

프로젝트 소개



회사 내의 가맹점과 본사의 유통을 간편하고 쉽게 파악할 수 있는 프로그램

좋은 ERP를 쓰면 업무가 효율화되고, 더 빨리 성장할 수 있습니다.

우리 회사 업무에 맞게 사용 환경을 최적화 할 수 있습니다.

간편하기 때문입니다.

거래 내역을 잘 기록해야 합니다.

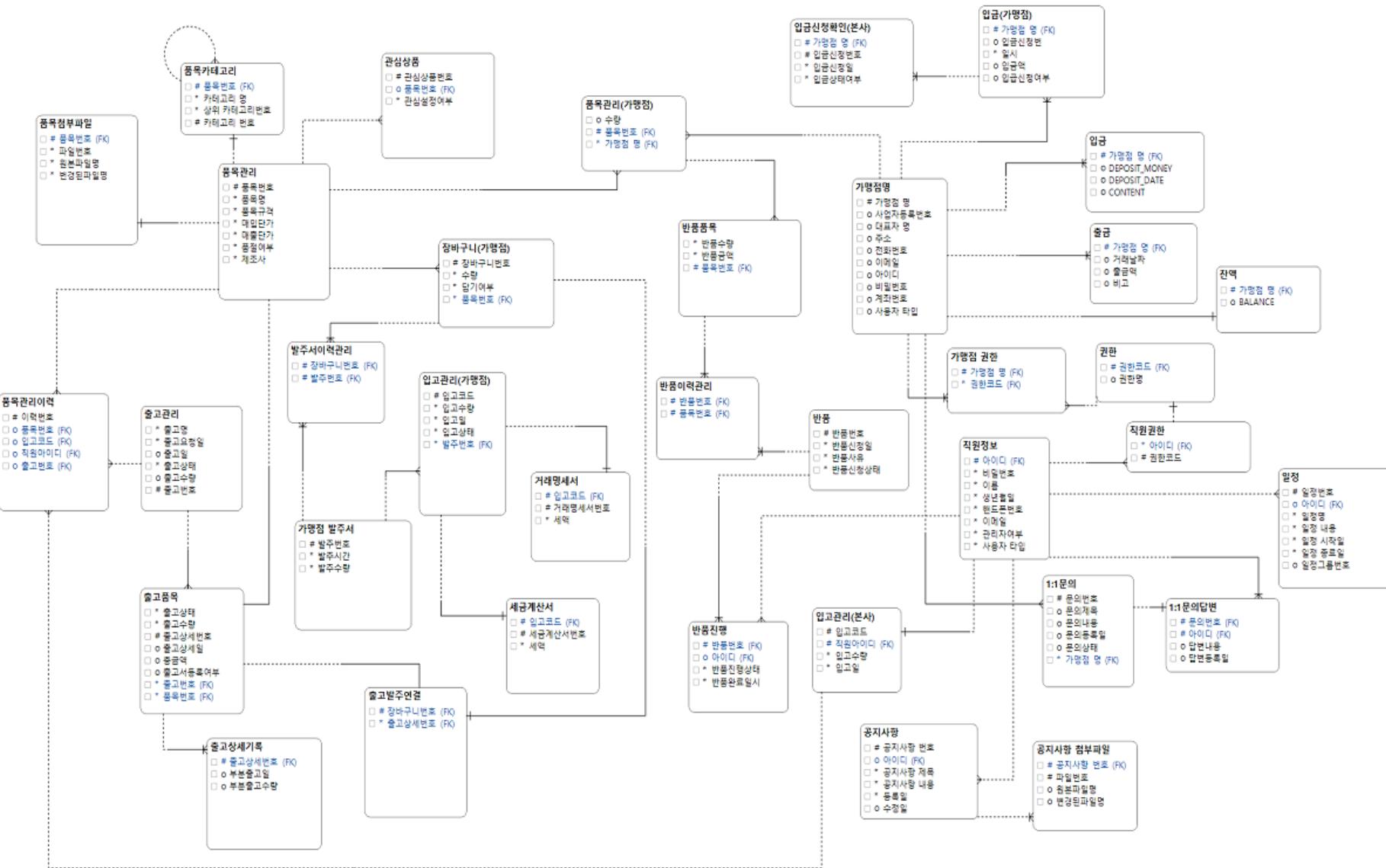
중복업무의 배제 및 실시간 정보처리 체계 구축!!

입력(기록)만 잘 하면 그 뒤는 치킨스톡이 모든 것을 처리해줍니다:)

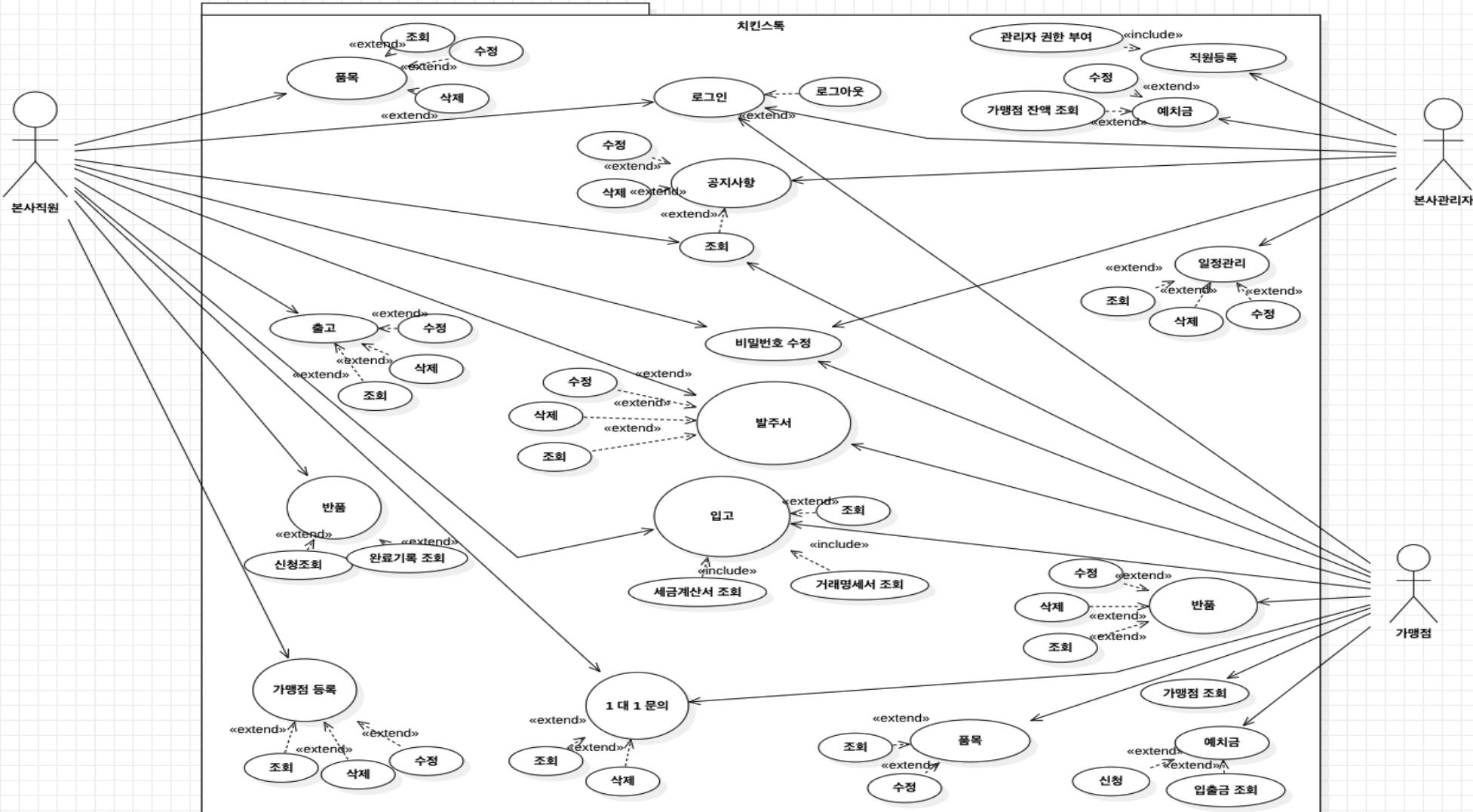
프로젝트 일정(WBS)

