

정성우_PostgreSQL소개 및 쿼리 작성

- 1. PostgreSQL 소개
- 2. PPAS 접속하기
- 3. Basic Selet
- 4. Table Joins
- 5. SubQuery
- 1. 그룹 함수, 다중열, 스칼라 SubQuery
- 2. Correlated SubQuery & 계층 쿼리
- 3. USING FUNCTION
- 4. 집합 연산자
- 1. DML, Truncate, Transaction
- 2. 대량 데이터 집합 조작
- 3. PARTITIONING

ADD

DROP

- 1. DDL

1. PostgreSQL 소개

1. 참고
 - 교육 자료 다운로드
 - 다음 전자 Wiki에서 "전자 개발 대상 데이터관련 교육 과정"로 검색
 - PostgreSQL(PPAS) 소개 및 쿼리 작성
2. RDBMS의 한 종류
3. ORDBMS
4. Open Source DBMS
5. 북미와 일본에서의 높은 인지도
6. 미국 버클리대학교에서 시작한 프로젝트
7. Version
 - PostgreSQL Community 9.3.1
 - Postgres Plus Advanced Server(PPAS) 9.2
8. 특징
 - PPAS는 Oracle과 유사한 syntax 기능 제공

2. PPAS 접속하기


1. User 생성
2. Tablespace 생성
 - 디렉토리는 생성되어 있어야 한다.
3. Database 생성
 - LC_COLLATE
 - 문자열 정렬 순서를 위한 로케일 설정을 위해 사용
 - LC_CTYPE
 - 문자 분류(알파벳, 숫자, 한글, 영문의 대소문자 등), 변환, 대소문자 비교를 위한 설정
4. Database 권한 부여
5. Schema 생성
 - 생성된 test 유저로 로그인
 - search_path 적용
6. Tools
 - GUI Tools for MAC
 - GUI Tools for Windows
7. pg_hba.conf
 - PPAC client 의 접속 관리 Configuration File
8. 실습

3. Basic Selet

1. 테이블 구조 확인
2. 연산자 사용, 우선순위
 - SELECT 절에 산술 연산자 사용 가능
 - ORACLE 과 우선 순위 동일
 - 우선순위 조절을 위한 괄호() 사용
3. NULL
 - ORACLE 과 동일
 - NULL 을 연산하면 결과를 NULL

4. ALIAS
 - ORACLE 과 동일
 - ALIAS에 공백, 대소문자 구분을 위해서는 "" 사용
5. DISTINCT, WHERE 절 조건 기술
 - ORACLE 과 동일
6. ORDER BY 동일
 - ORACLE 과 동일

4. Table Joins

1. CROSS JOIN
 - FROM 절에 기술된 테이블의 교차 곱을 생성
 - Caresian Product 라고도 함
 - 조인 연산이 없음
2. INNER JOIN
 - USING 사용(EQUIJOIN)
 - USING 에 사용된 칼럼의 중복은 제거됨
 - INNER 생략 가능
3. INNER JOIN
 - ON 사용 (EQUIJOIN)
 - ON 에 사용된 칼럼의 중복은 제거되지 않음
 - INNER 생략가능
4. INNER JOIN
 - ON 사용 (NON-EQUIJOIN)
 - ON 의 조건 연산자가 "=" 이 아닌 경우
 - INNER JOIN
 - NON-EQUIJOIN
5. INNER JOIN
 - 3개의 테이블 조인
6. LEFT JOIN (OUTER)
 - OUTER(LEFT) 테이블의 모든 행을 반환
 - 조건에 맞지 않는 데이터는 NULL 처리
 - ORACLE 과 동일한 문법으로 LEFT JOIN 
7. ON 과 WHERE 의 차이점
 - JOIN 의 ON 절은 조인 조건
 - WHERE 절은 조인 후 필터 조건으로 동작
 - OUTER JOIN ON 절에 기술
 - OUTER JOIN WHERE 절에 기술

