



데이터베이스

MM4220 게임서버 프로그래밍

정내훈

내용

- Database는 database 수업시간에 배움
- 여기서는 C++로 되어 있는 게임서버에서 database에 접속해서 SQL명령어를 실행하고 결과를 얻어오는 방법을 살펴봄
- 성능 문제도 같이 살펴봄

차례

- DB 기초
- DB 프로그래밍

데이터베이스

- 정의
 - 구조화된 데이터나 레코드의 모임
 - 컴퓨터에 저장
 - 쿼리(query)를 사용하여 프로그램에서 접근가능
- 종류
 - Hierarchical model
 - Relational model
 - DB2, Oracle, MySQL, MS-SQL
 - Object model

데이터베이스

- 게임에서 많이 사용되는 DB
 - CISAM (C Indexed Sequential Access Method)
 - DB가 아닌 단순한 파일 시스템
 - MySQL => Maria DB
 - Free Software
 - Simple but fast, 리눅스 운영체제에서 많이 사용
 - MSSQL
 - MicroSoft, 한국에서 가장 많이 사용
 - Oracle
 - Expensive, 수출이 어렵다
 - Memory DB
 - 고속 자료처리 가능, memcached
 - NoSQL
 - 자료의 분산, Cloud에 특화, Transaction 처리 곤란
 - HBase, mongo-db, Redis

데이터베이스

Type	Notable examples of this type
<u>Wide Column Store</u>	<u>Amazon DynamoDB</u> , <u>Bigtable</u> , <u>Cassandra</u> , <u>Scylla</u> , <u>HBase</u> , <u>Hypertable</u>
Tuple store	<u>Apache River</u> , <u>GigaSpaces</u>
Object database	<u>Objectivity/DB</u> , <u>Perst</u> , <u>ZopeDB</u>
Native multi-model database	<u>ArangoDB</u> , <u>Cosmos DB</u> , <u>OrientDB</u> , <u>MarkLogic</u>
Key–value store (ordered)	<u>FoundationDB</u> , <u>InfinityDB</u> , <u>LMDB</u> , <u>MemcacheDB</u>
Key–Value store (eventually consistent)	<u>Oracle NoSQL Database</u> , <u>Dynamo</u> , <u>Riak</u> , <u>Voldemort</u>
Key–Value store	<u>ArangoDB</u> , <u>Aerospike</u> , <u>Couchbase</u> , <u>Redis</u>
Key–value cache	<u>Apache Ignite</u> , <u>Couchbase</u> , <u>Coherence</u> , <u>eXtreme Scale</u> , <u>Hazelcast</u> , <u>Infinispan</u> , <u>Memcached</u> , <u>Redis</u> , <u>Velocity</u>
Document store	<u>ArangoDB</u> , <u>BaseX</u> , <u>Clusterpoint</u> , <u>Couchbase</u> , <u>CouchDB</u> , <u>DocumentDB</u> , <u>eXist-db</u> , <u>IBM Domino</u> , <u>MarkLogic</u> , <u>MongoDB</u> , <u>Qizx</u> , <u>RethinkDB</u> , <u>Elasticsearch</u>

데이터베이스

- 왜 Data를 보조 기억장치에 저장하는가?
 - 데이터의 크기
 - 서버 프로그램이 모든 데이터를 다 메모리에 갖고 있을 수 없다.
 - 데이터 보존
 - 서버가 종료되어도 게임 데이터가 남아 있어야 한다.
- 왜 DB를 사용하는가: 안정성과 효율성
 - 상용 데이터베이스 보다 데이터 관리를 더 잘하는 프로그램을 짤 수 있는가?

데이터베이스

- DB를 쓰면 좋은 점
 - 다른 프로그램으로도 데이터를 다룰 수 있다.
 - 각종 툴이나 browser를 사용할 수 있다
- DB의 단점
 - 잘 모르는 Black-box!
 - 잘 모르는 상태에서 사용하면 풀기 어려운 예기치 않은 문제를 만날 수 있다.