대분류	중분류	소 분류	작업	날짜
개발	업무분석	업무분석		
업무분석 및 오간장의	요구분석	요구사항정제		
업무분석 및 오간장의	요구확인	유스케이스디자인		
업무분석 및 오간장의	요구현안	프로토타이핑		
업무분석 및 오간장의	오간장의	전체업무흐름도		
업무분석 및 오간장의	오간장의	업무우록		
업무분석 및 오간장의	오간장의	업무업무우록		
기획 / 설계	D8설계	놀리모달문제		
기획 / 설계	D8설계	성간모달문제		
기획 / 설계	D8설계	물리모달문제		
기획 / 설계	D8설계	데이터정의서작성		
기획 / 설계	D8설계	D8구축		
개발	미별리싱	회면 구현		
개발	미별리싱	회면 검수		
개발	프로그램개발	기능 구현		
테스트	단위테스트	단위 테스트		
테스트	통합테스트	통합 테스트		
테스트	인수테스트	인수 테스트		
현장	신기술증강	신기술 정의		
현장	시연발표	시연 발표		
2022년 8월				
9월				
업무분석 2022년 8월 10일 ~ 2022년 8월 11일 개발 업무분석				
요구사항정제 2022년 8월 16일 ~ 2022년 8월 26일 업무분석 및 오간장의 요구분석				
유스케이스디자인 2022년 8월 17일 ~ 2022년 8월 21일 업무분석 및 오간장의 요구확인				
프로토타이핑 2022년 8월 19일 ~ 2022년 8월 23일 업무분석 및 오간장의 요구현안				
전체업무흐름도 2022년 8월 19일 ~ 2022년 8월 23일 업무분석 및 오간장의 오간장의				
업무우록 2022년 8월 19일 ~ 2022년 8월 23일 업무분석 및 오간장의 오간장의				
단위업무우록 2022년 8월 20일 ~ 2022년 8월 24일 업무분석 및 오간장의 오간장의				
논리모달분석 2022년 8월 22일 ~ 2022년 8월 24일 기획 / 설계 D8설계				
종간모달정 2022년 8월 22일 ~ 2022년 8월 26일 기획 / 설계 D8설계				
물리모달분석 2022년 8월 22일 ~ 2022년 8월 26일 기획 / 설계 D8설계				
데이터정의서작성 2022년 8월 22일 ~ 2022년 8월 27일 기획 / 설계 D8설계				
D8구축 2022년 8월 22일 ~ 2022년 8월 27일 기획 / 설계 D8설계				
화급 구현 2022년 9월 6일 ~ 2022년 9월 20일 개발 기획 / 설계				
회선 정수 2022년 9월 19일 ~ 2022년 9월 23일 개발 기획 / 설계				
기능 구현 2022년 9월 6일 ~ 2022년 9월 21일 개발 프로그램개발				
단계 테스트 2022년 9월 17일 ~ 2022년 9월 21일 테스트 단계테스트				
통합 테스트 2022년 9월 17일 ~ 2022년 9월 21일 테스트 통합테스트				
인수 테스트 2022년 9월 17일 ~ 2022년 9월 21일 테스트 인수테스트				
신기술증강 2022년 9월 19일 ~ 2022년 9월 21일 현장 신기술증강				
시연 발표 2022년 9월 22일 ~ 2022년 9월 23일				

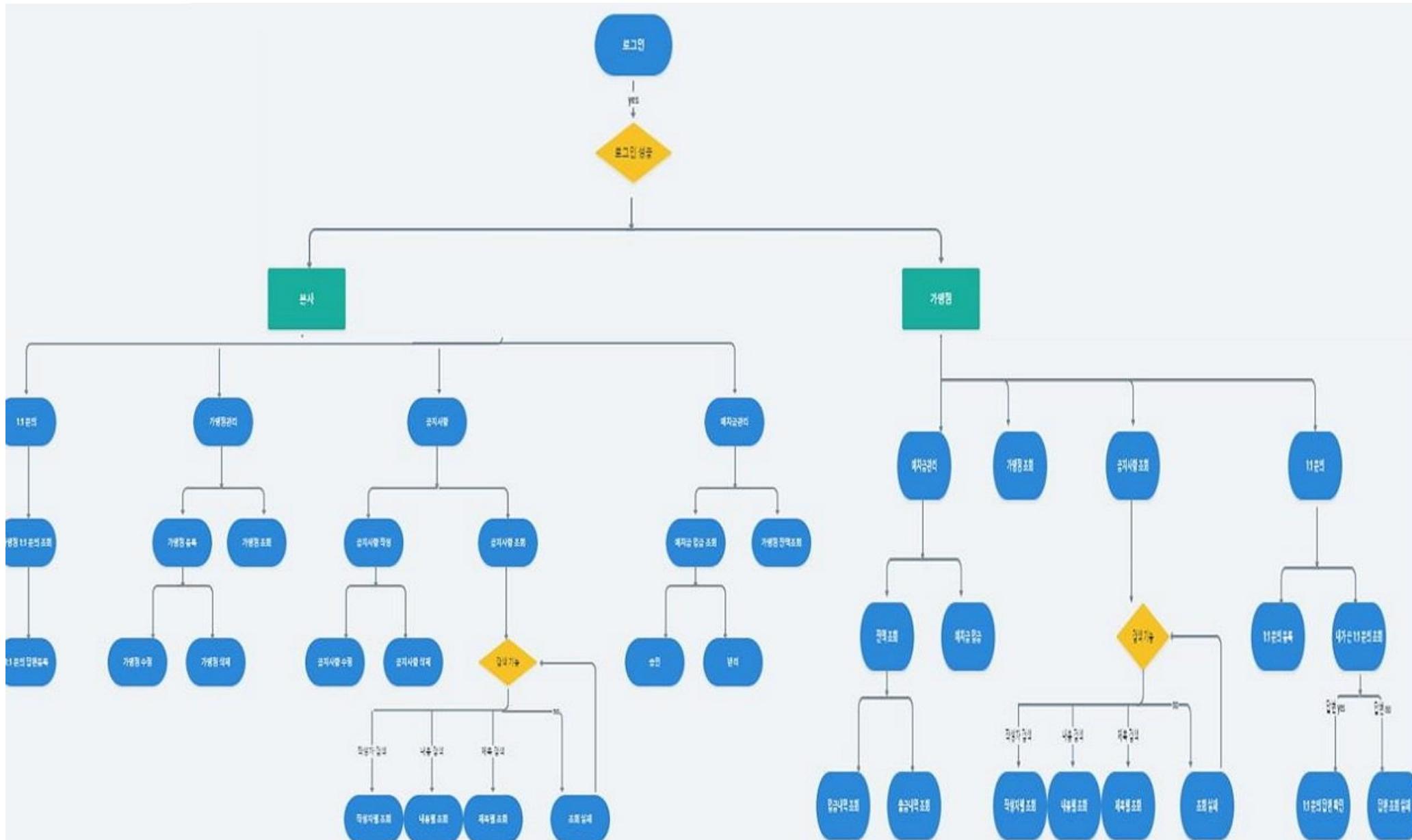
ERD (논리 데이터 베이스 모델)



유스케이스 모델



USER FLOW (담당 기능 : 예치금관리 / 1:1 문의 / 공지사항 / 가맹점관리)