5. SubQuery

1. SubQuery 란?
 - SELECT 절에 포함되는 SELECT 문
 - SubQuery 가 가능한 절
 - WHERE 절
 - FROM 절
 - HAVING 절
 - Operator 는 >, =, IN 등의 비교조건 포함
 - 단일행 연산자 (>, =, <)
 - 다중행 연산자 (IN, ALL , ANY)
2. SubQuery 유형
 - 단일행 SubQuery
 - SubQuery 에서 하나의 행만 반환
 - 다중행 SubQuery
 - SubQuery 에서 여러 행을 반환 (IN, ANY, ALL)
1. 그룹 함수, 다중열, 스칼라 SubQuery
2. Correlated SubQuery & 계층 쿼리
3. USING FUNCTION
4. 집합 연산자

1. 그룹 함수, 다중열, 스칼라 SubQuery

1. 그룹 함수
 - Group By 절을 이용해 그룹 당 하나의 결과를 반환
 - min, max, sum, count, avg
 - 한 개의 열로 Grouping
 - 두 개의 열로 Grouping
2. 그룹 함수, ROLLUP, CUBE
 - PPAS에서는 ROLLUP, CUBE, GROUPING 를 지원하지 않음

- 복제 테이블을 이용하여 "소계" 구하기
- 다중열 SubQuery
 - 앞서 단일행, 다중행 SubQuery에서는 단일열 사용
 - 다중열을 반환하는 SubQuery 가능
 - Scalar SubQuery
 - 단일행, 단일열 반환하는 SubQuery
 - Outer Table에 대한 참조 유무에 따른 분류
 - Uncorrelated SubQuery
 - Correlated SubQuery
 - Uncorrelated SubQuery
 - Correlated SubQuery
 - Outer Table에 대한 참조가 있어 모든 행에 대해 반복적 실행 필요
 - 대부분 Join으로 해결이 가능

2. Correlated SubQuery & 계층 쿼리

- Exists 연산자
 - SubQuery의 결과 집합에 행이 있는지 확인하는 역할
 - 하나 이상의 행을 반환하면 True, 그렇지 않다면 False
 - True가 반환되면 검색 중단
 - NOT EXISTS
 - NOT IN
- Correlated Update, Delete
 - Correlated Update
 - 한 테이블의 행을 기반으로 다른 테이블의 행을 Update
 - Correlated Delete
- With 절
 - 쿼리에서 동일한 Query 블록이 2회 이상 나타날 경우 With로 선언된 Query 블록을 그대로 사용
 - With로 선언된 블록은 메모리에 적재됨
- 계층 쿼리
 - RDBMS는 레코드를 계층적 방식으로 저장하지 않지만 테이블 ROW 사이에 계층적 관계가 있을 수 있음
 - START WITH CONNECT BY 절 이용하여 계층 데이터 조회
 - Oracle Syntax
 - LEVEL
 - Pseudo column으로 루트는 1, 하위형은 2, 3과 같은 값
 - FROM
 - 한 개의 테이블만 선택 가능
 - WHERE
 - 최종적으로 반환되는 행 필터
 - START WITH
 - 루트명을 지정, 조건절은 SubQuery 가능
 - CONNECT BY
 - 상, 하위의 PRIOR 행 사이의 관계가 있는 열 지정
 - 상, 하향식이 가능(컬럼의 위치에 따라 결정)
 - SubQuery 불가
 - PPAS Example
 - PPAS에서는 SELECT 절에 PRIOR을 사용할 수 없음
 - WITH RECURSIVE 이용

3. USING FUNCTION

- Numeric Functions
 - FLOOR
 - MOD
 - ROUND
 - ABS
 - CEIL
- Character Functions
 - LOWER
 - UPPER
 - LTRIM
 - RTRIM
 - TRIM
 - REPLACE
 - SUBSTR
- Conversion Functions
 - TO_CHAR
 - ADD_MONTHS
 - TO_DATE
 - CAST
- NULL-Related Functions

