



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년10월31일
(11) 등록번호 10-1670473
(24) 등록일자 2016년10월24일

(51) 국제특허분류(Int. Cl.)
G06F 11/14 (2006.01) G06F 17/30 (2006.01)
(52) CPC특허분류
G06F 11/1448 (2013.01)
G06F 11/1471 (2013.01)
(21) 출원번호 10-2015-0168811
(22) 출원일자 2015년11월30일
심사청구일자 2015년11월30일
(56) 선행기술조사문헌
KR101484882 B1
'MYSQL 데이터베이스의 삭제된 레코드 복구 기법', 디지털포렌식연구 제7권 제1호, 남궁재웅 외 4명, PP.1-12(2013.07)
KR1020140083556 A
'DB2 데이터베이스의 삭제된 레코드 복구 기법', 디지털포렌식연구 제8권 제2호, 이국현 외 3명, PP.1-14(2014.12)

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
대한민국(관리부서 대검찰청)
서울특별시 서초구 반포대로 157 (서초동)
(72) 발명자
이상진
서울특별시 종로구 사직로8길 20, 101동 1503호 (내수동, 파크팰리스)
박승범
경기도 안산시 상록구 정제로9길 5 (부곡동) (뒷면에 계속)
(74) 대리인
특허법인충현

전체 청구항 수 : 총 8 항

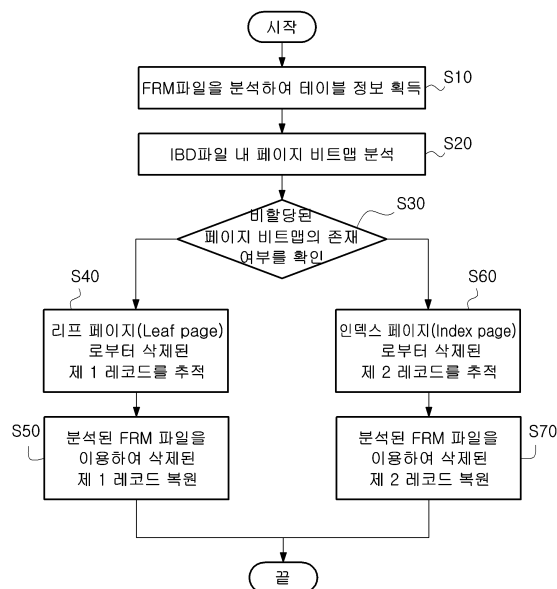
심사관 : 이동하

(54) 발명의 명칭 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법

(57) 요약

본 발명은 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 관한 것으로, MySQL InnoDB 데이터베이스의 FRM 파일을 분석하여 테이블 정보를 획득하는 단계와, 상기 MySQL InnoDB 데이터베이스의 IBD 파일 내 XDES 엔트리에서 페이지 비트맵을 획득하고, 상기 페이지 비트맵을 분석하여 비활당된 페이지 비트맵의 존재 여 (뒷면에 계속)

대표도 - 도12



부를 확인하는 단계와, 상기 비할당된 페이지 비트맵이 존재하는 경우 리프 페이지(Leaf page)로부터 삭제된 제 1 레코드를 추적하는 단계 및 상기 분석된 FRM 파일을 이용하여 상기 삭제된 제 1 레코드를 복원하는 단계를 포함하여 구성된다.

상기와 같이 구성된 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 의하면, 메타데이터 파일의 구조와 데이터베이스 파일의 페이지 정보를 이용함으로써, MySQL InnoDB 데이터베이스에서 삭제된 레코드 복구가 가능하며, 트랜잭션 로그를 일정기간만 보관하거나 없는 경우에도 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복구할 수 있다.