공지사항 - 메인페이지 공지사항 리스트 (관리자 / 가맹점)

```
<script>
$.ajax({
  url : "noticeMain",
  type : "GET",
  contentType : 'application/json',
  success:function (data){
    let notice = JSON.parse(data);
    console.log(notice);

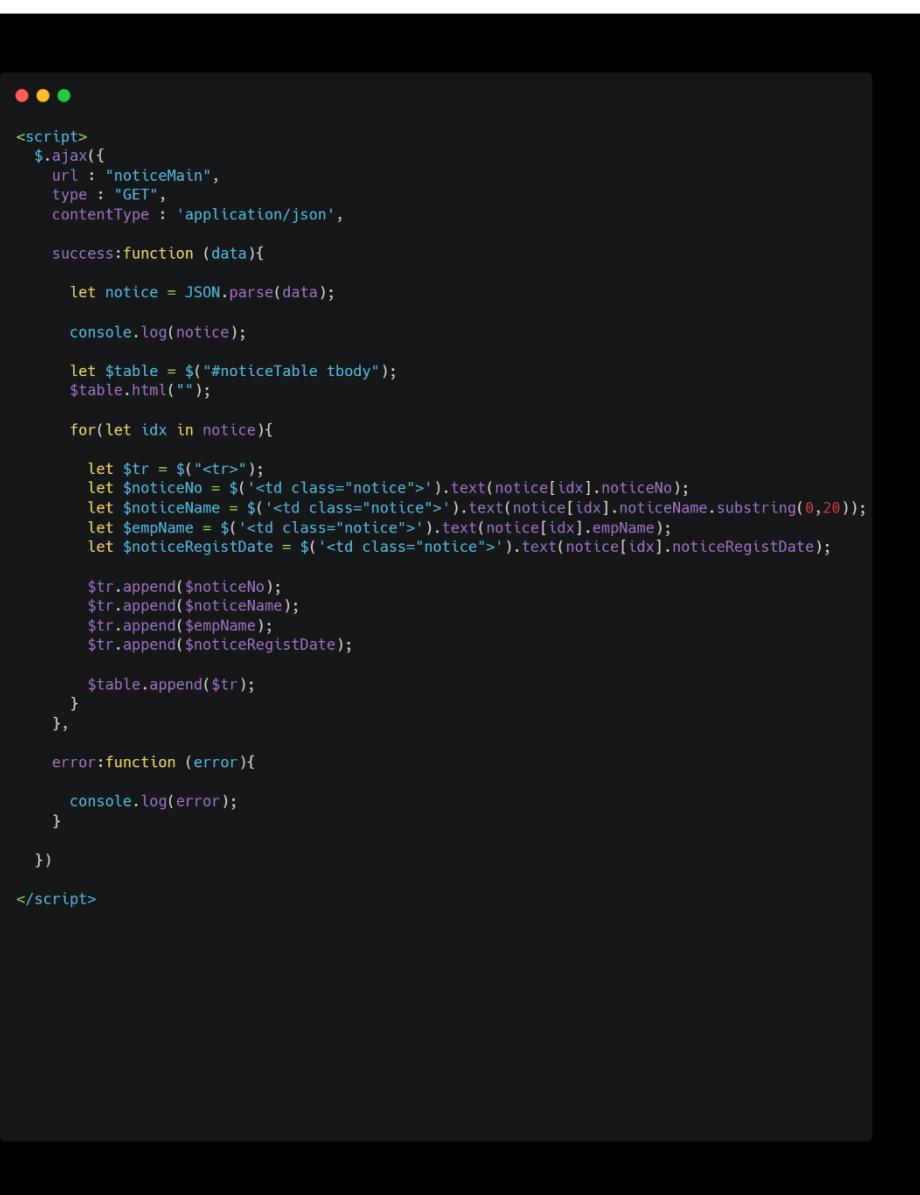
    let $table = $("#noticeTable tbody");
    $table.html("");

    for(let idx in notice){

      let $tr = $("<tr>");
      let $noticeNo = $('<td class="notice">').text(notice[idx].noticeNo);
      let $noticeName = $('<td class="notice">').text(notice[idx].noticeName.substring(0,20));
      let $empName = $('<td class="notice">').text(notice[idx].empName);
      let $noticeRegistDate = $('<td class="notice">').text(notice[idx].noticeRegistDate);

      $tr.append($noticeNo);
      $tr.append($noticeName);
      $tr.append($empName);
      $tr.append($noticeRegistDate);

      $table.append($tr);
    },
    error:function (error){
      console.log(error);
    }
  }
)</script>
```



```
@GetMapping("/noticeMain")
@ResponseBody
public String selectNoticeMain(){

  Gson gson = new Gson();

  List<NoticeDTO> noticeList = noticeService.selectMainNotice();

  return gson.toJson(noticeList);
}
```

```
<!-- 메인페이지 공지사항 조회 -->
<select id="selectMainNotice" resultMap="noticeResultMap">
  SELECT
    NOTICE_NO
    , NOTICE_NAME
    , EMP_NAME
    , NOTICE_COUNT
    , NOTICE_REGIST_DATE
  FROM NOTICE
  <! [CDATA[
  WHERE ROWNUM <= 5
  ]]>
  ORDER BY NOTICE_NO DESC
</select>
```

- Ajax를 사용하여 공지사항 글 5개를 출력, +더 보기 클릭 시 공지사항 리스트 페이지로 이동합니다

공지사항 - 공지사항 조회 (관리자 / 가맹점)

```
/* 관리자 공지사항 조회 */
@GetMapping("/admin/list")
public ModelAndView adminNoticeList(ModelAndView mv, HttpServletRequest request){

    /* 현재 페이지 인덱스 가져오기 */
    String currentPage = request.getParameter("currentPage");
    int pageNo = 1;

    if(currentPage != null && !"".equals(currentPage)) {
        pageNo = Integer.parseInt(currentPage);
    }

    /* 검색 조건 가져오기 */
    String searchCondition = request.getParameter("searchCondition");
    /* 검색 값 가져오기 */
    String searchValue = request.getParameter("searchValue");

    Map<String, String> searchMap = new HashMap<>();
    /* HashMap에 검색조건, 검색 값 담아 조회할 때 같이 넘겨주기 */
    searchMap.put("searchCondition", searchCondition);
    searchMap.put("searchValue", searchValue);

    log.info("[NoticeController] searchMap = " + searchMap);

    /* 페이징처리를 위해 전체 개수 조회 */
    int totalCount = noticeService.selectTotalCount(searchMap);

    log.info("[NoticeController] totalCount = " + totalCount);

    /* 한페이지당 최대 조회 개수 */
    int limit = 6;
    /* 페이지 이동 버튼 최대 개수 */
    int buttonAmount = 5;

    SelectCriteria selectCriteria = null;

    /* 페이지 이동시 인자값 같이 넘기기 */
    if(searchCondition != null && !"".equals(searchCondition)) {
        selectCriteria = Pagination.getSelectCriteria(pageNo, totalCount, limit, buttonAmount,
        searchCondition, searchValue);
    } else {
        selectCriteria = Pagination.getSelectCriteria(pageNo, totalCount, limit, buttonAmount);
    }

    log.info("[NoticeController] selectCriteria = " + selectCriteria);

    /* 공지사항 리스트 조회하기 */
    List<NoticeDTO> noticeList = noticeService.selectNoticeList(selectCriteria);

    log.info("[NoticeController] noticeList = " + noticeList);

    mv.addObject("noticeList", noticeList);
    mv.addObject("selectCriteria", selectCriteria);
    mv.setViewName("/notice/admin/adminNoticeList");

    return mv;
}
```

```
<!-- 공지사항 리스트 조회 -->
<select id="selectNoticeList" resultMap="noticeResultMap">
    SELECT
        A.RNUM
        , A.NOTICE_NO
        , A.NOTICE_NAME
        , A.EMP_NAME
        , A.NOTICE_CONTENT
        , A.NOTICE_REGIST_DATE
        , A.NOTICE_COUNT
        , D.EMP_NAME
    FROM (SELECT ROWNUM RNUM
        , B.NOTICE_NO
        , B.NOTICE_NAME
        , B.EMP_NAME
        , B.NOTICE_CONTENT
        , B.NOTICE_COUNT
        , B.NOTICE_REGIST_DATE
    FROM (SELECT C.NOTICE_NO
        , C.NOTICE_NAME
        , C.EMP_NAME
        , C.NOTICE_CONTENT
        , C.NOTICE_COUNT
        , C.NOTICE_REGIST_DATE
    FROM NOTICE C
    LEFT JOIN EMPLOYEE D ON(C.EMP_NAME = D.EMP_NAME)
    WHERE C.NOTICE_NO >= #{startRow}
    ORDER BY C.NOTICE_NO DESC
    ) B
    <!CDATA[
        WHERE ROWNUM <= #{endRow}
    ]>
) A
LEFT JOIN EMPLOYEE D ON (A.EMP_NAME = D.EMP_NAME)
WHERE A.RNUM >= #{startRow}
ORDER BY A.RNUM ASC
</select>
```

```
<!-- 공지사항 조회수 증가 -->
<update id="incrementNoticeCount">
    UPDATE
        NOTICE A
    SET A.NOTICE_COUNT = (SELECT B.NOTICE_COUNT
        FROM NOTICE B
        WHERE B.NOTICE_NO = #{noticeNo}
        ) + 1
    WHERE A.NOTICE_NO = #{noticeNo}
</update>
```

- 공지사항 글을 조회하여 한 페이지당 조회 개수를 6개로 설정하였고, 개수 초과시 다음 페이지로 이동하게 만들었습니다.

- 공지사항 클릭시 조회수 증가하게 만들었습니다.

- 키워드별(작성자,제목,내용)로 원하는 내용의 공지사항을 검색할 수 있습니다.

