분산 서버 아키텍처에서의 DB Server

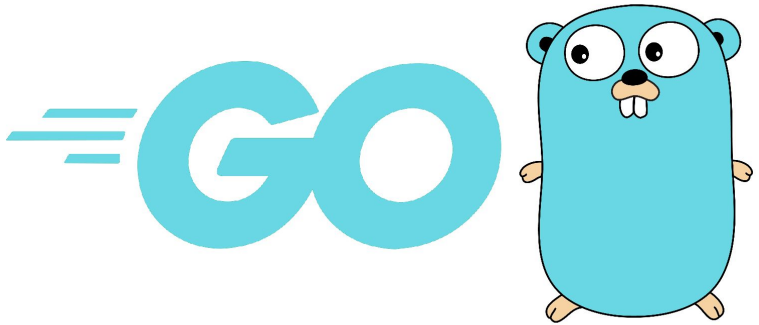
센트럴서버실
기술지원팀
최흥배 수석

DB Server의 장점

- 서버 부하 분산을 할 수 있다
- 데이베이스에 많은 연결이 발생하는 것을 방지할 수 있다
- 게임 서버의 언어나 플랫폼에 독립적으로 구현할 수 있다
- 게임 서버 개발에 영향을 주지 않으면서 개발할 수 있다
- 쉽게 DB 관련 기능을 추가할 수 있다

DB 프로그래밍에 추천하는 언어로는 C#, Golang

언어도 성숙 되었고, DB 프로그래밍 관련 라이브러리가 잘 준비 되어 있다.



C++은 최대한 안하는 것을 추천
너무 비효율 !

```
        // Set login timeout to 5 seconds
        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
           SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (SQLPOINTER)5, 0);

           // Connect to data source
           retcode = SQLConnect(hdbc, (SQLCHAR*) "SQLCMD", SQL_NTS, (SQLCHAR*) "Test1", 5,
(SQLCHAR*) "Password1", 9);

           // Allocate statement handle
           if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
              retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);

              retcode = SQLExecDirect (hstmt, (SQLCHAR *) "SELECT CustomerID, ContactName, Phone
FROM CUSTOMERS ORDER BY 2, 1, 3", SQL_NTS);
              if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {

                 // Bind columns 1, 2, and 3
                 retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, sCustID, 100, &cbCustID);
                 retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, szName, NAME_LEN, &cbName);
                 retcode = SQLBindCol(hstmt, 3, SQL_C_CHAR, szPhone, PHONE_LEN, &cbPhone);

                 // Fetch and print each row of data. On an error, display a message and exit.
                 for (int i=0 ; ; i++) {
                    retcode = SQLFetch(hstmt);
                    if (retcode == SQL_ERROR || retcode == SQL_SUCCESS_WITH_INFO)
                       show_error();
                    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
                       printf( "%d: %s %s %s\n", i + 1, sCustID, szName, szPhone);
                    else
                       break;
                 }
              }
```

https://uple.net/2412

이번은 C#으로 만들어 본다

CloudStructures를 이용한 C# Redis 프로그래밍

- 2020_10-CloudStructures를_이용한_CSharp_Redis_프로그래밍.mp4
- https://gist.github.com/jacking75/5f91f8cf975e0bf778508acdf79499c0

MySqlConnector + Dapper.NET을 이용하여 MySQL 프로그래밍

- 2020_10-CSharp_MySqlConnector_DapperNet.mp4
- MySqlConnector 간단 정리
- Dapper.NET을 이용하여 MySQL 프로그래밍

