대용량 IoT 데이터의 빠른 분석을 위한 OLAP 기반의 빅테이블 생성 방안

이도훈 · 조찬영 · 온병원*

군산대학교

OLAP-based Big Table Generation for Efficient Analysis of Large-sized IoT Data

Dohoon Lee · Chanyoung Jo · Byung-Won On*

Kunsan National University

E-mail : dhlee7@kunsan.ac.kr / whcksdudrns@naver.com / bwon@kunsan.ac.kr

요 약

최근 사물인터넷(IoT) 기술이 발전하면서 다양한 단말들이 인터넷에 연결되고 있다. 그로 인해 발생하는 IoT 데이터의 양 또한 증가하고 있는데, 이렇게 발생한 대용량 IoT 데이터를 빠르게 분석할 수 있는 인덱스 키를 제안한다. 기존 인덱스 키에는 시간과 공간의 정보만 존재하여 반복문이나, 조인 연산(Join operation)을 사용하여 인덱스 테이블과 인스턴스 테이블에 저장되어있는 데이터를 질의했다면, 제안방안의 인덱스 키에는 IoT 데이터를 임베딩(Embedding) 하여 시간이 지연되었던 반복문이나 조인횟수를 최소화하기 위하여 OLAP 기반의 빅테이블을 생성함으로써 시간을 단축하였다.

ABSTRACT

With the recent development of the Internet of Things (IoT) technology, various terminals are being connected to the Internet. As a result, the amount of IoT data is also increasing, and an index key that can efficient analyze the large-scale IoT data is proposed. Existing index keys have only time and space information, so if data stored in index tables and instance tables were queried using repetition or join operation, IoT data was embedded in the index key of the proposal to create OLAP-based big tables to minimize the number of repetitions or join times.

키워드

large IoT data, efficient query processing, Embedding, minimize time-delay

Ⅰ. 서 론

최근 4차 산업혁명 시대에 도래하면서 빅데이터(Big data), 사물인터넷(Internet of Things; IoT), 인공지능(Artificial Intelligence; AI) 등의 기술이 빠른 속도로 발전하고 있다. 그로 인해 다양한 단말들을 인터넷에 연결하려는 시도가 증가하면서 IoT 데이터의 양 또한 증가하고 있다. 하지만 대용량의 IoT 데이터를 분석하는 연구는 미비한 실정이다.

기존 연구로는 각 특성이 있는 테이블끼리 분리하여 관리하는 전통적인 데이터베이스 시스템 구조를 이용한 한진주의 제안방안[1]과 이도훈의 제안방안[2]이다. [1]의 방안은 시경계, 도로, 사용자 맞춤형(선, 원, 사각형, 다각형) 등 다양한 지역과 사용자가 원하는 날짜를 웹을 통해 입력받아 통합 시공간 인덱스 모델을 사용하여 분석한 결과를 웹을 통해 사용자에게 시각적으로 결과를 제공한다. [2]의 방안은 [1]의 방안의 개선 연구로 두 테이블을 사용할 때 조인의 횟수를 최소화하며 해시 함수(Hash function)와 블로킹(Blocking)기법을 사용하면서 선형적인 시간복잡도를 갖는 알고리즘을 제안하였다. 각 테이블은 시간과 공간의 정보를 저장

