학습목표

1. MariaDB 파티셔닝

학습내용

- 데이터베이스 파티셔닝에 대해 알 수 있습니다.
- MariaDB기반의 파티셔닝 실습

사전퀴즈

1. 파티션(Partition)을 사용하면 데이터베이스가 관리할 수 있는 데이터가 증가한다.
정답 : X
해설 : 파티셔닝(Partitioning)은 하나의 시스템 안에서 관리하는 방식을 다루는 관계로 데이터 관리용량과는 관계가 없다. 다만 데이터 검색과 업데이트 성능을 향상시킬 수 있다.

2. 파티션에서 외래키를 사용할 수 없다.
정답 : X
해설 : 파티션에서는 성능문제로 외래키를 사용할 수 없다.

수업

1. 파티셔닝

데이터베이스 파티셔닝(Partitioning)
- 하나의 테이블을 여러개의 파티션으로 분할 저장하고 관리하는 방법
- 외부에서는 내부에서 파티셔닝을 했는지 알 수 없음
- DBMS 자체에서 제공

장점
- 데이터 전체 검색시 필요한 부분만 탐색해 성능 증가
- 전체 데이터를 손실할 가능성이 줄어듦 → 가용성(Availability) 향상
- 파티션별 백업/복구 가능
- 파티션 단위로 I/O 분산 가능 → 업데이트 성능 증가

제약사항(Constraints)
- 테이블단위 연산이 힘들어짐(비용문제)
  조인연산 어려움(정규화 문제) → 역정규화(중복허용)로 해결
- 외래키(FK)의 효용문제
  레코드 추가시 참조무결성 조건 체크 → 시스템 부담증가로 지원하지 않음
  예) 참조무결성 - city 테이블에 레코드를 추가할 때 country 테이블에 해당 국가코드가 있는지 먼저 조회해본다

MariaDB 파티셔닝
- 최대 8192개
- 모든 파티션은 동일한 스토리지 엔진
- 파티션은 외래키(FK) / Full Text 인덱스 지원하지 않음

- 파티션 값은 정수

파티셔닝 방식

종류	설명	예시
범위(range)	- a-m / n-r / s-z - 일반적인 방식 - 기준값은 연속적 - 파티션 크기가 일정하지 않을 수 있음	직원 테이블을 입사년도 기준으로 파티션 분할 (입사연도는 2016, 2017, 2018... 연속적)
해시(hash)	- 해시함수 파티션별로 크기를 비슷하게 나눔 - 처음부터 새로 구성할 때는 문제가 되지 않으나 파티션을 분할하거나 파티션을 삭제할 때 대량의 데이터를 이동해야 하는 문제가 있을 수 있음	
리스트(list)	- 특정한 컬럼을 기준 - 기준값은 비연속적 - 파티션 크기가 일정하지 않을 수 있음	계절 - 봄/여름/가을/겨울 취미 - 영화/스키/수영
컴포지트(composite)	range-hash / range-list	

