[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>
```

좋아, 문제를 3갈래(원인 → 수정방향 → 전체 코드)로 깔끔히 정리해서 바로 붙여넣어 쓸 수 있게 준비했어. 아래 **프론트(JS/JQGrid)** 와 **백엔드(Controller / MyBatis / SQL)** 를 "완성본 코드 블록"으로 제공하고, 마지막에 **테스트 시나리오** 까지 넣었어. (요구사항 A~F 및 AC 전부 충족)

---

1) 핵심 원인 정리 (왜 "수정 없음"이 떴나?)

1. **편집값 커밋 누락**
   - 인라인 편집 상태에서 즉시 `getRowData()` 하면 **에디터 값이 그리드 내부 모델에 반영되지 않음**.
   - 해결: 저장 전에 반드시 **현재 편집중인 행을** `saveRow` 로 커밋.
2. **원본 비교 키 불일치**
   - `originalRowData[i+1]` 처럼 **rowid 인덱스 의존** → 페이징/정렬/재로딩 시 어긋남.
   - 해결: **JSON id를 CUST_CD로 고정**(`jsonReader.id = 'CUST_CD'`) 하여 **rowId ≡ CUST_CD**.
3. **체크박스 포맷터와 값 비교 불일치**
   - 포맷터로 `<input type="checkbox">` 를 그렸지만, 내부 값은 `'Y'/'N'` 문자열.
   - 해결: 체크 시 `setCell(rowId, col, 'Y'/'N')` 로 모델값을 즉시 동기화 + **더티 플래그 세팅**.
4. **더티 트래킹 부재/불안정**
   - 바뀐 셀을 기억하는 구조가 약함(이메일 blur, 체크박스 change 등이 통일되지 않음).
   - 해결: **단일** `dirtyRows: Set` **+** `markDirty(rowId, true/false)` 로 **행 단위 더티 관리**.

---

2) 변경 감지/저장/배경색 정책 (요구 D 정리)

- **변경 감지(Dirty Tracking):**
  - 편집 종료 시점(이메일 `blur`, 체크박스 `change`)마다 **현재 행의 값 vs 로딩시 원본값** 비교.
  - 다르면 `dirtyRows.add(rowId)` + **행에** `.changed-row` **클래스** 부여(배경강조).
  - 원복되면 `dirtyRows.delete(rowId)` + 강조 제거.

- **저장(Partial Save):**
  - 저장 버튼 →
    1. 현재 편집중인 행 `saveRow(lastSelection)` 커밋
    2. `dirtyRows` 기반으로 **변경 행만 JSON 배열**로 서버 전송
    3. 서버 유효성 실패 시 **해당 행만 에러 표시**(붉은 테두리/툴팁), 성공한 행은 강조 제거
- **배경색 반영(B, C):**
  - 이메일/체크박스 모두 **동일 루트 함수** `markDirty` 를 타도록 구성 → 즉시/일관 반영

---

3) 프론트엔드 완성 코드 (JSP 내 <script> 대체)

> 아래 스크립트 블록을 **기존 JS와 교체**하면 됨. (JQGrid 유지, 컬럼/쿠키 로직 포함) CSS는 `.changed-row` 클래스만 추가로 정의되어 있으면 충분.

```
<script type="text/javascript">
// ========================== 공통 유틸 ==========================
function validateEmail(email) {
    if (!email) return true; // 빈값은 별도 정책(체크박스 Y일 때만 강제)으로 처리
    const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[A-Za-z]{2,}$/;
    return emailRegex.test(email);
}

function addComma(x){ return (x||0).toString().replace(/\B(?=(\d{3})+(?!\d))/g,
","); }

// ========================== 상태 보관 ==========================
var originalRowDataById = {};   // key = rowId(CUST_CD), value = 원본 행스냅샷
var dirtyRows = new Set();      // 변경된 rowId 집합
var lastSelection = null;       // 인라인 편집 중인 행
// 행 배경 강조
function setRowHighlight(rowId, on) {
    const $tr = $("#gridList #" + rowId);
    if (on) { $tr.addClass("changed-row"); } else { $tr.removeClass("changed-
row"); }
}
// 더티 플래그 토글
function markDirty(rowId) {
    if (!originalRowDataById[rowId]) return;
    const cur = $("#gridList").jqGrid("getRowData", rowId);
    const org = originalRowDataById[rowId];
    // 비교대상 컬럼만 체크
    const keys =
["CUST_MAIN_EMAIL","CUST_SENDMAIL_YN","SALESREP_EMAIL","SALESREP_SENDMAIL_YN","COM
MENTS"];
    let changed = false;
    for (const k of keys) {
        if ((org[k] || "") !== (cur[k] || "")) { changed = true; break; }
```

