12

CHAPTER

전체 텍스트 검색과 파티션

Contents

01 전체 텍스트 검색

02 파티션

1-1 전체 텍스트 검색의 개요

- 전체 텍스트 검색(full-text search)
  - 긴 문장으로 구성된 열의 내용을 검색할 때 인덱스를 사용하여 검색 시간을 줄이는 개념

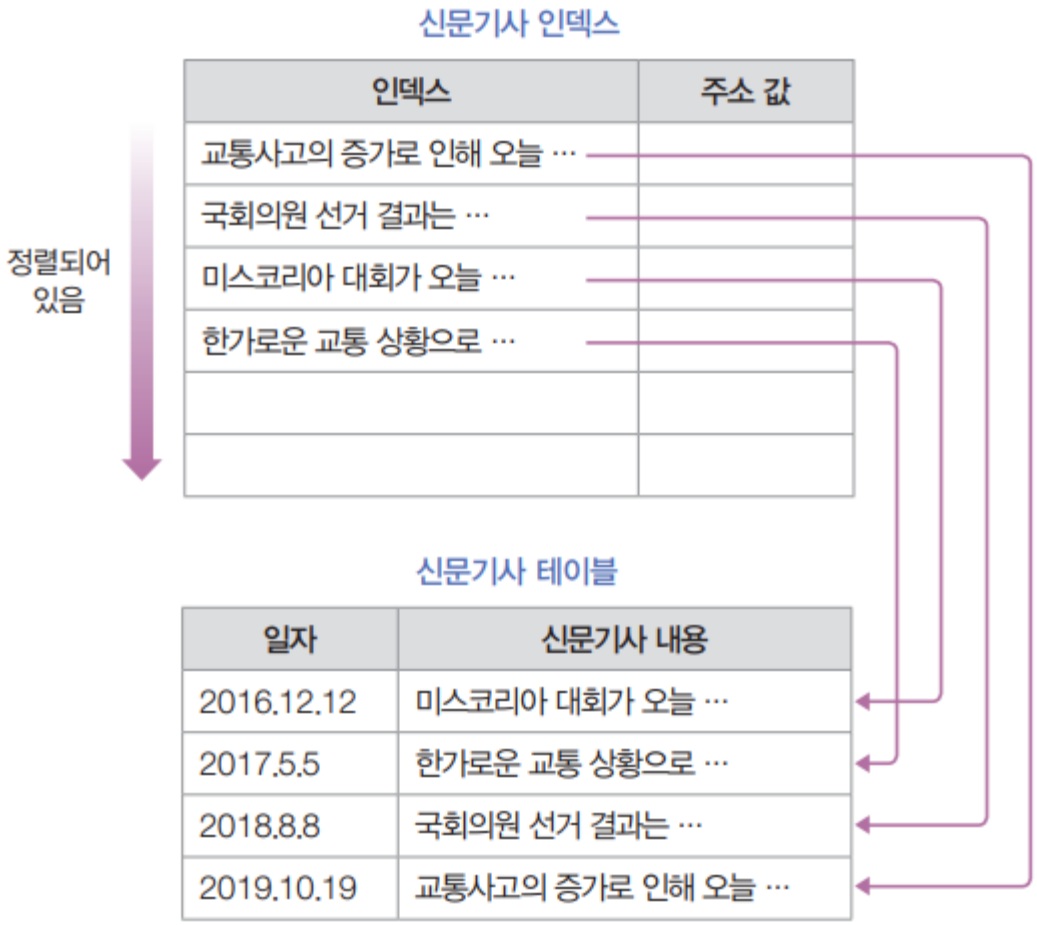


그림 12-1 신문기사 테이블과 신문기사 인덱스의 개념

1-1 전체 텍스트 검색의 개요

- [그림 12-1]의 테이블 구조

```
CREATE TABLE 신문기사테이블
( 일자 DATE,
  신문기사내용 VARCHAR(4000)
)
GO
CREATE INDEX 신문기사인덱스 ON 신문기사테이블(신문기사내용);
GO
```

- 신문기사 테이블에서 교통사고와 관련된 신문기사 검색

```
SELECT * FROM 신문기사테이블 WHERE 신문기사내용 = '교통사고의 증가로 인해 오늘 ...';
```

