6. 데이터 압축

↗ Books | Real MySQL 8.0

MySQL 서버에서 디스크에 저장된 데이터 파일의 크기는 일반적으로 쿼리 성능과 직결되기 때문에 데이터 압축 기능을 제공한다.

MySQL 서버에서 사용 가능한 압축 방식은 크게 테이블 압축과 페이지 압축의 두 가지 종류로 구분할 수 있다.

1. 페이지 압축

MySQL 서버가 디스크에 저장하는 시점에 페이지가 압축되어 저장되고, 반대로 디스크에서 데이터 페이지를 읽어올 때 압축이 해제된다.
즉, 버퍼 풀에 데이터 페이지가 한 번 적재되면 InnoDB 스토리지 엔진은 압축이 해제된 상태로만 데이터 페이지를 관리한다. (= 압축 여부와 관계 없이 투명(Tranparent)하게 작동한다)

여기서 한 가지 문제점은 압축한 후의 용량이 얼마나 될 지 모르는데 적어도 하나의 테이블은 동일한 크기의 페이지로 통일돼야 한다는 것이다. 그래서 운영체제별로 특정 버전의 파일 시스템에서만 지원되는 펀치 홀이라는 기능을 사용한다.

> 💡 이해한 바로는 16KB 페이지를 압축한 결과가 7KB라고 가정할 때 MySQL 서버는 압축 데이터 7KB + 펀치 홀 9KB로 관리를 한다.
> 생성된 펀치 홀의 공간은 다시 운영체제로 반납되긴 하지만 어쨌든 읽을 때 16KB를 읽어서 실제 데이터와 펀치 홀 공간을 구분하는 것 같다.

하지만 페이지 압축은 다음과 같은 이유로 많이 사용하지 않는다.

1. 펀치 홀 기능은 운영체제 뿐만 아니라 하드웨어 자체에서도 해당 기능을 지원해야 사용이 가능하다.

2. 아직 파일 시스템 관련 명령어가 펀치 홀을 지원하지 못한다는 것이다.
3. 백업과 복구를 포함한 여러 과정에서 파일 관련 유틸리티를 사용하는데 이 때 “cp” 같은 파일 복사 명령 또는 XtraBackup 같은 툴이 파일을 복사하면 펀치 홀이 다시 채워져서 압축을 했는데도 원본 크기만큼 차지할 수도 있다.

---

2. 테이블 압축

테이블 압축은 운영체제나 하드웨어에 대한 제약 없이 사용이 가능하고 디스크의 데이터 파일 크기를 줄일 수 있다는 장점이 있다.

하지만 다음과 같은 단점도 있다.

1. 버퍼 풀 공간 활용률이 낮다.
2. 쿼리 처리 성능이 낮다.
3. 빈번한 데이터 변경 시 압축률이 떨어진다.

1. 압축 테이블 생성

테이블 압축을 사용하기 위한 전제 조건으로 압축을 사용하려는 테이블이 별도의 테이블 스페이스를 사용해야 한다.

이를 위해서는

1. `innodb_file_per_table` 시스템 변수가 `ON`으로 설정된 상태에서 테이블이 생성돼야 한다.
2. 테이블을 생성할 때 `ROW_FORMAT=COMPRESSED` 옵션을 명시해야 한다.
3. (선택) `KEY_BLOCK_SIZE` 옵션을 이용해 압축된 페이지의 타깃 크기를 설정할 수 있는데, 2n으로만 (n 값은 2 이상) 설정할 수 있다.

3번을 보면 의문점이 들 수 있다. 압축한 용량이 얼마가 될지 알 수 없는데 어떻게 설정할 수 있을까?

InnoDB 스토리지 엔진이 압축을 적용하는 방식을 살펴보면 목표 크기(`KEY_BLOCK_SIZE`) 이하가 될 때까지 반복해서 페이지를 스플릿한다. 따라서 목표 크기 설정은 사용자의 선택이라고 볼 수 있을 것 같다.
하지만 잘못 설정되면 MySQL 서버의 처리 성능이 떨어질 수 있으니 조심해야 한다.

2. KEY_BLOCK_SIZE 결정

위에서 설명했듯이 압축된 결과가 어느 정도일지 예측해서 `KEY_BLOCK_SIZE`를 결정해야 한다. 테이블 압축을 적용하기 전에 4KB 또는 8KB로 테이블을 생성해서 샘플 데이터를 저장해보고 압축 실패율이 3~5% 미만으로 유지할 수 있게 선택하는 것이 좋다.

추가로 두 가지 알아두면 좋을 것이 있다.

1. 무조건 낮은 크기로 압축을 하는 것이 좋은 것도 아니다.
   책에 나온 예시를 보면 4KB와 8KB로 압축을 한 결과가 크게 다르지 않다. 즉, 오히려 압축 실패율은 낮으면서 효율은 상대적으로 높은 8KB를 선택하는 것이 훨씬 효율적일 것이다.
2. 압축 실패율이 높다고 해서 압축을 사용하지 말아야 한다는 것을 의미하지는 않는다.
   예를 들어, INSERT만 되는 로그 테이블의 경우 이후 변경되는 일이 없기 때문에 한 번 정도는 압축 시도가 실패해서 재압축한다고 하더라도 큰 손해는 아닐 것이다.
   하지만 반대로 압축 실패율은 그다지 높지 않지만 데이터가 빈번하게 조회되고 변경된다면 압축은 고려하지 않는 것이 좋다.

3. 압축된 페이지의 버퍼 풀 적재 및 사용

InnoDB 스토리지 엔진은 압축된 테이블의 데이터 페이지를 버퍼 풀에 적재하면 압축된 상태와 압축이 해제된 상태 2개의 버전을 관리한다. 그래서 디스크에서 읽은 상태 그대로의 데이터 페이지 목록을 관리하는 LRU 리스트와 압축된 페이지들의 압축 해제 버전인 Unzip_LRU 리스트를 별도로 관리하게 된다.

여기서 두 가지 문제점이 발생한다.
첫 번째는 압축된 테이블에 대해서는 버퍼 풀의 공간을 이중으로 사용하기 때문에 메모리를 낭비한다는 것이고 두 번째는 압축된 페이지에서 데이터의 조회나 변경 시에 압축을 해제해야 한다는 것이다.

이런 문제점을 보완하기 위해 Unzip_LRU 리스트를 별도로 관리하고 있다가 요청 패턴에 따라 적절한 수행을 한다.