(52) CPC특허분류

G06F 17/3007 (2013.01)

G06F 17/30289 (2013.01)

(72) 발명자

정두원

서울특별시 노원구 삼발로 123, 304동 903호 (공릉동, 라이프아파트)

강철훈

서울특별시 서초구 효령로2길 30, 104호 (방배동, 방배엔스위트)

이경민

경기도 화성시 동탄지성로 17, A동 2301호 (반송동, 풍성위버폴리스)

이 발명을 지원한 국가연구개발사업

과제고유번호 2015046342

부처명 미래창조과학부

연구관리전문기관 한국연구재단

연구사업명 원천기술개발사업

연구과제명 대용량 저장장치 조사용 포렌식 시스템 개발

기 여 율 1/1

주관기관 고려대학교 산학협력단

연구기간 2015.08.01 ~ 2016.07.31

공지예외적용 : 있음

명세서

청구범위

청구항 1

MySQL InnoDB 데이터베이스의 FRM 파일을 분석하여 테이블 정보를 획득하는 단계;

상기 MySQL InnoDB 데이터베이스의 IBD 파일 내 XDES 엔트리에서 페이지 비트맵을 획득하고, 상기 페이지 비트맵을 분석하여 비할당된 페이지 비트맵의 존재 여부를 확인하는 단계;

상기 비할당된 페이지 비트맵이 존재하는 경우 리프 페이지(Leaf page)로부터 삭제된 제 1 레코드를 추적하는 단계; 및

상기 분석된 FRM 파일을 이용하여 상기 삭제된 제 1 레코드를 복원하는 단계;를 포함하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 2

제 1 항에 있어서,

상기 비할당된 페이지 비트맵의 존재 여부를 확인하는 단계는,

상기 비할당된 페이지 비트맵이 존재하지 않는 경우 인덱스 페이지(Index page)로부터 삭제된 제 2 레코드를 추적하는 단계; 및

상기 분석된 FRM 파일을 이용하여 상기 삭제된 제 2 레코드를 복원하는 단계;를 더 포함하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 3

제 1 항에 있어서,

상기 삭제된 제 1 레코드를 추적하는 단계는,

상기 XDES 엔트리 영역을 해석하여 페이지 할당 여부를 확인하는 단계;

할당된 페이지들에만 접근하여 상기 할당된 페이지가 인덱스 페이지(Index page)인지 확인하는 단계;

상기 할당된 페이지가 상기 인덱스 페이지인 경우 페이지 레벨을 확인하는 단계; 및

상기 페이지 레벨이 0인 경우 삭제된 레코드의 시작 주소(First garbage record offset)와 삭제된 공간의 크기(Garbage space) 영역을 확인하는 단계;를 포함하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 4

제 1 항에 있어서,

상기 삭제된 제 1 레코드를 복원하는 단계는,

상기 삭제된 제 1 레코드의 시작 주소와 삭제된 공간의 크기 영역을 확인하여 상기 삭제된 제 1 레코드를 복원하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 5

제 2 항에 있어서,

상기 삭제된 제 2 레코드를 추적하는 단계는,

상기 IBD 파일의 스페이스(Space) 영역에서 인덱스 페이지 시작위치인 0xC000 주소를 확인하는 단계;

스페이스 시작지점에서 상기 인덱스 페이지 시작위치까지 이동 후 페이지를 해석하는 단계; 및

상기 인덱스 페이지 시작위치에서 x2C 주소만큼 추가 이동 후 삭제된 레코드의 시작 주소와 삭제된 공간의 크기 영역 값을 확인하는 단계;를 포함하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 6

제 2 항에 있어서,

상기 삭제된 제 2 레코드를 복원하는 단계는,

상기 삭제된 제 2 레코드의 시작 주소와 삭제된 공간의 크기 영역을 확인하여 상기 삭제된 제 2 레코드를 복원하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 7

제 1 항에 있어서,

상기 FRM 파일은,

MySQL 버전 정보, 테이블을 구성한 스토리지 엔진의 종류, 테이블 타입, 컬럼의 개수, 컬럼의 이름, 컬럼 메타 데이터, 데이터 타입, 캐릭터 셋(Character set)을 포함하는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 8

제 1 항에 있어서,

상기 IBD 파일은,

비-트리(B-Tree)로 구성되는 것을 특징으로 하는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법.

청구항 9

삭제

발명의 설명

기술 분야

[0001] 본 발명은 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 관한 것으로, 더욱 상세하게는 MySQL InnoDB 데이터베이스의 파일을 분석하여 삭제된 레코드를 추정하고, 추정된 레코드를 분석된 파일을 이용하여 복원하는 방법에 관한 것이다.