Game Server와 DB Server 통신 방법

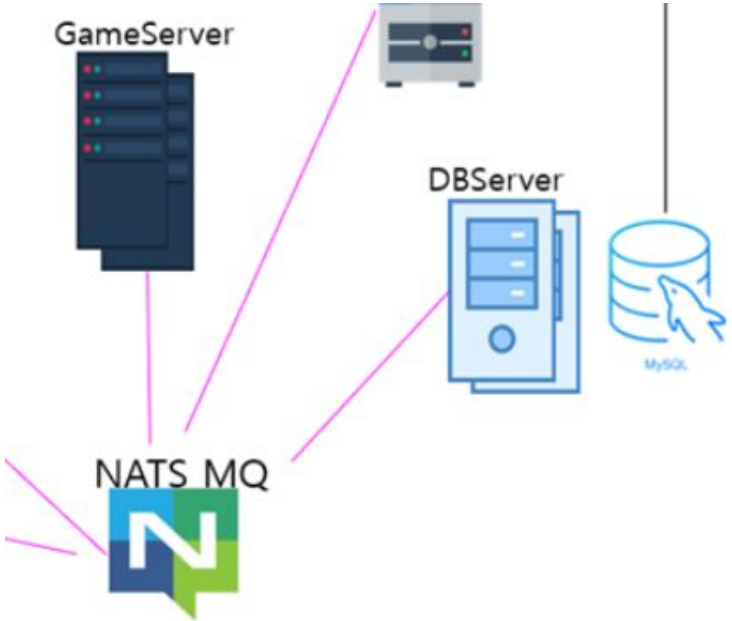
- **MQ**
- Redis
- 소켓으로 직접 통신

https://nats.io/

2020년 9월 월 세션. 분산 서버 구조의 온라인 오목 게임에서 사용

Game Server와 DB Server 간의 데이터 포맷

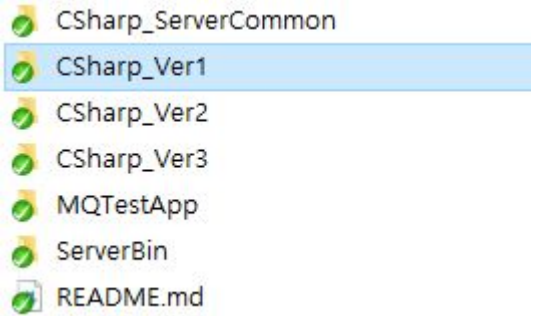
- 게임서버와 DB 서버 간 패킷을 주고 받을 때 사용하는 데이터 포맷
- 양 서버 모두 C#이면 MesagePack-CSharp
- 양 서버의 프로그래밍 언어가 다르다면 Google Protocol Buffers



DB 서버 구현 Type

- SQL Query만
- SQL Query + Cache(로컬)
- SQL Query + Cache(Redis)