* corresponding author

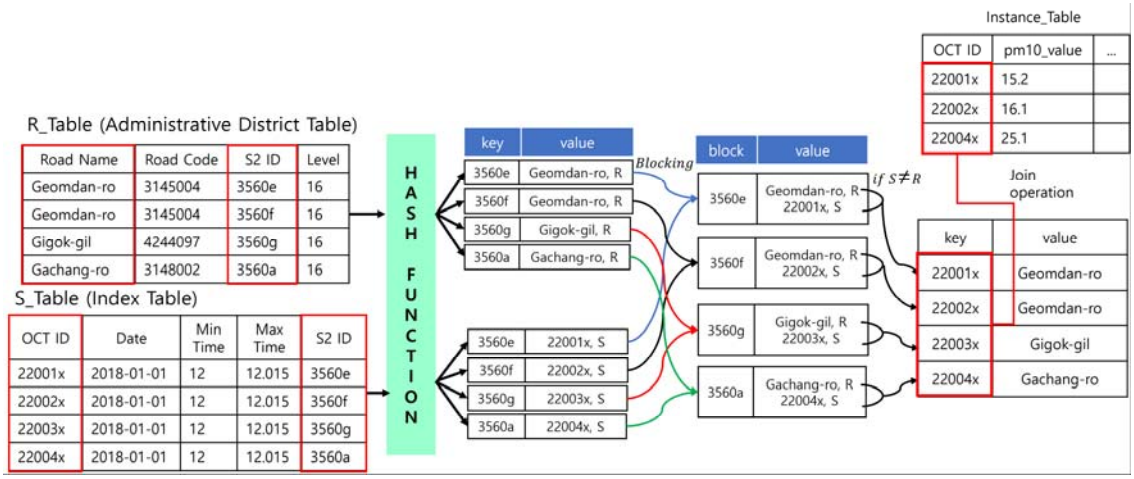

그림 1. 사용자 질의처리를 위한 해시 함수와 블로킹을 사용한 기존 방안 흐름도[2]

하고 있는 OCT ID를 외래키(Foreign key)로 사용한다. [2]의 제안 방안은 각 데이터에 특징에 맞게 테이블을 분리해놓아서 유지, 보수가 용이하다는 장점이 있지만, 사용자가 질의 시 조인 연산(Join operation)을 필요로 하여 시간이 지연된다는 단점이 있다.

본 논문에서는 기존 방안인 [1]과 [2]의 방안에서 병목현상이 발생하는 조인 연산을 최소화하는 OLAP(Online Analytical Processing) 방식의 알고리즘을 제안한다. [2]의 방안의 경우 해시 함수와 블로킹 과정보다 조인 연산 중 평균을 집계하는 과정이 응답시간을 많이 지연시키는 것을 확인하였다. 따라서 본 논문에서는 기존 시간과 공간의 정보만 담겨있던 OCT ID에 IoT 데이터를 추가하여 조인 연산을 줄이며, 기존에 여러 테이블로 분리되어있던 테이블을 하나로 합치며 응답시간을 단축하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 제안 방안의 이해를 돕기 위해 관련 연구에 대하여 설명한다. 제 3장에서는 제안 방안에 대하여 자세히 설명한다. 제 4장에서는 제안 방안에 대한 실험환경과 결과에 대하여 설명하고, 5장에서는 결론을 다룬다.

II. 관련 연구

본 연구의 기존 연구인 [1]과 [2]의 제안 방안은 사용자가 특정 도로, 특정 동, 특정 도로 히트맵, 전체 도로, 전체 동, 전체 도로 히트맵, 사용자 지정(선, 원, 사각형, 다각형) 등의 다양한 지역에 대한 IoT 데이터의 분석결과를 빠른 속도로 분석하여 사용자에게 웹을 통해 시각적으로 제공한다. 특히 전체 도로, 전체 동의 도로, 동 등의 행정구역에 대한 Google S2 cell[3]에 관한 정보가 저장되어 있는 R_Table과 OCT ID와 Google S2 cell이 저장되어있는 S_Table을 사용한다. 그림 1은 사용자 질의처리를 위한 해시 함수와 블로킹을 사용한 기존 방안의 알고리즘이다. 먼저 R_Table의 경우 도로명과 S2 ID가 해시 함수를 통과하게 되면 S2 ID를 키로 도로명과 테이블 명을 값으로 가지는 키, 값 쌍을 반환하고 S_Table의 경우 OCT ID와 S2 ID가 해시 함수를 통과하면 S2 ID를 키로 OCT ID와 테이블 명을 값으로 가지는 키, 값 쌍을 반환한다. 이후 각 키 별로 그룹화 해주는 블로킹 과정을 거치게 되고 값에 저장된 각 블록별 테이블 명을 확인하여 출처가 서로 다른 테이블이면 각 새로운 테이블에 한 행이 된다. 다음으로 집계 과정을 거치게 되는데 IoT 데이터가 저장된 인스턴스 테이블(Instance table)과 조인 연산을 통하여 각 행정구역 별 IoT 데이터의 평균 값을 구할 수 있다.