- LIKE 연산자를 사용하여 '교통'과 관련된 신문기사 검색

```
SELECT * FROM 신문기사테이블 WHERE 신문기사내용 LIKE '교통%'
```

- '교통'이라는 키워드가 텍스트의 앞에 있든 중간에 있든 상관없이 검색

```
SELECT * FROM 신문기사테이블 WHERE 신문기사내용 LIKE '%교통%'
```

1-1 전체 텍스트 검색의 개요

- 전체 텍스트 검색
  - 첫 글자뿐만 아니라 중간의 단어나 문장으로도 인덱스를 생성하고, 그 인덱스(정확히는 전체 텍스트 인덱스)를 사용하여 순식간에 검색 결과 출력
  - 긴 문자로 구성된 구조화되지 않은 텍스트 데이터(예를 들면 신문기사)를 빠르게 검색

1-2 전체 텍스트 인덱스 생성과 삭제

- 일반적인 인덱스와의 차이점
  - 전체 텍스트 인덱스는 CHAR, VARCHAR, TEXT 열에만 생성할 수 있음
  - 인덱스 힌트의 사용이 일부 제한됨
  - 여러 개의 열에 FULLTEXT 인덱스를 지정할 수 있음

1-2 전체 텍스트 인덱스 생성과 삭제

- 전체 텍스트 인덱스 생성
  - 형식 1

```
CREATE TABLE 테이블이름
(
 ...
 열이름 데이터형식,
 ...
FULLTEXT 인덱스이름(열이름)
);
```

  - 형식2

```
CREATE TABLE 테이블이름
(
 ...
 열이름 데이터형식,
 ...
);
ALTER TABLE 테이블이름
ADD FULLTEXT(열이름);
```

  - 형식 3

```
CREATE TABLE 테이블이름
(
 ...
 열이름 데이터형식,
 ...
);
CREATE FULLTEXT INDEX 인덱스이름
ON 테이블이름(열이름);
```

1-2 전체 텍스트 인덱스 생성과 삭제

- 전체 텍스트 인덱스 삭제

```
ALTER TABLE 테이블이름
   DROP INDEX FULLTEXT(열이름);
```

1-3 중지 단어

- 중지 단어(stopword)
  - 실제로 검색을 할 때 무시해도 되는 단어
  - MySQL은 INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD 테이블에 약 36개의 중지 단어가 있음

	value
▶	a
	about
	an
	are
	as
	at
	be
	by
	com
	de
	en

1-4 전체 텍스트 검색을 위한 쿼리

- 형식

```
MATCH(col1, col2, ...) AGAINST(expr [search_modifier])

search_modifier:
 {
     IN NATURAL LANGUAGE MODE
   | IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION
   | IN BOOLEAN MODE
   | WITH QUERY EXPANSION
 }
```

- 자연어 검색
  - 전체 텍스트 검색을 할 때 특별히 옵션을 지정하지 않거나 IN NATURAL LANGUAGE MODE를 붙이면 자연어 검색을 함
  - 자연어 검색은 단어가 정확한 것을 검색
  - '영화'라는 단어가 들어간 기사 검색

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화');
```

  - '영화' 또는 '배우'와 같이 두 단어 중 하나가 포함된 기사 검색

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화 배우');
```

1-4 전체 텍스트 검색을 위한 쿼리

- 불린 모드 검색
  - 단어나 문장이 정확히 일치하지 않는 것도 검색
  - 필수를 뜻하는 +, 제외를 뜻하는 -, 부분 검색을 위한 * 등의 다양한 연산자 지원
  - '영화를', '영화가', '영화는'과 같이 '영화'가 앞에 들어간 모든 결과 검색

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화*' IN BOOLEAN MODE);
```

  - '영화 배우'라는 단어가 들어 있는 기사 검색

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화 배우' IN BOOLEAN MODE);
```

  - '영화 배우'라는 단어가 들어 있는 기사 중에서 '공포'라는 단어가 포함된 것만 검색

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화 배우 +공포' IN BOOLEAN MODE);
```

  - '영화 배우'라는 단어가 들어 있는 기사 중에서 '남자'라는 단어가 포함된 기사를 검색 결과에서 제외