예제 1-1

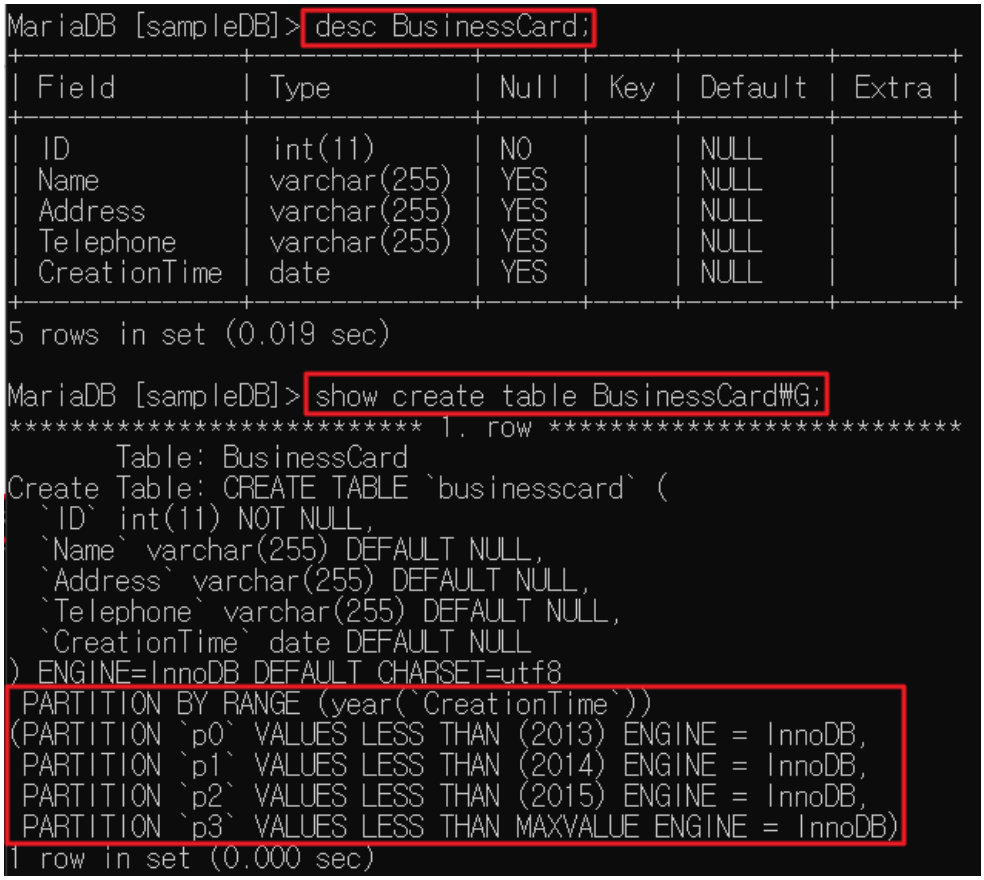
BusinessCard 테이블에 생성시간(CreationTime) 컬럼을 추가하고 파티셔닝하시오

1) BusinessCard 테이블 생성, 파티셔닝

```
CREATE TABLE BusinessCard(
   ID int NOT NULL,
   Name varchar(255),
   Address varchar(255),
   Telephone varchar(255),
   CreationTime Date
) PARTITION BY RANGE(YEAR(CreationTime))(
   PARTITION p0 VALUES LESS THAN (2013),
   PARTITION p1 VALUES LESS THAN (2014),
   PARTITION p2 VALUES LESS THAN (2015),
   PARTITION p3 VALUES LESS THAN MAXVALUE
)
;
```

- CreationTime 컬럼에서 YEAR 부분만 사용하여 4개의 파티션으로 분할
- 최근은 보통 MAXVALUE를 사용
- CreationTime 컬럼값이 null이면 제일 작은 값으로 판단하여 p0 파티션으로 이동
- CreationTime 컬럼값이 now이면 제일 큰 값으로 판단하여 p3 파티션으로 이동
- CreatonTime 컬럼값이 수정될 경우 파티션 사이에 이동이 있을 수 있음(외부에서는 변화 없음)
  예) 입사년도가 2013년이었는데 2015년으로 수정될 경우 p0에서 p2로 데이터 이동

2) BusinessCard 테이블 생성, 파티셔닝 결과 확인



```
MariaDB [sampleDB]> desc BusinessCard;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| ID           | int(11)      | NO   |     | NULL    |       |
| Name         | varchar(255) | YES  |     | NULL    |       |
| Address      | varchar(255) | YES  |     | NULL    |       |
| Telephone    | varchar(255) | YES  |     | NULL    |       |
| CreationTime | date         | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.019 sec)

MariaDB [sampleDB]> show create table BusinessCard\G;
*************************** 1. row ***************************
       Table: BusinessCard
Create Table: CREATE TABLE `businesscard` (
  `ID` int(11) NOT NULL,
  `Name` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Telephone` varchar(255) DEFAULT NULL,
  `CreationTime` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
 PARTITION BY RANGE (year(`CreationTime`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN MAXVALUE ENGINE = InnoDB)
1 row in set (0.000 sec)
```

desc BusinessCard;