Type 1 - SQL Query만



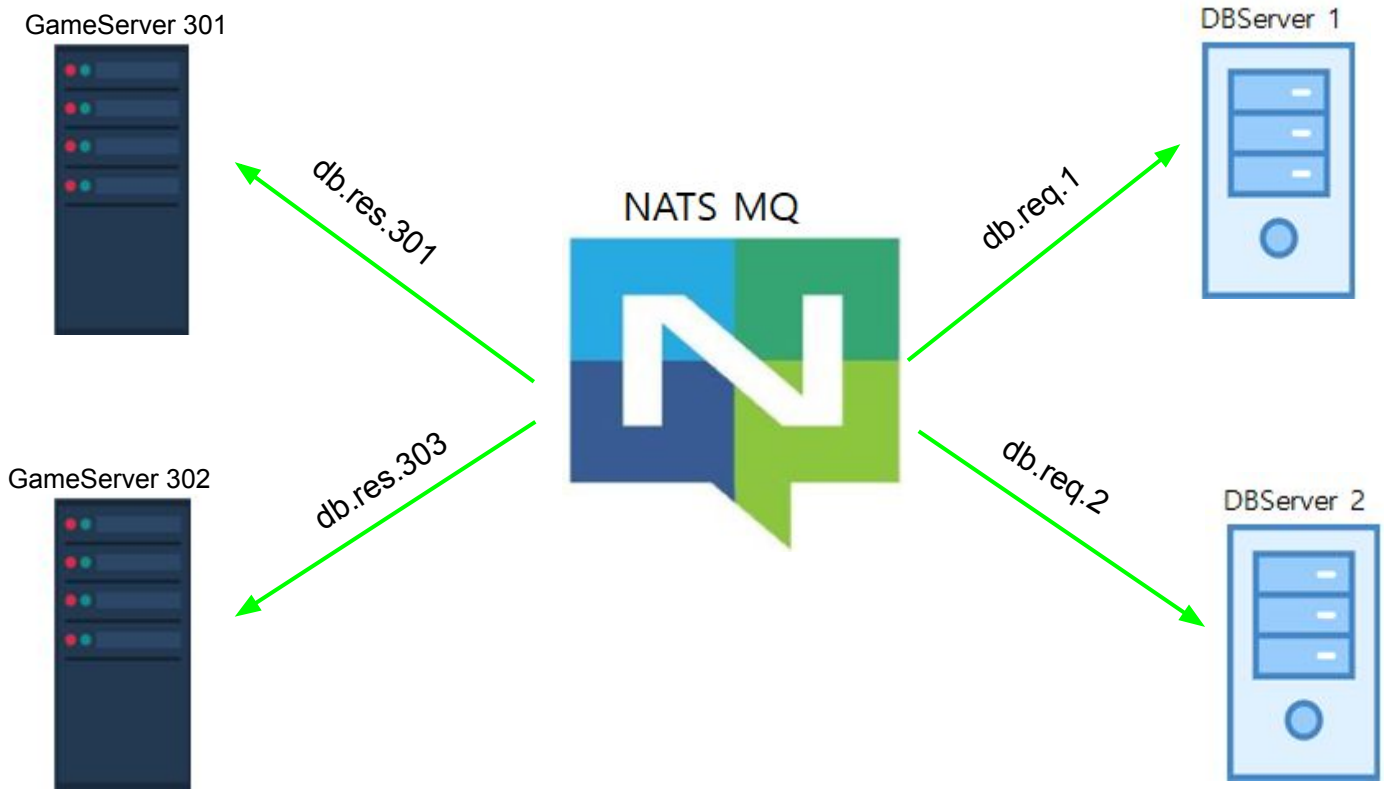
예제에서 사용할 **DB** 스키마

Database: GameDB

Table: OmokGameRecord

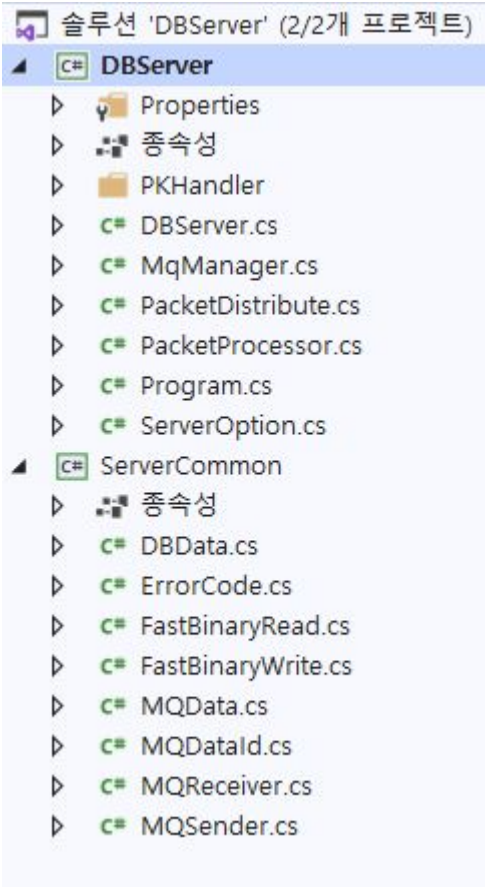
```
create table OmokGameRecord(
        UID bigint unsigned NOT NULL primary key,
        WinCount bigint NOT NULL,
        LoseCount bigint NOT NULL,
        DrawCount bigint NOT NULL
);
```