데이터베이스

- 필수 구현 요소 (왜 직접 구현하지 않나?)
 - 저장소 : 하드디스크, SSD, 메모리
 - 인덱스를 통한 고속 검색
 - 트랜잭션과 동시성(concurrency)
 - 복제(replication)
 - 안정성, 부하 분산
 - 보안(security)

데이터베이스

- 데이터베이스의 접속 및 사용
 - 네트워크를 통한 접속
 - Text Based Socket
 - ODBC
 - 쿼리 언어
 - 자료 검색/업데이트 부터 Table 생성, 소멸
 - SQL, QUEL, OQL,...

데이터베이스

- DB에 저장되는 데이터
 - 과금 정보
 - 캐릭터 정보 (이름, 레벨, 종족, Skill, Quest, 외형, ...)
 - 아이템 정보 (ID, Type, Enchant, 개수, Owner ID)
 - 월드 정보
 - 성주, 집주인, 대회 우승 기록, 게시판...
 - 경매장 정보
 - 우편정보
 - 길드 정보
- 게임 로그 (단순히 저장만 하고 다시 읽지 않음)
 - 상거래, PK, 채팅, 아이템 습득/사용
 - 파티, 레이드원 정보
 - 보스 드랍 아이템 기록

데이터베이스

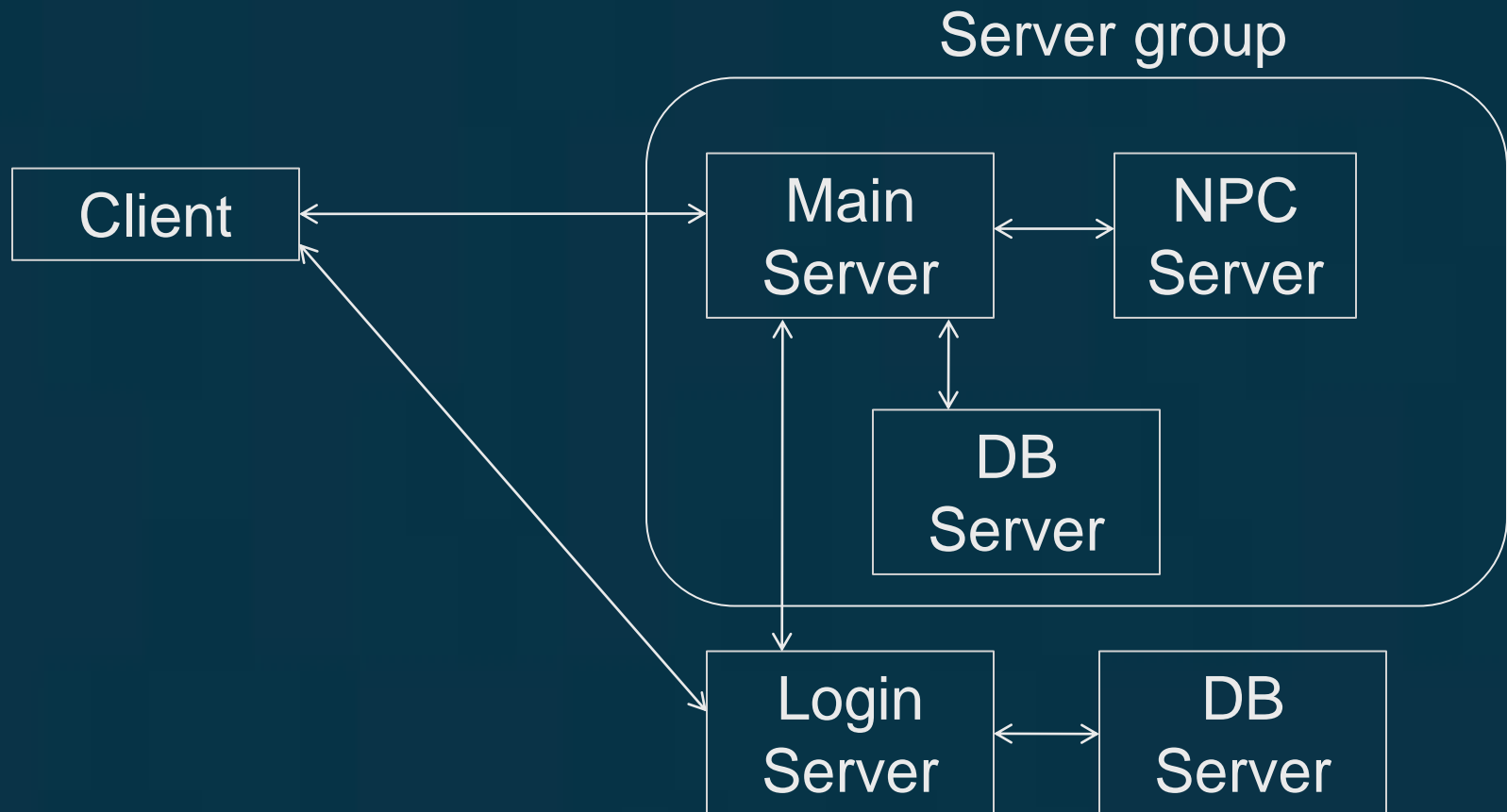
- DB에 저장되지 않는 데이터
 - 서버 재부팅 때 초기화 되는 정보
 - 몬스터의 위치, 몬스터의 HP, 바닥에 드랍된 아이템
 - 게임플레이 중에 변경되지 않는 정보
 - 파일로 저장되고 서버 부팅 시 메모리에 모두 로딩된다.
 - 예)
 - 지형데이터
 - 몬스터 배치, 드랍 테이블
 - Quest Script, AI Script, Skill Tree

