 차집합을 추출

```
SELECT 1 NUM EXCEPT SELECT 1;
```

연습문제 (1/3)

- 데이터베이스를 강의하는 교수의 코드번호, 이름, 전화번호, 강의실을 조회하라.

질의 결과

교수코드	이름	전화번호	강의실
P004	박현주	821-1202	R004

연습문제 (2/3)

- 박현주 교수님께서 강의하는 과목명, 강의 시수, 그리고 강의실을 조회하라.

질의 결과

과목이름	강의시수	강의실
OS	3	R003
Database	4	R004

연습문제 (3/3)

- 학생이 수강 신청한 과목에 대해 학생이름, 학생전화번호, 과목이름, 강의시수, 강의실을 조회하라. 단, 조회결과는 학생이름의 오름차순, 과목이름의 내림차순으로 정렬하라.

질의 결과

학생이름	학생 전화번호	과목이름	강의시수	강의실
신동성	567-8901	TCP/IP	3	R001
신동성	567-8901	OS	3	R003
신동성	567-8901	Database	4	R004
이정민	678-9012	TCP/IP	3	R001
이정민	678-9012	OS	3	R003
이정민	678-9012	Database	4	R004

QnA

hyeonsig@hanbat.ac.kr

https://blog.ngelmaum.org

Thank You