공지사항 - 공지사항 상세페이지 (관리자 / 가맹점)

```
/* 관리자 공지사항 상세페이지 */
@GetMapping("/admin/detail")
public ModelAndView adminNoticeDetail(ModelAndView mv, HttpServletRequest request){

    /* 공지사항 index 가져오기 */
    int noticeNo = Integer.parseInt(request.getParameter("noticeNo"));
    log.info("[noticeController] noticeNo : " + noticeNo);

    /* 공지사항 index로 해당 게시물 상세정보 조회 */
    NoticeDTO noticeDetail = noticeService.noticeDetailByNo(noticeNo);
    log.info("[noticeController] noticeDetail : " + noticeDetail);

    mv.addObject("noticeDetail", noticeDetail);
    mv.setViewName("/notice/admin/adminNoticeDetail");

    return mv;
}
```

```
<!-- 공지사항 상세페이지 조회 -->
<select id="noticeDetailByNo" resultMap="noticeResultMap" parameterType="_int">
    SELECT
        A.NOTICE_NO
        , A.NOTICE_NAME
        , A.NOTICE_CONTENT
        , A.NOTICE_REGIST_DATE
        , A.NOTICE MODIFY_DATE
        , A.EMP_NAME
        , A.NOTICE_COUNT
        , B.FILE_NO
        , B.ORIGIN_NAME
        , B.FILE_NAME
        , B.SAVED_PATH
        , B.NOTICE_NO
    FROM NOTICE A
    LEFT JOIN NOTICE_FILE B ON (A.NOTICE_NO = B.NOTICE_NO)
    WHERE A.NOTICE_NO = #{noticeNo}
</select>
```

```
<!-- 공지사항 조회수 증가 -->
<update id="incrementNoticeCount">
    UPDATE
        NOTICE A
    SET A.NOTICE_COUNT = (SELECT B.NOTICE_COUNT
        FROM NOTICE B
        WHERE B.NOTICE_NO = #{noticeNo}
    ) + 1
    WHERE A.NOTICE_NO = #{noticeNo}
</update>
```

```
<script>
    // 해당 게시물 클릭 시 상세페이지 이동
    if(document.getElementsByTagName("td")) {
        const $tds = document.getElementsByTagName("td");
        for (let i = 0; i < $tds.length; i++) {

            $tds[i].onclick = function () {
                const noticeNo = this.parentNode.children[0].innerText;
                location.href = "/notice/admin/detail?noticeNo=" + noticeNo
            }
        }
    }
</script>
```

- 공지사항 상세페이지 조회 및 첨부파일 다운로드가 가능합니다.
- 공지사항 클릭시 조회수 증가하게 만들었습니다.

공지사항 - 공지사항 작성 (관리자)

```
/* 관리자 - 공지사항 등록 */
@PostMapping("/admin/insert")
public String noticeInsert(@ModelAttribute NoticeDTO notice,
    @RequestParam(value="file", required=false) MultipartFile file,
    RedirectAttributes rtr
) throws Exception{
    NoticeFileDTO noticeFile = new NoticeFileDTO();
    /* 관지사항 등록 */
    int result = noticeService.noticeInsert(notice);
    log.info("[NoticeController] " + notice);
    log.info("[NoticeController] " + file);
    /* 첨부파일 첨부되었을 시 저장 부른다음 */
    String root = ResourceUtils.getURL("src/main/resources").getPath();
    String filePath = root + "static/uploadFiles";
    log.info("루트-----" + filePath);
    /* 저장 경로는 루트 경로가 됨 */
    File mkdir = new File(filePath);
    if(!mkdir.exists()) {
        mkdir.mkdirs();
    }
    String originFileName = "";
    String ext = "";
    String changeName = "";
    String savedPath = "";
    if(file.getSize() > 0) {
        originFileName = file.getOriginalFilename(); // 원본 파일명
        ext = originFileName.substring(originFileName.lastIndexOf(".")); // 첨부파일의 끝 확장자
        changeName = UUID.randomUUID().toString().replace("-", ""); // 첨부 파일의 랜덤 실제
        savedPath = filePath + "/" + changeName + ext; // 저장 경로 및 파일명
        /* 원본 파일이 있으면 파일을 첨부하는 경우 */
        noticeFile.setOriginName(originFileName);
        noticeFile.setFileName(changeName + ext);
        noticeFile.setSavedPath(savedPath);
        /* 첨부 파일 등록시 파일등록 등록 */
        noticeService.noticeFileInsert(noticeFile);
        try {
            /* 등록 성공 시 파일을 변경하여 저장 */
            file.transferTo(new File(filePath + "\\\" + changeName + ext));
        } catch (IOException e) {
            /* 등록 실패 시 해당 파일 삭제 */
            new File(filePath + "\\\" + changeName + ext).delete();
        }
    }
    /* 관리자 권한 시 alert창으로 message 출력해주기 위해 넘겨줌 */
    rtr.addFlashAttribute("message", "공지사항 등록 성공!");
    return "redirect:/notice/admin/list";
}
```

```
<!-- 관지사항 등록 -->
<insert id="noticeInsert">
    INSERT
        INTO NOTICE
    (
        NOTICE_NO
        , NOTICE_NAME
        , NOTICE_CONTENT
        , NOTICE_REGIST_DATE
        , NOTICE MODIFY_DATE
        , EMP_NAME
    )
    VALUES
    (
        SEQ_NOTICE_NO.NEXTVAL
        , #{ noticeName }
        , #{ noticeContent }
        , TO_CHAR(SYSDATE,'yyyy-MM-dd HH24:MI:SS')
        , NULL
        , #{ empName }
    )
</insert>
```

```
<!-- 관지사항 파일 등록 -->
<insert id="noticeFileInsert">
    <selectKey keyProperty="noticeNo" order="BEFORE" resultType="int">
        SELECT
            SEQ_NOTICE_NO.CURRVAL
        FROM DUAL
    </selectKey>
    INSERT
        INTO NOTICE_FILE
    (
        FILE_NO
        , ORIGIN_NAME
        , FILE_NAME
        , SAVED_PATH
        , NOTICE_NO
    )
    VALUES
    (
        SEQ_NOTICE_FILE_NO.NEXTVAL
        , #{ originName }
        , #{ fileName }
        , #{ savedPath }
        , #{ noticeNo }
    )
</insert>
```

- 관리자 권한이 있는 아이디만 공지사항을 작성할 수 있고 첨부파일 업로드가 가능합니다.

공지사항 - 공지사항 수정 / 삭제 (관리자)

```
/* 관리자 - 공지사항 수정 */
@PostMapping("/admin/update")
public String updateNotice(@ModelAttribute NoticeDTO notice,
    @RequestParam int noticeNo,
    @RequestParam(value="file", required=false) MultipartFile file,
    RedirectAttributes rtr) throws Exception{

    /* 관지사항 정보 수정하기 */
    int result = noticeService.updateNotice(notice);
    log.info("[NoticeController] " + notice);
    log.info("[NoticeController] " + file);

    /* 관지사항 첨부파일 정보 수정 */
    String root = ResourceUtils.getURL("src/main/resources").getPath();
    String filePath = root + "static/uploadfiles";
    log.info("루트-----" + filePath);

    /* 서버 폴더에 파일을 올려온 경로 */
    File mkdir = new File(filePath);
    if(!mkdir.exists()) {
        mkdir.mkdirs();
    }

    String originFileName = "";
    String ext = "";
    String changeName = "";
    String savedPath = "";

    if(file.getSize() > 0) {
        originFileName = file.getOriginalFilename(); // 원본 파일명
        ext = originFileName.substring(originFileName.lastIndexOf(".")); // 첨부파일명 끝 확장자
        changeName = UUID.randomUUID().toString().replace("-", ""); // 첨부 파일의 랜덤 확장
        savedPath = filePath + "/" + changeName + ext; // 서버 경로 및 파일명
    }

    NoticeFileDTO noticeFile = new NoticeFileDTO();

    if(result > 0) {

        if(notice.getNoticeFile() != null){
            /* 수정 할 공지사항에 파일 정보가 존재하면 삭제한다. */
            int result2 = noticeService.deleteNoticeFile(noticeNo);

            /* 파일 정보 삭제 후 파일 세부자료 DTO에 담아 새롭게 한다. */
            if(result2 > 0) {
                noticeFile.setNoticeNo(noticeNo);
                noticeFile.setOriginName(originFileName);
                noticeFile.setFileName(changeName);
                noticeFile.setSavedPath(savedPath);

                noticeService.updateNoticeFile(noticeFile);
            }
        } else {
            /* 파일 정보를 DTO에 담아 파일정보를 등록 */
            noticeFile.setNoticeNo(noticeNo);
            noticeFile.setOriginName(originFileName);
            noticeFile.setFileName(changeName);
            noticeFile.setSavedPath(savedPath);

            noticeService.updateNoticeFile(noticeFile);
        }
    }

    try {
        /* 파일 정보를 파일명 + ext로 변경하여 저장 */
        file.transferTo(new File(filePath + "\\\" + changeName + ext));
    } catch (IOException e) {
        /* 예외 발생 시 해당 파일 삭제 */
        e.printStackTrace();
        new File(filePath + "\\\" + changeName + ext).delete();
    }

    /* 알림 창으로 message를 띠워주기 위해 넘겨줌 */
    rtr.addFlashAttribute("message", "공지사항 수정 성공!");

    return "redirect:/notice/admin/list";
}
```

```
<!-- 공지사항 수정 -->
<update id="updateNotice">
    UPDATE
        NOTICE
    <trim prefix="SET" prefixOverrides=",">
        <if test="noticeName != null and noticeName != ''">
            , NOTICE_NAME = #{ noticeName }
        </if>
        <if test="noticeContent != null and noticeContent != ''">
            , NOTICE_CONTENT = #{ noticeContent }
        </if>
        , NOTICE MODIFY_DATE = TO_CHAR(SYSDATE, 'yyyy-MM-dd HH24:MI:SS')
    </trim>
    WHERE NOTICE_NO = #{ noticeNo }
</update>
```

```
<!-- 공지사항 첨부파일 수정 -->
<insert id="updateNoticeFile">
    INSERT
        INTO NOTICE_FILE
        (
            FILE_NO
            , ORIGIN_NAME
            , FILE_NAME
            , SAVED_PATH
            , NOTICE_NO
        )
    VALUES
        (
            SEQ_NOTICE_FILE_NO.NEXTVAL
            , #{ originName }
            , #{ fileName }
            , #{ savedPath }
            , #{ noticeNo }
        )
</insert>
```

```
/* 관리자 - 공지사항 삭제 */
@DeleteMapping("/admin/delete")
public String deleteNotice(HttpServletRequest request, RedirectAttributes rtr) throws NoticeDeleteException {

    /* 공지사항 index 가져오기 */
    int noticeNo = Integer.parseInt(request.getParameter("noticeNo"));

    /* 공지사항 index로 해당 게시물 삭제하기 */
    int result = noticeService.deleteNotice(noticeNo);

    /* 삭제 성공 시 alert창으로 message를 띠워주기 위해 넘겨줌 */
    rtr.addFlashAttribute("message", "공지사항 삭제 성공!");

    return "redirect:/notice/admin/list";
}
```

```
<!-- 공지사항 삭제 -->
<delete id="deleteNotice" parameterType="NoticeDTO">
    DELETE
        FROM NOTICE
    WHERE NOTICE_NO = #{ noticeNo }
</delete>

<!-- 공지사항 첨부파일 삭제 -->
<delete id="deleteNoticeFile" parameterType="NoticeFileDTO">
    DELETE
        FROM NOTICE_FILE
    WHERE NOTICE_NO = #{ noticeNo }
</delete>
```

