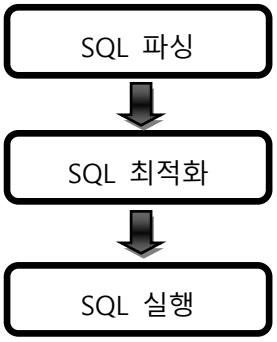
1.1 order by의 문제

데이터베이스를 이용할 때 웹이나 애플리케이션에 가장 신경 쓰는 부분은 1) 빠르게 처리되는 것 2) 필요한 양만큼만 데이터를 가져오는 것이다. 예를 들어 거의 모든 웹페이지에서 페이징을 하는 이유는 최소한의 필요한 데이터만을 가져와서 빠르게 화면에 보여주기 위함이다.

빠르게 동작하는 SQL을 위해서는 먼저 order by를 이용하는 작업을 가능하면 사용하지 말아야 한다. order by는 데이터가 많은 경우에 엄청난 성능의 저하를 가져오기 때문에 1) 데이터가 적을 경우와 2) 정렬을 빠르게 할 수 있는 방법이 있는 경우가 아니라면 order by는 주의해야만 한다.

1.2. 실행 계획과 order by

실행 계획은 말 그대로 'SQL을 데이터베이스에서 어떻게 처리할 것인가?'에 대한 것이다.



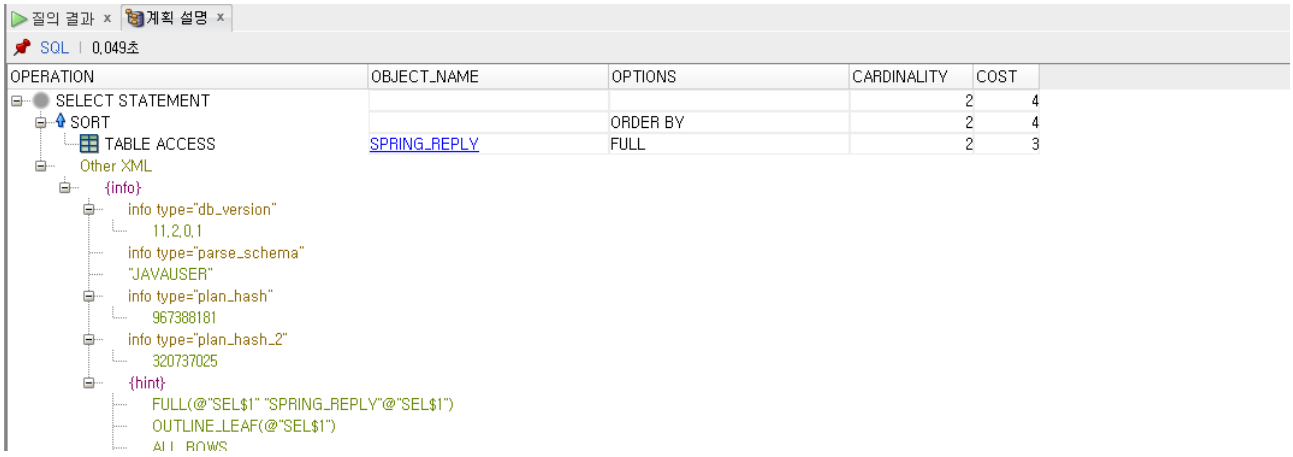
SQL 구문에 오류가 있는지 SQL을 실행해야 하는 대상가 존재하는지를 검사

SQL이 실행되는데 필요한 비용(COST)을 계산하게 되며 이 계산된 값을 기초로 어떤 방식으로 실행하는 것이 가장 좋다는 것을 판단하는 실행계획을 세운다