배경 기술

[0002] MySQL 데이터베이스는 오픈소스 형태로는 가장 많이 사용되는 데이터베이스이다. 오픈소스로 배포되기 때문에 중·소기업체에서 많이 사용되고 있으며 미디어 위키나 트루팔과 같은 인기 웹 애플리케이션에서 사용 중이다.

[0003] MySQL의 스토리지 엔진은 대표적으로 MyISAM과 InnoDB가 있으며 MySQL 5.6 버전에서 기본 스토리지 엔진은 InnoDB로 설정된다. 저장 방식에 따라 엔진 사용을 구분할 수 있으며 MyISAM 엔진의 경우 데이터 갱신이 적고 데이터 저장에 목적인 환경에서 사용된다. InnoDB 엔진의 경우 데이터 갱신이 잦고 에러 복구기능이 필요한 환경에서 사용된다. 특히 InnoDB는 다수의 사용자가 데이터베이스를 사용하는 환경에서 효과적이다.

[0004] 한편, 데이터베이스 포렌식 수사 시 조사할 데이터의 규모에 의해 수사 시간과 효율이 결정된다. 최근 스마트 기기를 이용한 애플리케이션 사용률이 증가하여 데이터베이스 용량은 계속해서 증가하고 처리할 데이터가 많아지는 실정이다. 이러한 상황으로 인해 수사관은 데이터베이스 포렌식 수사 시 증거 수집에 어려움을 겪게 된다. 특히 임의로 증거를 없앤 수사 환경의 경우 대용량 데이터베이스 전체를 수사해야 하는 문제가 발생한다. 그리고 로그를 기록하지 않는 환경의 경우 레코드를 삭제한 흔적을 특정하지 못하여 사건과 관련된 증거를 수집하지

못하는 상황을 겪게 된다.

[0005] 그러나, 현재 MySQL InnoDB 데이터베이스 파일에서 로그 파일, 트랜잭션 정보에 비종속적으로 삭제된 데이터를 복원하는 방안이 없어 이를 복원하기 위한 기술이 필요하다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 대한민국 공개특허공보 제10-2012-0097293호 (2012년 09월 03일)

발명의 내용

해결하려는 과제

[0007] 본 발명은 상술한 한계점을 극복하기 위해 안출된 것으로서, 메타데이터 파일의 구조와 데이터베이스 파일의 페이지 정보를 이용함으로써, MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법을 제공하는 것에 그 목적이 있다.

[0008] 또한, 본 발명은 트랜잭션 로그를 일정기간만 보관하거나 없는 경우에도 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원할 수 있는 방법을 제공하는 것에 그 목적이 있다.

과제의 해결 수단

[0009] 상기 목적을 달성하기 위해 본 발명은 MySQL InnoDB 데이터베이스의 FRM 파일을 분석하여 테이블 정보를 획득하는 단계와, 상기 MySQL InnoDB 데이터베이스의 IBD 파일 내 XDES 엔트리에서 페이지 비트맵을 획득하고, 상기 페이지 비트맵을 분석하여 비할당된 페이지 비트맵의 존재 여부를 확인하는 단계와, 상기 비할당된 페이지 비트맵이 존재하는 경우 리프 페이지(Leaf page)로부터 삭제된 제 1 레코드를 추적하는 단계 및 상기 분석된 FRM 파일을 이용하여 상기 삭제된 제 1 레코드를 복원하는 단계를 포함하여 구성된다.

[0010] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 비할당된 페이지 비트맵의 존재 여부를 확인하는 단계는 상기 비할당된 페이지 비트맵이 존재하지 않는 경우 인덱스 페이지(Index page)로부터 삭제된 제 2 레코드를 추적하는 단계 및 상기 분석된 FRM 파일을 이용하여 상기 삭제된 제 2 레코드를 복원하는 단계를 더 포함하는 것을 특징으로 한다.

[0011] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 삭제된 제 1 레코드를 추적하는 단계는 상기 XDES 엔트리 영역을 해석하여 페이지 할당 여부를 확인하는 단계와, 할당된 페이지들에만 접근하여 상기 할당된 페이지가 인덱스 페이지(Index page)인지 확인하는 단계와, 상기 할당된 페이지가 상기 인덱스 페이지인 경우 페이지 레벨을 확인하는 단계 및 상기 페이지 레벨이 0인 경우 삭제된 레코드의 시작 주소(First garbage record offset)와 삭제된 공간의 크기(Garbage space) 영역을 확인하는 단계를 포함하는 것을 특징으로 한다.