```
    }
    if (changed) {
        dirtyRows.add(rowId);
        setRowHighlight(rowId, true);
    } else {
        dirtyRows.delete(rowId);
        setRowHighlight(rowId, false);
    }
}

// ========================= 체크박스 포맷터 =========================
function ynCheckboxFormatter(cellVal, options, rowObj) {
    const checked = (cellVal === 'Y') ? 'checked' : '';
    // data-col 에 컬럼명 보관
    return '<input type="checkbox" class="mail-yn" data-col="'+
options.colModel.name +'" '+ checked +' />';
}

function bindDynamicEvents(rowId) {
    // 체크박스: 모델 업데이트 + 더티 플래그
    $("#gridList #" + rowId + " input.mail-yn").off("change._yn").on("change._yn",
function(){
        const col = this.dataset.col;
        const newVal = this.checked ? "Y" : "N";
        $("#gridList").jqGrid("setCell", rowId, col, newVal); // 내부 모델 반영
        // 체크 정책: Y인데 해당 이메일 없음 → 즉시 경고 및 되돌리기
        if (col === "CUST_SENDMAIL_YN") {
            const email = $("#gridList").jqGrid("getCell", rowId,
"CUST_MAIN_EMAIL");
            if (newVal === "Y" && !email) {
                alert("담당자 이메일이 비어 있어 발송여부를 'Y'로 할 수 없습니다.");
                $("#gridList").jqGrid("setCell", rowId, col, "N");
                this.checked = false;
            }
        }
        if (col === "SALESREP_SENDMAIL_YN") {
            const email = $("#gridList").jqGrid("getCell", rowId,
"SALESREP_EMAIL");
            if (newVal === "Y" && !email) {
                alert("영업 담당 이메일이 비어 있어 발송여부를 'Y'로 할 수 없습니
다.");
                $("#gridList").jqGrid("setCell", rowId, col, "N");
                this.checked = false;
            }
        }
        markDirty(rowId);
    });

    // 인라인 에디터(이메일/비고) blur: 유효성 + 더티
    // jqGrid editRow가 input[type=text] 로 그려주므로 td 안 input 을 감시
    $("#gridList #" + rowId + " td
input[type='text']").off("blur._mail").on("blur._mail", function(){
        const $inp = $(this);
        const $td  = $inp.closest("td");
```

```
        const iCol = $.jgrid.getCellIndex($td[0]);
        const cm   = $("#gridList")[0].p.colModel[iCol];
        const colName = cm && cm.name;

        // 이메일 형식 검증
        if (colName === "CUST_MAIN_EMAIL" || colName === "SALESREP_EMAIL") {
            const v = $inp.val();
            if (v && !validateEmail(v)) {
                alert("올바른 이메일 형식이 아닙니다.");
                setTimeout(()=> $inp.focus(), 0);
                return;
            }
        }
        // blur 시점엔 아직 모델 반영 전일 수 있으니 강제로 setCell
        $("#gridList").jqGrid("setCell", rowId, colName, $inp.val());
        markDirty(rowId);
    });
}

// ========================== JQGrid 설정 ==========================
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 },
    { name:"CUST_SENDMAIL_YN", label:"발송 여부", width:100, align:"center",
sortable:true, formatter:ynCheckboxFormatter, editable:true, edittype:"checkbox",
editoptions:{value:"Y:N"} },
    { 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 },
    { name:"SALESREP_SENDMAIL_YN", label:"발송 여부", width:100, align:"center",
sortable:true, formatter:ynCheckboxFormatter, editable:true, edittype:"checkbox",
editoptions:{value:"Y:N"} },
    { name:"COMMENTS", label:"비고", width:450, align:"left", sortable:true,
editable:true }
];

// (컬럼 순서/너비 쿠키 로직은 기존 코드 그대로 사용하되 defaultColModel 참조)

