[JQGrid 변경 감지/저장/배경색 표시 오작동 수정 요청]

개발환경:

- OS: Windows
- IDE: STS 4.5
- Java: Zulu-8 (zulu8.88.0.19)
- WAS: Tomcat 9.0.99
- DB: MSSQL 2019 (15.0.4415.2, RTM, Standard Edition 64-bit)
- Build: Maven
- ORM: MyBatis
- Grid: JQGrid

Requests:

```
  A. Please find and fix what is causing the "No modifications made" warning to
pop up incorrectly, and fix the whole flow so that if any cells are actually
changed, they are saved to the server correctly.
  B. Please fix the Contact Email/Salesperson Email columns so that when they are
modified, the background color of the row changes immediately/consistently.
  C. Please make the background color reflect the same when checking/unchecking
the sending status (CUST\_SENDMAIL\_YN / SALESREP\_SENDMAIL\_YN).
  D. Please clarify how changes are detected (dirty flag inside the grid, savedRow
comparison, getChangedCells, inline editing events, etc.
  E. Please present the required code modifications in the form of **complete code
blocks** for both frontend (Grid/JS) and backend (Controller/MyBatis/SQL).
  F. Please include unit test/acceptance test scenarios (reproduce → save →
reflect colors).
```

Constraints/Preferences:

- Keep the grid UI as JQGrid.
- Change detection: dirty tracking based on exit from cell edit mode (inline/cell)
- Save: partial save (change rows only), show error status in grid if server validation fails

Acceptance Criteria (AC):

1. after editing email/sent status, pressing save reflects actual DB (check with network tab & DB lookup).
2. changed rows are highlighted (background) before saving, de-highlighted after successful save.
3. "Nothing modified" warning pops up only when there is no change.