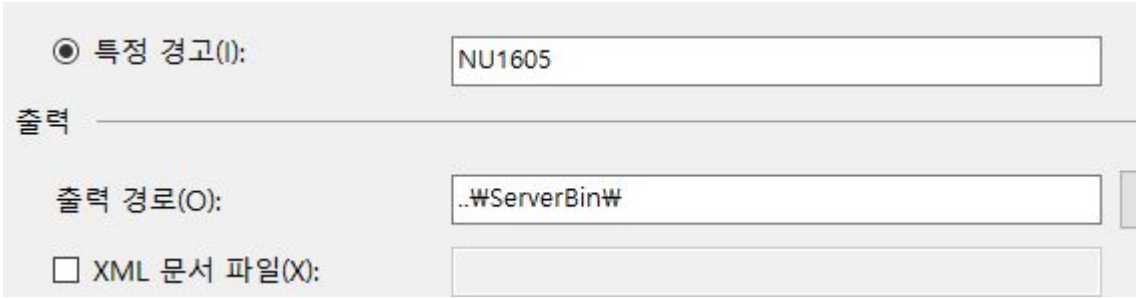
MQ를 사용한다고 가정했을 때의 연결 주소

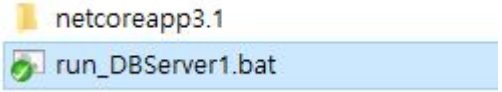


DB Server 인덱스 번호 구간: 1~256

코드 분석







dotnet netcoreapp3.1\DBServer1.dll --serverIndex 1 --name DBServer --threadCount 8 --mqServerAddress 127.0.0.1 --mySqlConnectionString server=127.0.0.1;user=root;password=123qwe();port=3306;database=GameDB;

dotnet netcoreapp3.1\DBServer1.dll --serverIndex 1 --name DBServer --threadCount 8 --mqServerAddress 127.0.0.1 --mySqlConnectionString server=127.0.0.1;user=root;password=123qwe();port=3306;database=GameDB;

```
using CommandLine;
using System;

namespace DBServer
{
    참조 6개
    public class ServerOption
    {
        [Option( "serverIndex", Required = true, HelpText = "DB Server Index")]
        참조 2개
        public UInt16 Index { get; set; }

        [Option("name", Required = true, HelpText = "Server Name")]
        참조 0개
        public string Name { get; set; }

        [Option("threadCount", Required = true, HelpText = "Max Packet Thread Count")]
        참조 1개
        public int ThreadCount { get; set; } = 0;

        [Option("mqServerAddress", Required = true, HelpText = "MQ Server Address")]
        참조 1개
        public string MQServerAddress { get; set; }

        [Option("mySqlConnectionString", Required = true, HelpText = "mySqlConnectionString")]
        참조 1개
        public string MySqlGameConnectionString { get; set; }
    }
}
```

찾아보기 설치됨 업데이트

검색(Ctrl+L) 시험판 포함

CommandLineParser 작성자: gsscoder,nemec,ericnewton76,moh-hassan

Terse syntax C# command line parser for .NET. For FSharp support see Comm
for manipulating command line arguments and related tasks.

Program.cs

```
using ZLogger;

using System;

namespace DBServer
{
    참조 5개
    class Program
    {
        public static ILogger GlobalLogger;

        참조 0개
        static void Main(string[] args)
        {
            GlobalLogger = CreateLogger();

            var serverOption = ParseCommandLine(args);

            var serverApp = new DBServer();
            serverApp.Start(serverOption);

            GlobalLogger.LogInformation("Start DBServer !");
            GlobalLogger.LogInformation("Press q to shut down the server");


            while (true)
            {
                System.Threading.Thread.Sleep(128);

                ConsoleKeyInfo key = Console.ReadKey(true);
                if (key.KeyChar == 'q')
                {
                    GlobalLogger.LogInformation("Server Stop ~~~");

                    serverApp.Stop();
                    break;
                }
                else
                {
                    GlobalLogger.LogInformation($"Preessed key:{key.KeyChar}");
                }
            }

            GlobalLogger.LogInformation("Server Terminate ~~~");
        }
```

