ACID Properties
&
Concurrency Control

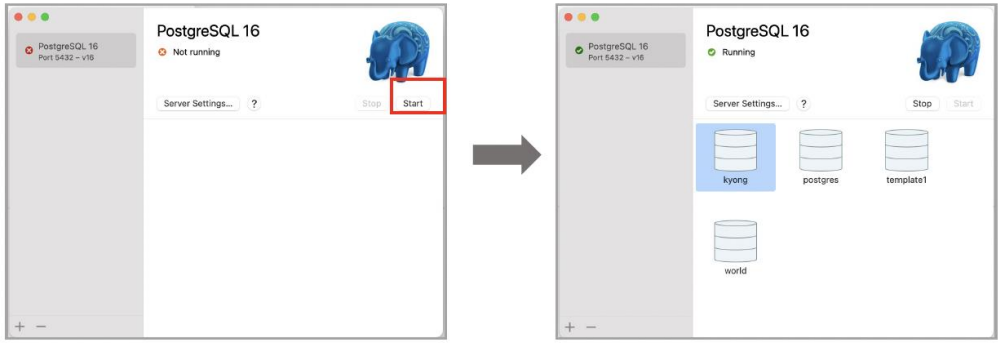
VLDB Lab.

Professor Sangwon Lee

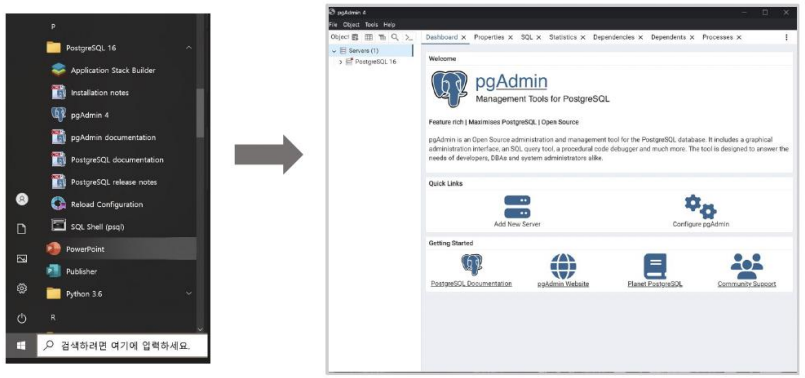
Start Postgres

1. Start Postgres.

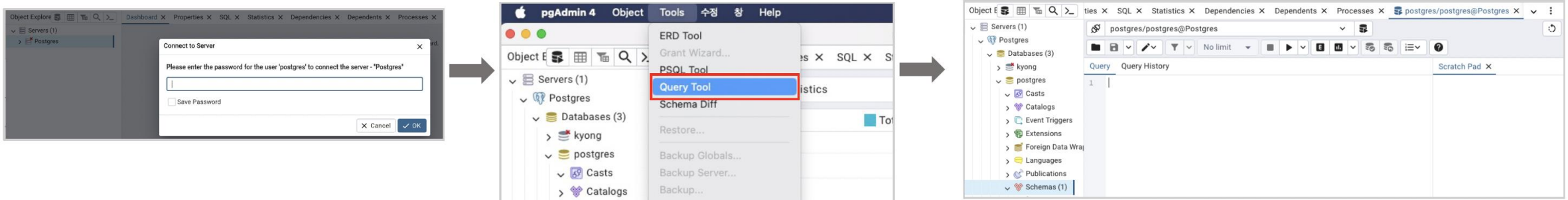
[Mac]



[Windows]



2. Start pgAdmin and connect postgres server.



Now you can write sql query!

ACID Properties

- **A (Atomicity, 원자성)**
  - Users should be able to regard the execution of each transaction as atomic: ALL-OR-NOTHING

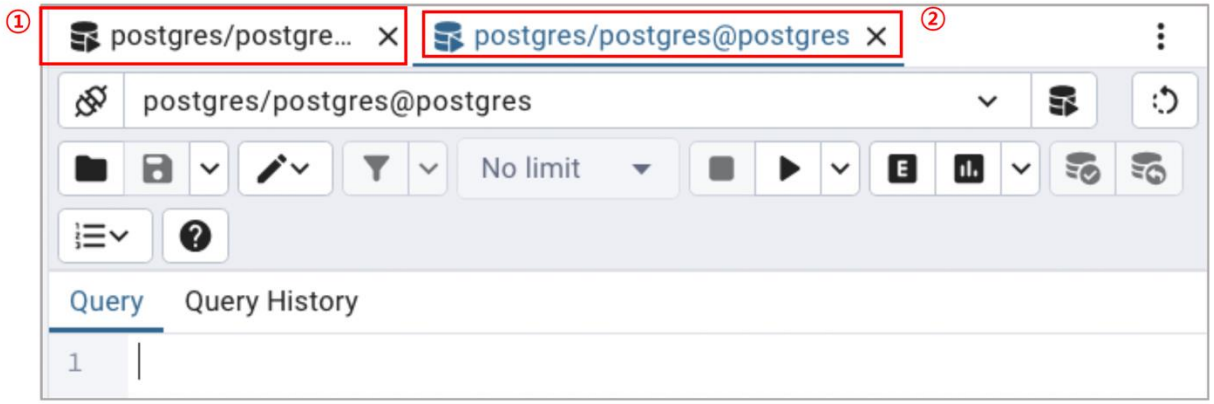
- **C (Consistency, 일관성)**
  - Each transaction, run by itself, must preserve the consistency of the database.

