□ OLAP Write-Back(배분 기능) 설명서

1. 컨트롤 기본 옵션의 설정하기

OlapGrid에서 Write-Back기능을 사용하시려면 아래의 옵션들을 통해서 기능을 활성화 하고 상세 동작을 제어하실 수 있습니다.

객체	하위 객체	속성명	설명
OlapGrid	Options	(bool) EnableWriteBack	Write-Back 기능을 활성화 할지 여부를 설정 합니다.
		(bool) EnableCreateRecord	입력 대상 셀에 레코드가 없는 경우 편집을 허용하고 해당 셀에 헤더(Row, Column)의 텍스트를 기준으로 자동으로 레코드를 생성할지 여부를 설정합니다. (이 설정이 false 일 경우 데이터가 없는 셀은 수정하실 수 없으며, true일 경우 서버에서 배분 시 자동으로 생성된 레코드는 현재 행/열에 배치된 항목의 값만 설정됩니다.)
		(bool) ManualUpdate	사용자가 데이터를 변경할 경우 서버에서 처리하는 자동 배분 기능을 수동으로 동작할지 여부를 설정합니다. (데이터가 많을 경우 이 옵션을 활성화하고 여러 개의 셀의 값을 편집한 뒤 "CalculateWriteBack" 함수를 호출하여 수정된 셀 모두를 한번에 서버에서 계산하도록 하여 편집 속도를 개선할 수 있습니다.)

<샘플 소스>

```
var OlapGrid = Matrix.getObject("OlapGrid");

OlapGrid.Options.EnableWriteBack = true; //Write-back 활성화
OlapGrid.Options.EnableCreateRecord = false; //레코드가 없는 셀의 편집및 레코드 자동 생성 여부
OlapGrid.Options.ManualUpdate = false; //수동 계산 실행 여부
```

2. 필드별 상세 배분 옵션 설정하기

데이터셀을 수정 시 수정된 값을 데이터셀의 상세 레코드에 배분작업을 하기위한 규칙은 아래의 옵션으로 조정 가능합니다.

객체	하위 객체	속성명	설명
OlapGrid	OlapField	(int) EditMethod	수정된 데이터를 레코드별로 배분하는 방식을 설정합니다. (3. 배분 방식별 설명을 참조) 1 : 가중치 배분 2 : 균등배분 3 : 가중치 배분(force) 4 : 균등배분 배분(force)
		(int) EditPrecision	데이터 배분 시 데이터의 최대 소수점 자릿 수를 설정합니다.
		(string) EditMethodRef	배분 방식이 가중치일 경우 가중치에 해당하는 값을 가지는 필드의 이름을 설정합니다.

<샘플 소스>

```
var fld = OlapGrid.getField("M1")
fld.EditMethod = 1; //가중치 배분
fld.EditPrecision = 0; //소수점 자리수(정수만 허용)
fld.EditMethodRef = "M2";//참조 필드(가중치 기준 필드)
```

3. 배분 방식별 처리 규칙

3.1. 가중치 배분

가중치 배분에서 가중치 기준 필드를 정의하지 않은 경우는 수정 대상 필드가 가지고 있는 값을 기준으로 가중치를 계산 합니다.

<수정 전 데이터>

시도	구군	매출 목표
서울 합계		1,500
서울	강남구	500
	서초구	400
	관악구	300
	도봉구	200
	노원구	100

<수정 값>

2,000

<증감>

500

<계산 수식>

{가중치}	{배분 값}	{계산 결과}
={구군별 값}/{서울 합계}	={증감} * {가중치}	={원본} + {배분 값}
0.33	167	667
0.27	133	533
0.20	100	400
0.13	67	267
0.07	33	133
		2,000

3.2. 균등 배분

균등 배분은 증감한 값을 레코드 수량으로 나누어서 배분하며, 이때 배분 이후에 값이 음수가 되지 않도록 자동 조정 합니다.

<수정 전 데이터>

시도	구군	매출 목표
서울 합계		1,500
서울	강남구	500
	서초구	400
	관악구	300
	도봉구	200
	노원구	100

<수정 값>

2,000

<증감>

500

<계산 수식>

{배분 값}	{계산 결과}
={증감} / {레코드 수량}	={원본} + {배분 값}
100	600
100	500
100	400
100	300
100	200
	2,000

3.3. 가중치 배분 (force)

가중치 배분을 수행하되 수정하기 전의 값은 무시되고 변경된 전체 값을 기준으로 재 배분 합니다.