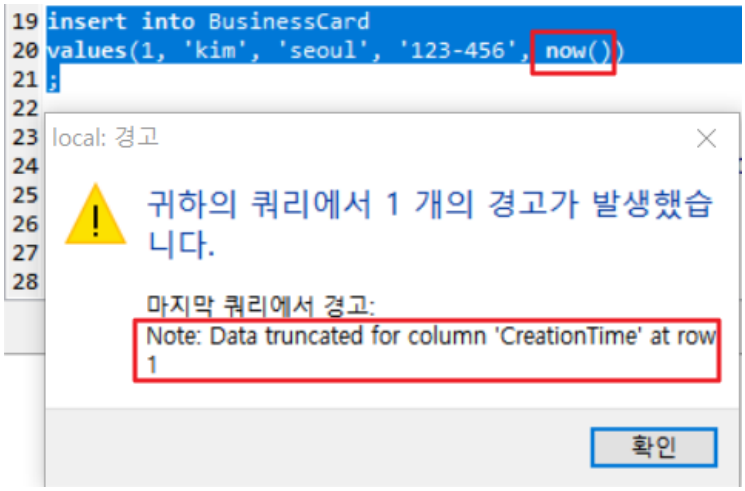
명령어로는 테이블이 파티셔닝 되었는지 알 수 없음

show create table BusinessCard;

명령어로 테이블 파티셔닝 확인

3) BusinessCard 테이블에 데이터 insert

```
19 insert into BusinessCard
20 values(1, 'kim', 'seoul', '123-456', now())
21 ;
22
23 insert into BusinessCard
24 values(2, 'lee', 'jeju', '231-456', '2000-01-01')
25 ;
26
27 insert into BusinessCard
28 values(3, 'park', 'busan', '321-456', '2014-01-01')
29 ;
30
31 insert into BusinessCard
32 values(4, 'jung', 'daegu', '3201-456', null)
33 ;
```



BusinessCard 테이블에 새로운 레코드를 insert하는데 위와 같은 경고 메시지 발생

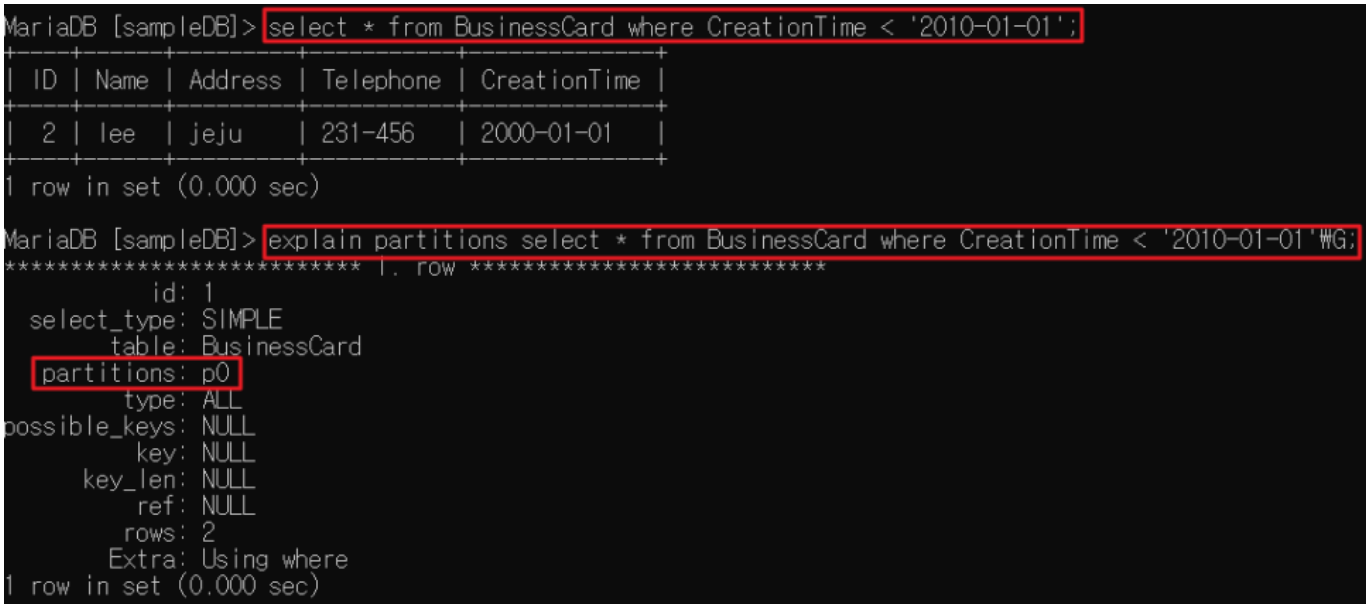
이유는?
CreationTime 컬럼은 DATE 타입인데 DATETIME 타입인 now()값을 insert하려고 해서 시/분/초 부분이 잘리고 년/월/일 부분만 입력이 된 것이다. 참고로 DATE 타입은 'YYYY-MM-DD'형식으로 '1000-01-01' 부터 '9999-12-31'까지만 입력이 된다. 반면에 DATETIME 타입은 'YYYY-MM-DD HH:MM:SS' 형식으로 '1000-01-01 00:00:00'부터 '9999-12-31 23:59:59'까지 데이터 입력이 가능하다.