- **I (Isolation, 고립성)**
  - Each transaction is isolated or protected from the effects of concurrent other transactions.

- **D (Durability, 내구성)**
  - Once completed, a transaction’s effect should persist in spite of system crashes.

Setting

1. 실습을 위해 Query Tool을 2개 열어놓습니다.



2. 첫번째 query tool 창에서 다음 SQL문을 실행합니다.

```
-- Drop and create table
DROP TABLE IF EXISTS account;
CREATE TABLE account (id INT, balance NUMERIC);
-- Insert initial data
INSERT INTO account (id, balance) VALUES (1, 1000);
INSERT INTO account (id, balance) VALUES (2, 2000);
COMMIT;
```

Atomicity

Rollback (Nothing)

postgres/postgres@postgres* | postgres/postgre...

postgres/postgres@postgres

No limit

Query | Query History

```
BEGIN;
UPDATE account SET balance = balance - 100 WHERE id = 1;
UPDATE account SET balance = balance + 100 WHERE id = 2;
rollback;

select * from account;
```

Data Output | Messages | Notifications

	id integer	balance numeric
1	1	1000
2	2	2000

Commit (All)

postgres/postgres@postgres* | postgres/postgre...

postgres/postgres@postgres

No limit

Query | Query History

```
BEGIN;
UPDATE account SET balance = balance - 100 WHERE id = 1;
UPDATE account SET balance = balance + 100 WHERE id = 2;
commit;

select * from account;
```

Data Output | Messages | Notifications

	id integer	balance numeric
1	1	900
2	2	2100

Consistency

Add Constraint

Query | Query History

```
ALTER TABLE account ADD CONSTRAINT positive_balance CHECK (balance >= 0);
```

Data Output | Messages | Notifications

```
ALTER TABLE
```

	id integer	balance numeric
1	1	1000
2	2	2000

1) Constraint 충족

Query | Query History

```
BEGIN;
UPDATE account SET balance = balance + 100 WHERE id = 2;
COMMIT;

SELECT * FROM account;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	id integer	balance numeric
1	1	1000
2	2	2100

2) Constraint 위반

Query | Query History

```
BEGIN;
UPDATE account SET balance = balance - 2000 WHERE id = 1; -- 실패 (balance < 0)
COMMIT;
```

Data Output | Messages | Notifications

```
ERROR:  current transaction is aborted, commands ignored until end of transaction block
```

Query | Query History

```
rollback;
```

Data Output | Messages | Notifications

```
ROLLBACK
```

👉 PostgreSQL에서는 트랜잭션 중에 하나의 명령이라도 실패하면 트랜잭션이 중단된(aborted) 상태로 전환됨. 이 상태에서 추가 명령을 실행할 수 없으며, 트랜잭션을 반드시 ROLLBACK 해야함.

Isolation

Timeline

Session 1

postgres/postgres@postgres* | postgres/postgres...

postgres/postgres@postgres

No limit

Query | Query History

```
1  begin;
2  update account set balance = balance + 100 where id = 1;
```

postgres/postgres@postgres* | postgres/postgres...

postgres/postgres@postgres

No limit

Query | Query History

```
1  commit;
```

Data Output | Messages | Notifications

```
COMMIT
```

Session 2

	id integer	balance numeric
1	1	1000
2	2	2000

postgres/postgres... | postgres/postgres@postgres*

postgres/postgres@postgres

No limit

Query | Query History

```
1  select * from account;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2 | Page No: 1 of 1

	id integer	balance numeric
1	1	1000
2	2	2000

	id integer	balance numeric
1	2	2000
2	1	1100

Isolation Level

- **트랜잭션 격리 수준:** 여러 트랜잭션이 동시에 처리될 때, 트랜잭션끼리 서로 얼마나 고립되어있는지를 나타내는 수준

	Dirty Read	Non-repeatable Read	Phantom Read
Read Uncommitted	o	o	o
Read Committed	x	o	o
Reapeatable Read	x	x	o
Serializable	x	x	x

Durability

1) Update

postgres/postgres@postgres

No limit

Query Query History

```
1  update account set balanace = balance + 100 where id = 1;
```

	id integer	balance numeric
1	1	1000
2	2	2000

	id integer	balance numeric
1	2	2000
2	1	1100

2) Postgres restart → 데이터가 잘 보존되어있음.

No limit

Query Query History

```
1  select * from account;
```

	id integer	balance numeric
1	2	2000
2	1	1100