- #authentication.getPrincipal()를 사용하여 글 작성한 사람과 로그인한 계정이 동일해야 수정 / 삭제 가능하게 만들었습니다.

1:1문의 - 1:1 문의 조회 (관리자 / 가맹점)

```
/*
 * 본인이 작성한 1:1 문의 조회 */
@GetMapping("/user/list")
public ModelAndView userCounselListPage(ModelAndView mv, @AuthenticationPrincipal User user){

    /* 현재 로그인 storeName 가져오기 */
    String storeName = ((StoreImpl) user).getStoreName();

    log.info("[CounselController] storeName : " + storeName);

    /* 현재 로그인 계정 1:1 문의 리스트 조회하기 */
    List<CounselDTO> counselList = counselService.selectCounselByStoreName(storeName);

    mv.addObject("counselList", counselList);
    mv.setViewName("/counsel/user/userCounselList");

    return mv;
}

/* 관리자 1:1 문의 조회 */
@GetMapping("/admin/list")
public ModelAndView adminCounselListPage(ModelAndView mv, HttpServletRequest request){

    /* 현재 페이지 인덱스 가져오기 */
    String currentPage = request.getParameter("currentPage");
    int pageNo = 1;

    if(currentPage != null && !"".equals(currentPage)) {
        pageNo = Integer.parseInt(currentPage);
    }

    /* 페이징처리를 위해 전체 개수 조회 */
    int totalCount = counselService.selectTotalCount();
    log.info("[CounselController] totalCount = " + totalCount);

    /* 한 페이지당 최대 조회 개수 */
    int limit = 6;
    /* 페이지 이동 버튼 최대 개수 */
    int buttonAmount = 5;

    SelectCriteria selectCriteria = null;

    /* 페이지 이동시 얻어갈 같이 넘기기*/
    selectCriteria = Pagination.getSelectCriteria(pageNo, totalCount, limit, buttonAmount);
    log.info("[CounselController] selectCriteria = " + selectCriteria);

    /* 1:1 문의 리스트 조회하기 */
    List<CounselDTO> counselList = counselService.selectCounsel(selectCriteria);

    mv.addObject("counselList", counselList);
    mv.addObject("selectCriteria", selectCriteria);
    mv.setViewName("/counsel/admin/adminCounselList");

    return mv;
}
```

```
<!-- 페이징을 위한 전체 개수 조회 -->
<select id="selectTotalCount" parameterType="CounselDTO" resultType="_int">
    SELECT
        COUNT(*)
    FROM COUNSEL
    ORDER BY COUNSEL_NO DESC
</select>

<!-- 1:1 문의 리스트 조회하기 -->
<select id="selectCounsel" resultMap="counselResultMap">
    SELECT
        A.RNUM
        , A.COUNSEL_NO
        , A.COUNSEL_NAME
        , A.COUNSEL_CONTENT
        , A.COUNSEL_REGIST_DATE
        , A.STORE_NAME
        , A.COUNSEL_YN
    FROM (SELECT ROWNUM RNUM
        , B.COUNSEL_NO
        , B.COUNSEL_NAME
        , B.COUNSEL_CONTENT
        , B.COUNSEL_REGIST_DATE
        , B.STORE_NAME
        , B.COUNSEL_YN
    FROM (SELECT C.COUNSEL_NO
        , C.COUNSEL_NAME
        , C.COUNSEL_CONTENT
        , C.COUNSEL_REGIST_DATE
        , C.STORE_NAME
        , C.COUNSEL_YN
    FROM COUNSEL C
    ORDER BY C.COUNSEL_NO DESC
    ) B
    <![CDATA[
        WHERE ROWNUM <= #{endRow}
    ]]>
    ) A
    WHERE A.RNUM >= #{startRow}
    ORDER BY 1 ASC
</select>
```

- 1:1문의 글을 조회하여 한 페이지당 조회 개수를 6개로 설정하였고, 개수 초과시 다음 페이지로 이동하게 만들었습니다.

1:1문의 - 1:1 문의 상세페이지 (관리자 / 가맹점)

```
/* 관리자 1:1 문의 등록 상세페이지 */
@GetMapping("/admin/detail")
public ModelAndView adminCounselDetail(ModelAndView mv, @RequestParam int counselNo){

    log.info("[CounselController] counselNo : " + counselNo);

    /* RequestParam으로 1:1 문의 인덱스를 가져와 해당 1:1 문의 내용 조회 */
    CounselDTO counsel = counselService.selectCounselByNo(counselNo);
    /* RequestParam으로 1:1 문의 인덱스를 가져와 해당 1:1 문의 답변내용 조회 */
    CounselApplyDTO counselApply = counselService.selectCounselApplyByNo(counselNo);

    mv.addObject("counsel", counsel);
    mv.addObject("counselApply", counselApply);
    mv.setViewName("/counsel/admin/adminCounselDetail");

    return mv;
}

/* 유저 1:1 문의 상세페이지 */
@GetMapping("user/detail")
public ModelAndView userCounselPage(ModelAndView mv, @RequestParam int counselNo){

    log.info("[CounselController] counselNo : " + counselNo);

    /* RequestParam으로 1:1 문의 인덱스를 가져와 해당 1:1 문의 내용 조회 */
    CounselDTO counsel = counselService.selectCounselByNo(counselNo);
    /* RequestParam으로 1:1 문의 인덱스를 가져와 해당 1:1 문의 답변내용 조회 */
    CounselApplyDTO counselApply = counselService.selectCounselApplyByNo(counselNo);

    mv.addObject("counsel", counsel);
    mv.addObject("counselApply", counselApply);

    mv.setViewName("/counsel/user/userCounselDetail");

    return mv;
}
```

```
<!-- 가맹점 현재 로그인 계정 1:1 문의 리스트 조회하기 -->
<select id="selectCounselByStoreName" resultMap="counselResultMap">
    SELECT
        COUNSEL_NO
        , COUNSEL_NAME
        , COUNSEL_CONTENT
        , COUNSEL_REGIST_DATE
        , STORE_NAME
        , COUNSEL_YN
    FROM COUNSEL
    WHERE STORE_NAME = #{storeName}
</select>

<!-- 관리자 1:1 문의 인덱스로 해당 게시글 내용 조회하기 -->
<select id="selectCounselByNo" resultMap="counselResultMap">
    SELECT
        COUNSEL_NO
        , COUNSEL_NAME
        , COUNSEL_CONTENT
        , COUNSEL_REGIST_DATE
        , STORE_NAME
        , COUNSEL_YN
    FROM COUNSEL
    WHERE COUNSEL_NO = #{counselNo}
</select>
```

```
<script>
    // 해당 게시글 클릭 시 상세페이지 이동
    if(document.getElementsByTagName("td")){
        const $tds = document.getElementsByTagName("td");
        for(let i = 0; i < $tds.length; i++) {

            $tds[i].onclick = function () {
                const counselNo = this.parentNode.children[0].innerText;
                location.href = "/counsel/admin/detail?counselNo=" + counselNo
            }
        }
    }
</script>
```

- 문의 답변 상태 대기 시 답변 내용이 없고 답변 완료 시 답변 내용이 보이게 만들었습니다.