DBServer.cs

```
using System.Text;

using ServerCommon;

namespace DBServer
{
    참조 2개
    class DBServer
    {
        public static ServerOption ServerOption;
        MqManager MQMgr = new MqManager();

        PacketDistributor Distributor = new PacketDistributor();

        참조 1개
        public ErrorCode Start(ServerOption option)
        {
            ServerOption = option;

            MQMgr.Init(ServerOption.MQServerAddress, ServerOption.Index, ReceivedMQData);

            Distributor.CreateAndStart(option);

            PacketProcessor.MQSendFunc = SendMqData;

            return ErrorCode.None;
        }


        참조 1개
        public void Stop()
        {
            Program.GlobalLogger.LogInformation("Server Stop <<<<");

            MQMgr.Destory();

            Distributor.Destory();

            Program.GlobalLogger.LogInformation("Server Stop >>>");
        }

        참조 1개
        void ReceivedMQData(byte[] data)
        {
            Distributor.Distribute(data);
        }

        참조 1개
        void SendMqData(int tagetServerIndex, byte[] data)
        {
            MQMgr.SendMQ($"db.res.{tagetServerIndex}", data);
        }


    }
}
```

PacketDistribute.cs

DBServer

DBServer.PacketDistributor

```
    참조 2개
11  public class PacketDistributor
12  {
13      List<PacketProcessor> PacketProcessorList = new List<PacketProcessor>();
14
15      int ThreadCount;
16
17      int PacketNumber;
18
19
20
        참조 1개
21      public void CreateAndStart(ServerOption option)
22      {
23          Program.GlobalLogger.LogInformation("[PacketDistributor.CreateAndStart] Start");
24
25          ThreadCount = option.ThreadCount;
26          PacketNumber = 0;
27          PKHandler.Base.DBServerIndex = option.Index;
28
29          for (int i = 0; i < ThreadCount; ++i)
30          {
31              var packetProcess = new PacketProcessor();
32              packetProcess.CreateAndStart();
33              PacketProcessorList.Add(packetProcess);
34          }
35
36          Program.GlobalLogger.LogInformation("[PacketDistributor.CreateAndStart] End");
37      }
38
        참조 1개
39      public void Destory()
40      {
41          PacketProcessorList.ForEach(preocess => preocess.Destory());
42          PacketProcessorList.Clear();
43      }
```

PacketProcessor.cs

DBServer | DBServer.PacketProcessor

```
참조 6개
class PacketProcessor
{
    bool IsThreadRunning = false;
    System.Threading.Thread ProcessThread;

    BufferBlock<byte[]> MsgBuffer = new BufferBlock<byte[]>();

    Dictionary<UInt16, PKHandler.Base> PacketHandlerMap = new Dictionary<UInt16, PKHandler.Ba

    public static Action<int, byte[]> MQSendFunc;

    MySqlConnection MysqlConnection = new MySqlConnection(DBServer.ServerOption.MySqlGameCon

    참조 1개
    public void CreateAndStart()
    {
        RegistPacketHandler();

        IsThreadRunning = true;
        ProcessThread = new System.Threading.Thread(this.Process);
        ProcessThread.Start();
    }

    참조 1개
    public void Destory()...


    참조 1개
    public void InsertMsg(byte[] mqData) => MsgBuffer.Post(mqData);


    참조 1개
    void Process()
    {
        MysqlConnection.Open();
        while (IsThreadRunning)
        {
            try
            {
                var mqData = MsgBuffer.Receive();
                var mqHeader = new MQPacketHeadInfo();
                mqHeader.Read(mqData);

                var mqId = mqHeader.Id;

                if (PacketHandlerMap.ContainsKey(mqId))
                {
                    PacketHandlerMap[mqId].Process(mqHeader, mqData);
                }
```