Sources:

```
<%@page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ include file="/WEB-INF/views/include/admin/commonimport.jsp" %>
```

```
<!DOCTYPE html>
<html>
<head>
<%@ include file="/WEB-INF/views/include/admin/commonhead.jsp" %>

<script type="text/javascript" src="${url}/include/js/common/select2/select2.js">
</script>
<link rel="stylesheet" href="${url}/include/js/common/select2/select2.css" />

<script type="text/javascript">
// 이메일 형식 유효성 검사 함수
function validateEmail(email) {
    const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
    return emailRegex.test(email);
}

// 배경색 변경 함수
function changeRowBackground(rowId, isChanged) {
    if (isChanged) {
        $('#gridList #' + rowId).css('background-color', '#ffebcd'); // 연한 주황색
    } else {
        $('#gridList #' + rowId).css('background-color', ''); // 원래 색상
    }
}

// 원본 데이터 저장용 전역 변수
var originalRowData = {};

    //
================================================================================
    // jqGrid Columns Order 설정
    //
================================================================================
//Start. Setting Jqgrid Columns Order.
    var ckNameJqGrid = 'admin/customer/customerList/jqGridCookie'; // 페이지별 쿠키
명 설정
    ckNameJqGrid += '/gridList'; // 그리드명별 쿠키명 설정

var globalColumnOrderStr = toStr(decodeURIComponent(getCookie(ckNameJqGrid)));
var globalColumnOrder = globalColumnOrderStr.split(',');

var defaultColModel = [
    {name:"CUST_CD", key:true, label:'거래처코드', width:120, align:'center',
sortable:true},
    {name:"CUST_NM", label:'거래처명', width:220, align:'left', sortable:true},
    {name:"CUST_MAIN_PERSON", label:'담당자', width:100, align:'center',
sortable:true},
    {name:"CUST_MAIN_EMAIL", label:'담당자 이메일', width:220, align:'center',
sortable:true, editable:true, editoptions:{dataEvents:[{type:'blur',
fn:validateEmailField}]}},
    {name:"CUST_SENDMAIL_YN", label:'발송 여부', width:100, align:'center',
sortable:true, formatter:checkboxFormatter, editable:true, edittype:'checkbox',
editoptions:{value:'Y:N', dataEvents:[{type:'change', fn:toggleCheckbox}]}},
    {name:"SALESREP_NM", label:'영업 담당', width:100, align:'center',
```

```
sortable:true},
    {name:"SALESREP_EMAIL", label:'영업 담당 이메일', width:300, align:'center',
sortable:true, editable:true, editoptions:{dataEvents:[{type:'blur',
fn:validateEmailField}]}},
    {name:"SALESREP_SENDMAIL_YN", label:'발송 여부', width:100, align:'center',
sortable:true, formatter:checkboxFormatter, editable:true, edittype:'checkbox',
editoptions:{value:'Y:N', dataEvents:[{type:'change', fn:toggleCheckbox}]}},
    {name:"COMMENTS", label:'비고', width:450, align:'left', sortable:true,
editable:true, editoptions:{dataEvents:[{type:'blur', fn:trackChanges}]}}
];

var defaultColumnOrder = writeIndexToStr(defaultColModel.length);
var updateComModel = [];

// 쿠키에서 컬럼 순서 복원
if (0 < globalColumnOrder.length) { // 쿠키값이 있을때
    if (defaultColModel.length == globalColumnOrder.length) {
        for (var i = 0, j = globalColumnOrder.length; i < j; i++) {
            updateComModel.push(defaultColModel[globalColumnOrder[i]]);
        }
        setCookie(ckNameJqGrid, globalColumnOrder, 365);
    } else {
        updateComModel = defaultColModel;
        setCookie(ckNameJqGrid, defaultColumnOrder, 365);
    }
} else { // 쿠키값이 없을때
    updateComModel = defaultColModel;
    setCookie(ckNameJqGrid, defaultColumnOrder, 365);
}

//
// ============================================================================
// jqGrid Column Width 설정
//
// ============================================================================
var ckNameJqGridWidth = ckNameJqGrid + '/width'; // 페이지별 쿠키명 설정
var globalColumnWidthStr =
toStr(decodeURIComponent(getCookie(ckNameJqGridWidth)));
var globalColumnWidth = globalColumnWidthStr.split(',');
var defaultColumnWidthStr = '';
var defaultColumnWidth;
var updateColumnWidth;

if ('' != globalColumnWidthStr) { // 쿠키값이 있을때
    if (updateComModel.length == globalColumnWidth.length) {
        updateColumnWidth = globalColumnWidth;
    } else {
        for (var j = 0; j < updateComModel.length; j++) {
            if ('rn' != updateComModel[j].name && 'cb' != updateComModel[j].name)
{
                var v = ('' != toStr(updateComModel[j].width)) ?
toStr(updateComModel[j].width) : '0';
                if ('' == defaultColumnWidthStr) {
                    defaultColumnWidthStr = v;
```

```
                } else {
                    defaultColumnWidthStr += ',' + v;
                }
            }
        }
        defaultColumnWidth = defaultColumnWidthStr.split(',');
        updateColumnWidth = defaultColumnWidth;
        setCookie(ckNameJqGridWidth, defaultColumnWidth, 365);
    }
} else { // 쿠키값이 없을때
    for (var j = 0; j < updateComModel.length; j++) {
        if ('rn' != updateComModel[j].name && 'cb' != updateComModel[j].name) {
            var v = ('' != toStr(updateComModel[j].width)) ?
toStr(updateComModel[j].width) : '0';
            if ('' == defaultColumnWidthStr) {
                defaultColumnWidthStr = v;
            } else {
                defaultColumnWidthStr += ',' + v;
            }
        }
    }