1:1문의 - 1:1 문의 답변 등록 (관리자)

```
/* 관리자 1:1 문의 답변 등록 페이지 이동 */
@GetMapping("/admin/insert")
public ModelAndView adminCounselInsertPage(ModelAndView mv, HttpServletRequest request){

    /* 1:1 문의 답변 등록을 하기 위하여 해당 1:1 문의 index 가져오기 */
    int counselNo = Integer.parseInt(request.getParameter("counselNo"));
    log.info("[CounselController] counselNo : " + counselNo);

    /* 가져온 index로 해당 1:1 문의 내용 조회하기 */
    CounselDTO counsel = counselService.selectCounselByNo(counselNo);
    log.info("[CounselController] counsel : " + counsel);

    mv.addObject("counsel", counsel);
    mv.setViewName("/counsel/admin/adminCounselInsert");

    return mv;
}

/* 관리자 1:1 문의 답변 등록 */
@PostMapping("/admin/insert")
public String adminCounselInsert(@RequestParam String answerContent,
                                 HttpServletRequest request,
                                 ModelAndView mv,
                                 RedirectAttributes rttr){

    /* 1:1 문의 답변 등록을 하기 위하여 해당 1:1 문의 index 가져오기 */
    int counselNo = Integer.parseInt(request.getParameter("counselNo"));
    log.info("[CounselController] counselNo : " + counselNo);

    /* 관리자 1:1 문의 답변 등록 empName 가져오기 */
    String empName = request.getParameter("empName");

    /* 관리자 1:1 문의 답변 등록하기 */
    counselService.insertCounselApply(answerContent, counselNo, empName);

    /* 1:1 문의 답변 성공 시 alert창으로 message 띄워주기 위해 넘겨줌 */
    rttr.addFlashAttribute("message", "1:1문의 답변 등록 완료!");

    return "redirect:/counsel/admin/list";
}
```

```
<!-- 관리자 1:1 문의 답변 등록하기 -->
<insert id="insertCounselApply">
    INSERT
        INTO COUNSEL_APPLY
    (
        COUNSEL_NO
        , ANSWER_CONTENT
        , ANSWER_REGIST_DATE
        , EMP_NAME
    )
    VALUES
    (
        #{ counselNo }
        , #{ answerContent }
        , SYSDATE
        , #{ empName }
    )
</insert>

<!-- 1:1 문의 답변 등록 성공 시 counsel 상태 '답변 완료'로 변경 -->
<update id="updateCounsel">
    UPDATE
        COUNSEL
    SET COUNSEL_YN = '답변 완료'
    WHERE COUNSEL_NO = #{ counselNo }
</update>
```

- 관리자는 가맹점이 등록한 1:1 문의 리스트를 조회하여 답변을 등록할 수 있습니다.

가맹점관리 - 가맹점 조회 (관리자 / 가맹점)

```
/* 관리자 가맹점 조회 */
@GetMapping("/admin/list")
public ModelAndView storeListPage(ModelAndView mv) {

    /* 가맹점 리스트 조회하기 */
    List<StoreDTO> storeList = storeService.storeList();
    log.info("[storeController] storeList : " + storeList);

    mv.addObject("storeList", storeList);
    mv.setViewName("/store/admin/adminStoreList");

    return mv;
}

/* 유저 가맹점 조회 */
@GetMapping("/user/list")
public ModelAndView userStoreListPage(ModelAndView mv) {

    /* 가맹점 리스트 조회하기 */
    List<StoreDTO> storeList = storeService.storeList();
    log.info("[storeController] storeList : " + storeList);

    mv.addObject("storeList", storeList);
    mv.setViewName("/store/user/userStoreList");

    return mv;
}
```

```
<!-- 가맹점 리스트 조회 -->
<select id="storeList" resultMap="storeResultMap">
    SELECT
        STORE_NAME
        , STORE_NO
        , STORE_REP
        , STORE_ADDRESS
        , STORE_PHONE
        , STORE_EMAIL
        , STORE_YN
    FROM STORE
    WHERE STORE_YN = 'Y'
</select>
```

- 전체 가맹점 리스트를 조회할 수 있습니다.

가맹점관리 - 가맹점 등록 (관리자)

```
/* 관리자 가맹점 등록 페이지 이동 */
@GetMapping("/admin/insert")
public String insertStorePage() {
    return "/store/admin/adminStoreInsert";
}

/* 관리자 가맹점 등록 */
@PostMapping("/admin/insert")
public String insertStore(@ModelAttribute StoreDTO store, HttpServletRequest request
, RedirectAttributes rttr, ModelAndView mv) throws StoreInsertException {

    /* 주소API를 사용하여 가져온 정보를 합쳐준다. */
    String storeAddress = request.getParameter("addr1") + " " + request.getParameter("addr2") + " "
    + request.getParameter("addr3");

    store.setStorePhone(store.getStorePhone().replace("-", ""));
    store.setStorePwd(passwordEncoder.encode(store.getStorePwd())); // 정보 보호를 위하여 password는
Encode 해줌.
    store.setStoreAddress(storeAddress);

    log.info("[StoreController] store : " + store);

    /* 가맹점 등록 */
    storeService.insertStore(store);

    /* 가맹점 등록 성공 시 alert창으로 message를 띄워주기 위해 넘겨줌 */
    rttr.addFlashAttribute("message", "가맹점 등록 성공!");

    mv.addObject("store", store);

    return "redirect:/store/admin/list";
}
```

```
<!-- 가맹점 등록 -->
<insert id="insertStore">
    INSERT
        INTO STORE
        (
            STORE_NAME, STORE_ACCOUNT, STORE_REP
            , STORE_ADDRESS, STORE_PHONE, STORE_EMAIL
            ; STORE_ID, STORE_PWD, STORE_NO, STORE_YN
        )
    VALUES
        (
            #{ storeName }, #{ storeAccount }, #{ storeRep }
            , #{ storeAddress }, #{ storePhone }, #{ storeEmail }
            ; #{ storeId }, #{ storePwd }, #{ storeNo }, DEFAULT
        )
</insert>

<!-- 가맹점 신설 대여금 설정 -->
<insert id="insertBalance">
    INSERT
        INTO BALANCE
        (
            STORE_NAME
            , BALANCE
        )
    VALUES
        (
            #{ storeName }
            ; DEFAULT
        )
</insert>

<!-- 가맹점 등록 시 권한 부여 -->
<insert id="insertStoreRole">
    INSERT
        INTO STORE_ROLE
        (
            AUTH_CODE
            , STORE_NAME
            , STORE_ID
        )
    VALUES
        (
            '3'
            , #{ storeName }
            ; #{ storeId }
        )
</insert>
```

```
<script th:inline="javascript">

    const $searchZipCode = document.getElementById("searchZipCode");
    const $goMain = document.getElementById("goMain");

    $searchZipCode.onclick = function() {
        /* 다음 우편번호 검색 창을 오픈하면서 동작할 콜백 메소드를 포함한 객체를 예약 변수로 전달한다. */
        new daum.Postcode({
            onComplete: function(data){
                /* 팝업에서 경도와緯度를 클릭했을 시 실행할 코드를 작성하는 부분 */
                document.getElementById("zipCode").value = data.zonecode;
                document.getElementById("address1").value = data.address;
                document.getElementById("address2").focus();
            }
        }).open();
    }

    $goMain.onclick = function() {
        location.href = "/";
    }
</script>
```

- 관리자는 신규 가맹점을 등록할 수 있고 가맹점의 초기 아이디와 비밀번호를 Encoding 하여 저장합니다.
- 가맹점 주소 등록은 카카오 지도 API를 사용하였습니다.