III. 제안 방안

[1]과 [2]의 연구는 각 특성에 알맞게 테이블을 분리하여 유지, 보수에 용이하지만, 사용자 질의처리시 시간이 지연된다는 단점이 있다. 그래서 본 논문에서는 해시, 블로킹, 조인이 필요하지 않게 한 테이블에 데이터를 병합하여 최대한 연산을 줄이고 기존에 날짜와 공간 정보가 저장된 OCT ID에 IoT 데이터를 추가하여 조인 연산을 필요로 하지 않는 테이블을 생성하였다. 테이블을 생성과정은 그림 2와 같은데 먼저 (1) 크로스 조인(Cross join)을 사용하여 데이터 손실 없이 조인 연산을 하며 행정구역 명(도로명, 동)컬럼을 추가하였다. 그리고 (2) 각 OCT ID별 IoT 데이터의 평균을 구하여 (3) 기존 OCT ID 뒤에 구분자 x에 이어 IoT 데이터의 평균 값을 임베딩 하였다.

제안 방안의 알고리즘은 Algorithm1과 같다.

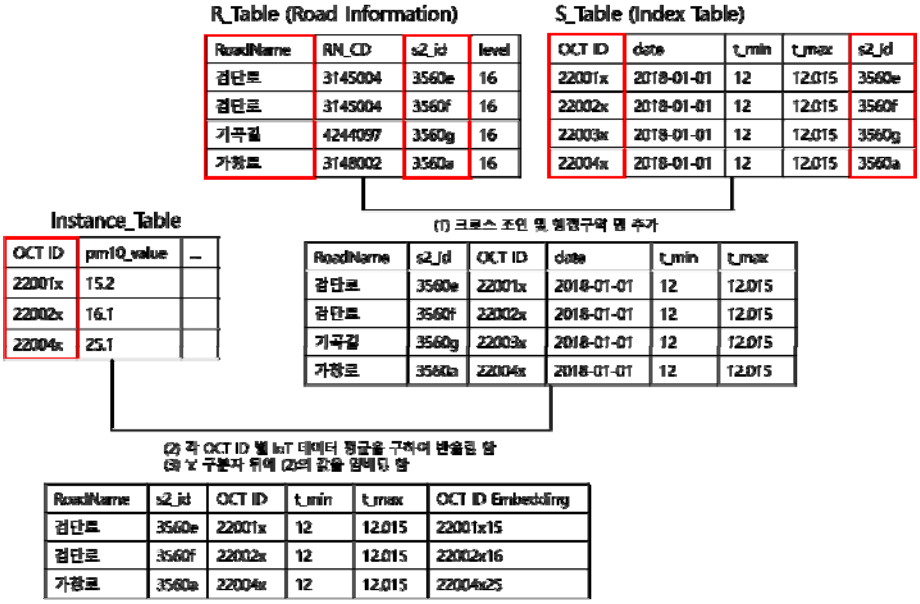

그림 2. 제안 방안의 테이블 생성 과정

Algorithm1. 제안 방안의 테이블 생성

```
1. CREATE VIEW JOIN(oct_district)
   SELECT OcT_info.*, IoT_index.*
   FROM IoT_index
   CROSS JOIN OcT_info
   ON OcT_info.oct_id = IoT_index.oct_id

2. CREATE VIEW JOIN(oct_pm10)
   SELECT ..., ROUND(AVG(IoT_value))
   FROM JOIN(oct_district)
   GROUP BY oct_id

3. CREATE TABLE OcT_embedding_IoT AS
   SELECT *, CONCAT(oct_id, `x`, IoT_value)
   FROM JOIN(oct_pm10)
```

Line1은 그림 2의 (1)의 과정으로 두 테이블을 조인 연산 한다. Line2는 그림 2의 (2)의 과정으로 oct_id 별로 평균을 소수점 첫째 자리에서 반올림하여 모두 정수형으로 통일한다. Line3은 (3)의 과정으로 새로운 OCT ID와 IoT 데이터의 값을 'x'구분자를 사이에 두고 결합하는 과정이다.