    defaultColumnWidth = defaultColumnWidthStr.split(',');
    updateColumnWidth = defaultColumnWidth;
    setCookie(ckNameJqGridWidth, defaultColumnWidth, 365);
}

// 컬럼 너비 적용
if (updateComModel.length == globalColumnWidth.length) {
    for (var j = 0; j < updateComModel.length; j++) {
        updateComModel[j].width = toStr(updateColumnWidth[j]);
    }
}


// 체크박스 포맷터
function checkboxFormatter(cellVal, options, rowObj) {
    var checked = (cellVal === 'Y') ? 'checked' : '';
    return '<input type="checkbox" class="mail-checkbox" ' + checked + '
onchange="handleCheckboxChange(this, \'' + options.rowId + '\', \'' +
options.colModel.name + '\')" />';
}

// 체크박스 변경 처리
function handleCheckboxChange(checkbox, rowId, fieldName) {
    var newValue = checkbox.checked ? 'Y' : 'N';
    var originalValue = originalRowData[rowId] ? originalRowData[rowId][fieldName]
: 'N';

    // 그리드 데이터 업데이트
    $('#gridList').setRowData(rowId, {[fieldName]: newValue});

    // 변경 사항 추적
    checkRowChanges(rowId);
}
```

```
// 이메일 필드 유효성 검사
function validateEmailField(e) {
    var email = $(e.target).val();
    var rowId = $(e.target).closest('tr').attr('id');

    if (email && !validateEmail(email)) {
        alert('올바른 이메일 형식을 입력해주세요.');
        $(e.target).focus();
        return false;
    }

    trackChanges(e);
}

// 체크박스 토글 처리
function toggleCheckbox(e) {
    var rowId = $(e.target).closest('tr').attr('id');
    checkRowChanges(rowId);
}

// 변경 사항 추적
function trackChanges(e) {
    var rowId = $(e.target).closest('tr').attr('id');
    checkRowChanges(rowId);
}

// 행 변경 사항 확인
function checkRowChanges(rowId) {
    if (!originalRowData[rowId]) return;

    var currentRowData = $('#gridList').getRowData(rowId);
    var hasChanges = false;

    // 각 필드별 변경 사항 확인
    for (var field in originalRowData[rowId]) {
        if (originalRowData[rowId][field] !== currentRowData[field]) {
            hasChanges = true;
            break;
        }
    }

    changeRowBackground(rowId, hasChanges);
}

$(function(){
    getGridList();
});

function getGridList(){
    var searchData = getSearchData();
    $('#gridList').jqGrid({
        url: "${url}/admin/customer/getOrderEmailAlarmAjax.lime", // 서버 호출 URL
        editurl: 'clientArray',
```

```
        datatype: "json",
        mtype: 'POST',
        postData: searchData,
        colModel: updateComModel,
        height: '360px',
        autowidth: false,
        multiselect: true,
        rownumbers: true,
        pagination: true,
        pager: "#pager",
        actions : true,
        pginput : true,
        // 열 순서 변경 이벤트
        sortable: {
            update: function(relativeColumnOrder) {
                var grid = $('#gridList');

                // 기본 컬럼 이름 배열
                var defaultColIndicies = [];
                for (var i = 0; i < defaultColModel.length; i++) {
                    defaultColIndicies.push(defaultColModel[i].name);
                }

                // 새로운 컬럼 순서 계산
                globalColumnOrder = [];
                var columnOrder = [];
                var currentColModel = grid.getGridParam('colModel');

                for (var j = 0; j < relativeColumnOrder.length; j++) {
                    // Row 번호(rn)나 Checkbox(cb) 제외
                    if ('rn' != currentColModel[j].name && 'cb' !=
currentColModel[j].name) {

columnOrder.push(defaultColIndicies.indexOf(currentColModel[j].name));
                    }
                }
                globalColumnOrder = columnOrder;

                // 변경된 순서를 쿠키로 저장
                setCookie(ckNameJqGrid, globalColumnOrder, 365);

                // 열 너비도 함께 저장
                var tempUpdateColumnWidth = [];
                for (var j = 0; j < currentColModel.length; j++) {
                    if ('rn' != currentColModel[j].name && 'cb' !=
currentColModel[j].name) {
                        tempUpdateColumnWidth.push(currentColModel[j].width);
                    }
                }
                updateColumnWidth = tempUpdateColumnWidth;
                setCookie(ckNameJqGridWidth, updateColumnWidth, 365);
            }
        },
```

```
        // 열 크기 조정 후 실행되는 이벤트
        resizeStop: function(width, index) {
            console.log('globalColumnOrder : ', globalColumnOrder);
            var minusIdx = 0;
            var grid = $('#gridList');
            var currentColModel = grid.getGridParam('colModel');

            // row number, row checkbox 컬럼이 맨 앞에 있으면 index 조정
            if ('rn' == currentColModel[0].name || 'cb' ==
currentColModel[0].name) minusIdx--;
            if ('rn' == currentColModel[1].name || 'cb' ==
currentColModel[1].name) minusIdx--;

            // 실제 조정된 컬럼 인덱스 계산
            var resizeIdx = index + minusIdx;

            // 변경된 너비 배열 반영
            updateColumnWidth[resizeIdx] = width;

            // 쿠키에 저장
            setCookie(ckNameJqGridWidth, updateColumnWidth, 365);
        },

        sortorder: 'desc', // 정렬 순서 기본값

        jsonReader: {
            root: 'list' // 서버 응답 JSON에서 데이터 배열 경로
        },

        loadComplete: function(data) {
            $('#listTotalCountSpanId').html(addComma(data.listTotalCount));

            // 원본 데이터 저장