- NVL
5. Analytic Functions
- ROWNUM
 - RANK
 - PERCENT_RANK
 - SUM

4. 집합 연산자

1. 집합 연산자란?
 - UNION, UNION ALL, INTERSECT 등
 - 둘 이상의 QUERY 를 하나의 결과로 조합
 - SELECT 리스트의 표현식의 개수, 데이터 유형은 동일해야 함
 - ORDER BY 절
 - 전체 QUERY의 맨 끝에만 기술 가능
 - 첫 번째 SELECT 문의 ALIAS 로 기술
 - UNION ALL 을 제외하면 중복 행은 자동으로 제거됨
 - 첫 번째 QUERY 의 칼럼 이름이 결과에 나타남
2. UNION
 - 중복 행이 제거된 합집합
3. UNION ALL
 - 중복 행이 제거되지 않은 합집합
4. INTERSECT
 - 중복 행이 제거된 교집합
 - NULL 값을 무시하지 않음
 - 순차적인 연산 진행
 - NULL도 연산에 포함됨
5. MINUS
 - 중복 행이 제거된 차집합
6. 실습
 1. DML, Truncate, Transaction
 2. 대량 데이터 집합 조작
 3. Partitioning

1. DML, Truncate, Transaction

1. INSERT
 - Syntax

```
INSERT INTO table [ ( column [, ...] ) ]
{ DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
[ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

2. UPDATE
 - Syntax

```
UPDATE [ ONLY ] table [ [ AS ] alias ] SET { column = { expression | DEFAULT } |
( column [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
[ FROM fromlist ]
[ WHERE condition ]
[ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

3. DELETE
 - Syntax

```
DELETE FROM [ ONLY ] table [ [ AS ] alias ]
[ USING usinglist ]
[ WHERE condition ]
[ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

4. TRUNCATE
 - Syntax

```
TRUNCATE [ TABLE ] name [, ...] [ CASCADE | RESTRICT ]
```

5. TRANSACTION

- Autocommit이 Default
- Begin and commit 으로 트랜잭션을 설정

2. 대량 데이터 집합 조작

1. SELECT INTO

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ] * | expression [ AS output_name ] [, ...]  
INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table  
[ FROM from_item [, ...] ]  
[ WHERE condition ]  
[ GROUP BY expression [, ...] ]  
[ HAVING condition [, ...] ]  
[ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]  
[ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]  
[ LIMIT { count | ALL } ] [ OFFSET start ]  
[ FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ] [...]
```

2. DEFAULT

3. MERGE

4. LIMIT, OFFSET

- LIMIT
 - Select * from merge_test1 order by x limit 5;
 - 결과값 중 5개 Row만 자른 값
- OFFSET
 - Select * from merge_test1 order by x offset 5;
 - 결과값 중 5개 Row 다음 값들
- LIMIT, OFFSET 결합
 - Select * from merge_test1 order by x limit 3 offset 5;
 - 결과값 중 5개 Row 다음 값들 중 3개만 자른 값
 - limit, offset이 같이 있을 경우 자리를 바꿔도 무조건 offset 실행 후 limit을 실행한다

3. PARTITIONING

- Summary
- Range, List, Subpartition
- 각각의 Partition은 부모 테이블의 자식 테이블로 생성되며, 부모 테이블에는 데이터가 저장되지 않는다.
- 내부적으로 자식 테이블은 부모 테이블을 Inheritance 받아 구현
- 데이터가 키 값을 기준으로 적절한 파티션에 저장되도록 Trigger가 생성된다.
- Index는 Local방식만 지원된다.
- Partition 추가시 Index도 직접 추가해주어야 한다.

ADD

```
ALTER TABLE table_name ADD PARTITION partition_table_name VALUES [LESS THAN]{key value};
```

DROP

```
ALTER TABLE table_name DROP PARTITION partition_table_name;
```

1. DDL

```
a. CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name ( [ { column_name  
data_type [ DEFAULT default_expr ]  
[ column_constraint [ ... ] ] | table_constraint | LIKE parent_table  
[ { INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES } ] ... } [, ... ] )  
[ INHERITS ( parent_table [, ...] ) ]  
[ WITH ( storage_parameter [= value] [, ...] ) | WITH OIDS | WITHOUT OIDS ]  
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ][ TABLESPACE tablespace ]
```

- Temporart, Temp : 임시 테이블을 생성하며 세션 종료시 자동으로 drop된다. ON COMMIT 옵션을 사용하여, 트랜잭션 단위로 보존, delete, drop을 제어 할 수 있다.
- UNLOGGED : 변경사항 등을 기록하는 WAL 로그가 활성화되지 않는다. 비정상적인 crash나 shutdown 시 truncate가 되어