데이터베이스

- 서버는 언제(얼마나 자주) DB에 접근하는가?
 - 필수
 - 캐릭터 login/logout
 - 서버 shutdown/booting
 - 추가적 : “비 정상적인 서버다운에도 안전하게!”
 - 그때 그때 : 캐릭터 사이의 아이템 거래
 - 주기적 : auto-save
 - 중요도와 주기는 반비례
 - 그 이외
 - 중요한 캐릭터 데이터 변경
 - 캐릭터의 zone이동 시

데이터베이스

- DB서버의 위치



데이터베이스

- 게임서버에서 DB를 사용할 때의 문제점
 - DB 접근은 처리에 시간이 걸린다.
 - 블럭킹 API
 - 게임서버는 클라이언트 패킷에 real-time으로 반응해야 한다
- 해결
 - DB 접근과 게임 메인 쓰레드(worker thread)의 분리
 - DB 접근 Thread
 - DB 접근 Process
 - Cache Server, Query Server

데이터베이스

- 분리의 효과

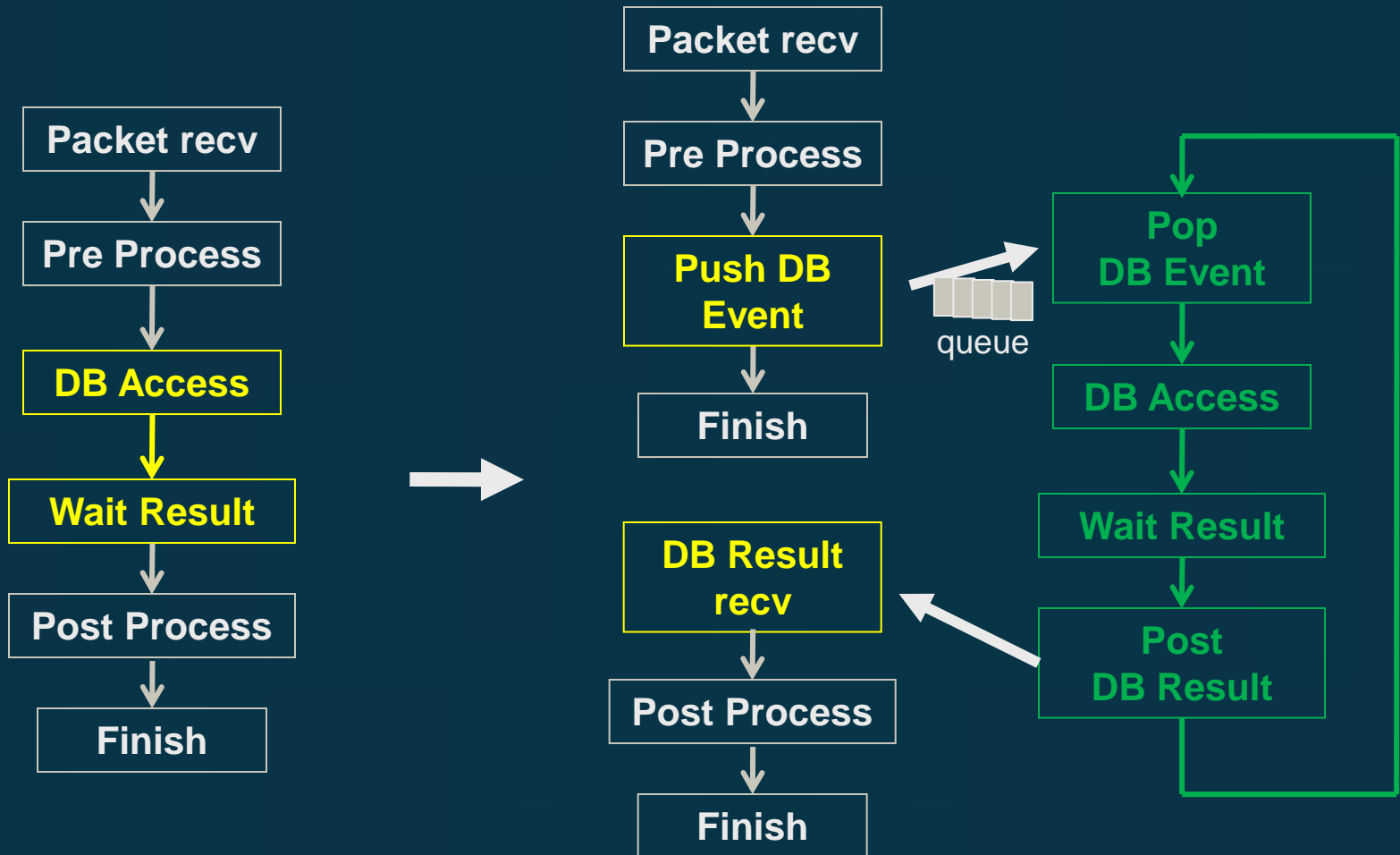
- 분리전

- 아이템 클릭 -> 아이템 수집 패킷 -> 패킷 프로세스 -> DB 업데이트 요청 -> [DELAY] -> DB 업데이트 응답 -> 인벤토리 업데이트 -> 인벤토리 업데이트 패킷 -> 아이템 창 업데이트 -> 끝

- 분리후

- 아이템 클릭 -> 아이템 수집 패킷 -> 패킷 프로세스 -> DB 업데이트 이벤트 생성 -> 끝
 - DB 업데이트 이벤트 POP -> DB 업데이트 요청 -> [DELAY] -> DB 업데이트 응답 -> 결과 PQCS -> 반복
 - GQCS -> 결과 해석 -> 인벤토리 업데이트 -> 인벤토리 업데이트 패킷 -> 아이템 창 업데이트

