Software Engineering

Dr. Young-Woo Kwon

KNU

Why Use Database?

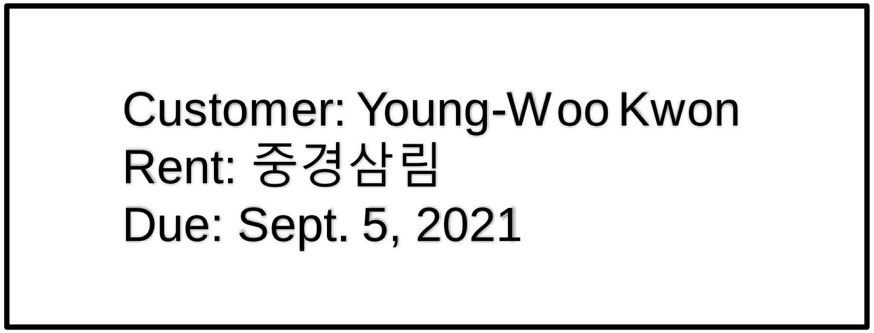
- Behind every successful website, there is a powerful database.
- • Examples:
  - – UPS / FedEx tracking
  - – Amazon's/eBay's websites
  - – Wal-Mart's inventory system
  - – Dell's ordering system
  - – Google's search engine

Data Management Example

- Scenario
  - You run a movie rental startup.
  - Your customers rent DVD copies of movies.
  - Several copies of each movie.
- Requirements
  - Which DVD disks have a customer rented?
  - Are any disks overdue?
  - When will a disk become available?

Solution: A “File-based” System

- (Create an) Edit rented.txt file



```
Customer: Young-Woo Kwon
Rent: 중경삼림
Due: Sept. 5, 2021
```

- Advantages?
  - – Text editors are easy to use
  - – Simple to insert a record (really?)
  - – Simple to delete a record (really?)

Complication: Queries?

- Does not address requirements
  - Query 1: Which movies have been rent by ‘Young-Woo Kwon’?
    - Search for ‘Young-Woo Kwon’
    - Read a movie rent by ‘Kwon’
    - Repeat it until there is no movie rent by ‘Kwon’
  - Query 2: Are there overdue disks?
    - Hmm, repeat query 1 and check date? Too complicate!!!

Complication: Integrity

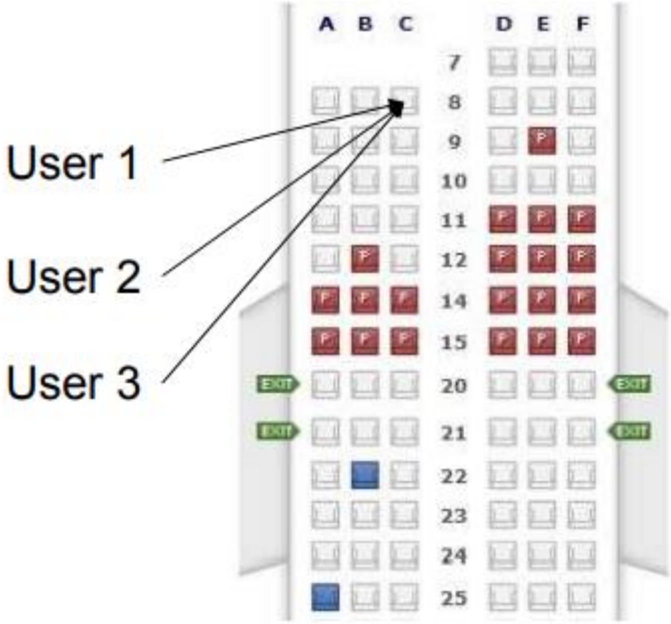
- Lacks data integrity, consistency
  - Clerk misspells value/field
    - Customer: Young-Koo Kwon, Rent: 중경삼림, Deu: Sep. 5, 2020
  - Inputs improper value, same value differently
    - Customer: Young-Woo Kwon, Rent: Chungking Express, Due: Sep. 5, 2018
  - Forgets/adds/reorders field
    - Terms: weekly special Due: Sep. 5, 2020, Rented: 중경 삼림

Complication: Update

- Add/delete/update fields in every record
  - Record store location.
    - Customer: Young-Woo Kwon, Rent: 중경삼림, Due: Sep. 5, 2020, Store: Bukgu
    - Modify the customer field to the first and last name fields
      - First: Young-Woo, Last: Kwon, Rented: 중경삼림, Due: Sep. 5, 2020, Store: Bukgu
  - Add/delete/update new information collections
    - customer.txt file to record information
    - customer: Young-Woo Kwon, Phone: 7566

Complication: Multiple Users

- Two clerks edit rent.txt file at the same time.
  - 1) Alice starts to edit rent.txt, reads it into memory.
  - 2) Bob starts to edit rent.txt.
  - 3) Alice adds a record.
  - 4) Alice saves rent.txt to disk.
  - 5) Bob saves rented.txt to disk



Complication: Crashes

- Crash during update may lead to inconsistent state.
  - You deposit $100 at an ATM
  - Before the ATM returns the deposit result, the network was disconnected or the banking system was shut down
  - Where is your $$$?

