DBMS 및 SQL 활용 #7

▼ 문제

```
# 1. DB 생성, 데이터 삽입
-- DB 생성
CREATE DATABASE company;

-- DB 접속
\c company

-- 테이블 생성
CREATE TABLE employee (
  emp_id SERIAL PRIMARY KEY,
  name VARCHAR(50),
  department VARCHAR(50),
  hire_date DATE,
  salary NUMERIC
);

-- 샘플 데이터 삽입
INSERT INTO employee (name, department, hire_date, salary) VALUES
('Alice', 'IT', '2021-01-01', 55000),
('Bob', 'HR', '2020-06-15', 49000),
('Charlie', 'Sales', '2022-03-20', 60000);

# 2. 백업
백업
pg_dump -U postgres -d company -f company_backup.sql

복원
createdb company_restore
psql -U postgres -d company_restore -f company_backup.sql

Custom Format 백업
pg_dump -U postgres -Fc -d company -f company_custom.backup
createdb company_restore_custom
pg_restore -U postgres -d company_restore_custom company_custom.backup

Direct Format 백업
pg_dump -U postgres -Fd -j 4 -d company -f company_dir_backup/
createdb company_restore_dir
pg_restore -U postgres -d company_restore_dir company_dir_backup/

전체 백업
-- pg_dumpall -U postgres -f all_dbs_backup.sqlpg_dumpall -U postgres --clean -f all_dbs_backup.sql


//복원시
psql -U postgres -f all_dbs_backup.sql
```

▼ 실습 개요

- 목적

    - DB 및 테이블을 생성하고, SQL 파일, Custom Format, Directory Format, 전체 DB 클러스터 백업, 복원 수행
- 구현
    - DB 및 테이블 생성
        - company 데이터베이스 생성 → employee 테이블 생성 (사번, 이름, 부서, 입사일, 연봉 필드) → Alice, Bob, Charlie 샘플 데이터 삽입
    - SQL 파일 백업 & 복원
        - 백업: pg_dump -f company_backup.sql
        - 복원: createdb company_restore → psql -f
    - Custom Format 백업 & 복원
        - 백업: .backup 파일 생성 (-Fc)
        - 복원: pg_restore
    - Directory Format 백업 & 복원
        - 백업: 병렬 처리되는 디렉토리 형식 백업 (-Fd -j 4)
        - 복원: pg_restore
    - 전체 DB 클러스터 백업 & 복원
        - 백업: 모든 DB + 권한 + 유저 정보 백업 (pg_dumpall)
        - 복원: psql -f

▼ 코드

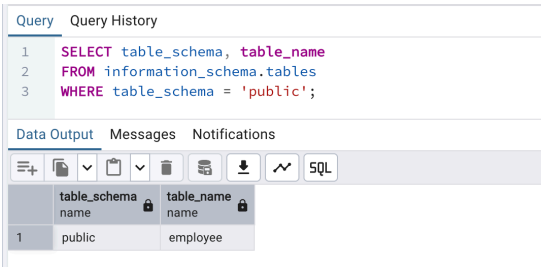
```
-- 1. DB 및 테이블 생성
-- DB 생성
CREATE DATABASE company;

-- company DB 접속

-- 테이블 생성
CREATE TABLE employee (
    emp_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    hire_date DATE,
    salary NUMERIC
);

-- 샘플 데이터 삽입
INSERT INTO employee (name, department, hire_date, salary) VALUES
('Alice', 'IT', '2021-01-01', 55000),
('Bob', 'HR', '2020-06-15', 49000),
('Charlie', 'Sales', '2022-03-20', 60000);
```