```
SELECT * FROM newspaper
   WHERE MATCH(article) AGAINST('영화 배우 -남자' IN BOOLEAN MODE);
```

[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

1 시스템 변수 값 변경하기

1-1 innodb_ft_min_token_size 시스템 변수의 값 확인

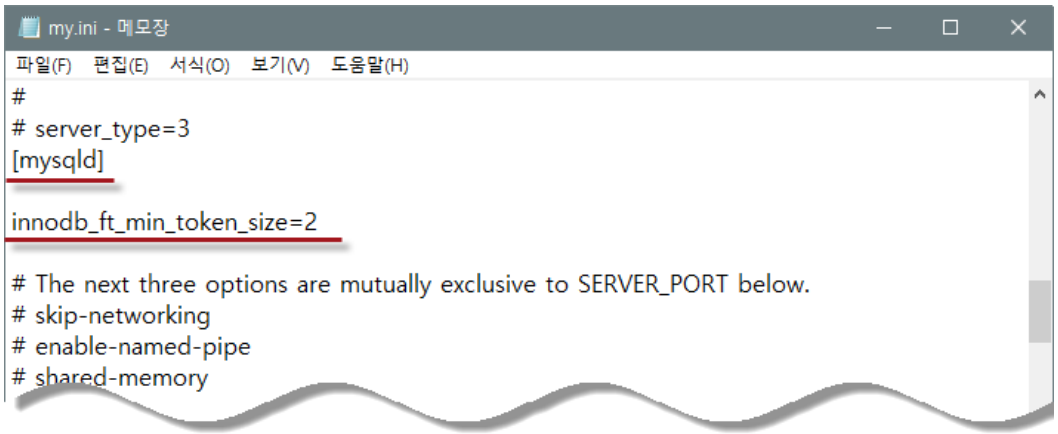
```
SHOW VARIABLES LIKE 'innodb_ft_min_token_size';
```

1-2 메모장에 my.ini 파일 열기

```
cmd
CD %PROGRAMDATA%
CD MySQL
CD "MySQL Server 8.0"
NOTEPAD my.ini
```

1-3 다음 내용 추가

```
innodb_ft_min_token_size=2
```



[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

1-4 MySQL 재시작

```
NET STOP MySQL
NET START MySQL
```

1-5 다시 Workbench 실행

2 데이터베이스 생성하기

2-1 데이터베이스와 테이블 생성

```
CREATE DATABASE IF NOT EXISTS FulltextDB;
USE FulltextDB;
DROP TABLE IF EXISTS FulltextTbl;
CREATE TABLE FulltextTbl
(  id int AUTO_INCREMENT PRIMARY KEY,  -- 고유 번호
   title VARCHAR(15) NOT NULL,  -- 영화 제목
   description VARCHAR(1000)  -- 영화 내용 요약
);
```

[실습 12-1] 전체 텍스트 검색하기

2-2 샘플 데이터 몇 건 입력