데이터베이스



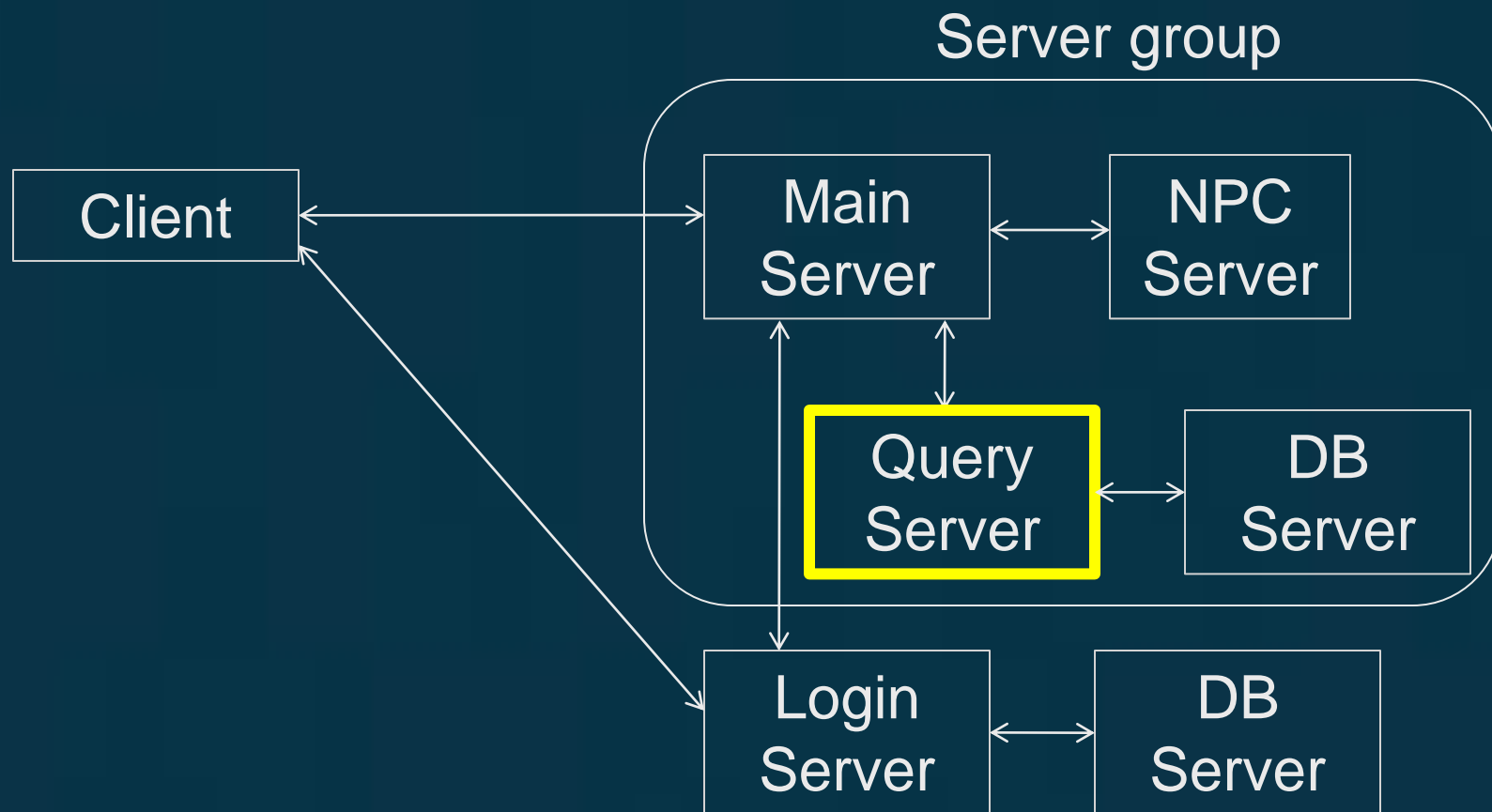
데이터베이스

- DB 접근 Thread
 - DB 접근을 위한 thread를 따로 돌림
 - Worker thread에서는 DB 접근으로 인한 Delay가 없음
 - DB 관련 event의 정의가 필요
 - Multiple Write/Single Read Concurrent Queue가 필요.

데이터베이스

- DB 접근 Process
 - DB 접근을 위한 프로그램을 따로 돌림
 - Main Server에서는 DB 접근으로 인한 Delay가 없음
 - Process간의 communication overhead가 있음
 - DB 관련 protocol 의 정의가 필요
 - 다른 컴퓨터에서 수행 가능
 - 부하와 메모리 사용 분산
 - Cache Server/Query Server로 불리기도 함

데이터베이스



DB Programming

- SQL (Structured Query Language)
 - A database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management

<http://en.wikipedia.org/wiki/SQL>

DB Programming

- SQL (Structured Query Language)
 - 실제 DB의 data를 조작하는 언어
 - 예)
 - `SELECT * FROM items WHERE owner_id = 42`
 - `INSERT INTO items (id, item_type, owner_id, count) VALUES (1043, 21, 24, 1);`
 - `UPDATE user_table SET exp = 129020 WHERE user_id = 42;`
 - `DELETE FROM items WHERE id = 1043;`
 - `CREATE TABLE items (id INT, item_type INT, owner_id INT, count INT);`