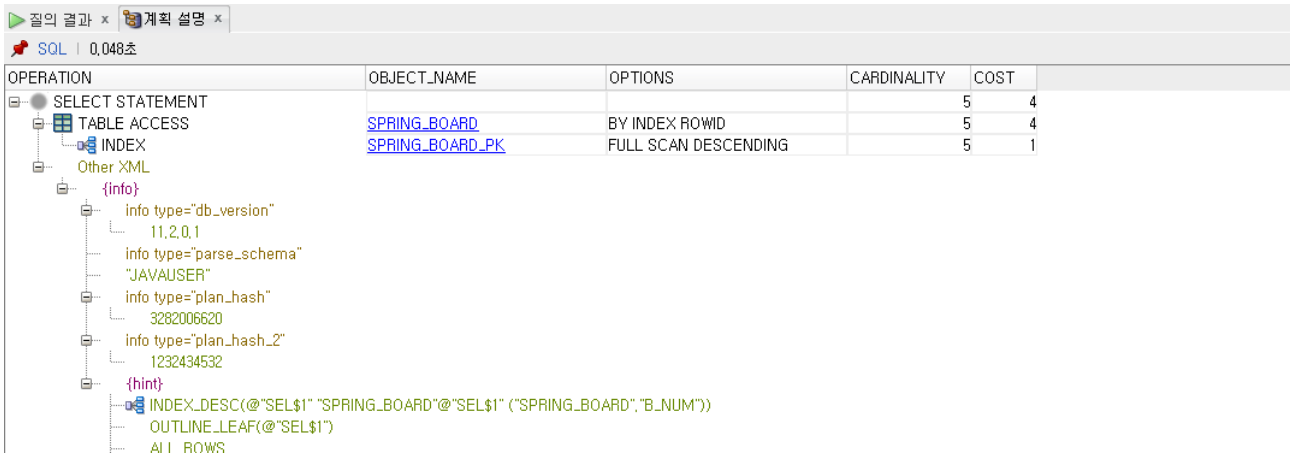
세워진 실행 계획을 통해서 메모리상에 데이터를 읽거나 물리적인 공간에서 데이터를 로딩하는 등의 작업을 하게 된다.

```
select * from spring_reply order by b_num desc;
```

상단의 버튼 중에서 SQL에 대해서 '실행 계획'을 쉽게 볼수 있도록 버튼을 제공한다. (F10 클릭)



```
select * from spring_board order by b_num desc;
```



2.1 order by보다는 인덱스

데이터가 많은 경우 order by가 문제가 된다면 이를 해결하기 위한 방법이 인덱스(index)를 이용해서 정렬을 생략하는 방법이다. 왜냐면 인덱스라는 존재가 이미 정렬된 구조이므로 이를 이용해서 별도의 정렬을 하지 않는 방법이다.

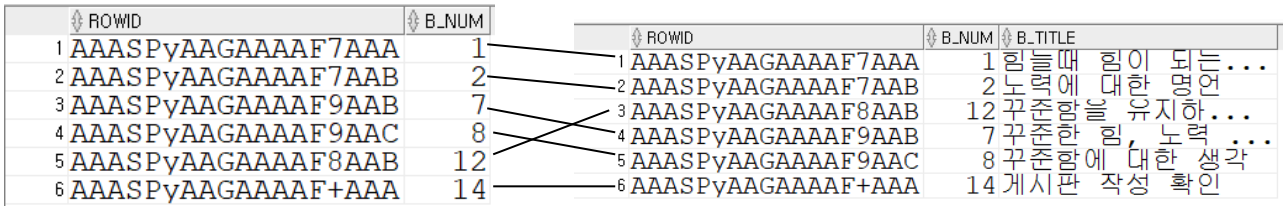
```
select /*+ INDEX_DESC(spring_board spring_board_pk) */
        *
from spring_board where b_num > 0;
```

1) SORT를 하지 않는다는 점.