```
INSERT INTO FulltextTbl VALUES
(NULL, '광해, 왕이 된 남자', '왕위를 둘러싼 권력 다툼과 당쟁으로 혼란이 극에 달한 광해군 8년'),
(NULL, '간첩', '남한 내에 고장간첩 5만 명이 암약하고 있으며 특히 권력 핵심부에도 침투해 있다.'),
(NULL, '남자가 사랑할 때', '대책 없는 한 남자이야기. 형 집에 얹혀 살며 조카한테 무시당하는 남자'),
(NULL, '레지던트 이블 5', '인류 구원의 마지막 퍼즐, 이 여자가 모든 것을 끝낸다.'),
(NULL, '파괴자들', '사랑은 모든 것을 파괴한다! 한 여자를 구하기 위한, 두 남자의 잔인한 액션 본능!'),
(NULL, '킹콩을 들다', '역도에 목숨을 건 시골소녀들이 만드는 기적 같은 신화.'),
(NULL, '테드', '지상 최대 황금찾기 프로젝트! 500년 전 사라진 황금도시를 찾아라!'),
(NULL, '타이타닉', '비극 속에 침몰한 세기의 사랑, 스크린에 되살아날 영원한 감동'),
(NULL, '8월의 크리스마스', '시한부 인생 사진사와 여자 주차 단속원과의 미묘한 사랑'),
(NULL, '늑대와 춤을', '늑대와 친해져 모닥불 아래서 함께 춤을 추는 전쟁 영웅 이야기'),
(NULL, '국가대표', '동계올림픽 유치를 위해 정식 종목인 스키점프 국가대표팀이 급조된다.'),
(NULL, '쇼생크 탈출', '그는 누명을 쓰고 쇼생크 감옥에 감금된다. 그리고 역사적인 탈출.'),
(NULL, '인생은 아름다워', '귀도는 삼촌의 호텔에서 웨이터로 일하면서 또다시 도라를 만난다.'),
(NULL, '사운드 오브 뮤직', '수녀 지망생 마리아는 명문 트랩가의 가정교사로 들어간다'),
(NULL, '매트릭스', '2199년. 인공 두뇌를 가진 컴퓨터가 지배하는 세계.');
```

2-3 '남자'라는 단어 검색

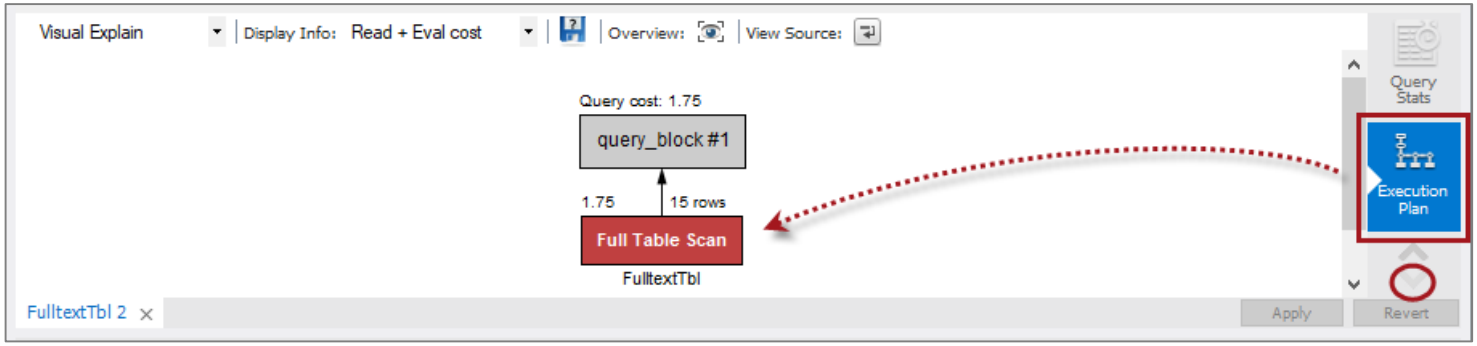
```
SELECT * FROM FulltextTbl WHERE description LIKE '%남자%';
```

	id	title	description
▶	3	남자가 사랑할 때	대책 없는 한 남자이야기. 형 집에 얹혀 살며 조카한테 무시당하는 남자
	5	파괴자들	사랑은 모든 것을 파괴한다! 한 여자를 구하기 위한, 두 남자의 잔인한 액션 본능!

[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

2-4 실행 계획 확인



3 전체 텍스트 인덱스 생성하기

3-1 전체 텍스트 인덱스 생성

```
CREATE FULLTEXT INDEX idx_description ON FulltextTbl(description);
```

3-2 생성한 인덱스의 정보 확인

```
SHOW INDEX FROM FulltextTbl;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
fulltexttbl	0	PRIMARY	1	id	A	15	NULL	NULL		BTREE	
fulltexttbl	1	idx_description	1	description	NULL	15	NULL	NULL	YES	FULLTEXT	

[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

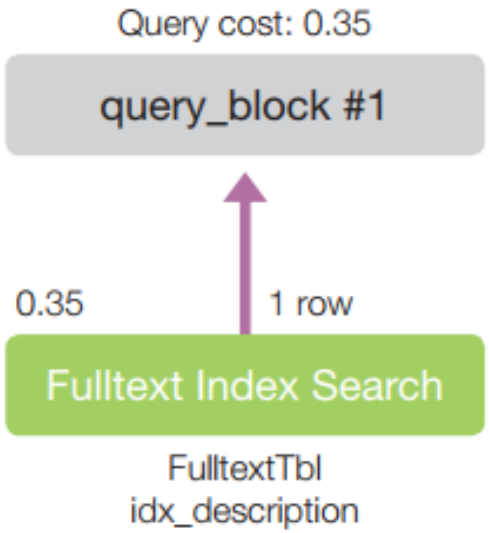
4 전체 텍스트 인덱스를 활용하여 검색하기

4-1 '남자'라는 단어가 들어 있는 행 검색

```
SELECT * FROM FulltextTbl WHERE MATCH(description) AGAINST('남자*' IN BOOLEAN MODE);
```

	id	title	description
▶	3	남자가 사랑할 때	대책 없는 한 남자이야기. 형 집에 얹혀 살며 조카한테 무시당하는 남자
	5	파괴자들	사랑은 모든 것을 파괴한다! 한 여자를 구하기 위한, 두 남자의 잔인한 액션 본능!