가맹점관리 - 가맹점 정보 수정/삭제 (관리자)

```
/* 관리자 가맹점 삭제 */
@GetMapping("/admin/delete")
public String deleteStore(HttpServletRequest request, RedirectAttributes rttr) throws Exception {
    /* 가맹점 명 가져오기 */
    String storeName = request.getParameter("storeName");
    log.info("[storeController] storeName : " + storeName);

    /* 가져온 가맹점 명으로 해당 가맹점 정보 삭제 */
    storeService.deleteStore(storeName);

    /* 가맹점 삭제 시 alert창으로 message 띄워주기 위해 넘겨줌 */
    rttr.addFlashAttribute("message", "가맹점 정보 삭제 성공!");

    return "redirect:/store/admin/list";
}

/* 관리자 가맹점 정보 수정 페이지 이동 */
@GetMapping("/admin/update")
public ModelAndView updateStorePage(HttpServletRequest request, RedirectAttributes rttr, ModelAndView mv) {
    /* 가맹점 정보 수정을 위하여 해당 가맹점 명을 가져옴 */
    String storeName = request.getParameter("storeName");
    log.info("[StoreController] storeName : " + storeName);

    /* 가맹점 명으로 해당 가맹점 정보 조회 */
    StoreDTO store = storeService.selectStoreByName(storeName);

    mv.addObject("store", store);
    mv.setViewName("/store/admin/adminStoreUpdate");

    return mv;
}

/* 관리자 가맹점 정보 수정 */
@PostMapping("/admin/update")
public String updateStore(@ModelAttribute StoreDTO store, RedirectAttributes rttr) throws StoreUpdateException {
    log.info("[StoreController] store : " + store);
    log.info("[StoreController] storeId : " + store.getStoreId());

    /* 해당 가맹점 정보 수정 */
    storeService.updateStore(store);

    /* 수정 성공 시 alert창으로 message를 띄워주기 위해 넘겨줌 */
    rttr.addFlashAttribute("message", "가맹점 정보 수정 성공!");

    return "redirect:/store/admin/list";
}
```

```
<!-- 가맹점 명을 빼고 가맹점 정보 조회 -->
<select id="selectStoreByName" resultMap="storeResultMap">
    SELECT
        STORE_NAME
        , STORE_ACCOUNT
        , STORE_REP
        , STORE_ADDRESS
        , STORE_PHONE
        , STORE_EMAIL
        , STORE_ID
        , STORE_PWD
        , STORE_NO
        , STORE_YN
    FROM STORE
    WHERE STORE_NAME = #{storeName}
</select>

<!-- 가맹점 정보 수정 -->
<update id="updateStore">
    UPDATE
        STORE
        <trim prefix="SET" prefixOverrides=",">
            <if test="storeAccount != null and storeAccount != ''">
                , STORE_ACCOUNT = #{storeAccount}
            </if>
            <if test="storeRep != null and storeRep != ''">
                , STORE_REP = #{storeRep}
            </if>
            <if test="storeAddress != null and storeAddress != ''">
                , STORE_ADDRESS = #{storeAddress}
            </if>
            <if test="storePhone != null and storePhone != ''">
                , STORE_PHONE = #{storePhone}
            </if>
            <if test="storeEmail != null and storeEmail != ''">
                , STORE_EMAIL = #{storeEmail}
            </if>
            <if test="storeNo != null and storeNo != ''">
                , STORE_NO = #{storeNo}
            </if>
        </trim>
        WHERE STORE_ID = #{storeId}
</update>

<!-- 가맹점 정보 삭제 -->
<update id="deleteStore" parameterType="StoreDTO">
    UPDATE
        STORE
        SET STORE_YN = 'N'
        WHERE STORE_NAME = #{storeName}
</update>
```

- 관리자는 가맹점 정보를 수정할 수 있고 폐점 시 가맹점 정보를 삭제할 수 있습니다.

예치금관리 - 입금신청 (가맹점)

```
/* 유저 입금신청 */
@GetMapping("/user/insert")
public String userAccountInsert(@RequestParam int accountDeposit, RedirectAttributes rttr,
@AuthenticationPrincipal User user){

    log.info("[AccountController] accountDeposit : " + accountDeposit);

    /* 현재 로그인 storeName 가져오기 */
    String storeName = ((StoreImpl) user).getStoreName();

    log.info("[AccountController] storeName : " + storeName);

    /* 입금신청 하기 */
    accountService.accountInsert(accountDeposit, storeName);

    /* 입금신청 시 alert창으로 message 띄워주기 위해 넘겨줌 */
    rttr.addFlashAttribute("message", "입금 신청을 하였습니다.");

    return "redirect:/account/user/list";
}
```

```
<!-- 가맹점 입금신청 시 입금신청 정보 등록 -->
<insert id="accountInsert">
    INSERT
        INTO ACCOUNT
    (
        STORE_NAME, DEPOSIT_NUM, ACCOUNT_DATE
        , ACCOUNT_DEPOSIT, ACCOUNT_YN
    )
    VALUES
    (
        #{ storeName }, SEQ_DEPOSIT_NUM.NEXTVAL, SYSDATE
        , #{ accountDeposit }, '대기'
    )
</insert>

<!-- 가맹점 입금신청 성공 시 관리자 입금신청 정보 등록 -->
<insert id="accountApplyInsert">
    INSERT
        INTO ACCOUNT_APPLY
    (
        STORE_NAME
        , DEPOSIT_NUM
        , DEPOSIT_REGIST_DATE
        , DEPOSIT_YN
    )
    VALUES
    (
        #{ storeName }
        , SEQ_DEPOSIT_NUM.CURRVA
        , SYSDATE
        , '대기'
    )
</insert>
```

```
<script type="text/javascript">
    // 입금신청 버튼 클릭 시 modal창 띄우기
    $(function(){

        const accountPop = $('.dimm, .account_popup')
        $('.account_popup_open').on('click', function(){
            $(accountPop).fadeIn(300);
        });
        $('.btn-close, .btn-cancel').on('click', function(){
            $(accountPop).fadeOut(300);
        });
    });
</script>
```

- 모달창을 사용하여 본사에 입금 신청하여 승인 시 잔액이 충전됩니다.

예치금관리 - 입금신청 조회 (관리자)

```
/* 관리자 입금신청 내역 조회 */
@GetMapping("/admin/list")
public ModelAndView accountAdminList(ModelAndView mv, HttpServletRequest request) {

    /* 현재 페이지 인덱스 가져오기 */
    String currentPage = request.getParameter("currentPage");
    int pageNo = 1;

    if(currentPage != null && !"".equals(currentPage)) {
        pageNo = Integer.parseInt(currentPage);
    }

    /* 페이지당 최대 조회 개수 */
    int totalCount = accountService.selectTotalCount();

    log.info("[AccountController] totalCount = " + totalCount);

    /* 한 페이지당 최대 조회 개수 */
    int limit = 6;
    /* 페이지 이동 버튼 최대 개수 */
    int buttonAmount = 5;

    SelectCriteria selectCriteria = null;

    /* 페이지 이동시 인자값 같이 넘기기 */
    selectCriteria = Pagination.getSelectCriteria(pageNo, totalCount, limit, buttonAmount);
    log.info("[AccountController] selectCriteria = " + selectCriteria);

    /* 입금 신청 리스트 조회하기 */
    List<AccountApplyDTO> accountApplyList = accountService.selectAccountApplyList(selectCriteria);
    mv.addObject("accountApplyList", accountApplyList);
    mv.addObject("selectCriteria", selectCriteria);
    mv.setViewName("/account/admin/adminAccountList");

    return mv;
}
```

```
<!-- 페이지를 위한 전체 개수 조회 -->
<select id="selectTotalCount" resultType="_int" parameterType="AccountDTO">
    SELECT
        COUNT(*)
    FROM ACCOUNT
    ORDER BY DEPOSIT_NUM DESC
</select>

<!-- 본사 가맹점 입금신청 조회 -->
<select id="selectAccountApplyList" resultMap="accountApplyResultMap">
    SELECT
        A.RNUM
        , A.STORE_NAME
        , A.DEPOSIT_NUM
        , A.DEPOSIT_REGIST_DATE
        , A.DEPOSIT_YN
        , D.ACCOUNT_DEPOSIT
    FROM(SELECT ROWNUM RNUM
        , B.STORE_NAME
        , B.DEPOSIT_NUM
        , B.DEPOSIT_REGIST_DATE
        , B.DEPOSIT_YN
        , C.SELECT C.STORE_NAME
            , C.DEPOSIT_NUM
            , C.DEPOSIT_REGIST_DATE
            , C.DEPOSIT_YN
        FROM ACCOUNT_APPLY C
        LEFT JOIN ACCOUNT D ON (C.DEPOSIT_NUM = D.DEPOSIT_NUM)
        ORDER BY C.DEPOSIT_NUM DESC
    )B
    <![CDATA[
        WHERE RNUM <= #{endRow}
    ]]>
    )A
    LEFT JOIN ACCOUNT D ON (A.DEPOSIT_NUM = D.DEPOSIT_NUM)
    WHERE A.RNUM >= #{startRow}
    ORDER BY 1 ASC
</select>
```

- 가맹점에서 입금 신청한 내역을 조회하여 한 페이지당 조회 개수를 6개로 설정하여 초과 시 다음 페이지로 이동하게 만들었습니다.