Complication: Physically Separate Data

- Need: want to inform Avengers' fans of that the 'Avengers: End Game' movie has been released
- Solution
  - customer.txt contains addresses of customers
  - Merge with rent.txt to create mailing list
- Problem
  - How to merge using a text editor?
  - What if there are several 'Kwon'?

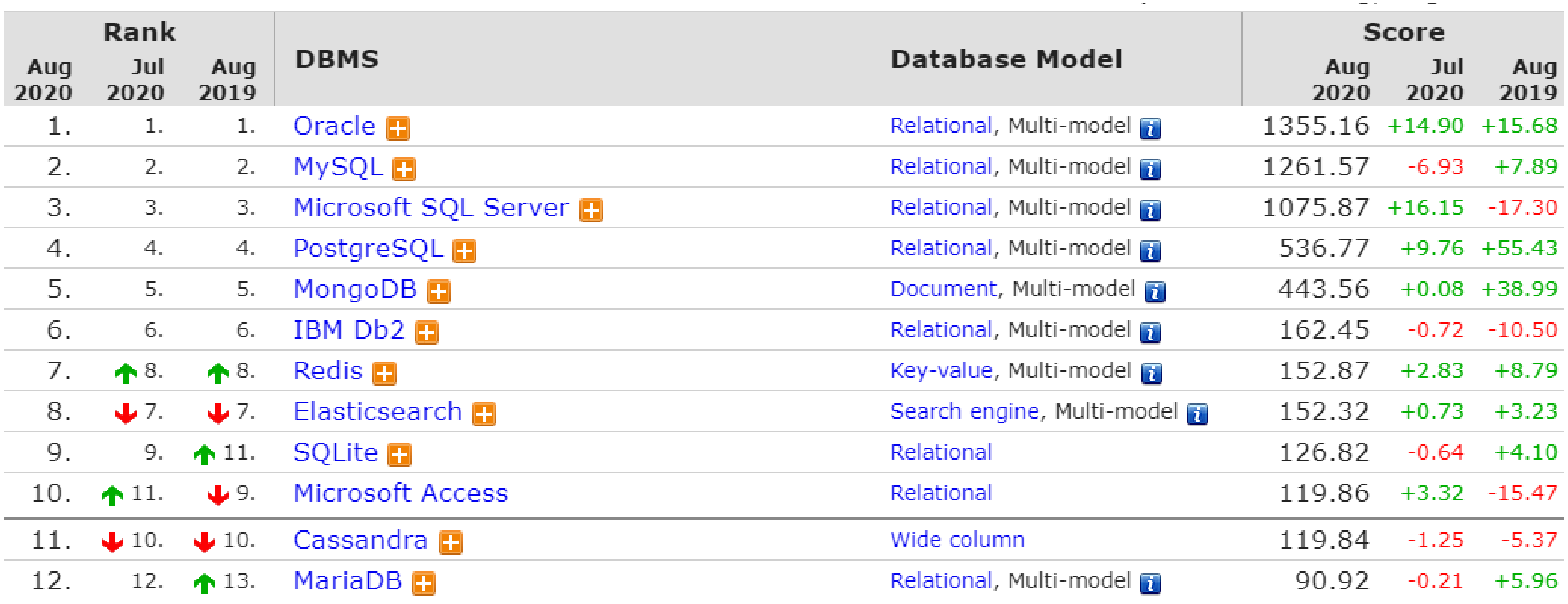
Complication: Security

- Customers want to know how many times a movie has been rented.
  - Provide access to rented.txt, but not to customer field, how I do that in an editor?
- Customers under 19 cannot see the list of R-rated movies
  - Add a new field? Keep two movie lists?