            originalRowData = {};
            if (data.list) {
                $.each(data.list, function(i, row) {
                    originalRowData[i+1] = {
                        CUST_MAIN_EMAIL: row.CUST_MAIN_EMAIL || '',
                        CUST_SENDMAIL_YN: row.CUST_SENDMAIL_YN || 'N',
                        SALESREP_EMAIL: row.SALESREP_EMAIL || '',
                        SALESREP_SENDMAIL_YN: row.SALESREP_SENDMAIL_YN || 'N',
                        COMMENTS: row.COMMENTS || ''
                    };
                });
            }
        },
        onSelectRow: function(rowId){
            editRow(rowId);
        }
    });
}

var lastSelection;
function editRow(id){
```

```
    if (id && id !== lastSelection) {
        var grid = $('#gridList');
        grid.jqGrid('editRow',id, {keys: false});
        lastSelection = id;
    }
}

// 저장 처리
function dataSave(obj) {
    $(obj).prop('disabled', true);

    // 수정된 행 찾기
    var modifiedRows = [];
    var grid = $('#gridList');
    var ids = grid.getDataIDs();

    $.each(ids, function(i, rowId) {
        var rowData = grid.getRowData(rowId);
        var original = originalRowData[rowId];

        if (original) {
            var hasChanges = false;
            for (var field in original) {
                if (original[field] !== (rowData[field] || '')) {
                    hasChanges = true;
                    break;
                }
            }
            if (hasChanges) {
                modifiedRows.push(rowId);
            }
        }
    });

    if (modifiedRows.length === 0) {
        alert('수정된 내용이 없습니다.');
        $(obj).prop('disabled', false);
        return false;
    }

    // 유효성 검사
    var validationFailed = false;
    $.each(modifiedRows, function(i, rowId) {
        var rowData = grid.getRowData(rowId);

        // 이메일 형식 검사
        if (rowData.CUST_MAIN_EMAIL && !validateEmail(rowData.CUST_MAIN_EMAIL)) {
            alert('거래처 담당자 이메일 형식이 올바르지 않습니다. (행: ' + rowId +
')');
            validationFailed = true;
            return false;
        }

        if (rowData.SALESREP_EMAIL && !validateEmail(rowData.SALESREP_EMAIL)) {
```

```
            alert('영업 담당 이메일 형식이 올바르지 않습니다. (행: ' + rowId + ')');
            validationFailed = true;
            return false;
        }

        // 발송 여부 체크 시 이메일 존재 확인
        if (rowData.CUST_SENDMAIL_YN === 'Y' && !rowData.CUST_MAIN_EMAIL) {
            alert('담당자 이메일 발송이 체크되어 있지만 담당자 이메일이 비어있습니다.
(행: ' + rowId + ')');
            validationFailed = true;
            return false;
        }

        if (rowData.SALESREP_SENDMAIL_YN === 'Y' && !rowData.SALESREP_EMAIL) {
            alert('영업 담당 이메일 발송이 체크되어 있지만 영업 담당 이메일이 비어있습
니다. (행: ' + rowId + ')');
            validationFailed = true;
            return false;
        }
    });

    if (validationFailed) {
        $(obj).prop('disabled', false);
        return false;
    }

    // 데이터 준비
    var iFormObj = $('form[name="iForm"]');
    iFormObj.empty();

    $.each(modifiedRows, function(i, rowId) {
        var rowData = grid.getRowData(rowId);

        iFormObj.append('<input type="hidden" name="custCd" value="' +
rowData.CUST_CD + '" />');
        iFormObj.append('<input type="hidden" name="custMainEmail" value="' +
(rowData.CUST_MAIN_EMAIL || '') + '" />');
        iFormObj.append('<input type="hidden" name="custSendmailYn" value="' +
(rowData.CUST_SENDMAIL_YN || 'N') + '" />');
        iFormObj.append('<input type="hidden" name="salesrepEmail" value="' +
(rowData.SALESREP_EMAIL || '') + '" />');
        iFormObj.append('<input type="hidden" name="salesrepSendmailYn" value="' +
(rowData.SALESREP_SENDMAIL_YN || 'N') + '" />');
        iFormObj.append('<input type="hidden" name="comments" value="' +
(rowData.COMMENTS || '') + '" />');
    });

    if (confirm('저장 하시겠습니까?')) {
        var iFormData = iFormObj.serialize();
        var url = '${url}/admin/system/updateOrderMailAlarmAjax.lime';
        $.ajax({
            async : false,
            data : iFormData,
            type : 'POST',
```

```
            url : url,
            success : function(data) {
                if (data.RES_CODE == '0000') {
                    alert(data.RES_MSG);
                    dataSearch();
                }else{
                    alert(data.RES_MSG);
                }
                $(obj).prop('disabled', false);
            },
            error : function(request,status,error){
                alert('Error');
                $(obj).prop('disabled', false);
            }
        });
    }else{
        $(obj).prop('disabled', false);
    }
}

function getSearchData(){
    var searchData = {
        custCd : $('input[name="searchCustCd"]').val(),
        custNm : $('input[name="searchCustNm"]').val(),
        salesrepNm : $('input[name="searchSalesrepNm"]').val()
    };
    return searchData;
}

// 조회
function dataSearch() {
    var searchData = getSearchData();
    $('#gridList').setGridParam({
        postData : searchData
    }).trigger("reloadGrid");
}

// 엑셀다운로드
function excelDown(obj){
    $('#ajax_indicator').show().fadeIn('fast');
    var token = getFileToken('excel');