2) spring_board를 바로 접근하는 것이 아니라 spring_board_pk를 이용해서 접근한 점

select rowid, b_num from spring_board;	select rowid, b_num, b_title, b_name, b_date from spring_board;

식별자로 만들어진 인덱스 테이블의 데이터

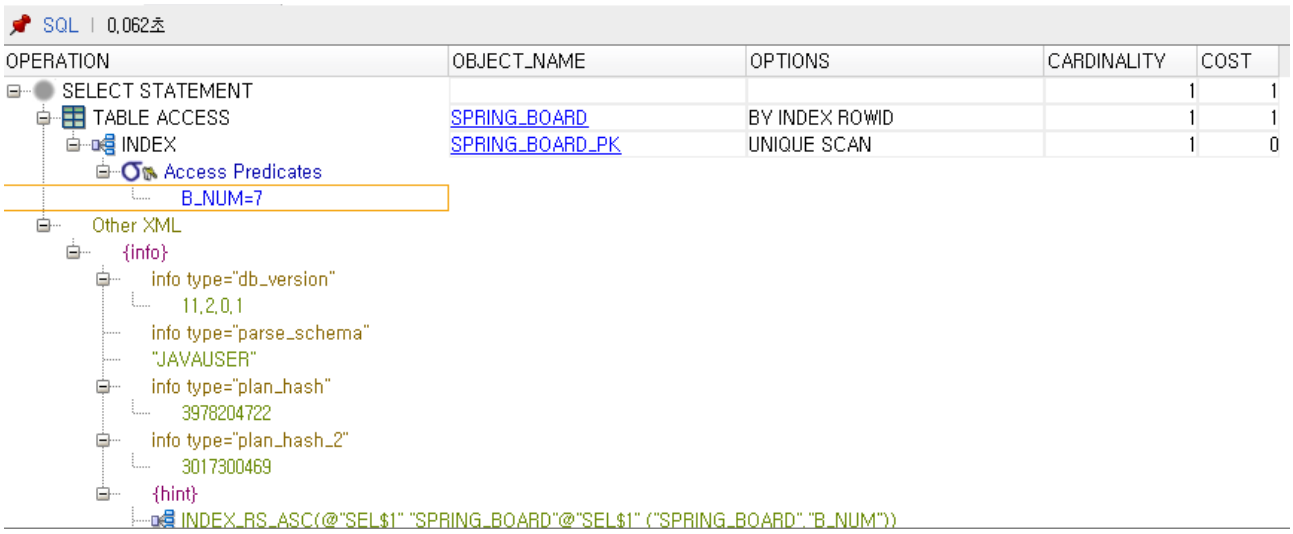


	ROWID	B_NUM
1	AAASPyAAGAAAAF7AAA	1
2	AAASPyAAGAAAAF7AAB	2
3	AAASPyAAGAAAAF9AAB	7
4	AAASPyAAGAAAAF9AAC	8
5	AAASPyAAGAAAAF8AAB	12
6	AAASPyAAGAAAAF+AAA	14

	ROWID	B_NUM	B_TITLE
1	AAASPyAAGAAAAF7AAA	1	힘들때 힘이 되는...
2	AAASPyAAGAAAAF7AAB	2	노력에 대한 명언
3	AAASPyAAGAAAAF8AAB	12	꾸준함을 유지하...
4	AAASPyAAGAAAAF9AAB	7	꾸준한 힘, 노력 ...
5	AAASPyAAGAAAAF9AAC	8	꾸준함에 대한 생각
6	AAASPyAAGAAAAF+AAA	14	게시판 작성 확인

rowid는 데이터베이스 내의 주소에 해당하는데 모든 데이터는 자신만의 주소를 가지고 있다.

```
select * from spring_board where b_num = 7;
```



SQL | 0.062초

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	SPRING_BOARD	BY INDEX ROWID	1	1
INDEX	SPRING_BOARD_PK	UNIQUE SCAN	1	0

Access Predicates
B_NUM=7

Other XML
{info}
info type="db_version"
11.2.0.1
info type="parse_schema"
"JAVAUSER"
info type="plan_hash"
3978204722
info type="plan_hash_2"
3017300469
{hint}
INDEX_RS_ASC(@"SEL$1" "SPRING_BOARD"@"SEL$1" ("SPRING_BOARD"."B_NUM"))

3.1 인덱스와 오라클 힌트(hint)

오라클은 select문을 전달할 때 힌트(hint)라는 것을 사용할 수 있다. 힌트(hint)는 개발자가 데이터베이스에 어떤 방식으로 실행해 줘야하는지를 명시하기 때문에 조금 강제성이 부여되는 방식이다.

select문을 작성할 때 힌트는 잘못 작성되어도 실행할 때는 무시되기만 하고 별도의 에러는 발생하지 않는다. 우선 힌트를 사용할 때에는 다음과 같은 문법을 사용한다.