Complication: Efficiency

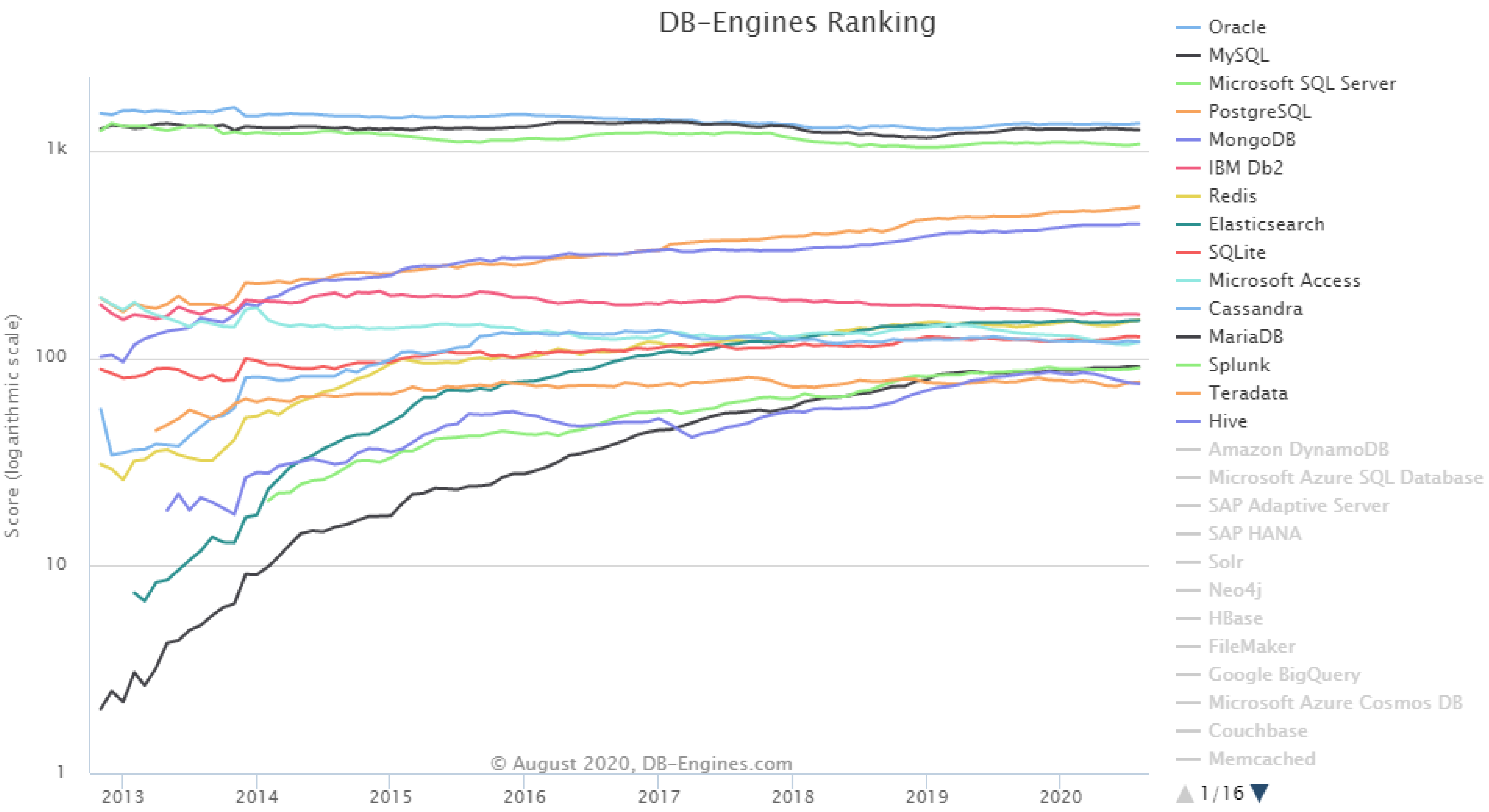
- Your customer list grows enormously.
  - rent.txt file gets huge (gigabytes, terabytes, or more of data).
  - Slow to open and edit
  - Slow to query for customer information.

DB Engine Rankings



Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.84	-1.25	-5.37
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	90.92	-0.21	+5.96

DB Engine Rankings



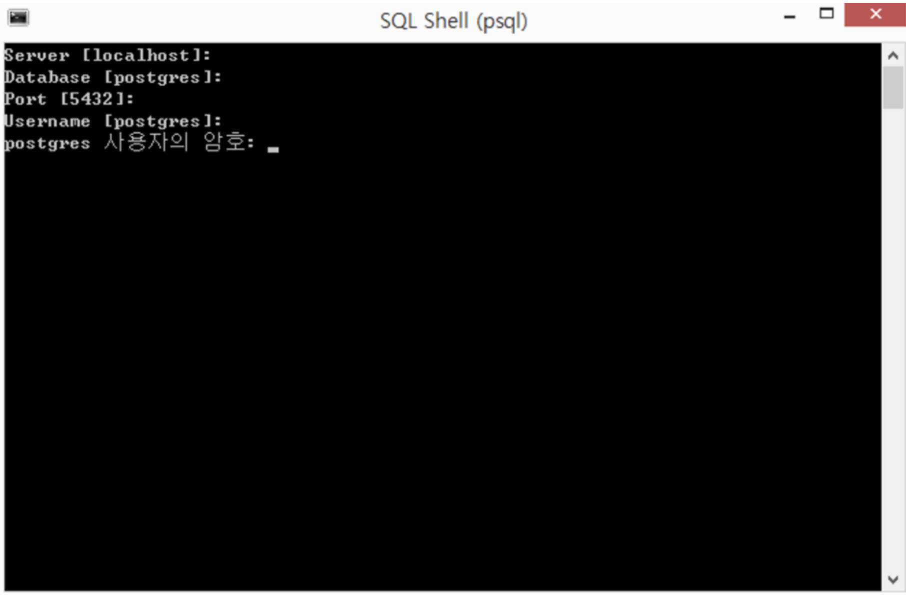
So, which database engines do you want to learn?