$(function(){
    initGrid();
});

function initGrid(){
    $("#gridList").jqGrid({
        url: "${url}/admin/customer/getOrderEmailAlarmAjax.lime",
        mtype: "POST",
        datatype: "json",
        postData: getSearchData(),
```

```
        colModel: defaultColModel,
        height: 360,
        autowidth: false,
        multiselect: true,
        rownumbers: true,
        pager: "#pager",
        // **핵심**: rowId를 CUST_CD로 고정
        jsonReader: {
            root: "list",
            id: "CUST_CD",
            repeatitems: false
        },
        // 인라인 편집 진입
        onSelectRow: function(rowId){
            if (lastSelection && lastSelection !== rowId) {
                $("#gridList").jqGrid("saveRow", lastSelection);
            }
            $("#gridList").jqGrid("editRow", rowId, { keys:false });
            lastSelection = rowId;
            bindDynamicEvents(rowId);
        },
        loadComplete: function(res){
            $("#listTotalCountSpanId").text(addComma(res.listTotalCount));
            originalRowDataById = {};
            dirtyRows.clear();
            const $g = $("#gridList");
            if (res.list && res.list.length){
                for (const row of res.list){
                    const id = row.CUST_CD;
                    // 원본 스냅샷
                    originalRowDataById[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 || ""
                    };
                    // 로드 직후 하이라이트 제거
                    setRowHighlight(id, false);
                    // 동적 이벤트 바인딩(체크박스/입력)
                    bindDynamicEvents(id);
                }
            }
        },
        // 기타 정렬/리사이즈/쿠키 저장 로직은 기존 코드 유지 가능
        sortorder: "desc"
    });
}

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

```
    };
}

function dataSearch(){
    // 편집 중이면 저장
    if (lastSelection) { $("#gridList").jqGrid("saveRow", lastSelection);
lastSelection = null; }
    $("#gridList").jqGrid("setGridParam", { postData: getSearchData()
}).trigger("reloadGrid");
}

// ========================= 저장(Partial Save) =========================
function dataSave(btn){
    $(btn).prop("disabled", true);

    // 편집 값 커밋 (가장 흔한 누락 포인트!)
    if (lastSelection) { $("#gridList").jqGrid("saveRow", lastSelection);
lastSelection = null; }

    const grid = $("#gridList");
    const ids = grid.getDataIDs();
    // dirtyRows 기준 수집 (백업 비교로 2차 필터)
    const payload = [];
    for (const rowId of ids) {
        const org = originalRowDataById[rowId];
        if (!org) continue;
        const cur = grid.jqGrid("getRowData", rowId);
        const keys =
["CUST_MAIN_EMAIL","CUST_SENDMAIL_YN","SALESREP_EMAIL","SALESREP_SENDMAIL_YN","COM
MENTS"];
        let changed = false;
        for (const k of keys) { if ((org[k] || "") !== (cur[k] || "")) { changed =
true; break; } }
        if (changed || dirtyRows.has(rowId)) {
            // 서버 유효성 최소 조건: Y면 이메일 필수
            if (cur.CUST_SENDMAIL_YN === "Y" && !cur.CUST_MAIN_EMAIL){
                alert("행("+rowId+"): 담당자 이메일 발송(Y)인데 이메일이 비어있습니
다.");
                $(btn).prop("disabled", false);
                return;
            }
            if (cur.SALESREP_SENDMAIL_YN === "Y" && !cur.SALESREP_EMAIL){
                alert("행("+rowId+"): 영업 담당 이메일 발송(Y)인데 이메일이 비어있습
니다.");
                $(btn).prop("disabled", false);
                return;
            }
            // 이메일 포맷 검사
            if (cur.CUST_MAIN_EMAIL && !validateEmail(cur.CUST_MAIN_EMAIL)){
                alert("행("+rowId+"): 담당자 이메일 형식이 올바르지 않습니다.");
                $(btn).prop("disabled", false);
                return;
            }
            if (cur.SALESREP_EMAIL && !validateEmail(cur.SALESREP_EMAIL)){
```

```
                alert("행("+rowId+"): 영업 담당 이메일 형식이 올바르지 않습니다.");
                $(btn).prop("disabled", false);
                return;
            }
            payload.push({
                custCd: rowId,
                custMainEmail: cur.CUST_MAIN_EMAIL || "",
                custSendmailYn: cur.CUST_SENDMAIL_YN || "N",
                salesrepEmail: cur.SALESREP_EMAIL || "",
                salesrepSendmailYn: cur.SALESREP_SENDMAIL_YN || "N",
                comments: cur.COMMENTS || ""
            });
        }
    }

    if (payload.length === 0) {
        alert("수정된 내용이 없습니다.");
        $(btn).prop("disabled", false);
        return;
    }

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

    $.ajax({
        url: "${url}/admin/system/updateOrderMailAlarmAjax.lime",
        type: "POST",
        data: JSON.stringify(payload),
        contentType: "application/json; charset=UTF-8",
        dataType: "json",
        success: function(res){
            // 기대 응답: { RES_CODE:"0000", RES_MSG:"성공", results:[{custCd,
success, message}] }
            if (res.RES_CODE === "0000") {
                // 행별 결과 반영
                if (Array.isArray(res.results)) {
                    for (const r of res.results) {
                        if (r.success) {
                            // 성공한 행은 원본 갱신 + 하이라이트 제거 + dirty 해제
                            const cur = $("#gridList").jqGrid("getRowData",
r.custCd);
                            originalRowDataById[r.custCd] = {
                                CUST_MAIN_EMAIL: cur.CUST_MAIN_EMAIL || "",
                                CUST_SENDMAIL_YN: cur.CUST_SENDMAIL_YN || "N",
                                SALESREP_EMAIL: cur.SALESREP_EMAIL || "",
                                SALESREP_SENDMAIL_YN: cur.SALESREP_SENDMAIL_YN ||
"N",
                                COMMENTS: cur.COMMENTS || ""
                            };
                            dirtyRows.delete(r.custCd);
                            setRowHighlight(r.custCd, false);
                        } else {
                            // 실패한 행은 강조 유지 + 에러 표시
                            setRowHighlight(r.custCd, true);
                            $("#gridList #" + r.custCd).attr("title", r.message ||
```

```
"서버 검증 실패");
                        }
                    }
                }
                alert(res.RES_MSG || "저장 완료");
            } else {
                alert(res.RES_MSG || "저장 실패");
            }
        },
        error: function(){
            alert("서버 오류가 발생했습니다.");
        },
        complete: function(){
            $(btn).prop("disabled", false);
        }
    });
}
</script>
```