    $('form[name="frm"]').append('<input type="hidden" name="filetoken"
value="'+token+'" />');

    formPostSubmit('frm', '${url}/admin/system/orderMailAlarmExcelDown.lime');
    $('form[name="frm"]').attr('action', '');

    $('input[name="filetoken"]').remove();
    var fileTimer = setInterval(function() {
        if('true' == getCookie(token)){
            $('#ajax_indicator').fadeOut();
            delCookie(token);
            clearInterval(fileTimer);
        }
```

```
    }, 1000 );
}
</script>
</head>

<body class="page-header-fixed compact-menu">
    <div id="ajax_indicator" style="display:none;">
        <p style="position: absolute; top: 50%; left: 50%; margin: -110px 0 0
-110px;">
            <img src="${url}/include/images/common/loadingbar.gif" />
        </p>
    </div>

    <!-- Page Content -->
    <main class="page-content content-wrap">

        <%@ include file="/WEB-INF/views/include/admin/header.jsp" %>
        <%@ include file="/WEB-INF/views/include/admin/left.jsp" %>

        <%-- 임의 form --%>
        <form name="iForm" method="post"></form>

        <form name="frm" method="post">

        <!-- Page Inner -->
        <div class="page-inner">
            <div class="page-title">
                <h3>
                    주문메일알람 관리
                    <div class="page-right">
                        <button type="button" class="btn btn-line f-black"
title="검색" onclick="dataSearch();"><i class="fa fa-search"></i><em>검색</em>
</button>
                        <button type="button" class="btn btn-line f-black"
title="새로고침" onclick="window.location.reload();"><i class="fa fa-refresh"></i>
<em>새로고침</em></button>
                        <button type="button" class="btn btn-line f-black"
title="엑셀다운로드" onclick="excelDown(this);"><i class="fa fa-file-excel-o"></i>
<em>엑셀다운로드</em></button>
                    </div>
                </h3>
            </div>

            <!-- Main Wrapper -->
            <div id="main-wrapper">
                <!-- Row -->
                <div class="row">
                    <div class="col-md-12">
                        <div class="panel panel-white">
                            <div class="panel-body no-p">
                                <div class="tableSearch">
                                    <div class="topSearch">
                                        <ul>
                                            <li>
```

```
                                                <label class="search-h">거래처코드
</label>
                                                <div class="search-c">
                                                    <input type="text"
class="search-input" name="searchCustCd" value="${param.searchCustCd}"
onkeypress="if(event.keyCode == 13){dataSearch();}" />
                                                </div>
                                            </li>
                                            <li>
                                                <label class="search-h">거래처명
</label>
                                                <div class="search-c">
                                                    <input type="text"
class="search-input" name="searchCustNm" value="${param.searchCustNm}"
onkeypress="if(event.keyCode == 13){dataSearch();}" />
                                                </div>
                                            </li>
                                            <li>
                                                <label class="search-h">영업담당
</label>
                                                <div class="search-c">
                                                    <input type="text"
class="search-input" name="searchSalesrepNm" value="${param.searchSalesrepNm}"
onkeypress="if(event.keyCode == 13){dataSearch();}" />
                                                </div>
                                            </li>
                                        </ul>
                                    </div>
                                </div>
                            </div>

                            <div class="panel-body">
                                <h5 class="table-title listT">TOTAL <span
id="listTotalCountSpanId">0</span>EA</h5>
                                <div class="btnList writeObjectClass">
                                    <button type="button" class="btn btn-info"
onclick="dataSave(this);">저장</button>
                                </div>
                                <div class="table-responsive in">
                                    <table id="gridList" class="display table
tableList nowrap" width="100%" border="0" cellpadding="0" cellspacing="0"></table>
                                    <div id="pager"></div>
                                </div>
                            </div>

                        </div>
                    </div>
                </div>
                <!-- //Row -->
            </div>
            <!-- //Main Wrapper -->

            <%@ include file="/WEB-INF/views/include/admin/footer.jsp" %>
```

```
        </div>

        </form>
        <!-- //Page Inner -->
    </main>
    <!-- //Page Content -->

</body>

</html>
```