경고메시지 없이 데이터를 insert하려면
- CreationTime 컬럼의 타입을 DATETIME으로 변경하고 now()값 입력
- CreationTime 컬럼의 타입은 DATE로 하고 now() 값을 DATE 타입으로 변경해서 입력
  DATE_FORMAT(NOW(),'%Y-%m-%d')
- CreationTime 컬럼의 타입은 DATE로 하고 값을 DATE 형식에 맞게 입력
  예) '2000-01-01'

4) 쿼리 실행시 어떤 파티션을 사용했는지 확인

```
MariaDB [sampleDB]> explain partitions select * from BusinessCard₩G;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: BusinessCard
   partitions: p0,p1,p2,p3
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 5
        Extra:
1 row in set (0.000 sec)
```

→ BusinessCard 테이블의 전체 레코드를 조회했기 때문에 모든 파티션(p0, p1, p2, p3)을 사용



→ CreationTime 값이 2010-01-01보다 작은 값만을 조회했기 때문에 파티션 p0만 사용

*** 참고 1 ***

파티션관련 최적화(검색)

예) 입사년도 별로 나눠진 직원 테이블

검색할 때 전체 파티션을 검색하는 것이 아니라 조건에 맞는 파티션만 검색

- 2005년에 입사한 직원들만 검색하는 경우
- 2005년 ~ 2007년(여러 개의 파티션) 사이에 입사한 직원들을 검색하는 경우

파티션관련 최적화(업데이트)

예) 입사년도 별로 나눠진 직원 테이블

- 2006년에 입사한 사원의 입사일자가 2005년으로 변경된 경우
  2006년 파티션에서 2005년 파티션으로 이동

*** 참고 2 ***

구분	문법	예시
쿼리 실행시 어떤 인덱스를 사용했는지 확인	explain 쿼리문	explain select * from BusinessCard;
쿼리 실행시 어떤 파티션을 사용했는지 확인	explain partitions 쿼리문	explain partitions select * from BusinessCard;

예제 1-2

파티션 삭제/추가

1) 파티션 삭제

```
MariaDB [sampleDB]> ALTER TABLE BusinessCard DROP PARTITION p3;
Query OK, 0 rows affected (0.026 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [sampleDB]> show create table BusinessCard₩G;
*************************** 1. row ***************************
       Table: BusinessCard
Create Table: CREATE TABLE `businesscard` (
  `ID` int(11) NOT NULL,
  `Name` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Telephone` varchar(255) DEFAULT NULL,
  `CreationTime` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
 PARTITION BY RANGE (year(`CreationTime`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB)
1 row in set (0.016 sec)
```