Ⅳ. 실험 환경 및 결과

실험에 사용된 데이터는 택시 34대가 2011년 1월 1일부터 2018년 8월 29일까지 대구광역시를 이동하며 수집한 IoT 데이터를 사용했다. 각 택시의 센서는 시간, 경도, 위도, 미세먼지, 초미세먼지, 온도, 습도, 이산화황 등 15개의 컬럼을 1초마다 수집하였다. 본 논문에서는 시간, 위도, 경도, 미세먼지, 초미세먼지 컬럼을 사용하였고, 인덱싱 기간은 2018년 1월 1일 ~ 2018년 8월 29일까지 약 8개월분의 데이터를 사용하였다. 도로 데이터는 국가공간정보포털에서 행정안전부의 도로 구간 데이터 사이트에서 제공하는 shape파일 형식으로 도로 구간 데이터 셋을 사용하였다. 하지만 제공된 데이터는 국내 좌표계로 이루어져 있어 위경도 좌표계 기반으로 이루어져 있는 Google S2 ID와 통일하기 위해 위경도 좌표계로 변환하는 작업을 거쳤다. 시경계 데이터의 경우 행정자치부, 통계청에서 제공하는 대한민국 행정구역 읍면동 데이터를 다운로드 받았다. 이는 전국에 있는 모든 도시의 시경계(시, 구, 동)에 대한 공간 정보를 위도와 경도로 표시한 데이터이다.

표 1. 실험환경

CPU	Intel(R)Core(TM)i7-7700 CPU@3.60GHz
RAM	16GB
SSD	512GB
OS	Ubuntu 18.04 LTS

본 논문은 실험은 Python 3.6.5를 사용하여 입력받은 사용자의 질의에 해당하는 지역의 IoT 데이

터를 분석하여 그 결과를 빠르게 사용자에게 시각적으로 제공하였고, 데이터는 MySQL 5.7을 사용하여 저장해놓았다. 실험은 [1]의 제안 방안, [2]의 제안 방안과 본 논문의 제안 방안을 비교 실험 하였다. 실험 결과는 표 2와 같다. 표의 질의종류는 다음과 같다. q1은 특정 동, q2는 특정도로, q3은 특정도로의 히트맵, q4는 도시 전체의 도로 검색, q5는 도시 전체의 도로 히트맵, q6은 도시 전체의 동이고, q7~q10은 사용자가 마우스로 지도에 (선, 원, 사각형, 다각형) 등을 그려서 원하는 지역을 탐색하는 질의방법이다. 실험 방법은 24시간, 1주일, 1개월, 8개월의 결과를 비교하였다. 이 중에 24시간, 1주일, 1개월은 무작위로 10회 질의하여 얻은 응답시간의 평균이다. 8개월은 10회 질의 시 동일한 질의를 10회 하는 것으로 1회만 질의하였다.

V. 결 론

기존 방안인 [1]과 [2]는 유지보수에 초점을 맞춘방면 본 논문의 제안 방안은 응답시간에 초점을 맞췄다. 향후 연구로는 Hadoop, Apache Spark와 같은 빅데이터 분석 플랫폼을 사용하여 의미있는 분석 결과와 유지보수와 응답속도를 모두 고려한 알고리즘을 구상할 예정이다.

Acknowledgement

이 논문은 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2019R1F1A106075213).