[0012] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 삭제된 제 1 레코드를 복원하는 단계는 상기 삭제된 제 1 레코드의 시작 주소와 삭제된 공간의 크기 영역을 확인하여 상기 삭제된 제 1 레코드를 복원하는 것을 특징으로 한다.

[0013] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 삭제된 제 2 레코드를 추적하는 단계는 상기 IBD 파일의 스페이스(Space) 영역에서 인덱스 페이지 시작위치인 0xC000 주소를 확인하는 단계와, 스페이스 시작지점에서 상기 인덱스 페이지 시작위치까지 이동 후 페이지를 해석하는 단계 및 상기 인덱스 페이지 시작위치에서 x2C 주소만큼 추가 이동 후 삭제된 레코드의 시작 주소와 삭제된 공간의 크기 영역 값을 확인하는 단계를 포함하는 것을 특징으로 한다.

[0014] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 삭제된 제 2 레코드를 복원하는 단계는 상기 삭제된 제 2 레코드의 시작 주소와 삭제된 공간의 크기 영역을 확인하여 상기 삭제된 제 2 레코드를 복원하는 것을 특징으로 한다.

[0015] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 FRM 파일은

MySQL 버전 정보, 테이블을 구성한 스토리지 엔진의 종류, 테이블 타입, 컬럼의 개수, 컬럼의 이름, 컬럼 메타데이터, 데이터 타입, 캐릭터 셋(Character set)을 포함하는 것을 특징으로 한다.

[0016] 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 있어서, 상기 IBD 파일은 비트리(B-Tree)로 구성되는 것을 특징으로 한다.

발명의 효과

[0017] 상기와 같이 구성된 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 의하면, 메타데이터 파일의 구조와 데이터베이스 파일의 페이지 정보를 이용함으로써, MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원 가능하다.

[0018] 또한, 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 의하면, 트랜잭션 로그를 일정기간만 보관하거나 없는 경우에도 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원할 수 있다.

도면의 간단한 설명

[0019] 도 1은 본 발명의 일 실시 예에 따른 FRM 파일에서 획득할 수 있는 정보를 나타내는 도면이다.
 도 2는 본 발명의 일 실시 예에 따른 FRM 파일 내 버전 정보를 나타내는 도면이다.
 도 3은 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 타입 정보를 나타내는 도면이다.
 도 4는 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 개수 정보를 나타내는 도면이다.
 도 5는 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 이름 정보를 나타내는 도면이다.
 도 6은 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 메타데이터 정보를 나타내는 도면이다.
 도 7은 본 발명의 일 실시 예에 따른 IBD 파일의 구조를 나타내는 도면이다.
 도 8은 본 발명의 일 실시 예에 따른 XDES 엔트리의 상태 정보를 나타내는 도면이다.
 도 9는 본 발명의 일 실시 예에 따른 IBD 파일 내 인덱스 페이지의 구조를 나타내는 도면이다.
 도 10은 본 발명의 일 실시 예에 따른 리프 페이지의 키 필드(Key Field)의 구조를 나타내는 도면이다.
 도 11은 본 발명의 일 실시 예에 따른 논-리프 페이지의 논-키 필드(Non-Key Field)의 구조를 나타내는 도면이다.
 도 12는 본 발명의 바람직한 실시 예에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법을 나타내는 순서도이다.
 도 13은 리프 페이지로부터 삭제된 제 1 레코드를 추적하는 방법을 나타내는 도면이다.
 도 14는 인덱스 페이지로부터 삭제된 제 2 레코드를 추적하는 방법을 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 본 발명은 다양한 변형 및 여러 가지 실시 예를 가질 수 있는바, 특정 실시 예들을 도면에 예시하고 상세한 설명에 보다 상세하게 설명하고자 한다. 본 발명을 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

[0021] 이하, 본 발명의 바람직한 실시 예를 첨부된 도면과 참조하여 상세히 설명하기로 한다.

[0022] 먼저, 데이터를 복원하기에 앞서 MySQL InnoDB 데이터베이스에 대해서 설명하겠다. MySQL InnoDB 데이터베이스는 FRM, IBD 2개의 파일을 통해 테이블을 관리한다. 여기서 FRM 파일에 대해서는 도 1을 참조하여 구체적으로 설명하겠다.