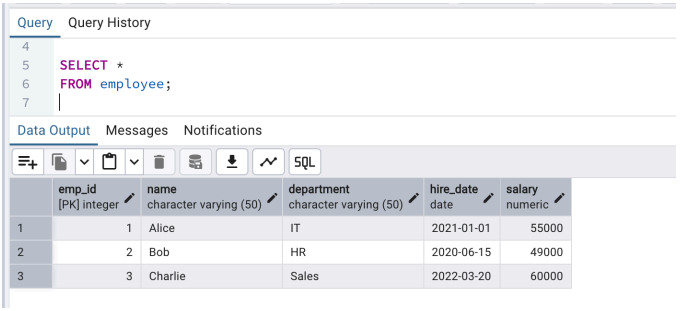
```
# 2. 백업
# $ pwd
# /Users/yshmbid/Documents/home/github/SQL/backup

# SQL 백업 / 복원
$ pg_dump -U postgres -d company -f company_backup.sql
$ createdb company_restore
$ psql -U postgres -d company_restore -f company_backup.sql
```

```
# Custom Format 백업 / 복원
$ pg_dump -U postgres -Fc -d company -f company_custom.backup
$ createdb company_restore_custom
$ pg_restore -U postgres -d company_restore_custom company_custom.backup

# Directory Format 백업 / 복원
$ pg_dump -U postgres -Fd -j 4 -d company -f company_dir_backup/
$ createdb company_restore_dir
$ pg_restore -U postgres -d company_restore_dir company_dir_backup/

# 전체 DB 백업 / 복원
#$ pg_dumpall -U postgres -f all_dbs_backup.sql
$ pg_dumpall -U postgres --clean -f all_dbs_backup.sql
$ psql -U postgres -f all_dbs_backup.sql
```

▼ 결과

▼ 1. DB 생성, 데이터 삽입





▼ 2. 백업 / 복원

#1 SQL 백업 / 복원

```
(skala) yshmbid:backup yshmbid$ pg_dump -U postgres -d company -f company_backup.sql
(skala) yshmbid:backup yshmbid$
(skala) yshmbid:backup yshmbid$ createdb company_restore
psql -U postgres -d company_restore -f company_backup.sql(skala) yshmbid:backup yshmbid$ psql -U postgres -d company_restore -f company_backup.sql
SET
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER TABLE
COPY 3
 setval
--------
      3
(1 row)

ALTER TABLE
```

#2 Custom Format 백업 / 복원

```
(skala) yshmbid:backup yshmbid$ pg_dump -U postgres -Fc -d company -f company_custom.backup
(skala) yshmbid:backup yshmbid$ createdb company_restore_custom
pg_restore -U postgres -d company_restore_custom company_custom.backup(skala) yshmbid:backup yshmbid$ pg_restore -U postgres -d company_restore_custom company_custom.backup
```

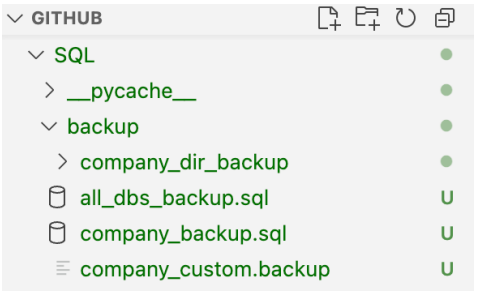
#3 Directory Format 백업 / 복원

```
(skala) yshmbid:backup yshmbid$ pg_dump -U postgres -Fd -j 4 -d company -f company_dir_backup/
(skala) yshmbid:backup yshmbid$ createdb company_restore_dir
pg_restore -U postgres -d company_restore_dir company_dir_backup/(skala) yshmbid:backup yshmbid$ pg_restore -U postgres -d company_restore_dir company_dir_backup/
```