[구문]

/*+ 로 시작하고 */로 끝난다.

FULL(테이블명): /*+ FULL(spring_board) */

INDEX_ASC, INDEX_DESC : /*+ INDEX_DESC(spring_board spring_board_pk) */

```
select /*+ INDEX_DESC(spring_board spring_board_pk) */
            b_num, b_name, b_title, b_date
from spring_board;
```

4.1 rownum과 인라인 뷰

전체가 아닌 필요한 만큼의 데이터를 가져오는 방식에 대해 정리해 보자. 오라클 데이터베이스는 페이지 처리를 위해서 rownum이라는 키워드를 사용해서 데이터에 순번을 붙여 사용한다.

```
select rownum, b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
from spring_board ;
```

한 페이지당 10개의 데이터를 출력한다고 가정하면 rownum 조건을 where 구문에 추가해서 다음과 같이 작성할 수 있다. 1번부터 10번까지의 데이터를 출력한다.

```
select /*+ INDEX_DESC(spring_board spring_board_pk) */
        rownum, b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
from spring_board where rownum <= 10;
```

2페이지 데이터를 구하기 위해 11번부터 20번까지의 데이터를 출력하도록 다음과 같이 코딩한다.

```
select /*+ INDEX_DESC(spring_board spring_board_pk) */
        rownum, b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
from spring_board where rownum > 10 and rownum <= 20;
```

그런데 결과는 아무 데이터도 출력하지 않는다. rownum 조건은 반드시 1이 포함되어야 한다.

```
select /*+ INDEX_DESC(spring_board spring_board_pk) */
        rownum, b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
from spring_board where rownum <= 20; -- 반드시 1이 포함되도록 해야 한다.
```

4.2 인라인뷰(In-line View) 처리

```
select
          rnum, b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
    from (
              select /*+ INDEX_DESC(spring_board spring_board_pk) */
                        rownum as rnum, b_num, b_name, b_title, b_date
              from spring_board
              where rownum <= 20
         ) boardlist
    where rnum > 10;
```

이 과정을 정리하면 다음과 같은 순서이다

- 필요한 순서로 정렬된 데이터에 rownum을 붙인다.
- 처음부터 해당 페이지의 데이터를 rownum<=20과 같은 조건을 이용해서 구한다.
- 구해놓은 데이터를 하나의 테이블처럼 간주하고 인라인뷰로 처리한다.
- 인라인뷰에서 필요한 데이터만을 남긴다.

5.1 페이징 처리

페이징 처리를 위해서 필요한 파라미터는 1) 페이지 번호(pageNum), 2) 한 페이지당 몇 개의 데이터(amount)를 보여줄 것인지가 결정되어야 한다.

```
SELECT
        b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
    FROM (
            SELECT /*+ INDEX_DESC(spring_board spring_board_pk) */
                    rownum as rnum, b_num, b_name, b_title, b_date
            FROM spring_board
            WHERE rownum <= 1 * 10        <-- pageNum * amount
        ) boardlist
    WHERE rnum > (1 - 1) * 10;
                                          <-- (pageNum -1) * amount
```

com.spring.common.vo.CommonVO.java

```
package com.spring.common.vo;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

//import lombok.Data;

//@Data
@ToString
@Setter
@Getter
public class CommonVO {
        private int pageNum = 0;       //페이지 번호
        private int amount = 0;        //페이지에 보여줄 데이터 수

        //조건검색시 사용할 필드(검색대상, 검색단어)
        private String search = "";
        private String keyword = "";

        //날자검색시 사용할 필드(시작일, 종료일)
        private String start_date = "";
        private String end_date = "";
```

```
	public CommonVO() {
		this(1, 10);
	}

	public CommonVO(int pageNum, int amount) {
		this.pageNum = pageNum;
		this.amount = amount;
	}
}
```

기본 쿼리문을 변경하여 보자.