PostgreSQL

PostgreSQL: The World's Most Advanced Open Source Relational Database

SQL Shell (PSQL)

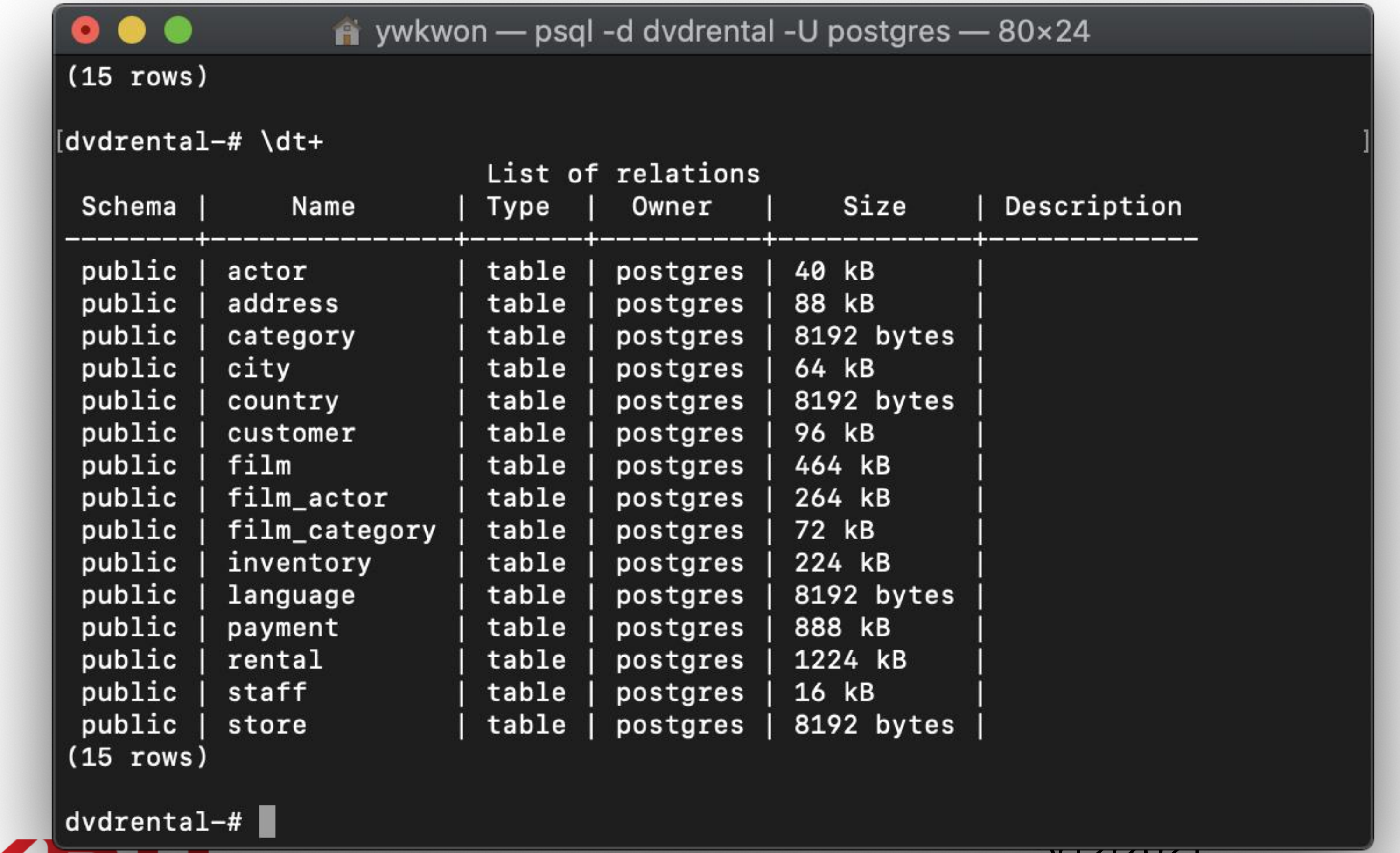
- PostgreSQL를 사용하기 위한 다른 도구
  - Server: 155.230.118.120
  - Database: hustarsedb
  - Port: 5432
  - Username: hustarse
  - Password: hustarse2021



```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
postgres 사용자의 암호: _
```