[0023] 도 1은 본 발명의 일 실시 예에 따른 FRM 파일에서 저장된 정보를 나타내는 도면으로, FRM 파일로부터 MySQL 버전 정보, 테이블을 구성한 스토리지 엔진의 종류, 테이블 타입, 컬럼의 개수, 컬럼의 이름, 컬럼 메타데이터, 데이터 타입, 캐릭터 셋(Character set) 등에 대한 정보를 확인할 수 있다.

[0024] 여기서 MySQL 버전 정보 확인 방법은 도 2를 참조하여 설명하겠다. 도 2는 본 발명의 일 실시 예에 따른 FRM 파

일 내 버전 정보를 나타내는 도면으로 FRM 파일 내 버전 정보를 나타내는 영역을 확인할 수 있다. 버전 정보는 0x33-0x34 주소에 리틀 엔디언 방식으로 저장되며, 10진수 형으로 변환하여 해석한다.

- [0025] 이러한 MySQL 버전 정보는 앞의 한 자리 값이 Major version을 의미하고, 한 자리를 건너뛰고 나머지 세 자리 값은 Minor version을 의미한다. 도 2에서 버전 정보인 2바이트 값을 10진수로 나타내면 50613을 나타내고 있어, 도 2의 MySQL 버전은 5.6.13을 의미함을 확인할 수 있다.
- [0026] 다음으로 테이블 타입 정보에 대해서는 도 3을 참조하여 설명하겠다. 도 3은 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 타입 정보를 나타내는 도면이다. 도 3을 참조하면, 테이블 타입 정보는 0x1E 주소에 존재하는 2바이트 값이며, 이 값이 짝수일 경우 Static 테이블을 의미하고, 홀수일 경우 Dynamic 테이블을 의미한다. 도 3에서는 Static 테이블을 의미하고 있음을 확인할 수 있다.
- [0027] 더불어 컬럼의 개수, 컬럼의 이름, 컬럼 메타데이터 정보에 대해서는 도 4 내지 도 6을 참조하여 설명하겠다. 먼저, 도 4는 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 개수 정보를 나타내는 도면이다.
- [0028] 도 4를 참조하면 컬럼의 개수 정보는 0x2102 주소에 존재하는 2바이트 값이며, 값 그대로 테이블의 컬럼 수를 의미한다. 따라서, 도 4에서 컬럼의 개수가 0x0002 개를 나타내고 있음을 확인할 수 있다.
- [0029] 다음으로 도 5는 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 이름 정보를 나타내는 도면으로, 컬럼의 이름 정보는 0x2150번지부터 저장된다. 특히, 컬럼의 개수와 컬럼 이름의 길이에 따라 저장된 영역의 크기가 다르다. Sequence number는 0x04부터 시작하여 순차적으로 증가한다.
- [0030] 마지막으로, 도 6은 본 발명의 일 실시 예에 따른 FRM 파일 내 테이블의 컬럼 메타데이터 정보를 나타내는 도면이다. 도 6을 참조하면, 컬럼 메타데이터 정보가 컬럼 이름 정보 다음에 저장되고 있으며, 0x00 주소에는 Sequence number, 0x01 주소에는 컬럼 이름의 길이, 0x03 주소부터 2바이트에는 컬럼의 길이, 0x0c 주소에는 데이터 타입, 0x0E 주소에는 캐릭터 셋 정보가 저장된다.