4-2 실행 계획 확인



[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

4-3 ' 남자' 또는 '여자'라는 단어가 들어 있는 행 검색

```
SELECT * , MATCH(description) AGAINST('남자 * 여자 * ' IN BOOLEAN MODE) AS 점수
    FROM FulltextTbl WHERE MATCH(description) AGAINST('남자 * 여자 * ' IN BOOLEAN MODE);
```

id	title	description	점수
5	파괴자들	사랑은 모든 것을 파괴한다! 한 여자를 구하기 위한, 두 남자의 잔인한 액션 본능!	0.9771181344985962
3	남자가 사랑할 때	대책 없는 한 남자 이야기. 형 집에 얹혀 살며 조카한테 무시당하는 남자	0.4885590672492981
4	레지던트 이블 5	인류 구원의 마지막 퍼즐, 이 여자가 모든 것을 끝낸다.	0.4885590672492981
9	8월의 크리스마스	시한부 인생 사진사와 여자 주차 단속원과의 미묘한 사랑	0.4885590672492981

4-4 ' 남자'와 '여자'라는 단어가 둘 다 포함된 영화 출력

```
SELECT * FROM FulltextTbl
    WHERE MATCH(description) AGAINST('+남자 * +여자 * ' IN BOOLEAN MODE);
```

4-5 ' 남자'라는 단어가 들어 있는 영화 중에서 '여자'가 포함된 영화 제외

```
SELECT * FROM FulltextTbl
    WHERE MATCH(description) AGAINST('남자 * -여자 * ' IN BOOLEAN MODE);
```

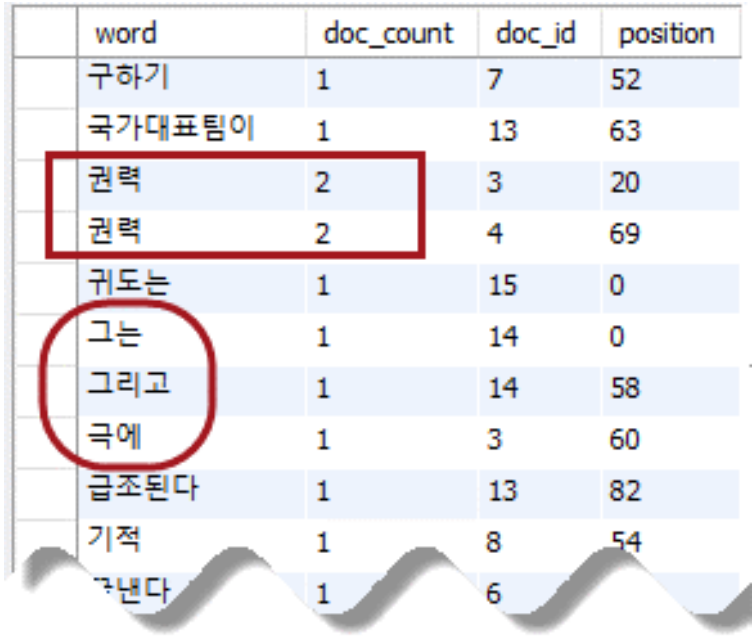
[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

5 전체 텍스트 인덱스가 생성된 단어 확인하기

5-1 전체 텍스트 인덱스로 만들어진 단어 확인

```
SET GLOBAL innodb_ft_aux_table = 'fulltextdb/fulltexttbl'; -- 모두 소문자
SELECT word, doc_count, doc_id, position
      FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE;
```



word	doc_count	doc_id	position
구하기	1	7	52
국가대표팀이	1	13	63
권력	2	3	20
권력	2	4	69
귀도는	1	15	0
그는	1	14	0
그리고	1	14	58
극에	1	3	60
급조된다	1	13	82
기적	1	8	54
깐다	1	6	