예치금관리 - 입금 승인 및 반려 (관리자)

```
/* 관리자 입금신청 승인 */
@GetMapping("/admin/update")
public String adminAccountApplyUpdate(@ModelAttribute AccountApplyDTO accountApply, HttpServletRequest request, RedirectAttributes rtr){

    /* 입금신청 금액 가져오기 */
    int accountDeposit = Integer.parseInt(request.getParameter("accountDeposit"));
    /* 입금신청한 storeName 가져오기 */
    String storeName = request.getParameter("storeName");

    log.info("[AccountController] accountDeposit : " + accountDeposit);
    log.info("[AccountController] storeName : " + storeName);
    log.info("[AccountController] accountApply : " + accountApply);

    /* 입금 승인시 입금신청한 store 잔액 update 해주기 */
    accountService.balanceUpdate(accountApply, accountDeposit, storeName);

    /* 입금 승인시 alert창으로 message 띄워주기 위해 넘겨줌 */
    rtr.addFlashAttribute("message", "입금을 승인 하였습니다.");

    return "redirect:/account/admin/list";
}

/* 관리자 입금신청 반려 */
@GetMapping("/admin/update2")
public String adminAccountApplyUpdate2(HttpServletRequest request, RedirectAttributes rtr){

    /* 입금신청한 index 가져오기 */
    int depositNum = Integer.parseInt(request.getParameter("depositNum"));

    /* 입금신청한 index 반려하기 */
    accountService.accountApplyUpdate2(depositNum);

    /* 입금신청 반려시 alert으로 message 띄워주기 위해 넘겨줌 */
    rtr.addFlashAttribute("message", "입금을 반려 하였습니다.");

    return "redirect:/account/admin/list";
}
```

```
<!-- 기행점 입금 승인 시 입금신청 상태 '승인'으로 변경 -->
<update id="accountApplyUpdate">
    UPDATE
        ACCOUNT_APPLY
        SET DEPOSIT_YN = '승인'
        WHERE DEPOSIT_NUM = #{ depositNum }
    </update>

<!-- 가맹점 입금 승인 시 관리자 입금신청 상태 '승인'으로 변경 -->
<update id="accountUpdate">
    UPDATE
        ACCOUNT
        SET ACCOUNT_YN = '승인'
        WHERE DEPOSIT_NUM = #{ depositNum }
    </update>

<!-- 입금 승인 시 해당 store 잔액 수정 -->
<update id="balanceUpdate">
    UPDATE
        BALANCE A
        SET A.BALANCE = (SELECT B.BALANCE
                            FROM BALANCE B
                            WHERE B.STORE_NAME = #{ storeName }
                        ) + #{ accountDeposit }
        WHERE A.STORE_NAME = #{ storeName }
    </update>

<!-- 기행점 입금 반려 시 입금신청 상태 '반려'로 변경 -->
<update id="accountApplyUpdate2">
    UPDATE
        ACCOUNT_APPLY
        SET DEPOSIT_YN = '반려'
        WHERE DEPOSIT_NUM = #{ depositNum }
    </update>

<!-- 가맹점 입금 반려 시 관리자 입금신청 상태 '반려'로 변경 -->
<update id="accountUpdate2">
    UPDATE
        ACCOUNT
        SET ACCOUNT_YN = '반려'
        WHERE DEPOSIT_NUM = #{ depositNum }
    </update>
```

- 가맹점에서 입금 신청한 내역을 조회하여 입금 승인 및 반려를 할 수 있다.

예치금관리 - 입/출금 내역 조회 (가맹점)

```
/* 가맹점 입출금 내역 조회 */
@GetMapping("/user/bank")
public ModelAndView userBankPage(ModelAndView mv, @AuthenticationPrincipal User user){

    /* 현재 로그인 storeName 가져오기 */
    String storeName = ((StoreImpl) user).getStoreName();

    /* 현재 로그인 잔액 조회 */
    BalanceDTO balance = accountService.selectBalance(storeName);
    /* 현재 로그인 입금내역 조회 */
    List<StoreDepositDTO> storeDeposit = accountService.selectStoreDeposit(storeName);
    /* 현재 로그인 출금내역 조회 */
    List<StoreBreakdownDTO> storeBreakdown = accountService.selectStoreBreakdown(storeName);

    log.info("[AccountController] balance : " + balance);

    mv.addObject("balance", balance);
    mv.addObject("storeDeposit", storeDeposit);
    mv.addObject("storeBreakdown", storeBreakdown);
    mv.setViewName("/account/user/userBank");

    return mv;
}
```

```
<!-- 관리자 가맹점 산액 조회 -->
<select id="balanceSelect" resultMap="balanceResultMap">
    SELECT
        STORE_NAME
        , BALANCE
    FROM BALANCE
</select>

<!-- 관리자 가맹점 입금액 조회 -->
<select id="selectStoreDeposit" resultMap="storeDepositResultMap">
    SELECT
        STORE_NAME
        , DEPOSIT_MONEY
        , DEPOSIT_DATE
        , CONTENT
    FROM STORE_DEPOSIT
    WHERE STORE_NAME = #{ storeName }
    ORDER BY DEPOSIT_DATE
</select>

<!-- 가맹점 출금액 조회 -->
<select id="selectStoreBreakdown" resultMap="storeBreakdownResultMap">
    SELECT
        STORE_NAME
        , BREAKDOWN_DATE
        , WITHDRAWAL
        , CONTENT
    FROM STORE_BREAKDOWN
    WHERE STORE_NAME = #{ storeName }
    ORDER BY BREAKDOWN_DATE
</select>
```

```
<-- 반품시 반품금액만큼 잔액 증가 -->
<update id="updateAccount" parameterType="ReItemDTO">
    UPDATE
        STORE_ITEM_INFO
    SET STORE_AMOUNT = STORE_AMOUNT + (SELECT
        #{} returnCount
    FROM RETURN_ITEM_INFO
    WHERE REF_R_NO = #{ rNo }
    AND ITEM_NO = #{ itemNo })
```

WHERE ITEM_NO = (SELECT
 #{} itemNo
FROM RETURN_ITEM_INFO
WHERE REF_R_NO = #{ rNo }
AND ITEM_NO = #{ itemNo })

```
</update>

<!-- 수불 상권 시 카드 수불금액만큼 -해수거 -->
<update id="updateStoreBalance">
    UPDATE BALANCE
    SET BALANCE = BALANCE - (SELECT
        B.TOTAL_PRICE
    FROM ORDER_HANDLER B
    WHERE B.CART_NO = #{ cartNo })
    WHERE STORE_NAME = #{ storeName }
    AND BALANCE >= #{ totalPrice }
</update>

<!-- 캐시 내역 -->
<insert id="insertStoreBreakdown">
    INSERT
        INTO STORE_BREAKDOWN
    (
        STORE_NAME
        , BREAKDOWN_DATE
        , WITHDRAWAL
        , CONTENT
    )
    VALUES
    (
        #{} storeName
        , TO_CHAR(SYSDATE, 'yyyy-MM-dd HH24:MT:SS')
        , #{} totalPrice
        , '출금'
    )
</insert>
```

- 가맹점은 입금, 발주, 반품을 할 시 입출금 기록을 남겨 조회할 수 있습니다.

예치금관리 - 가맹점 잔액 조회 (관리자)

```
/* 관리자 가맹점 잔액 조회 */
@GetMapping("/admin/balance")
public ModelAndView adminBalanceSelect(ModelAndView mv){

    /* 전체 가맹점 잔액 조회 */
    List<BalanceDTO> balanceSelect = accountService.balanceSelect();

    mv.addObject("balanceSelect", balanceSelect);
    mv.setViewName( "/account/admin/adminBalanceSelect" );

    return mv;
}
```

```
<!-- 관리자 가맹점 잔액 조회 -->
<select id="balanceSelect" resultMap="balanceResultMap">
    SELECT
        STORE_NAME
        , BALANCE
    FROM BALANCE
</select>
```

- 관리자는 전체 가맹점의 잔액을 조회할 수 있습니다.