테이블 보기

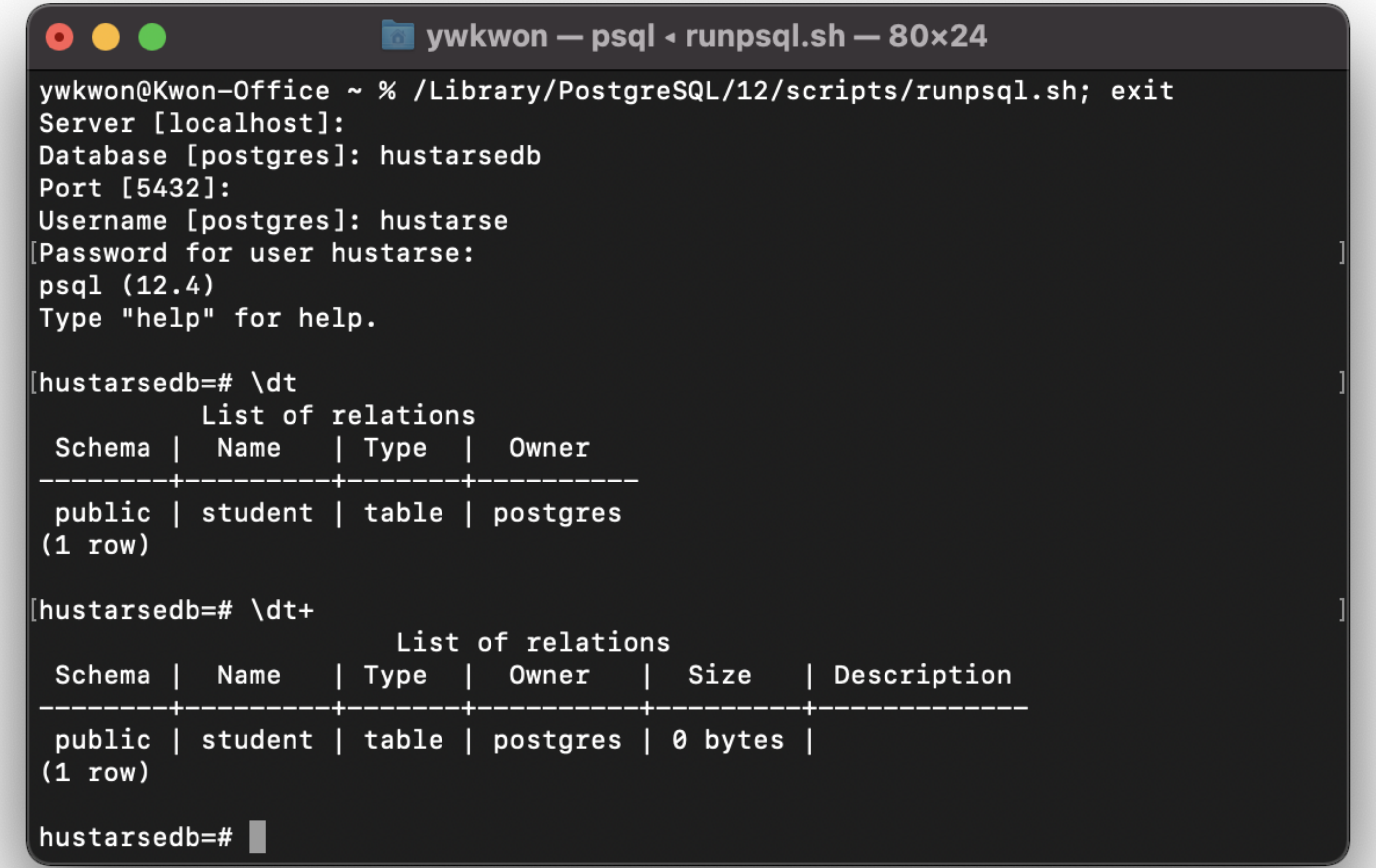
- ¥dt



```
(15 rows)

[dvdrental-# \dt+
                          List of relations
 Schema |     Name      | Type  |  Owner   |    Size    | Description
--------+---------------+-------+----------+------------+-------------
 public | actor         | table | postgres | 40 kB      |
 public | address       | table | postgres | 88 kB      |
 public | category      | table | postgres | 8192 bytes |
 public | city          | table | postgres | 64 kB      |
 public | country       | table | postgres | 8192 bytes |
 public | customer      | table | postgres | 96 kB      |
 public | film          | table | postgres | 464 kB     |
 public | film_actor    | table | postgres | 264 kB     |
 public | film_category | table | postgres | 72 kB      |
 public | inventory     | table | postgres | 224 kB     |
 public | language      | table | postgres | 8192 bytes |
 public | payment       | table | postgres | 888 kB     |
 public | rental        | table | postgres | 1224 kB    |
 public | staff         | table | postgres | 16 kB      |
 public | store         | table | postgres | 8192 bytes |
(15 rows)

dvdrental-# 
```