JQGrid에서 변경 감지, 저장, 배경색 표시의 오작동 문제를 다음과 같이 완전한 코드 개선 방안으로 해결할 수 있습니다.

변경 감지 및 배경색 표시 개선 원칙

1. **변경 감지는 `originalRowData`와 현재 그리드 데이터(`getRowData`)를 철저히 비교하여 boolean 판단.**
2. **체크박스 등 모든 editable 컬럼 변경 시 즉시 배경색 변경 함수 호출로 상태 반영.**
3. **저장 요청 시 변경된 행의 데이터만 서버 전송, 저장 성공 후에만 `originalRowData` 갱신 및 배경색 초기화.**
4. **수정 없을 경우 '수정된 내용이 없습니다' 알림만 띄움.**
5. **이메일 유효성 및 발송 여부에 따른 유효성 검사 강화.**

프론트엔드 (JavaScript / JQGrid) 수정 완전 코드

```
// 이메일 형식 유효성 검사 함수
function validateEmail(email) {
    const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
    return emailRegex.test(email);
}

// 행 배경색 변경 함수
function changeRowBackground(rowId, isChanged) {
    if (isChanged) {
        $('#gridList #' + rowId).css('background-color', '#ffebcd'); // 연한 주황색
    } else {
        $('#gridList #' + rowId).css('background-color', ''); // 원래 색상
    }
}

// 원본 데이터 저장용 전역 변수
var originalRowData = {};

// 체크박스 포맷터
function checkboxFormatter(cellVal, options, rowObj) {
    var checked = (cellVal === 'Y') ? 'checked' : '';
```

```
    return '<input type="checkbox" class="mail-checkbox" ' + checked + '
onchange="handleCheckboxChange(this, \'' + options.rowId + '\', \'' +
options.colModel.name + '\')" />';
}

// 체크박스 변경 처리 및 배경색 즉시 변경
function handleCheckboxChange(checkbox, rowId, fieldName) {
    var newValue = checkbox.checked ? 'Y' : 'N';

    $('#gridList').jqGrid('setCell', rowId, fieldName, newValue);

    checkRowChanges(rowId);
}

// 입력 필드 변경 추적 (email blur 등)
function validateEmailField(e) {
    var email = $(e.target).val().trim();
    var rowId = $(e.target).closest('tr').attr('id');

    if (email && !validateEmail(email)) {
        alert('올바른 이메일 형식을 입력해주세요.');
        $(e.target).focus();
        return false;
    }

    trackChanges(rowId);
}

// 변경 추적 호출 함수
function trackChanges(rowId) {
    checkRowChanges(rowId);
}

// 행 변경 사항 확인 및 배경색 반영
function checkRowChanges(rowId) {
    if (!originalRowData[rowId]) return;

    var currentRowData = $('#gridList').jqGrid('getRowData', rowId);
    var hasChanges = false;

    for (var field in originalRowData[rowId]) {
        var originalVal = (originalRowData[rowId][field] || '').toString().trim();
        var currentVal = (currentRowData[field] || '').toString().trim();

        if (originalVal !== currentVal) {
            hasChanges = true;
            break;
        }
    }

    changeRowBackground(rowId, hasChanges);
}

// 편집행 호출 시
```

```
var lastSelection;
function editRow(id) {
    if (id && id !== lastSelection) {
        if (lastSelection) {
            $('#gridList').jqGrid('saveRow', lastSelection, {aftersavefunc:
function() {
                checkRowChanges(lastSelection);
            }});
        }
        $('#gridList').jqGrid('editRow', id, { keys: false, focusField: true });
        lastSelection = id;
    }
}

// 저장 처리 함수
function dataSave(obj) {
    $(obj).prop('disabled', true);

    var grid = $('#gridList');
    var ids = grid.getDataIDs();
    var modifiedRows = [];

    $.each(ids, function(i, rowId) {
        var rowData = grid.getRowData(rowId);
        var original = originalRowData[rowId];
        if (!original) return true; // continue

        var hasChanges = false;

        for (var field in original) {
            var originalVal = (original[field] || '').toString().trim();
            var currentVal = (rowData[field] || '').toString().trim();

            if (originalVal !== currentVal) {
                hasChanges = true;
                break;
            }
        }

        if (hasChanges)
            modifiedRows.push(rowId);
    });

    if (modifiedRows.length === 0) {
        alert('수정된 내용이 없습니다.');
        $(obj).prop('disabled', false);
        return false;
    }

    // 유효성 검사
    for (var i=0; i<modifiedRows.length; i++) {