<수정 전 데이터>

시도	구군	매출 목표
서울 합계		1,500
	강남구	500
	서초구	400

<수정 값>

2,000

<계산 수식>

{가중치}	{배분 값}	{계산 결과}
={구군별 값}/{서울 합계}	={수정값} * {가중치}	={배분 값}
0.33	667	667
0.27	533	533

서울	관악구	300	0.20	400	400
	도봉구	200	0.13	267	267
	노원구	100	0.07	133	133
					2,000

3.4. 균등 배분 (force)

수정된 최종 값으로 전체 레코드에 동일한 값으로 배분하며, 수정되기 전의 값은 무시되고 최종 값으로 변경됩니다.

<수정 전 데이터>

시도	구군	매출 목표
서울 합계		1,500
서울	강남구	500
	서초구	400
	관악구	300
	도봉구	200
	노원구	100

<수정 값>

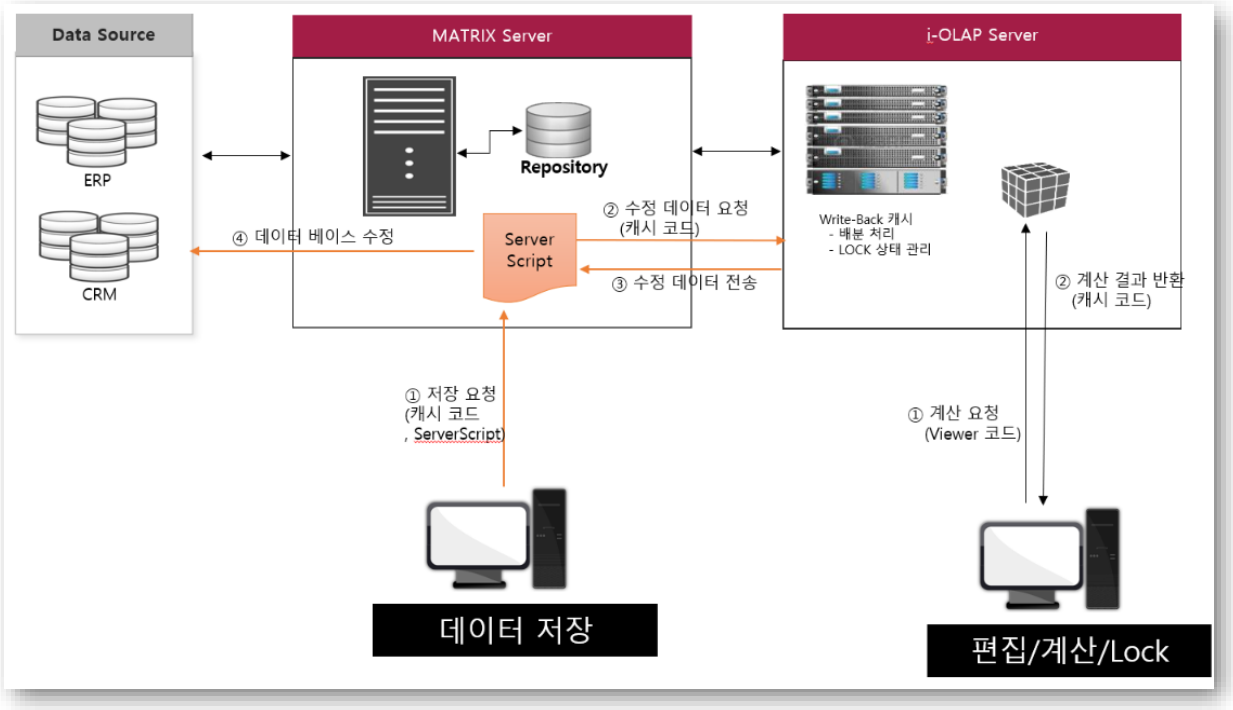
2,000

<계산 수식>

{배분 값}	{계산 결과}
={수정값} / {레코드 수량}	={배분 값}
400	400
400	400
400	400
400	400
400	400
	2,000

4. Write-Back 연산 및 저장

Write-Back의 연산및 상태 관리는 서버에 캐시 파일로 관리하며, 해당 캐시는 현재 사용자의 OlapGrid별로 고유한 코드로 관리됩니다.
캐시는 사용자가 새로 조회를 하면 다시 생성됩니다. 새로고침 전에 피벗팅, 필터링 등의 OlapGrid 고유 작업시에는 캐시가 유지 됩니다.
데이터베이스에 최종 수정을 위해서는 해당 business를 ServerScript로 작성해야 합니다.