2) 파티션 추가

```
46 ALTER TABLE BusinessCard ADD PARTITION(
47     PARTITION p3 VALUES LESS THAN (2016),
48     PARTITION p4 VALUES LESS THAN MAXVALUE
49 )
50 ;
```

```
MariaDB [sampleDB]> show create table BusinessCard₩G;
*************************** 1. row ***************************
       Table: BusinessCard
Create Table: CREATE TABLE `businesscard` (
  `ID` int(11) NOT NULL,
  `Name` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Telephone` varchar(255) DEFAULT NULL,
  `CreationTime` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
 PARTITION BY RANGE (year(`CreationTime`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN (2016) ENGINE = InnoDB,
 PARTITION `p4` VALUES LESS THAN MAXVALUE ENGINE = InnoDB)
1 row in set (0.023 sec)
```

예제 1-3

파티션 분할/병합

1) 파티션 분할

```
52 ALTER TABLE BusinessCard REORGANIZE PARTITION p4 INTO(
53     PARTITION p5 VALUES LESS THAN (2017),
54     PARTITION p6 VALUES LESS THAN MAXVALUE
55 )
56 ;
```

```
MariaDB [sampleDB]> show create table BusinessCard₩G;
*************************** 1. row ***************************
       Table: BusinessCard
Create Table: CREATE TABLE `businesscard` (
  `ID` int(11) NOT NULL,
  `Name` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Telephone` varchar(255) DEFAULT NULL,
  `CreationTime` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
 PARTITION BY RANGE (year(`CreationTime`))
(PARTITION `p0` VALUES LESS THAN (2013) ENGINE = InnoDB,
 PARTITION `p1` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN (2016) ENGINE = InnoDB,
 PARTITION `p5` VALUES LESS THAN (2017) ENGINE = InnoDB,
 PARTITION `p6` VALUES LESS THAN MAXVALUE ENGINE = InnoDB)
1 row in set (0.027 sec)
```

2) 파티션 병합

```
58 ALTER TABLE BusinessCard REORGANIZE PARTITION p0, p1 INTO(
59    PARTITION p01 VALUES LESS THAN (2014)
60 )
61 ;
```

```
MariaDB [sampleDB]> show create table BusinessCard₩G;
*************************** 1. row ***************************
       Table: BusinessCard
Create Table: CREATE TABLE `businesscard` (
  `ID` int(11) NOT NULL,
  `Name` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Telephone` varchar(255) DEFAULT NULL,
  `CreationTime` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
 PARTITION BY RANGE (year(`CreationTime`))
(PARTITION `p01` VALUES LESS THAN (2014) ENGINE = InnoDB,
 PARTITION `p2` VALUES LESS THAN (2015) ENGINE = InnoDB,
 PARTITION `p3` VALUES LESS THAN (2016) ENGINE = InnoDB,
 PARTITION `p5` VALUES LESS THAN (2017) ENGINE = InnoDB,
 PARTITION `p6` VALUES LESS THAN MAXVALUE ENGINE = InnoDB)
1 row in set (0.023 sec)
```

파티셔닝이란?

전문가의견

데이터의 용량이 증가해 하나의 시스템을 넘어가는 정도로 커지면 시스템을 나눠서 용량 및 성능에 대응하는 경우가 많습니다.

파티셔닝은 시스템(DBMS 시스템 자체)에서 관리대상을 내부적으로 나누는 것입니다. 즉, 사용자가 보기에는 파티션이 보이지 않습니다.

DMBS 시스템이 파티셔닝에 대해 자체적인 관리를 하며, 읽기나 쓰기를 할 때 특정 파티션만 검색/업데이트 하기 때문에 검색이나 업데이트 성능을 향상시킬 수 있습니다.