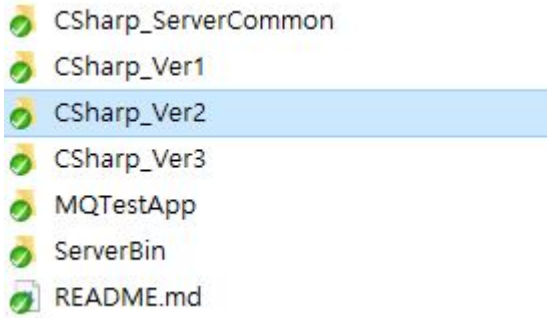
GameRecord.cs

DBServer

DBServer.PKHandler.GameRecord

```
11  namespace DBServer.PKHandler
12  {
        참조 2개
13      class GameRecord : Base
14      {
            참조 1개
15          public GameRecord(MySqlConnection mysqlConnection)
16          {
17              MySqlConnection = mysqlConnection;
18          }
19
            참조 3개
20          public override void Process(MQPacketHeadInfo mqHead, byte[] mqData)
21          {
22              try
23              {
24                  ProcessImpl(mqHead, mqData);
25              }
26              catch (Exception ex)
27              {
28                  Console.WriteLine(ex.ToString());
29              }
30          }
31
32
            참조 1개
33          void ProcessImpl(MQPacketHeadInfo mqReqHeader, byte[] mqReqData )
34          {
35              var reqServerIndex = mqReqHeader.SenderIndex;
36              var reqUserUniqueId = mqReqHeader.UserUniqueId;
37
38              var reqData = MessagePackSerializer.Deserialize<MQReqGameRecord>(mqReqData);
39
40              var reply = MySqlConnection.Query<MQResGameRecord>("select WinCount,LoseCount,DrawC
41
42
43              var resPacket = MessagePackSerializer.Serialize(reply);
44
45              var mqresHeader = new MQPacketHeadInfo();
46              mqresHeader.Id = (UInt16)MqPacketId.MQ_RES_GAME_RECORD;
47              mqresHeader.SenderIndex = Base.DBServerIndex;
48              mqresHeader.UserUniqueId = reqUserUniqueId;
49              mqresHeader.Write(resPacket);
50
51              PacketProcessor.MQSendFunc(reqServerIndex, resPacket);
52
53          }
```

Type 2 - SQL Query + Cache

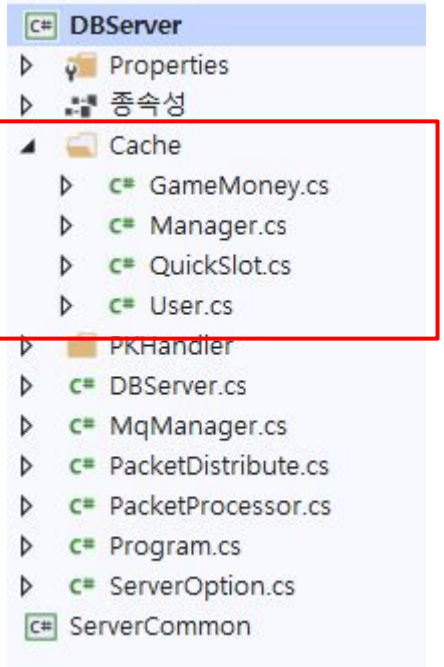


원칙적으로 Cache 기능은 사용하지 않는 것을 추천
여기에 버그가 발생하면 큰 문제가 된다

그러나 DB 부하가 성능에 큰 이슈가 된다면 사용할 수 밖에 없음...