- [0031] 이와 같이 FRM 파일 내에서 MySQL 버전 정보, 테이블 타입, 컬럼의 이름, 길이, 데이터 타입, 캐릭터 셋 정보 등을 얻어 추후 IBD 파일에서 레코드를 복구할 때 사용할 수 있다.
- [0032] 다음으로 MySQL InnoDB 데이터베이스의 IBD 파일은 InnoDB의 데이터에 대한 기록을 저장하고 있으며, 비-트리(B-tree)로 구성되어 있다. IBD 파일 안에는 트랜잭션을 위한 데이터, 각 트리들에 대한 인덱스 정보를 다루는 노드 정보, 그리고 각 데이터베이스가 각 테이블과 어떠한 연관성을 가지는지에 대한 맵핑 정보 등, 모든 데이터가 하나의 파일에 집약되어 있고 하나의 파일 안에서 모든 내용이 유기적으로 연결된다.
- [0033] IBD 파일은 각각의 데이터의 구분을 쉽게 하기 위하여 페이지 단위로 데이터를 구성하고 있으며, 각 페이지는 0x4000(16,384)bytes로 이루어져 있다. 또한 64개의 페이지를 extent라는 단위로 관리하며, 256개의 extent를 하나의 space로 관리한다. 각 페이지들은 자신의 정보를 헤더를 통해 따로 가지고 있으며, 페이지에 대한 헤더 정보는 모든 페이지가 동일하게 적용된다.
- [0034] 따라서 IBD 파일의 각 페이지들의 헤더 값이 어떠한 값을 가지는지 주목할 필요가 있으며, 그 헤더 값을 통해 각 페이지의 속성이 달라진다. 이러한 IBD 파일에 대해서는 도 7을 참조하여 보다 구체적으로 설명하겠다.
- [0035] 도 7은 본 발명의 일 실시 예에 따른 IBD 파일의 구조를 나타내는 도면이다. 도 7을 참조하면 IBD 파일의 전체적인 구조와 페이지 헤더, space의 정보를 나타내는 FSP 헤더(FSP Header), 페이지 비트맵(Page state bitmap)이 존재하는 XDES 엔트리를 나타낸다.
- [0036] 첫 번째 space의 시작 페이지인 IBD 파일의 첫 페이지는 38바이트의 페이지 헤더 이후에 첫 번째 space에 관한 정보를 FSP 헤더에서 확인할 수 있다.
- [0037] 다음으로 XDES 엔트리에서는 State, 페이지 비트맵 등을 확인할 수 있으며, State는 상태 정보 값을 가지고 상태 정보 값의 의미는 도 8과 같이 4개의 값으로 구분된다.
- [0038] 먼저, XDES_FREE는 어떤 페이지도 할당되지 않은 상태의 extent를 의미하고 있으며, XDES_FREE_FRAG는 일부 페이지들이 할당되었으나, 할당되지 않은 페이지가 존재하는 상태의 extent를 의미한다.
- [0039] 다음으로 XDES_FULL_FRAG는 프리 페이지(Free page)가 존재하지 않는 상태의 extent를 의미하고, XDEF_FSEG는 해당 extent가 특정 파일 세그먼트에 존재하고 있음을 의미한다.
- [0040] 더불어 XDES 엔트리의 페이지 비트맵은 추후 리프 페이지(Leaf page)에서 삭제된 레코드를 추적할지 인덱스 페