Board.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="com.spring.client.board.dao.BoardDao">
	<!-- 게시판 리스트 검색 부분 추가 -->
	<select id="boardList" parameterType="board" resultType="board">
		SELECT b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
		FROM spring_board
		<where>
			<if test="search=='b_title'">
				<![CDATA[ b_title LIKE '%'|| #{keyword} ||'%' ]]>

			</if>
			<if test="search=='b_content'">
				<![CDATA[ b_content LIKE '%'|| #{keyword} ||'%' ]]>

			</if>
			<if test="search=='b_name'">
				<![CDATA[ b_name LIKE '%'|| #{keyword} ||'%' ]]>

			</if>
		</where>
		ORDER BY b_num desc
	</select>
...중략
</mapper>
```

Board.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="com.spring.client.board.dao.BoardDao">
	<!-- 게시판 리스트 조회(페이징 처리와 검색 처리) -->
	<!--
	힌트(hint): 개발자가 데이터베이스에 어떤 방식으로 실행해 줘야 하는지를 명시하기
	때문에 강제성을 부여한다.
	구문: /*+로 시작하고 */ 로 종료된다.
	/*+ INDEX_DESC(테이블명 인덱스명[기본키 설정시 제약조건명으로 인덱스 자동 생성]) */ -->
	<!--rownum: SQL이 실행된 결과에 넘버링해준다. -->
	<select id="boardList" parameterType="board" resultType="board">
		<![CDATA[
		SELECT
		b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
		FROM (
		SELECT /*+ INDEX_DESC(spring_board spring_board_pk) */
			rownum as rnum, b_num, b_name, b_title, b_date
		FROM spring_board
		WHERE ]]>
		<trim prefix="(" suffix=") AND " prefixOverrides="AND">
			<if test="search=='b_title'">
				<![CDATA[ b_title LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
			<if test="search=='b_content'">
				<![CDATA[ b_content LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
			<if test="search=='b_name'">
				<![CDATA[ b_name LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
		</trim>
	    <![CDATA[    rownum <= #{pageNum} * #{amount}
		    ) boardlist
	 WHERE rnum > (#{pageNum} - 1) * #{amount}
	 ]]>
	</select>
...중략
</mapper>
```

페이징 처리를 하기 위해서는 전체 레코드 개수가 필요하다. 그래서 BoardDao 인터페이스에 메서드 선언하고 Board.xml에 추가 작업을 한다.

com.spring.client.board.dao.BoardDao

```
package com.spring.client.board.dao;

import java.util.List;
import org.apache.ibatis.annotations.Param;
import com.spring.client.board.vo.BoardVO;

public interface BoardDao {
	public List<BoardVO> boardList(BoardVO bvo);
	public int boardListCnt(BoardVO bvo);		// 추가
	public int boardInsert(BoardVO bvo);
	public BoardVO boardDetail(BoardVO bvo);
	public int pwdConfirm(BoardVO bvo);
	public int boardUpdate(BoardVO bvo);
	public int boardDelete(int b_num);
}
```

Board.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="com.spring.client.board.dao.BoardDao">
.. 중략
	<!-- 전체 레코드 수 조회  -->
	<select id="boardListCnt" parameterType="board" resultType="int">
		SELECT count(*) FROM spring_board
		<trim prefix=" where (" suffix=")" >
			<if test="search=='b_title'">
				<![CDATA[ b_title LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
			<if test="search=='b_content'">
				<![CDATA[ b_content LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
			<if test="search=='b_name'">
				<![CDATA[ b_name LIKE '%'|| #{keyword} ||'%' ]]>
			</if>
		</trim>
	</select>
```