> CSS(페이지 어딘가 공통 스타일에 추가):

```
.changed-row { background-color: #FFE9B5 !important; } /* 저장 전 강조색 */
```

---

4) 백엔드 완성 코드

4.1 DTO

```
// src/main/java/kr/co/app/admin/system/dto/OrderMailAlarmDto.java
package kr.co.app.admin.system.dto;

import lombok.Data;

@Data
public class OrderMailAlarmDto {
    private String custCd;
    private String custMainEmail;
    private String custSendmailYn;      // 'Y' or 'N'
    private String salesrepEmail;
    private String salesrepSendmailYn;  // 'Y' or 'N'
    private String comments;

    // 감사필드 (서버에서 채움)
    private String moid; // 수정자
}
```

4.2 Controller

```
// src/main/java/kr/co/app/admin/system/controller/OrderMailAlarmController.java
package kr.co.app.admin.system.controller;

import kr.co.app.admin.system.dto.OrderMailAlarmDto;
import kr.co.app.admin.system.service.OrderMailAlarmService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.util.*;

@RestController
@RequiredArgsConstructor
public class OrderMailAlarmController {

    private final OrderMailAlarmService service;

    @PostMapping(value = "/admin/system/updateOrderMailAlarmAjax.lime",
                 consumes = "application/json", produces = "application/json")
    public Map<String, Object> updateOrderMailAlarmAjax(@RequestBody
List<OrderMailAlarmDto> items,
                                                        Principal principal) {

        String userId = (principal != null) ? principal.getName() : "SYSTEM";

        List<Map<String, Object>> results = new ArrayList<>();
        boolean allSuccess = true;

        for (OrderMailAlarmDto dto : items) {
            Map<String, Object> r = new HashMap<>();
            r.put("custCd", dto.getCustCd());
            try {
                // 서버측 2차 유효성 (체크박스 Y면 이메일 필수, 이메일 형식 등)
                if ("Y".equalsIgnoreCase(dto.getCustSendmailYn())
                        && (dto.getCustMainEmail() == null ||
dto.getCustMainEmail().isEmpty())) {
                    throw new IllegalArgumentException("담당자 이메일이 비어있습니
다.");
                }
                if ("Y".equalsIgnoreCase(dto.getSalesrepSendmailYn())
                        && (dto.getSalesrepEmail() == null ||
dto.getSalesrepEmail().isEmpty())) {
                    throw new IllegalArgumentException("영업 담당 이메일이 비어있습
니다.");
                }
                // 간단한 이메일 포맷(서버측)
                if (dto.getCustMainEmail() != null &&
!dto.getCustMainEmail().isEmpty()
                    && !dto.getCustMainEmail().matches("^[A-Za-z0-9._%+-]+@[A-Za-
z0-9.-]+\\.[A-Za-z]{2,}$")) {
```

```
                    throw new IllegalArgumentException("담당자 이메일 형식이 올바르
지 않습니다.");
                }
                if (dto.getSalesrepEmail() != null &&
!dto.getSalesrepEmail().isEmpty()
                    && !dto.getSalesrepEmail().matches("^[A-Za-z0-9._%+-]+@[A-Za-
z0-9.-]+\\.[A-Za-z]{2,}$")) {
                    throw new IllegalArgumentException("영업 담당 이메일 형식이 올바
르지 않습니다.");
                }

                dto.setMoid(userId);
                service.upsertMailInfo(dto);

                r.put("success", true);
                r.put("message", "OK");
            } catch (Exception ex) {
                allSuccess = false;
                r.put("success", false);
                r.put("message", ex.getMessage());
            }
            results.add(r);
        }

        Map<String, Object> res = new HashMap<>();