#4 전체 DB 백업 / 복원

```
(skala) yshmbid:backup yshmbid$ pg_dumpall -U postgres -f all_dbs_backup.sql
(skala) yshmbid:backup yshmbid$ psql -U postgres -f all_dbs_backup.sql
SET
SET
SET
psql:all_dbs_backup.sql:16: ERROR:  role "postgres" already exists
ALTER ROLE
psql:all_dbs_backup.sql:18: ERROR:  role "yshmbid" already exists
ALTER ROLE
You are now connected to database "template1" as user "postgres".
SET
SET
SET
SET
SET
SET
 set_config
------------

(1 row)



psql:all_dbs_backup.sql:739: ERROR:  relation "design_doc" already exists
ALTER TABLE
psql:all_dbs_backup.sql:754: ERROR:  relation "design_doc_id_seq" already exists
ALTER SEQUENCE
ALTER SEQUENCE
psql:all_dbs_backup.sql:776: ERROR:  relation "issues" already exists
ALTER TABLE
psql:all_dbs_backup.sql:791: ERROR:  relation "issues_id_seq" already exists
ALTER SEQUENCE
ALTER SEQUENCE
psql:all_dbs_backup.sql:809: ERROR:  relation "temp_test" already exists
ALTER TABLE
psql:all_dbs_backup.sql:826: ERROR:  relation "user_behavior" already exists
ALTER TABLE
psql:all_dbs_backup.sql:838: ERROR:  relation "user_embeddings" already exists
ALTER TABLE
ALTER TABLE
ALTER TABLE
psql:all_dbs_backup.sql:982: ERROR:  duplicate key value violates unique constraint "design_doc_pkey"
DETAIL:  Key (id)=(1) already exists.
CONTEXT:  COPY design_doc, line 1
psql:all_dbs_backup.sql:1390: ERROR:  duplicate key value violates unique constraint "issues_pkey"
DETAIL:  Key (id)=(1) already exists.
CONTEXT:  COPY issues, line 1
COPY 24
psql:all_dbs_backup.sql:1930: ERROR:  duplicate key value violates unique constraint "user_behavior_pkey"
DETAIL:  Key (user_id)=(U0001) already exists.
CONTEXT:  COPY user_behavior, line 1
psql:all_dbs_backup.sql:2438: ERROR:  duplicate key value violates unique constraint "user_embeddings_pkey"
DETAIL:  Key (user_id)=(U0001) already exists.
CONTEXT:  COPY user_embeddings, line 1
 setval
--------
    120
(1 row)

 setval
--------
    400
(1 row)

psql:all_dbs_backup.sql:2460: ERROR:  multiple primary keys for table "design_doc" are not allowed
psql:all_dbs_backup.sql:2468: ERROR:  multiple primary keys for table "issues" are not allowed
psql:all_dbs_backup.sql:2476: ERROR:  multiple primary keys for table "user_behavior" are not allowed
psql:all_dbs_backup.sql:2484: ERROR:  multiple primary keys for table "user_embeddings" are not allowed
psql:all_dbs_backup.sql:2491: ERROR:  relation "design_doc_cosine_idx" already exists
psql:all_dbs_backup.sql:2498: ERROR:  relation "design_doc_embedding_vector_idx" already exists
psql:all_dbs_backup.sql:2505: ERROR:  relation "design_doc_embedding_vector_idx1" already exists
psql:all_dbs_backup.sql:2512: ERROR:  relation "design_doc_embedding_vector_idx2" already exists
psql:all_dbs_backup.sql:2519: ERROR:  relation "design_doc_embedding_vector_idx3" already exists
psql:all_dbs_backup.sql:2526: ERROR:  relation "design_doc_embedding_vector_idx4" already exists
psql:all_dbs_backup.sql:2533: ERROR:  relation "design_doc_embedding_vector_idx5" already exists
psql:all_dbs_backup.sql:2540: ERROR:  relation "design_doc_embedding_vector_idx6" already exists
psql:all_dbs_backup.sql:2547: ERROR:  relation "design_doc_embedding_vector_idx7" already exists
psql:all_dbs_backup.sql:2554: ERROR:  relation "design_doc_embedding_vector_idx8" already exists
psql:all_dbs_backup.sql:2561: ERROR:  relation "design_doc_embedding_vector_idx9" already exists
psql:all_dbs_backup.sql:2568: ERROR:  relation "design_doc_l2_idx" already exists
ALTER TABLE
psql:all_dbs_backup.sql:2581: ERROR:  policy "user_issues_policy" for table "issues" already exists
(skala) yshmbid:backup yshmbid$ pg_dumpall -U postgres --clean -f all_dbs_backup.sql
```

▼ 3. 파일 생성된 목록



▼ 개념

- DB 백업 목적?