```
.. 중략
</mapper>
```

그런데 소스를 확인해 보면 검색 후 데이터 조회와 검색 후 레코드 개수 조회를 진행하기 위해서 if 요소가 반복되고 있다. 그래서 그 부분을 수정하면 다음과 같다.

Board.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="com.spring.client.board.dao.BoardDao">
	<sql id="boardSearch">
		<if test="search=='b_title'">
			<![CDATA[ b_title LIKE '%'|| #{keyword} ||'%' ]]>
		</if>
		<if test="search=='b_content'">
			<![CDATA[ b_content LIKE '%'|| #{keyword} ||'%' ]]>
		</if>
		<if test="search=='b_name'">
			<![CDATA[ b_name LIKE '%'|| #{keyword} ||'%' ]]>
		</if>
	</sql>
	<!-- 게시판 리스트 조회(페이징 처리와 검색 처리) -->
	<select id="boardList" parameterType="board" resultType="board">
	<![CDATA[
		SELECT
		b_num, b_name, b_title, to_char(b_date,'YYYY-MM-DD') as b_date
		FROM (
			SELECT /*+ INDEX_DESC(spring_board spring_board_pk) */
				rownum as rnum, b_num, b_name, b_title, b_date
			 FROM spring_board
		WHERE ]]>
			<trim prefix="(" suffix=") AND " prefixOverrides="AND">
				<include refid="boardSearch"></include>
			</trim>
	 <![CDATA[	rownum <= #{pageNum} * #{amount}
		) boardlist
		WHERE rnum > (#{pageNum} - 1) * #{amount}
	 ]]>
	</select>
```

```
        <!-- 전체 레코드 수 조회 -->
        <select id="boardListCnt" parameterType="board" resultType="int">
                SELECT count(*) FROM   spring_board
                <trim prefix=" where (" suffix=")" >
                        <include refid="boardSearch"></include>
                </trim>
        </select>
.. 중략
</mapper>
```

예전에 생성해 놓은 BoardMapperTests 클래스를 통해서 확인해 보자.

com.spring.client.board.dao.BoardMapperTests.java

```
package com.spring.client.board.dao;

import java.util.List;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.spring.client.board.vo.BoardVO;

import lombok.Setter;
import lombok.extern.log4j.Log4j;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardMapperTests {

        @Setter(onMethod_ = @Autowired)
        private BoardDao boardDao;

        @Test
        public void testBoardList() {
                BoardVO bvo = new BoardVO();

                bvo.setPageNum(1);
```

```
                bvo.setAmount(10);

                // 검색 조건 부여
                //bvo.setSearch("b_title");
                //bvo.setKeyword("노력");

                List<BoardVO> list =   boardDao.boardList(bvo);
                for(BoardVO vo : list) {
                        log.info(vo);
                }
        }
}
```

5.2 화면에 페이징 처리

화면에 페이징 처리를 하기 위해서는 우선적으로 여러 가지 필요한 정보들이 존재한다. 화면에 페이지는 크게 다음과 같은 정보들이 필요하다.

- 현재 페이지 번호(page)
- 이전과 다음으로 이동 가능한 링크의 표시 여부(prev, next)
- 화면에서 보여지는 페이지의 시작번호와 끝번호(startPage, endPage)

5.2.1 끝 페이지 번호와 시작 페이지 번호

페이징 처리를 하기 위해서 우선적으로 필요한 정보는 현재 사용자가 보고 있는 페이지(page)의 정보이다. 보통 바로가기 페이지 번호를 1페이지부터 10페이지로 한다면 5페이지를 본다면 화면의 페이지 번호는 1페이지부터 시작해야 하고 만일 15페이지를 본다면 11페이지부터 시작해야 한다.

그때 보통 끝번호는 다음과 같은 방식으로 처리한다.

페이징의 끝번호(endPage) 계산
this.endPage = (int) (Math.ceil(cvo.getPageNum() / 10.0)) * 10;

Math.ceil() 메서드는 소수점을 올림으로 처리하기 때문에 다음과 같은 상황이 가능하다.