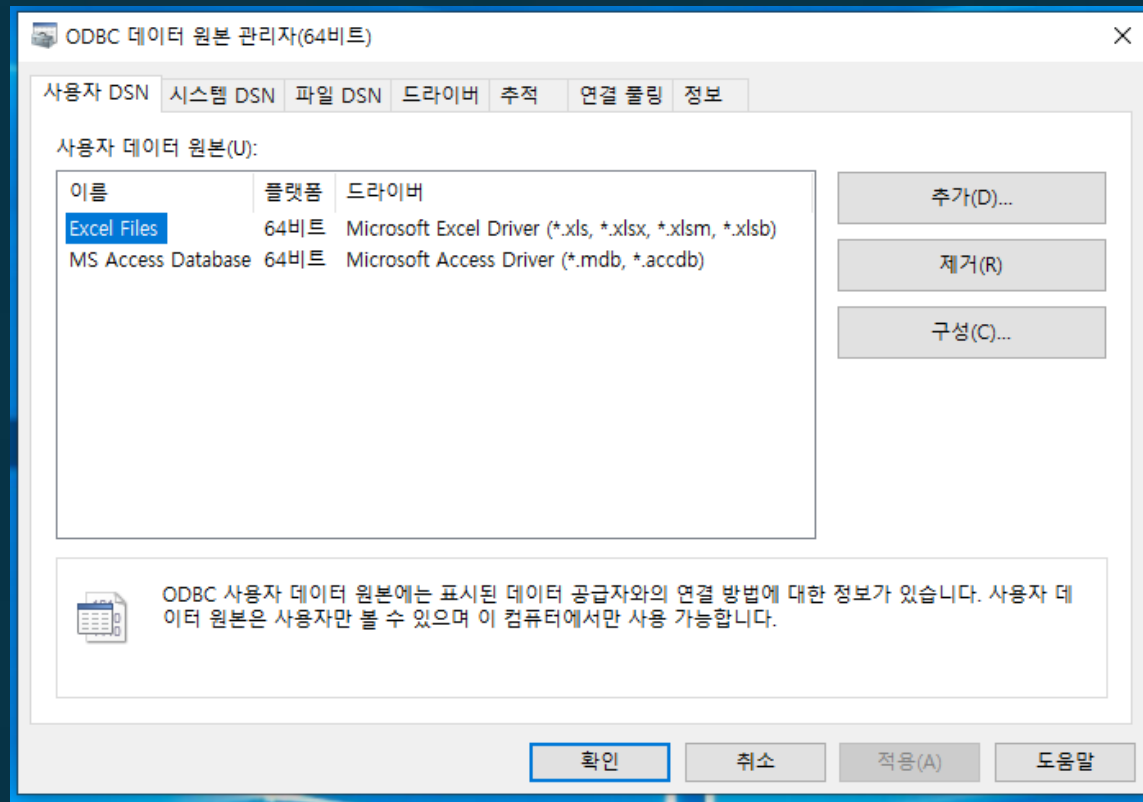
DB Programming

- ODBC (Open Database Connectivity)
 - C같은 프로그래밍 언어에서 SQL을 사용하기 위한 표준
 - It provides a standard software API method for using database management systems (DBMS). The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems.