        var rowId = modifiedRows[i];
        var rowData = grid.getRowData(rowId);
```

```
        if (rowData.CUST_MAIN_EMAIL && !validateEmail(rowData.CUST_MAIN_EMAIL)) {
            alert('거래처 담당자 이메일 형식이 올바르지 않습니다. (행: ' + rowId +
')');
            $(obj).prop('disabled', false);
            return false;
        }
        if (rowData.SALESREP_EMAIL && !validateEmail(rowData.SALESREP_EMAIL)) {
            alert('영업 담당 이메일 형식이 올바르지 않습니다. (행: ' + rowId + ')');
            $(obj).prop('disabled', false);
            return false;
        }
        if (rowData.CUST_SENDMAIL_YN === 'Y' && !rowData.CUST_MAIN_EMAIL) {
            alert('담당자 이메일 발송이 체크되어 있지만 담당자 이메일이 비어있습니다.
(행: ' + rowId + ')');
            $(obj).prop('disabled', false);
            return false;
        }
        if (rowData.SALESREP_SENDMAIL_YN === 'Y' && !rowData.SALESREP_EMAIL) {
            alert('영업 담당 이메일 발송이 체크되어 있지만 영업 담당 이메일이 비어있습
니다. (행: ' + rowId + ')');
            $(obj).prop('disabled', false);
            return false;
        }
    }

    // 변경 데이터 폼에 셋팅
    var iFormObj = $('form[name="iForm"]');
    iFormObj.empty();

    $.each(modifiedRows, function(i, rowId) {
        var rowData = grid.getRowData(rowId);
        iFormObj.append('<input type="hidden" name="custCd" value="' +
rowData.CUST_CD + '" />');
        iFormObj.append('<input type="hidden" name="custMainEmail" value="' +
(rowData.CUST_MAIN_EMAIL || '') + '" />');
        iFormObj.append('<input type="hidden" name="custSendmailYn" value="' +
(rowData.CUST_SENDMAIL_YN || 'N') + '" />');
        iFormObj.append('<input type="hidden" name="salesrepEmail" value="' +
(rowData.SALESREP_EMAIL || '') + '" />');
        iFormObj.append('<input type="hidden" name="salesrepSendmailYn" value="' +
(rowData.SALESREP_SENDMAIL_YN || 'N') + '" />');
        iFormObj.append('<input type="hidden" name="comments" value="' +
(rowData.COMMENTS || '') + '" />');
    });

    if (!confirm('저장 하시겠습니까?')) {
        $(obj).prop('disabled', false);
        return false;
    }

    $.ajax({
        async: false,
        data: iFormObj.serialize(),
        type: 'POST',
```

```
        url: '${url}/admin/system/updateOrderMailAlarmAjax.lime',
        success: function(data) {
            if (data.RES_CODE === '0000') {
                alert(data.RES_MSG);
                // 저장 성공 시 원본 데이터 갱신 및 배경색 해제
                $.each(modifiedRows, function(i, rowId) {
                    originalRowData[rowId] = $.extend(true, {},
grid.getRowData(rowId));
                    changeRowBackground(rowId, false);
                });
                dataSearch();
            } else {
                alert(data.RES_MSG);
            }
            $(obj).prop('disabled', false);
        },
        error: function() {
            alert('Error');
            $(obj).prop('disabled', false);
        }
    });
}

// 초기 데이터 로드 완료 시 원본 데이터 저장
$('#gridList').jqGrid({
    // ... 기존 옵션 생략 ...
    loadComplete: function(data) {
        $('#listTotalCountSpanId').html(addComma(data.listTotalCount));

        originalRowData = {};
        if (data.list) {
            $.each(data.list, function(i, row) {
                // 행 ID는 jqGrid 기본 아이디가 1-base 인 경우
                var id = i + 1;
                originalRowData[id] = {
                    CUST_MAIN_EMAIL: row.CUST_MAIN_EMAIL || '',
                    CUST_SENDMAIL_YN: row.CUST_SENDMAIL_YN || 'N',
                    SALESREP_EMAIL: row.SALESREP_EMAIL || '',
                    SALESREP_SENDMAIL_YN: row.SALESREP_SENDMAIL_YN || 'N',
                    COMMENTS: row.COMMENTS || ''
                };
                changeRowBackground(id, false); // 초기엔 배경색 없음
            });
        }
    }
});

// onSelectRow 이벤트 등에서 editRow 호출 유지

// 이메일 필드 blur event 등에서 validateEmailField 호출 유지
```