테이블 보기

- ¥dt+: 상세 정보 보기

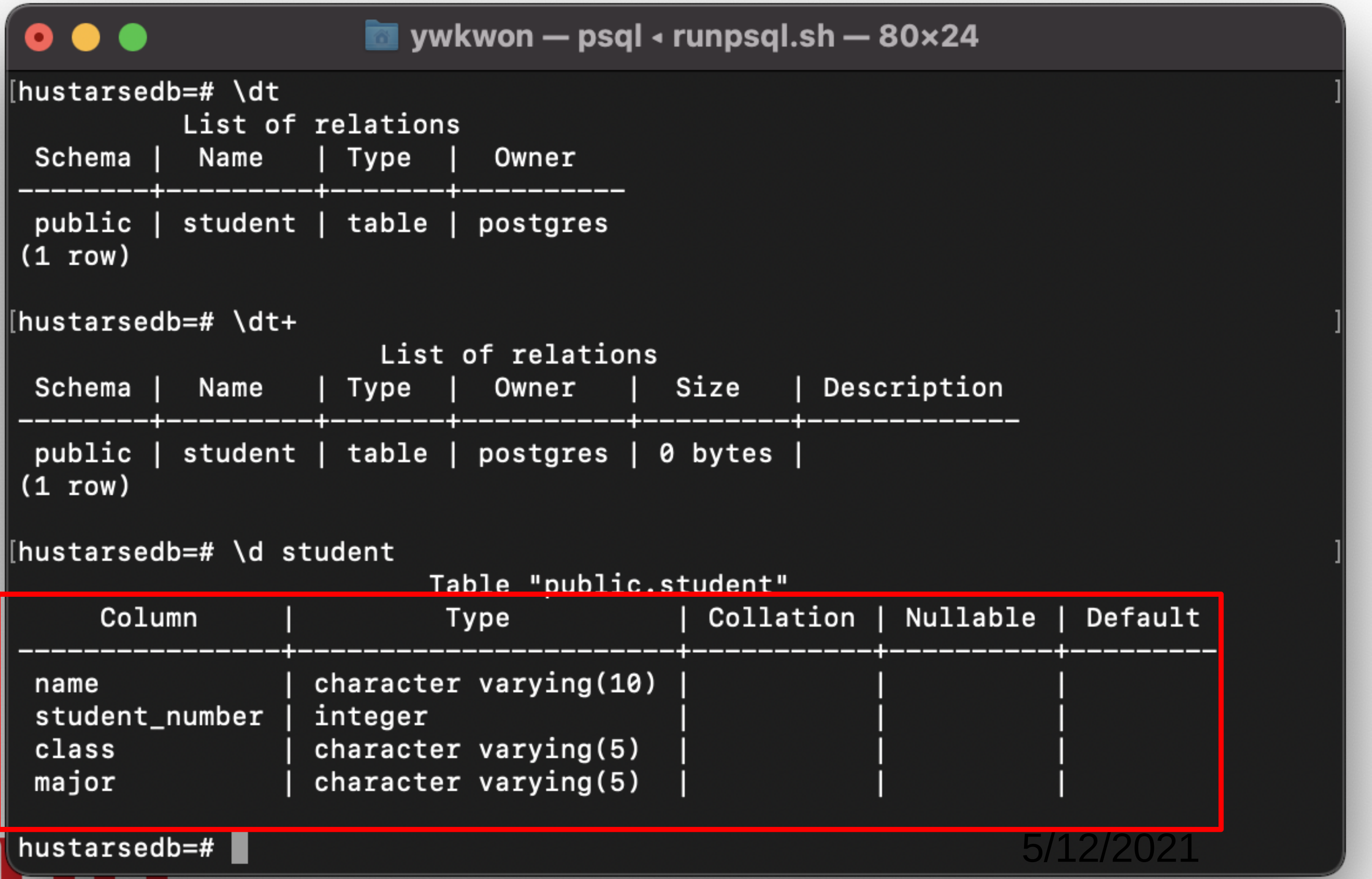


ywkwon — psql ◂ runpsql.sh — 80×24

```
ywkwon@Kwon-Office ~ % /Library/PostgreSQL/12/scripts/runpsql.sh; exit
Server [localhost]:
Database [postgres]: hustarsedb
Port [5432]:
Username [postgres]: hustarse
Password for user hustarse:
psql (12.4)
Type "help" for help.

hustarsedb=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | student | table | postgres
(1 row)

hustarsedb=# \dt+
                      List of relations
 Schema |  Name   | Type  |  Owner   |  Size   | Description
--------+---------+-------+----------+---------+-------------
 public | student | table | postgres | 0 bytes |
(1 row)

hustarsedb=# 
```