- 1페이지의 경우 : Math.ceil(0.1) * 10 = 10
- 2페이지의 경우 : Math.ceil(0.2) * 10 = 10
- 10페이지의 경우 : Math.ceil(1) * 10 = 10
- 11페이지의 경우 : Math.ceil(1.1) * 10 = 20
- 20페이지의 경우 : Math.ceil(2) * 10 = 20

이때 전체 데이터 수가 적다면 10페이지로 끝나면 안되는 상황이 생길 수도 있기에 아직은 개선이 더 필요하다. 그리고 화면에 10개씩 보여준다면 시작번호(startPage)는 무조건 끝번호(endPage)에서 9를 빼주면 된다.

페이징의 시작번호(startPage) 계산
this.startPage = this.endPage - 9;

끝번호(endPage)는 전체 데이터수(total)에 의해서 영향을 받는다.

전체 데이터수(total)를 통한 endPage의 재계산

```
int realEnd = (int) (Math.ceil((total * 1.0) / cvo.getAmount()));

if (realEnd <= this.endPage) {
            this.endPage = realEnd;
}
```

5.2.2 이전과 다음

이전의 경우는 시작번호가 1보다 큰 경우라면 존재하게 된다.

이전(prev) 계산

```
this.prev = this.startPage > 1;
```

다음은 realEnd가 끝번호(endPage)보다 큰 경우에만 존재하게 된다.

다음(next) 계산

```
this.next = this.endPage < realEnd;
```

위 내용을 가지고 페이징 처리를 위한 클래스를 생성한다.

com.spring.common.vo.PageDTO

```
package com.spring.common.vo;

import lombok.Getter;
import lombok.ToString;

@Getter
@ToString
public class PageDTO {
        private int startPage; // 화면에서 보여지는 페이지의 시작 번호
        private int endPage;    // 화면에서 보여지는 페이지의 끝번호
        private boolean prev, next; // 이전과 다음으로 이동한 링크의 표시 여부

        private int total;
        private CommonVO cvo;

        public PageDTO(CommonVO cvo, int total) {
            this.cvo = cvo;
            this.total = total;

            /* 페이징의 끝번호(endPage) 구하기
                this.endPage = (int) (Math.ceil(페이지번호 / 10.0)) * 10;*/
```

```
            this.endPage = (int) (Math.ceil(cvo.getPageNum() / 10.0)) * 10;

            /* 페이징의 시작번호(startPage) 구하기 */
            this.startPage = this.endPage - 9;

            /* 끝 페이지 구하기*/
            int realEnd = (int) (Math.ceil((total * 1.0) / cvo.getAmount()));

            if (realEnd <= this.endPage) {
               this.endPage = realEnd;
            }

            /* 이전(prev) 구하기 */
            this.prev = this.startPage > 1;

            /* 다음(next) 구하기 */
            this.next = this.endPage < realEnd;
         }
}
```

com.spring.client.board.controller.BoardController

```
package com.spring.client.board.controller;

import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.spring.client.board.service.BoardService;
import com.spring.client.board.vo.BoardVO;
import com.spring.common.vo.PageDTO;

import lombok.AllArgsConstructor;
import lombok.extern.log4j.Log4j;
```

```
@Controller
@Log4j
@RequestMapping("/board/*")
@AllArgsConstructor
public class BoardController {

	private BoardService boardService;

	/**************************************************************
	 * 글목록 구현하기(페이징 처리 목록 조회)
	 **************************************************************/
	@RequestMapping(value="/boardList", method = RequestMethod.GET)
	//@GetMapping("/boardList")
	public String boardList(@ModelAttribute("data") BoardVO bvo, Model model) {
		log.info("boardList 호출 성공");
		// 전체 레코드 조회
		List<BoardVO> boardList = boardService.boardList(bvo);
		model.addAttribute("boardList", boardList);

		// 전체 레코드수 구현
		int total = boardService.boardListCnt(bvo);
		// 페이징 처리
		model.addAttribute("pageMaker", new PageDTO(bvo, total));

		return "board/boardList";
	}
... 중략
}
```