- 유저가 게임서버에 접속하면 게임 서버에서 메모리에 저장할 데이터를 로딩한다.
  - 이 데이터가 주로 DB Server에 캐시로 저장된다.
  - 만약 로딩할 데이터가 많으면 한번의 요청으로 다 로딩하지 말고, 나누어서 로딩한다.
- 유저는 자신의 게임 데이터를 다 로딩을 한 후에 게임을 시작할 수 있다.
- 원칙적으로 게임서버는 유저의 DB처리는 순차적으로 요청한다(요청에 대한 결과를 받으면 다음 요청을 한다)
- 로그아웃 하기 전까지는 cache에 데이터를 저장하고, 정해진 정책에 의해 DB에 저장한다.
  - 로그아웃을 할 때는 모든 데이터를 다 DB에 저장한다.
- 유저의 cache 데이터가 동시에 변경될 수 있는 경우는 lock을 건다.
  - 대부분은 lock을 사용하지 않을 것이다.

코드 분석




```
namespace DBServer.Cache
{
    참조 5개
    public class User
    {
        object LockObj = new object();

        public GameMoney GameMoneyObj = new GameMoney();
        public QuickSlot QuickSlotObj = new QuickSlot();


        참조 0개
        public void Lock()
        {
            System.Threading.Monitor.Enter(LockObj);
        }

        참조 0개
        public void UnLock()
        {
            System.Threading.Monitor.Exit(LockObj);
        }

        참조 1개
        public void SaveDB(MySqlConnection sqlDB)
        {
            // 이 함수는 로그아웃할 때 사용할 예정이므로 lock을 걸지 않는다.
        }
    }
}
```