- 데이터가 유실된다.
- IF NOT EXISTE : MySQL과 동일하게 테이블의 존재유무에 따라 생성한다.
- COLLATE : 컬럼단위로 collation을 지정할 수 있다.
- LIKE
 - 테이블 생성시 참조되는 컬럼의 이름, 데이터 타입, not null 제약조건에 대해서 동일하게 생성한다.
 - including/excluding 옵션을 통해 필요한 테이블 정보를 추가 반영할 수 있다. (CONSTRAINTS, INDEXES, STORAGE, COMMENTS)
- TABLESPACE : 특정 tablespace를 지정할 수 있다.
- USING INDEX TABLESPACE : PK나 UK의 tablespace를 지정할 수 있다.
- #CONSTRAINTS
- NOT NULL
- UNIQUE KEY
- PRIMARY KEY
- FOREIGN KEY
- CHECK
 - 입력되는 컬럼값이 조건을 만족해야 한다.

a. INDEX

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] name ON table
[ USING method ] ( { column | ( expression ) } [ opclass ] [, ... ] )
[ WITH ( storage_parameter = value [, ... ] ) ]
[ TABLESPACE tablespace ]
[ WHERE predicate ]
```

b. DATA TYPE

- Numeric Types

이름	저장 크기	설명	범위
smallint	2바이트	작은 범위의 정수	-32768에서 +32767까지
integer	4바이트	자주 사용하는 정수	-2147483648에서 +2147483647까지
bigint	8바이트	광범위 정수	-9223372036854775808에서 9223372036854775807까지
decimal	변수	사용자 지정 정밀도, 정확함	제한 없음
numeric	가변	사용자 지정 정밀도, 정확함	제한 없음
real	4바이트	가변 정밀도, 부정확	6자리수 정밀도
double precision	8바이트	가변 정밀도, 부정확	15 자리수 정밀도
serial	4바이트	자동 증가하는 정수	1에서 2147483647까지
bigserial	8바이트	광범위 자동 증가 정수	1에서 9223372036854775807까지

- Character Types

이름	설명
character varying(n), varchar(n)	상한 첨부 가변길이
character(n), char(n)	공백에서 패드 된 고정길이
text	무제한 가변길이

- Date/Time Types

이름	저장 크기	설명	낮은 값	높은 값	해결
----	-------	----	------	------	----

timestamp [(p)] [without time zone]	8 바이트	일자와 시각 양쪽 모두	4713 BC 5874897 AD	1μ초, 14 자리수	
timestamp [(p)] with time zone	8바이트	일자와 시각 양쪽 모두, 시간대 첨부	4713 BC	5874897 AD	1μ초, 14 자리수
interval [(p)]	12바이트	시간 간격	-178000000년	178000000년	1μ초, 14 자리수
date	4바이트	일자만	4713 BC	5874897 AD	1일
time [(p)] [without time zone]	8바이트	그 날의 시각만	00:00:00	24:00:00	1μ초, 14 자리수
time [(p)] with time zone	12바이트	그 날의 시각만, 시간대 첨부	00:00:00+1459	24:00:00-1459	1μ초, 14 자리수

- Boolean Type

Name	Storage Size	Description	True	False
boolean	1 byte	state of true or false	TRUE, 't', 'true', 'y', 'yes', 'on', '1'	FALSE, 'f', 'false', 'n', 'no', 'off', '0'

2.

a. ALTER TABLE

- 칼럼 추가/삭제 및 칼럼명 변경
- Constraint 추가/삭제
- Default값 변경
- 데이터 타입 변경/테이블명 변경

a. DROP TABLE

```
DROP TABLE [ IF EXISTS ] name [ , ... ] [ CASCADE | RESTRICT ]
```

- IF EXISTS : 존재하지 않는 테이블을 삭제하러 해도 에러를 발생시키지 않는다.
- CASCADE : 테이블을 참조하는 모든 오브젝트들도 함께 삭제한다.
- RESTRICT : 테이블에 dependency가 걸려있을 경우 Drop되지 않도록 한다.

a. VIEW

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW name [ ( column_name [ , ... ] ) ] AS query
```

b. SEQUENCE

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE name [ INCREMENT [ BY ] increment ]  
[ MINVALUE minvalue | NO MINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE ]  
[ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
```