

## 4장. AgensSQL의 동시성 제어

### 4.1 ACID 지원

PostgreSQL은 관계형 DBMS의 기본적인 기능인 트랜잭션과 ACID 속성을 준수합니다.

[참고] ACID

#### Atomicity(원자성)

Transaction은 모두 반영되거나 모두 반영되지 않아야 합니다. 여러 쿼리를 하나의 실행 단위로 반영하기 때문에 트랜잭션내의 쿼리들의 결과는 모두 반영되거나 모두 취소하여 일부만 반영되는 일이 없어야 합니다.

#### Consistency(일관성)

Transaction의 결과는 실행 시점에 따라 데이터의 일관성을 유지하여야 합니다. 가령 트랜잭션의 처리도중 데이터의 변경이 발생한다 하여도 트랜잭션의 결과 데이터는 처리시점을 기준으로 합니다.

#### Isolation(격리성)

각 Transaction은 수행될 때 서로의 연산에 간섭할 수 없습니다. 즉 한 Transaction의 중간 결과가 다른 트랜잭션에 영향을 주어서는 안됩니다. 격리성을 보장할수 있는 가장 좋은 방법은 각각의 Transaction을 순차적으로 수행하는 것입니다.

#### Durability(지속성)

완료된 Transaction의 결과는 데이터베이스에 저장하여 이후 어떤 소프트웨어나 하드웨어 장애에도 데이터를 보존하여야 합니다.

## 4.2 AgensSQL의 MVCC

RDBMS는 데이터의 일관성을 확보하기 위해 데이터에 대한 버전관리를 통해 이를 해결하는데 이를 MVCC (Multi Version Concurrency Control)라 말합니다. AgensSQL에서는 각 데이터별로 4Byte의 버전정보(XID)를 두어 시점을 식별 합니다. 쿼리 수행 시점의 버전 정보와 데이터의 버전정보의 비교를 통해 MVCC를 구현합니다.

이전 버전의 데이터를 테이블 블록 내에 저장함으로써, MVCC를 매우 단순하게 구현할 수 있다는 장점이 있는 반면, 불필요한 데이터로 인해 테이블의 공간 사용 효율이 떨어진다는 단점이 있습니다. 이를 해결하기 위해 Vacuum 작업을 자동 및 수동적으로 수행해야 합니다.

### 4.2.1 MVCC 작동 예제

- **Transaction A (Terminal 1)**  
# 접속  
asql -U agens -d postgres  
  
# Test Table 생성  
CREATE TABLE test (a int);  
  
# Data Insert 진행  
BEGIN;  
INSERT INTO test VALUES (1);
- **Transaction B (Terminal 2)**  
# 접속  
asql -U agens -d postgres  
  
# Data Insert 진행  
BEGIN;  
INSERT INTO test VALUES (2);
- **Transaction C (Terminal 3)**  
# 접속  
asql -U agens -d postgres  
  
# pg\_stat\_activity View를 이용한 XID 확인  
SELECT backend\_xid, backend\_xmin, query FROM pg\_stat\_activity;

backend_xid	backend_xmin	
495		INSERT INTO test VALUES (1);
	495	SELECT username,backend_xid,ba
496		insert into test VALUES(2);

쿼리 별 **XID, XMIN**을 이용하여 **Data Version** 관리

### 4.2.2 세션별 데이터 확인 예제

transaction\_isolation이 default 값인 **'read committed'** 인경우 MVCC 작동방식

Session A (asql -U agens -d postgres)		Session B (asql -U agens -d postgres)	
①	create table test (id numeric, name varchar(20)); insert into test select 1, 'session1'; insert into test select 2, 'session2';		
②	begin; update test set name='session10' where id=1;		
		③	select name from test where id=1; => session1
		④	update test set name='session20' where id=1; => LOCKED WAITING...
⑤	commit;		
			UPDATED 1 => success updated
		⑥	select name from test where id=1; => session20