4. Server Script 작성 가이드

수정된 데이터를 대상 데이터 베이스에 업데이트 하기 위해서는 ServerScript를 작성하시면 되며,
이때 주의 사항은 수정된 데이터의 레코드 개수가 많을 수 있으므로 전체를 메모리에 올리지 않고 행별로 SQL을 수행 합니다.

Target DataBase 연결 및 Transactionsaction 시작

OlapGrid 컨트롤에서 수정한 데이터 테이블 가져오기

OlapGrid에서 수정한 데이터는 개수가 많을 수가 있어 전체를 메모리에 올리지 않고 행별로

```
1
2   var req = Matrix.getRequest(); // request
3   var con = Matrix.getConnection(); // dbms connection
4   var util = Matrix.getUtility();
5   var gen = Matrix.getQueryGenerator(); // query generator
6   var sql = "";
7   var status = "";
8   var row = null;
9   var stmt = null;
10  var stmtInsert = null;
11  var ROW_IDX  = 0;
12  try{
13      //connection
14      con.Connect("MTXRPTY");// set target dbms connection code
15      con.BeginTransaction();  // begin transaction
16
17      //------------------------------------------------------
18      // save table data
19      //------------------------------------------------------
20      var table = req.getTable("OlapGrid"); //get grid's work data
21
22      sql = "UPDATE MEX_USER_CRUD_DATA SET  M1=?, M2=?, M3=?  WHERE P1=?";
23      stmt = con.PreparedStatement(sql);
24
25      sql = " INSERT INTO MEX_USER_CRUD_DATA(P1,D1,D2,D3,D4,D5,D6,D7,D8,M1,M2,M3) "
26          + "                    VALUES (?, ?,?,?,?,?,?,?,?,?,?,?)";
27      stmtInsert    = con.PreparedStatement(sql);
28
29      /* OLAP은 데이터가 많을 수 있으므로 Callback를 수행한다.  */
30      table.FetchRows(CALL_BACK(function(row){
31          ROW_IDX ++;
32          status = row.getRowStatus();
```

처리 합니다.

```
33          var IDX = 0;
34          if(status == "U"){
35              stmt.setDouble(++IDX , row.getDouble("M1"));
36              stmt.setDouble(++IDX , row.getDouble("M2"));
37              stmt.setDouble(++IDX , row.getDouble("M3"));
38              stmt.setString(++IDX , row.getString("P1"));
39              stmt.addBatch();
40          }
41          else if(status == "N"){
42              stmtInsert.setString(++IDX , util.getUniqueKey("P")); //primary key
43              stmtInsert.setString(++IDX , row.getString("D1"));
44              stmtInsert.setString(++IDX , row.getString("D2"));
45              stmtInsert.setString(++IDX , row.getString("D3"));
46              stmtInsert.setString(++IDX , row.getString("D4"));
47              stmtInsert.setString(++IDX , row.getString("D5"));
48              stmtInsert.setString(++IDX , row.getString("D6"));
49              stmtInsert.setString(++IDX , row.getString("D7"));
50              stmtInsert.setString(++IDX , row.getString("D8"));
51              stmtInsert.setDouble(++IDX , row.getDouble("M1"));
52              stmtInsert.setDouble(++IDX , row.getDouble("M2"));
53              stmtInsert.setDouble(++IDX , row.getDouble("M3"));
54              stmtInsert.addBatch();
55          }
56          return null;
57      }));
58
59      stmt.executeBatch();
60      stmtInsert.executeBatch();
61
62      // COMMIT
63      con.CommitTransaction();
64      con.DisConnect();
65      con = null;
66  }catch(e){
67      Matrix.WriteLog("ERROR" + e.message);
68      if(con != null){
69          try{
70              con.RollBackTransaction();
71              con.DisConnect();
72              con = null;
73          }catch(e){
74          }
75      }
76      Matrix.ThrowException("Server Exception:" + e.message);
77  }finally{
78      if(stmt != null){
79          try{
80              stmt.Close();
81              stmt = null;
82          }catch(e){
83          }
84      }
85      if(con != null){
86          try{
87              con.DisConnect();
88              con = null;
89          }catch(e){
90          }
91      }
92  }
```

Transaction Comit 및 데이터 베이스 연결 종료

end of document