    - 단순히 내용을 저장하는 것이 아니라 필요할 때 원하는 상태로 되돌릴 수 있도록 하기 위해서이고
    - 같은 데이터를 다루더라도 상황에 따라 어떤 방식으로 백업했는지가 복구의 편의성과 속도를 크게 좌우한다.
- SQL 파일 방식
    - 사람이 읽을 수 있는 텍스트 스크립트 형태로 데이터를 저장해서 단순하고 직관적이다
    - employee 테이블 전체를 날렸을때 SQL 파일에는 CREATE TABLE employee ...와 INSERT INTO employee ... (Alice, Bob, Charlie) 같은 명령어가 그대로 기록되어 있기 때문에 이 파일을 실행하면 테이블과 세 명의 데이터가 다시 만들어짐. 학습용이나 소규모 환경에서는 유용한데 데이터가 수백만 건으로 늘어나면 이 방식은 느리고 전체 단위 복원만 가능하다는 한계가 있다.
- Custom Format
    - 특정 부분만 복원할 수 있다.
    - employee 테이블에서 Alice와 Bob은 잘 보관되어 있는데, 누군가 실수로 Charlie 행을 지운 경우 SQL 파일 방식이라면 전체를 복원하면서 기존 데이터와 충돌이 날 수 있지만 하지만 Custom Format을 사용하면 employee 테이블 중 Charlie 데이터만 선택적으로 복원할 수 있다 즉 전체 DB를 건드리지 않고 필요한 부분만 다시 살려낼 수 있다.
- Directory Format
    - 데이터 양이 많을 때 효과적
    - employee 테이블이 단 세 명이 아니라 수백만 명인 경우 백업 파일이 하나라면 읽고 쓰는 속도가 느려지는데 Directory Format은 데이터를 여러 파일로 나누어 저장하고 병렬 처리로 동시에 복원할 수 있기 때문에 대규모 직원 데이터를 더 빠르게 되살릴 수 있다.
- 전체 DB 클러스터 백업
    - 단일 테이블이나 특정 DB가 아니라 서버 전체를 복원해야 할때.
    - employee 테이블만이아니라 payroll, attendance 같은 다른 테이블까지 포함된 여러 데이터베이스가 모두 날아갔거나 사용자 계정과 접근 권한까지 함께 손상됐다면 단순한 테이블 복원만으로는 부족하고 이때는 클러스터 백업을 복원하면 employee 테이블 + 다른 모든 DB와 사용자 권한까지 한 번에 되살릴 수 있다.
- 전체 DB 백업 / 복원 단계에서 오류 발생이유와 해결방법

```
# 오류 코드
$ pg_dumpall -U postgres -f all_dbs_backup.sql
```

```
psql:all_dbs_backup.sql:2460: ERROR:  multiple primary keys for table "design_doc" are not allowed
psql:all_dbs_backup.sql:2468: ERROR:  multiple primary keys for table "issues" are not allowed
psql:all_dbs_backup.sql:2476: ERROR:  multiple primary keys for table "user_behavior" are not allowe
d
psql:all_dbs_backup.sql:2484: ERROR:  multiple primary keys for table "user_embeddings" are not allo
wed
psql:all_dbs_backup.sql:2491: ERROR:  relation "design_doc_cosine_idx" already exists
psql:all_dbs_backup.sql:2498: ERROR:  relation "design_doc_embedding_vector_idx" already exists
psql:all_dbs_backup.sql:2505: ERROR:  relation "design_doc_embedding_vector_idx1" already exists
psql:all_dbs_backup.sql:2512: ERROR:  relation "design_doc_embedding_vector_idx2" already exists
...
```

    - 발생 이유
        - 기존 DB 객체(유저, 테이블, 인덱스, 제약조건 등)가 남아 있는 상태에서 그대로 복원 명령을 실행해서.
    - 기존 DB 객체가 남아 있는 상태에서 그대로 복원하면 안되는이유?
        - 데이터베이스 복원은 단순히 데이터를 덮어쓰는 작업이 아니라 백업 시점의 구조와 내용을 그대로 재현하는 과정이라서
        - 복원 스크립트에는 CREATE ROLE, CREATE TABLE, ALTER TABLE, ADD CONSTRAINT INSERT INTO VALUES CREATE INDEX 등이 포함돼있다 즉 복원 스크립트는 새로 만들겠다는 전제를 갖고 있음.

        - 그래서 복원 대상 DB에 이미 같은 이름의 객체가 존재하면 충돌이 발생한다. (ex. 같은 이름의 테이블이 있으면 CREATE TABLE 구문에서 에러 / 기본키 제약조건이 걸린 상태에서 같은 데이터(예: emp_id=1, Alice)를 또 삽입하려 하면 중복 에러 / 인덱스·제약조건: 이미 있는 인덱스나 PK를 다시 만들면 에러)
    - 해결 방법
        - `pg_dumpall -U postgres -f all_dbs_backup.sql` 대신 `pg_dumpall -U postgres --clean -f all_dbs_backup.sql`을 사용.
        - -clean 옵션은 복원 전에 DROP 구문을 포함시켜 기존 객체를 먼저 삭제함. DROP TABLE IF EXISTS design_doc; → CREATE TABLE design_doc; 순으로 실행되기 때문에 중복 충돌이 사라진다.
    - 결론
        - 복원은 DB를 "백업 시점과 동일한 빈 상태"로 가정하고 실행되는데 대상 DB에 기존 객체가 남아 있으면 CREATE, ALTER, ADD, INSERT등이 실행되는 과정에서 이름 충돌, 제약조건 위반, 중복 데이터 삽입같은 오류가 발생하기 때문에 에러가 발생함.