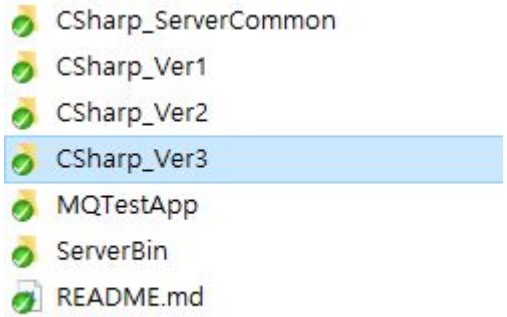
```
namespace DBServer.Cache
{
    참조 9개
    public class Manager
    {
        ConcurrentDictionary<UInt64, User> Users = new ConcurrentDictionary<

        참조 1개
        public User AddUser(UInt64 uid)
        {
            var user = new User();

            if (Users.TryAdd(uid, user) == false)
            {
                return null;
            }

            return user;
        }

        참조 0개
        public void RemoveUser(UInt64 uid)
        {
            Users.TryRemove(uid, out var temp);
        }


        참조 3개
        public User GetUser(UInt64 uid)
        {
            if(Users.TryGetValue(uid, out var user))
            {
                return user;
            }

            return null;
        }
```

```
namespace DBServer.Cache
{
    // 돈, 다이아몬드 등의 재화

    // DB 저장 정책
    // 다이아몬드: 즉시
    // 돈: 이전 저장에 비해 지정 크기 이상의 변동이 있으면 저장한다.

    참조 2개
    public class GameMoney
    {
        Int32 MAX_CHANGE_MONEY = 10_000;

        참조 3개
        public Int64 OldMoney { get; private set; }
        참조 6개
        public Int64 Money { get; private set; }
        참조 4개
        public Int32 Diamond { get; private set; }

        참조 1개
        public void Set(Int64 money, Int32 diamond)
        {
            OldMoney = Money = money;
            Diamond = diamond;
        }
```

```
class LoadUserGameData : Base
{
    참조 1개
    public LoadUserGameData(MySqlConnection mysqlConnection, Cach
    {
        MySqlConnection = mysqlConnection;
        CacheMgr = cacheMgr;
    }

    참조 4개
    public override void Process(MQPacketHeadInfo mqReqHeader, by
    {
        try...
    }

    참조 1개
    void ProcessImpl(MQPacketHeadInfo mqReqHeader, byte[] mqData)
    {
        ReqServerIndex = mqReqHeader.SenderIndex;
        ReqUserUniqueId = mqReqHeader.UserUniqueId;

        var reqData = MessagePackSerializer.Deserialize<MQReqUser

        //TODO MySQL에서 유저의 게임 데이터를 읽어 온다

        Int64 userMoney = 23456;
        Int32 userDiamond = 110;
        var slot1 = new DBSlotInfo() { Index = 0, SkillCode = 21
        var slot2 = new DBSlotInfo() { Index = 3, SkillCode = 34
        var slot3 = new DBSlotInfo() { Index = 7, SkillCode = 55


        var user = CacheMgr.AddUser(reqData.UID);
        user.GameMoneyObj.Set(userMoney, userDiamond);
        user.QuickSlotObj.UpdateSlot(slot1);
        user.QuickSlotObj.UpdateSlot(slot2);
        user.QuickSlotObj.UpdateSlot(slot3);


        var resData = new MQResUserGameDataLoad();
        resData.Result = ErrorCode.None;
```

```
class BuyItem : Base
{
    참조 1개
    public BuyItem(MySqlConnection mysqlConnection, Cache.Manager cacheMg
    {
        MySqlConnection = mysqlConnection;
        CacheMgr = cacheMgr;
    }

    참조 4개
    public override void Process(MQPacketHeadInfo mqReqHeader, byte[] mqD


    참조 1개
    void ProcessImpl(MQPacketHeadInfo mqReqHeader, byte[] mqData)
    {
        ReqServerIndex = mqReqHeader.SenderIndex;
        ReqUserUniqueId = mqReqHeader.UserUniqueId;

        var reqData = MessagePackSerializer.Deserialize<MQReqBuyItem>(mqD

        var userCacheData = CacheMgr.GetUser(ReqUserUniqueId);
        if(userCacheData == null)
        {
            SendRespnsePacket(ErrorCode.BuyItem_InvalidUser, reqData.Iter
            return;
        }


        var item = GetItem(reqData.ItemCode);
        var newItemUID = InsertItem(reqData.UID, reqData.ItemCode);

        var moneyUpdateRet = userCacheData.GameMoneyObj.UpdateMoney(item
        var diamondUpdateRet = userCacheData.GameMoneyObj.UpdateDiamond(

        if(moneyUpdateRet == Cache.DBPolicy.UPDATE ||
            diamondUpdateRet == Cache.DBPolicy.UPDATE)
        {
            userCacheData.GameMoneyObj.SaveDB(MySqlConnection);
        }

        SendRespnsePacket(ErrorCode.None, reqData.ItemCode, newItemUID);
```

Type 3 - SQL Query + Cache(Redis)



- 유저의 cache 데이터를 Redis에 저장한다.
- 상대적으로 Redis가 DB Server 보다 더 안전해서, 혹시 DB Server가 죽더라도 cache 데이터를 안전하게 DB에 저장할 수 있다.

```
public DBPolicy UpdateMoneyDiamond(UInt64 uid, Int64 money, Int32 diamond)
{
    var redisId = new RedisString<CacheGameMoney>(Manager.RedisConn,
      RedisKeyGameMoney, null);
    var cacheData = redisId.GetAsync().Result.Value;

    var dbPolicy = DBPolicy.PASS;

    var moneyUpdate = DBPolicy.PASS;
    if (money ≠ 0)
    {
        moneyUpdate = UdateMoney(cacheData, money);
    }

    if(diamond ≠ 0)
    {
        cacheData.Diamond += diamond;
    }


    if(moneyUpdate == DBPolicy.UPDATE || diamond ≠ 0)
    {
        OldMoney = cacheData.Money;
        SaveDB(SqlDBConn);
        dbPolicy = DBPolicy.UPDATE;
    }

    SetCache(uid, money, diamond);

    return dbPolicy;
}
```