References

[1] Jinju Han, A Unified Spatio-temporal Index Model for Different Types of Things, Kunsan National University Graduate School, August, 2020

[2] Dohoon Lee, Byung-Won On, Hash and Blocking-based Query Processor for Efficient Analysis of Large-Sized IoT Data, JKIIT, Vol. 19, No. 8, p. 41-50, August, 2021

[3] S2Geometry[Internet] Available: https://s2geometry.io

표 2. 사용자 질의시간 범위에 따른 응답시간 비교 실험 결과 (단위: 초)

		1day				1week				1month				8month
		min	max	avg	σ	min	max	avg	σ	min	max	avg	σ	
q1	[1], [2]	0.68	0.74	0.72	0.02	4.32	5.01	4.71	0.26	20.10	30.25	27.17	3.24	219.38
	제안방안	1.55	1.60	1.57	0.01	1.55	1.61	1.57	0.02	1.56	1.61	1.58	0.02	1.60
q2	[1], [2]	0.71	0.75	0.73	0.02	4.33	4.99	4.71	0.25	20.94	30.92	26.68	3.27	223.16
	제안방안	1.56	1.60	1.58	0.01	1.56	1.62	1.58	0.02	1.56	1.61	1.58	0.02	1.59
q3	[1], [2]	1.76	1.87	1.82	0.02	4.43	4.99	4.71	0.20	21.12	31.19	27.36	3.12	215.74
	제안방안	1.56	1.61	1.59	0.02	1.56	1.62	1.59	0.02	1.54	1.61	1.57	0.02	1.60
q4	[1]	34.71	47.71	44.54	3.53	108.05	118.76	111.63	3.23	142.09	177.71	156.07	13.48	862.96
	[2]	25.39	30.51	27.32	1.60	26.15	30.30	28.54	1.37	31.57	37.13	35.09	1.98	448.52
	제안방안	1.55	1.61	1.57	0.02	1.56	1.61	1.58	0.02	1.58	1.65	1.61	0.02	3.10
q5	[1]	8.67	11.88	10.24	1.07	41.00	48.51	45.89	2.43	331.84	352.51	341.43	9.06	2881.48
	[2]	3.72	5.62	5.02	0.54	10.04	15.51	13.49	1.78	27.22	38.12	29.25	3.23	996.7
	제안방안	1.56	1.60	1.58	0.01	1.56	1.62	1.58	0.02	1.57	1.64	1.60	0.02	3.14
q6	[1]	8.88	13.25	11.02	1.33	41.33	47.37	44.29	2.47	332.81	360.05	342.83	9.78	2862.66
	[2]	3.87	5.62	5.12	0.51	10.83	15.57	13.78	1.64	27.19	40.20	29.46	3.87	981.32
	제안방안	1.54	1.61	1.57	0.02	1.56	1.62	1.59	0.02	1.59	1.67	1.63	0.02	9.42
q7	[1], [2]	0.84	1.09	0.97	1.35	4.50	5.26	4.82	0.27	25.44	30.01	27.51	1.84	216.00
	제안방안	1.57	1.62	1.61	0.02	1.57	1.65	1.62	0.03	1.56	1.61	1.58	0.03	1.60
q8	[1], [2]	0.68	0.98	0.86	1.35	3.29	5.12	4.25	0.48	25.48	34.14	29.42	2.68	230.01
	제안방안	1.57	1.64	1.61	0.02	1.57	1.64	1.62	0.03	1.57	1.61	1.58	0.03	1.61
q9	[1], [2]	0.78	1.11	1.01	0.38	3.48	5.22	4.28	0.46	26.45	35.00	29.71	2.60	214.46
	proposal	1.58	1.64	1.62	0.02	1.58	1.65	1.61	0.03	1.57	1.65	1.62	0.03	1.60
q10	[1], [2]	0.79	1.10	0.98	0.44	3.94	5.41	4.36	0.43	26.71	31.74	29.44	1.72	195.33
	proposal	1.58	1.64	1.62	0.02	1.58	1.65	1.61	0.03	1.57	1.65	1.62	0.03	1.61