테이블 상세 보기

- ¥d [table name]
  - ¥d actor



```
[hustarsedb=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | student | table | postgres
(1 row)

[hustarsedb=# \dt+
                      List of relations
 Schema |  Name   | Type  |  Owner   |  Size   | Description
--------+---------+-------+----------+---------+-------------
 public | student | table | postgres | 0 bytes |
(1 row)

[hustarsedb=# \d student
                         Table "public.student"
     Column     |         Type          | Collation | Nullable | Default
----------------+-----------------------+-----------+----------+---------
 name           | character varying(10) |           |          |
 student_number | integer               |           |          |
 class          | character varying(5)  |           |          |
 major          | character varying(5)  |           |          |

hustarsedb=# 
```

Lab 1

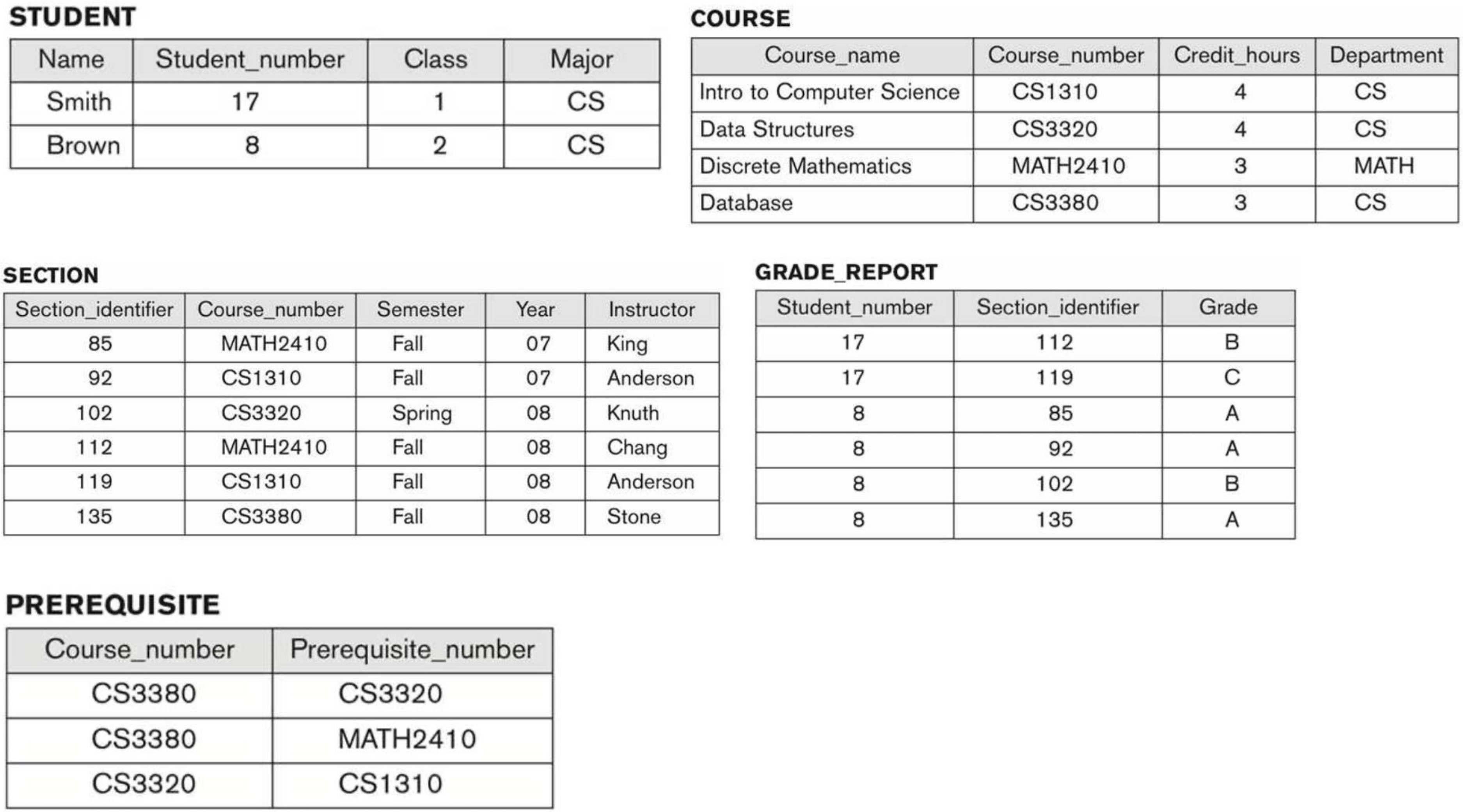
- Hustarse 데이터베이스에서 employee 테이블 정보를 psql을 사용하여 확인하세요.
  - employee 테이블의 컬럼의 개수는 몇 개인가요?
  - employee 테이블에서 사용되는 자료형의 종류를 나열하세요.