5-2 중지 단어 추가

```
DROP INDEX idx_description ON FulltextTbl;
```

[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

5-3 사용자가 추가할 중지 단어를 저장할 테이블 생성

```
CREATE TABLE user_stopword (value VARCHAR (30));
```

5-4 중지 단어를 테이블에 삽입

```
INSERT INTO user_stopword VALUES ('그는'), ('그리고'), ('극에');
```

5-5 중지 단어용 테이블을 시스템 변수 innodb_ft_server_stopword_table에 설정

```
SET GLOBAL innodb_ft_server_stopword_table = 'fulltextdb/user_stopword'; -- 모두 소문자
SHOW GLOBAL VARIABLES LIKE 'innodb_ft_server_stopword_table';
```

Variable_name	Value
innodb_ft_server_stopword_table	fulltextdb/user_stopword

5-6 다시 전체 텍스트 인덱스 생성

```
CREATE FULLTEXT INDEX idx_description ON FulltextTbl(description);
```

[실습 12-1] 전체 텍스트 검색하기

교재 429~435p 참고

5-7 전체 텍스트 인덱스에 생성된 단어 확인

```
SELECT word, doc_count, doc_id, position
    FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE;
```

word	doc_count	doc_id	position
국가대표팀이	1	13	63
권력	2	3	20
권력	2	4	69
귀도는	1	15	0
급조된다	1	13	82
기적	1	8	54
끝낸다	1	6	63
남자	1	5	89
남자의	1	7	74
남자이야기	1	5	18
남한	1	4	0
내에	1	4	

2-1 파티션의 개요

- 파티션(partition)
  - 대량의 데이터를 한 테이블에 저장할 때 그 내용을 물리적으로 별도의 테이블에 분리해서 저장하는 기법
  - 몇 개의 파티션으로 분리되었든 사용자 입장에서는 하나의 테이블로 보이기 때문에 테이블을 사용하는 방법은 같음
  - MySQL 내부적으로는 데이터가 분리되어 처리되기 때문에 시스템의 성능 향상에 큰 도움이 됨
  - 테이블을 분리할 때는 테이블의 범위에 따라서 서로 다른 파티션에 저장하는 것이 가장 보편적