com.spring.client.board.service.BoardService

```
package com.spring.client.board.service;

import java.util.List;

import com.spring.client.board.vo.BoardVO;

public interface BoardService {
	public List<BoardVO> boardList(BoardVO bvo);
	public int boardListCnt(BoardVO bvo);
```

```
	public int boardInsert(BoardVO bvo);
	public BoardVO boardDetail(BoardVO bvo);
	public int pwdConfirm(BoardVO bvo);
	public BoardVO updateForm(BoardVO bvo);
	public int boardUpdate(BoardVO bvo);
	public int boardDelete(int b_num);
	public int replyCnt(int b_num);
}
```

com.spring.client.board.service.BoardServiceImpl

```
package com.spring.client.board.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.spring.client.board.dao.BoardDao;
import com.spring.client.board.vo.BoardVO;
import com.spring.client.reply.dao.ReplyDao;
import lombok.Setter;

@Service
public class BoardServiceImpl implements BoardService {

	@Setter(onMethod_=@Autowired)
	private BoardDao boardDao;

	@Setter(onMethod_=@Autowired)
	private ReplyDao replyDao;

	// 글목록 구현
	@Override
	public List<BoardVO> boardList(BoardVO bvo) {
		List<BoardVO> list = null;
		list = boardDao.boardList(bvo);
		return list;
	}

	// 전체 레코드 수 구현
```

```
        @Override
        public int boardListCnt(BoardVO bvo) {
                return boardDao.boardListCnt(bvo);
        }
... 중략
}
```

/WEB-INF/views/board/boardList.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<!DOCTYPE html>
<html>
        <head>
                <script type="text/javascript">
                    $(function(){
                        ... 중략
                        $(".paginate_button a").click(function(e) {
                            e.preventDefault();
                            $("#f_search").find("input[name='pageNum']").val($(this).attr("href"));
                            goPage();
                        });
                    });

                    /* 검색을 위한 실질적인 처리 함수 */
                    function goPage(){
                        if($("#search").val()=="all"){
                                $("#keyword").val("");
                        }
                        $("#f_search").attr({
                                "method":"get",
                                "action":"/board/boardList"
                        });
                        $("#f_search").submit();
                    }
                </script>
        </head>
        <body>
                <div class="contentContainer container">
```

```
            ... 중략
          <%-- ============== 검색기능 시작 ==================== --%>
          <div id="boardSearch" class="text-right">
              <form id="f_search" name="f_search" class="form-inline">
                    <input type="hidden" name="pageNum" value="${pageMaker.cvo.pageNum}">
                    <input type="hidden" name="amount" value="${pageMaker.cvo.amount}">
          ... 중략
          <%-- ============== 리스트 종료  ============= --%>
          <%-- ============== 페이징 출력 시작 =========== --%>
           <div class="text-center">
                <ul class="pagination">
                   <c:if test="${pageMaker.prev}">
                        <li class="paginate_button previous">
                            <a href="${pageMaker.startPage -1}">Previous</a>
                        </li>
                   </c:if>
                   <c:forEach var="num" begin="${pageMaker.startPage}"
                                        end="${pageMaker.endPage}">
                      <li class="paginate_button   ${pageMaker.cvo.pageNum == num ? 'active':''}">
                          <a href="${num}">${num}</a>
                      </li>
                   </c:forEach>
                   <c:if test="${pageMaker.next}">
                              <li class="paginate_button next">
                                      <a href="${pageMaker.endPage +1 }">Next</a>
                              </li>
                        </c:if>
                    </ul>
           </div>
          <%-- ============== 페이징 출력 종료 ============= --%>

          <%-- ============= 글쓰기 버튼 출력 시작 =========== --%>
          <div class="contentBtn   text-right">
                <input type="button" value="글쓰기" id="insertFormBtn" class="btn btn-success">
           </div>
          <%-- ============== 글쓰기 버튼 출력 종료 ========== --%>
      </div>
    </body>
</html>
```