UNIVERSITY DATABASE

Creating a database schema (university)

Creating tables in the database schema

UNIVERSITY DB Schema



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

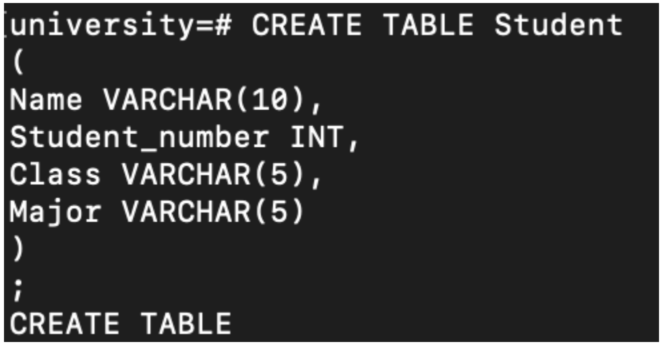
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Lab 2

- 테이블 생성

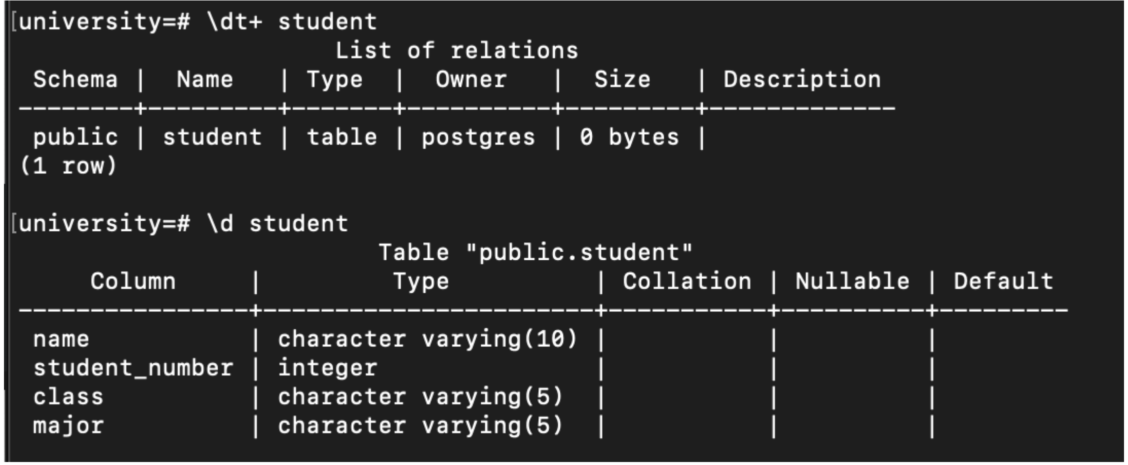
테이블 생성

- 테이블 생성: 다음 코드를 사용하여 테이블 생(테이블 명은 student_학번)



```
[university=# CREATE TABLE Student
(
Name VARCHAR(10),
Student_number INT,
Class VARCHAR(5),
Major VARCHAR(5)
)
;
CREATE TABLE
```

- 테이블 생성 확인 후 ¥dt 명령어와 ¥d student 사용하여 테이블과 각 컬럼 확인



```
[university=# \dt+ student
                       List of relations
 Schema |  Name   | Type  |  Owner   |  Size   | Description
--------+---------+-------+----------+---------+-------------
 public | student | table | postgres | 0 bytes |
(1 row)

[university=# \d student
                          Table "public.student"
     Column     |         Type          | Collation | Nullable | Default
----------------+-----------------------+-----------+----------+---------
 name           | character varying(10) |           |          |
 student_number | integer               |           |          |
 class          | character varying(5)  |           |          |
 major          | character varying(5)  |           |          |
```

Lab 2 제출물

- 테이블 생성 SQL
- 데이터베이스와 테이블 화면 캡쳐