<http://en.wikipedia.org/wiki/ODBC>

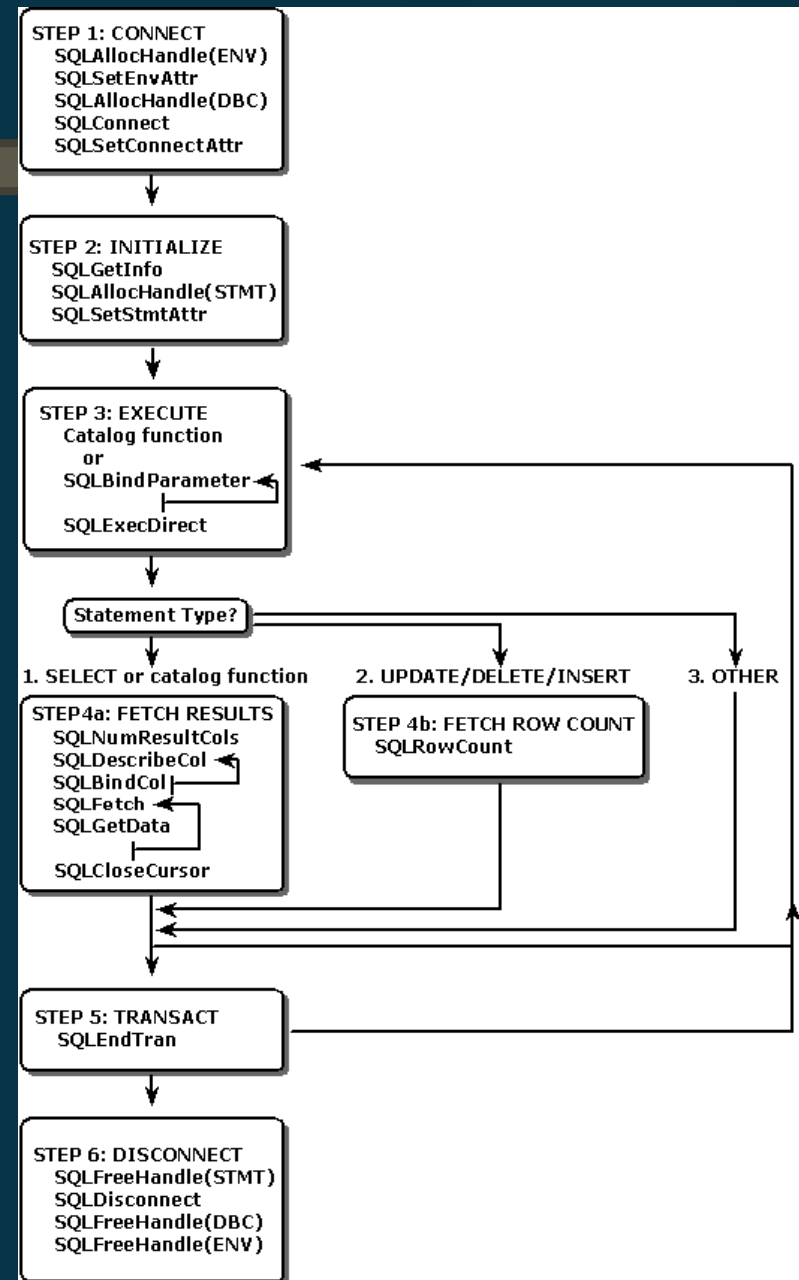
DB Programming

- ODBC (Open Database Connectivity)
 - 세팅



DB Programming

- ODBC
 - SQL 인터페이스와 연결
 - SQL 명령 실행
 - 결과 수집



DB Programming

- ODBC

```
#include <sqlext.h>

RETCODE rc;    // ODBC return code
HENV henv;     // Environment
HDBC hdbc;     // Connection handle
HSTMT hstmt;   // Statement handle

SQLAllocEnv(&henv);
SQLAllocConnect(henv, &hdbc);
rc = SQLConnect(hdbc, "mydb", SQL_NTS, NULL, 0, NULL, 0);

rc = SQLAllocStmt(hdbc, &hstmt);
```

DB Programming

- ODBC

```
SQLHSTMT      hstmt1;
SQLUINTEGER   OrderID;
SQLINTEGER     OrderIDInd = 0;

// Prepare a statement to delete orders from the Orders table.
SQLPrepare(hstmt1, "DELETE FROM Orders WHERE OrderID = ?",
SQL_NTS);

// Bind OrderID to the parameter for the OrderID column.
SQLBindParameter(hstmt1, 1, SQL_PARAM_INPUT, SQL_C_ULONG,
SQL_INTEGER, 5, 0, &OrderID, 0, &OrderIDInd);

// Repeatedly execute hstmt1 with different values of OrderID.
while ((OrderID = GetOrderID()) != 0) {
    SQLExecute(hstmt1);
}
```

DB Programming (2023 화목)

- Stored Procedure (내장 함수)
 - Stored Procedure를 사용하지 않고 SQL을 직접 사용하는 것은 최악 – from NDC2008
 - 일련의 동작들을 SQL로 프로그래밍 해서 DB 서버에 저장해 놓은 것
 - 장점
 - 성능향상 : Pre-compile, Pre-scheduling
 - 보안
 - Transaction 구현
 - 네트워크 트래픽 감소
 - 사용법
 - EXEC proc_name (param1) (param2) ...

Summary

- 다음 시간
 - DB 프로그래밍 실습
 - 속제