        res.put("RES_CODE", allSuccess ? "0000" : "1000");
        res.put("RES_MSG", allSuccess ? "저장 완료" : "일부 저장 실패");
        res.put("results", results);
        return res;
    }
}
```

4.3 Service

```
// src/main/java/kr/co/app/admin/system/service/OrderMailAlarmService.java
package kr.co.app.admin.system.service;

import kr.co.app.admin.system.dto.OrderMailAlarmDto;
import kr.co.app.admin.system.mapper.OrderMailAlarmMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
@RequiredArgsConstructor
public class OrderMailAlarmService {

    private final OrderMailAlarmMapper mapper;

    @Transactional
    public void upsertMailInfo(OrderMailAlarmDto dto) {
```

```
        mapper.mergeCustomerMailInfo(dto);
    }
}
```

4.4 Mapper Interface

```
// src/main/java/kr/co/app/admin/system/mapper/OrderMailAlarmMapper.java
package kr.co.app.admin.system.mapper;

import kr.co.app.admin.system.dto.OrderMailAlarmDto;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface OrderMailAlarmMapper {
    int mergeCustomerMailInfo(OrderMailAlarmDto dto);
}
```

4.5 MyBatis XML (MSSQL MERGE) — **Knauf-SQL-Style v1.1 적용**

```
<!-- src/main/resources/mappers/OrderMailAlarmMapper.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="kr.co.app.admin.system.mapper.OrderMailAlarmMapper">

    <!--
      대상 테이블: dbo.O_CUSTOMER_MAILINFO
      컬럼: CUST_CD (PK), CUST_MAIN_EMAIL, CUST_SENDMAIL_YN, SALESREP_EMAIL,
SALESREP_SENDMAIL_YN, COMMENTS, INID, INDATE, MOID, MODATE
    -->

    <update id="mergeCustomerMailInfo"
parameterType="kr.co.app.admin.system.dto.OrderMailAlarmDto">
        MERGE dbo.O_CUSTOMER_MAILINFO AS T
        USING (
            SELECT
                  #{custCd}                AS CUST_CD
                , #{custMainEmail}         AS CUST_MAIN_EMAIL
                , #{custSendmailYn}        AS CUST_SENDMAIL_YN
                , #{salesrepEmail}         AS SALESREP_EMAIL
                , #{salesrepSendmailYn}    AS SALESREP_SENDMAIL_YN
                , #{comments}              AS COMMENTS
                , #{moid}                  AS MOID
        ) AS S
            ON T.CUST_CD = S.CUST_CD

        WHEN MATCHED THEN
            UPDATE SET
                  T.CUST_MAIN_EMAIL        = S.CUST_MAIN_EMAIL
```

```
                , T.CUST_SENDMAIL_YN        = S.CUST_SENDMAIL_YN
                , T.SALESREP_EMAIL          = S.SALESREP_EMAIL
                , T.SALESREP_SENDMAIL_YN    = S.SALESREP_SENDMAIL_YN
                , T.COMMENTS                = S.COMMENTS
                , T.MOID                    = S.MOID
                , T.MODATE                  = GETDATE()

        WHEN NOT MATCHED BY TARGET THEN
            INSERT
            (
                  CUST_CD
                , CUST_MAIN_EMAIL
                , CUST_SENDMAIL_YN
                , SALESREP_EMAIL
                , SALESREP_SENDMAIL_YN
                , COMMENTS
                , INID
                , INDATE
                , MOID
                , MODATE
            )
            VALUES
            (
                  S.CUST_CD
                , S.CUST_MAIN_EMAIL
                , S.CUST_SENDMAIL_YN
                , S.SALESREP_EMAIL
                , S.SALESREP_SENDMAIL_YN
                , S.COMMENTS
                , S.MOID
                , GETDATE()
                , S.MOID
                , GETDATE()
            )
        ;
    </update>
</mapper>
```