2-2 파티션 구현

- cookDB의 회원 테이블(partTBL)을 출생 연도별로 3개의 파티션으로 구분

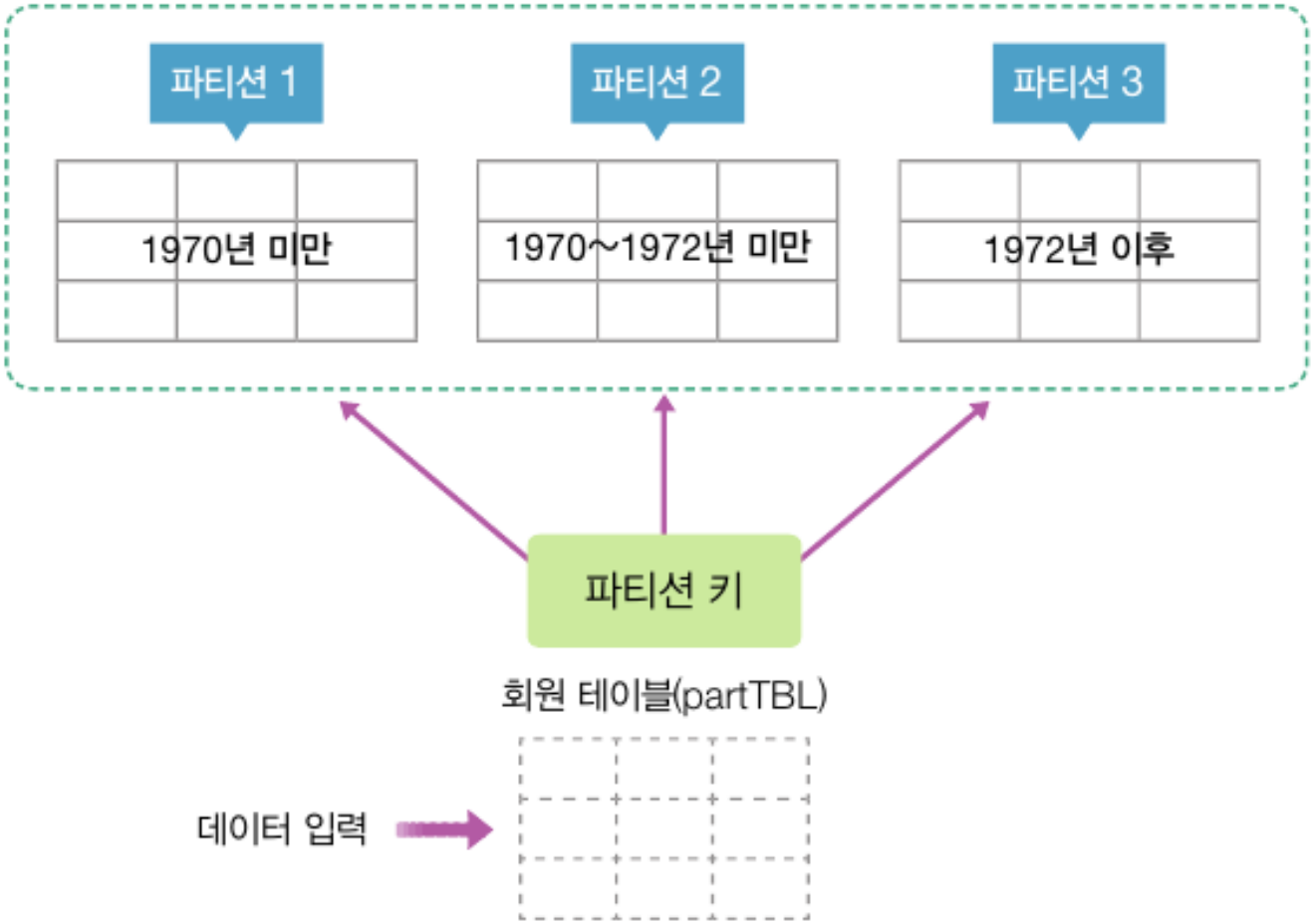


그림 12-13 파티션의 개념

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

1 cookDB 초기화하기

1-1 cookDB.sql 파일을 열어 실행

1-2 열린 쿼리 창을 모두 닫고 새 쿼리 창 염

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

2 파티션으로 분리되는 테이블 생성하기

2-1 파티션으로 분리되는 테이블 정의

표 12-1 회원 테이블의 파티션 구성

파티션 이름	part1	part2	part3
파티션 일련번호	1	2	3
값	값 < 1970	1970 ≤ 값 < 1972	1972 ≤ 값 < 최대

```
CREATE DATABASE IF NOT EXISTS partDB;
USE partDB;
DROP TABLE IF EXISTS partTBL;
CREATE TABLE partTBL
( userID CHAR(8) NOT NULL, -- 기본키로 설정하면 안 됨
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  addr CHAR(2) NOT NULL
)
PARTITION BY RANGE(birthYear)
(
  PARTITION part1 VALUES LESS THAN (1970),
  PARTITION part2 VALUES LESS THAN (1972),
  PARTITION part3 VALUES LESS THAN MAXVALUE
);
```

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

2-2 회원 테이블(partTBL)에 데이터 삽입

```
INSERT INTO partTBL
    SELECT userID, userName, birthYear, addr FROM cookDB.userTBL;
```

2-3 삽입한 데이터 확인

```
SELECT * FROM partTBL;
```

userID	userName	birthYear	addr	
KKJ	김국진	1965	서울	파티션 1(1970년 미만)
KYM	김용만	1967	서울	
LKK	이경규	1960	경남	
KHD	강호동	1970	경북	파티션 2(1970~1972년 미만)
NHS	남희석	1971	충남	
PSH	박수홍	1970	서울	
SDY	신동엽	1971	경기	
KJD	김제동	1974	경남	파티션 3(1972년 이후)
LHJ	이휘재	1972	경기	
YJS	유재석	1972	서울	

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

3 파티션 확인하기

3-1 파티션의 정보 확인

```
SELECT TABLE_SCHEMA, TABLE_NAME, PARTITION_NAME, PARTITION_ORDINAL_POSITION, TABLE_ROWS
    FROM INFORMATION_SCHEMA.PARTITIONS
    WHERE TABLE_NAME = 'parttbl';
```