백엔드 (Spring Controller + MyBatis) 예시

```
// Controller 측 저장 메서드 예시 (요청 수신 및 MyBatis 호출)
@PostMapping("/admin/system/updateOrderMailAlarmAjax.lime")
@ResponseBody
public Map<String, Object> updateOrderMailAlarm(@ModelAttribute OrderMailAlarmDTO
dto) {
    Map<String, Object> result = new HashMap<>();

    try {
        // 여러 행 데이터가 넘어온 경우 List 처리. (Map(List<Object>)도 가능)
        int updatedCount = service.updateOrderMailAlarms(dto);

        if (updatedCount > 0) {
            result.put("RES_CODE", "0000");
            result.put("RES_MSG", "저장되었습니다.");
        } else {
            result.put("RES_CODE", "9999");
            result.put("RES_MSG", "저장할 데이터가 없습니다.");
        }
    } catch (Exception e) {
        result.put("RES_CODE", "9999");
        result.put("RES_MSG", "에러가 발생하였습니다.");
    }
    return result;
}
```

```
<!-- Mapper SQL 예시 -->
<update id="updateOrderMailAlarms" parameterType="map">
    <foreach collection="list" item="item" index="index" separator=";">
        UPDATE CUSTOMER_ORDER_MAIL_ALARM
        SET
            CUST_MAIN_EMAIL = #{item.custMainEmail},
            CUST_SENDMAIL_YN = #{item.custSendmailYn},
            SALESREP_EMAIL = #{item.salesrepEmail},
            SALESREP_SENDMAIL_YN = #{item.salesrepSendmailYn},
            COMMENTS = #{item.comments}
        WHERE CUST_CD = #{item.custCd}
    </foreach>
</update>
```

테스트 시나리오

1. 그리드 데이터 로드 후, 이메일/체크박스 값 일부 수정 → 행 배경색 즉시 변경 확인
2. 변경된 값으로 저장 버튼 클릭 → 저장 성공 메시지 및 서버 DB 반영 확인 (네트워크 탭, DB 직접 조회)
3. 저장 완료 후 해당 행의 배경색 해제 및 `originalRowData` 갱신 확인
4. 저장 시 변경된 데이터 없으면 '수정된 내용이 없습니다.' 알림 발생 확인

5. 이메일 형식 오류 또는 발송 여부 조건 불일치 시 저장 차단 및 사용자 알림 동작 확인

위 방식으로 하면 변경 감지 로직이 정확해지고, 체크박스 포함 모든 변경 사항에 대해 즉각 배경색 반영 및 저장 흐름도 올바르게 동작하여 요구 조건(AC)를 충족합니다. 궁금한 점 있으면 알려주시길 바랍니다.