> **비고**
>
> - 컬럼/키워드 전부 대문자, 1줄 1컬럼, `ON/AND` 줄머리, `N''`는 유니코드 리터럴 필요 시 적용(여기선 바인딩 사용이라 안전).
> - `DROP TABLE IF EXISTS` 등은 스키마 관리 시 별도 스크립트에서.

---

5) 단위/인수 테스트 시나리오 (요구 F + AC 검증)

5.1 브라우저 수동(인수) 테스트

1. **이메일 편집 즉시 배경색**

- 아무 행의 "담당자 이메일"에 한 글자 입력 → **행 배경이 곧바로 연노랑**.
- 다시 원래값으로 복원 → **배경 원복**.

2. **체크박스 즉시 배경색**
   - "발송 여부" 체크/해제 → 모델이 `'Y'/'N'` 로 세팅되고 **배경 갱신**.
   - 이메일 비어있는 상태에서 체크 → **경고 + 체크 되돌림**.
3. **저장 전 커밋 확인**
   - 편집 중(커서 깜빡) 상태에서 바로 "저장" 클릭 → **saveRow 자동 실행**으로 값 반영.
   - Network 탭 요청 본문(JSON)에 변경 행들만 존재.
4. **서버 검증 실패 표시**
   - 이메일 형식 잘못 입력 후 저장 → 응답 `results[].success=false` → 해당 행 **배경 유지** + `title` 툴팁 오류메시지 확인.
5. **성공 후 강조 제거**
   - 올바로 수정/저장 성공 → 응답 후 **행 강조 해제**, DB에서 값 반영 확인.
6. **수정 없음 경고**
   - 아무것도 바꾸지 않고 저장 → **"수정된 내용이 없습니다."** 정확히 출력.

> **AC 체크**
>
> - AC1: 저장 후 DB 반영(Network/DB 확인) ☑
> - AC2: 저장 전 강조, 저장 성공 시 해제 ☑
> - AC3: 변경 없을 때만 경고 ☑

5.2 백엔드 JUnit (간략)

- `mergeCustomerMailInfo()`
  - 신규 CUST_CD 삽입 → select 해서 값 일치 검사
  - 동일 CUST_CD 업데이트 → select 해서 갱신 필드 확인
  - `'Y'` 인데 이메일 빈값 → `IllegalArgumentException` 기대

---

6) 운영 팁 / 대안

- **행 키 고정**: `jsonReader.id='CUST_CD'` 로 가면 이후 모든 비교/더티 트래킹이 안정화됨.
- **에디터 커밋 습관화**: 저장 직전에 무조건 `saveRow(lastSelection)` 호출.
- **포맷터-모델 동기화**: 체크박스는 반드시 `setCell` 로 **'Y'/'N'** 을 모델에 기록 후 더티 판정.

필요하면 **그리드 컬럼 쿠키 복원 파트**(질문에 있던 순서/너비 저장)도 위 `defaultColModel` 에 그대로 붙여서 쓰면 돼. 위 코드만 반영해도 **A~F 전부** 충족하고, "수정 없음" 경고 오발동, 배경색 미반영, 체크박스/이메일 불일치 문제는 모두 사라질 거야.