TABLE_SCHEMA	TABLE_NAME	PARTITION_NAME	PARTITION_ORDINAL_POSITION	TABLE_ROWS
partdb	parttbl	part1	1	3
partdb	parttbl	part2	2	4
partdb	parttbl	part3	3	3

3-2 1970년 전에 출생한 회원 조회

```
SELECT * FROM partTBL WHERE birthYear < 1970;
```

3-3 어떤 파티션을 사용했는지 확인

```
EXPLAIN
    SELECT * FROM partTBL WHERE birthYear < 1970;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	partTBL	part1	ALL	NULL	NULL	NULL	NULL	3	33.33	Using where

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

3-4 작다(<)를 작거나 같다(<=)로 수정해서 실행

```
EXPLAIN
    SELECT * FROM partTBL WHERE birthYear <= 1970;
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	partTBL	part1,part2	ALL	NULL	NULL	NULL	NULL	7	33.33	Using where

4 파티션 관리하기

4-1 파티션 3을 1972~1974년 미만(파티션 3)과 1974년 이후(파티션 4)로 분리

```
ALTER TABLE partTBL
    REORGANIZE PARTITION part3 INTO (
        PARTITION part3 VALUES LESS THAN (1974),
        PARTITION part4 VALUES LESS THAN MAXVALUE
    );
OPTIMIZE TABLE partTBL;
```

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

4-2 INFORMATION_SCHEMA 데이터베이스의 PARTITIONS 테이블을 다시 조회

```
SELECT TABLE_SCHEMA, TABLE_NAME, PARTITION_NAME, PARTITION_ORDINAL_POSITION, TABLE_ROWS
    FROM INFORMATION_SCHEMA.PARTITIONS
    WHERE TABLE_NAME = 'parttbl';
```

TABLE_SCHEMA	TABLE_NAME	PARTITION_NAME	PARTITION_ORDINAL_POSITION	TABLE_ROWS
partdb	parttbl	part1	1	3
partdb	parttbl	part2	2	4
partdb	parttbl	part3	3	2
partdb	parttbl	part4	4	1

4-3 파티션 합치기

```
ALTER TABLE partTBL
      REORGANIZE PARTITION part1, part2 INTO (
            PARTITION part12 VALUES LESS THAN (1972)
);
OPTIMIZE TABLE partTBL;
```

[실습 12-2] 파티션 구현하기

교재 437~442p 참고

4-4 INFORMATION_SCHEMA 데이터베이스의 PARTITIONS 테이블 다시 조회

```
SELECT TABLE_SCHEMA, TABLE_NAME, PARTITION_NAME, PARTITION_ORDINAL_POSITION, TABLE_ROWS
    FROM INFORMATION_SCHEMA.PARTITIONS
    WHERE TABLE_NAME = 'parttbl';
```

TABLE_SCHEMA	TABLE_NAME	PARTITION_NAME	PARTITION_ORDINAL_POSITION	TABLE_ROWS
partdb	parttbl	part12	1	7
partdb	parttbl	part3	2	2
partdb	parttbl	part4	3	1

4-5 파티션 12 삭제

```
ALTER TABLE partTBL DROP PARTITION part12;
OPTIMIZE TABLE partTBL;
```

4-6 회원 테이블 조회

```
SELECT * FROM partTBL;
```

userID	userName	birthYear	addr
LHJ	이휘재	1972	경기
YJS	유재석	1972	서울
KJD	김제동	1974	경남

2-3 파티션의 제한 사항

- 파티션에서 부가적으로 기억해야 할 사항
  - 파티션 테이블에 외래키를 설정할 수 없음. 파티션은 단독으로 사용되는 테이블에만 설정할 수 있음
  - 스토어드 프로시저, 스토어드 함수, 사용자 변수 등을 파티션 함수나 식에 사용할 수 없음
  - 임시 테이블에는 파티션 기능을 사용할 수 없음
  - 파티션 키에는 일부 함수만 사용 가능
  - 파티션의 개수는 최대 8192개까지 지원됨
  - 레인지 파티션에는 숫자형의 연속된 범위를 지정하여 사용하고, 리스트 파티션에는 숫자형 또는 문자형의 연속되지 않은 파티션 키 값을 하나하나 지정하여 사용함
  - 리스트 파티션에는 MAXVALUE를 사용할 수 없음. 모든 경우의 파티션 키 값을 지정해야 함