이지(Index page)에서 삭제된 레코드를 추적할지에 대해서 분기하는 역할을 하는 값으로, 하나의 XDES 엔트리에 존재하는 페이지 비트맵은 16바이트 값을 가지며, 각각 2비트 단위로 페이지의 할당을 나타내어 하나의 extent에 존재하는 64개 페이지의 할당 여부를 나타낼 수 있다.

- [0041] 한편 InnoDB는 인덱스 페이지를 관리하기 위하여 비-트리틀을 사용할 수 있다. 실제로 데이터가 저장되는 영역이며 루트 페이지에서 레벨을 확장하며 저장할 수 있다.
- [0042] 또한, 인덱스 페이지는 리프 페이지와 논-리프 페이지(Non-Leaf page)로 구분하며, 논-리프 페이지의 경우 다른 페이지로 연결할 수 있는 포인터를 저장한다. 리프 페이지는 실제 데이터를 저장하는 페이지이며, 논-키 필드(Non-Key Field)에 데이터를 저장한다. 여기서 인덱스 페이지의 구조는 도 9를 참조하여 설명하겠다.
- [0043] 도 9는 본 발명의 일 실시 예에 따른 IBD 파일 내 인덱스 페이지의 구조를 나타내는 도면이다. 도 9를 참조하면 인덱스 페이지의 구조는 space의 시작으로부터 4번째 페이지(시작주소 0xC000)에서 시작한다.
- [0044] 도 9와 같이 인덱스 페이지의 인덱스 헤더(Index Header)에는 인덱스 페이지의 헤더에는 리프 페이지와 논-리프 페이지를 구분할 수 있는 페이지 레벨 값, 삭제된 레코드의 시작 주소(First garbage record offset)와 삭제된 공간의 크기(garbage space) 정보가 존재한다.
- [0045] 또한, 페이지 레벨(Page level)은 리프 페이지일 경우 0을 나타내고, 논-리프 페이지인 경우에는 가변적으로 증가하기 때문에 이 값을 이용하여 리프 페이지를 나타내는지 여부를 확인할 수 있다.
- [0046] 도 9에서 레코드 포맷(Record Format) 구조는 리프 페이지, 논-리프 페이지가 각각 다르므로, 페이지의 레벨에서 리프 페이지와 논-리프 페이지를 구분한 후 분석한다. 해당 페이지가 리프 페이지일 경우 실제 데이터가 저장되는 페이지로써 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법을 이용하여 삭제된 레코드를 추적 이후 삭제된 레코드 복원에 사용되는 페이지이다. 이러한 리프 페이지의 레코드가 저장되는 키 필드(Key Field) 구조는 도 10과 같다.
- [0047] 도 10은 본 발명의 일 실시 예에 따른 리프 페이지의 키 필드(Key Field)의 구조를 나타내는 도면이다. 도 10을 참조하면, 리프 페이지의 키 필드는 가변 필드 길이(Variable Fields Lengths), 플래그 정보(Info Flags), 소유하고 있는 레코드의 수(Number of Records Owned), 오더 및 레코드 타입(Order, Record Type), 다음 레코드의 위치(Next Record offset), 클러스터 키 필드(Cluster Key Fields), 트랜잭션 ID(Transaction ID), 롤 포인터(Roll Pointer), 논-키 필드 등을 포함하고 있다.
- [0048] 가변 필드 길이는 레코드에서 차지하는 데이터 길이를 의미하며 처음 1비트는 공백 비트이고 나머지 7 비트는 값에 따라 변동하는 1바이트 이상의 크기를 가진다. 레코드에 할당된 데이터가 127개 이상인 경우 처음 1비트는 공백 비트 2번째 비트는 페이지를 넘어가면 1, 넘어가지 않으면 0으로 설정된다.
- [0049] 다음으로 플래그 정보는 Extra 바이트 중 첫 1 바이트에서 최상위 4비트를 의미하며, 여기서 최상위 3번째 비트가 1일 경우 삭제된 레코드를 의미하며, 예를 들면 플래그 정보 값이 0010 0000인 경우 최상위 비트에서 3번째 비트가 1이므로 삭제된 레코드를 의미한다.
- [0050] 소유하고 있는 레코드의 수는 해당 레코드의 Directory 슬롯에서 다른 레코드들을 포함한 개수를 의미하고, 오더 및 레코드 타입은 인덱스 페이지의 힙(Heap)에서 현재 레코드의 순서 번호를 의미하며 상위 13비트를 의미한다.
- [0051] 여기서 최하위 3비트는 레코드 타입으로 000은 conventional, 001은 node pointer(비-트리 상에서), 010은 infimum, 011은 supremum, 1xx는 reserved를 의미한다.
- [0052] 더불어 다음 레코드의 위치는 다음 레코드 위치를 가지는 포인터를 의미하며 다음 레코드 위치는 레코드 시작 위치부터 포인터 오프셋만큼 더하여 계산한다. 삭제된 레코드에서는 두 번째로 삭제된 레코드를 가리킨다.
- [0053] 클러스터 키 필드는 레코드에 할당된 키 값이 저장되는데 키 값의 타입에 따라 크기 및 형식이 가변적이다. 예를 들면 tinyint, bigint 등의 타입에 따라 크기 및 형식이 가변적으로 저장된다.
- [0054] 마지막으로 논-키 필드에서 레코드 해석은 FRM 정보를 이용하여 데이터 타입에 맞게 바이트 해석을 한다.
- [0055] 상기와 같이 리프 페이지의 키 필드 구조에 대해서 설명하였다. 다음으로 논-리프 페이지의 논-키 필드 구조에 대해서 설명하겠다.
- [0056] 도 11은 본 발명의 일 실시 예에 따른 논-리프 페이지의 논-키 필드(Non-Key Field)의 구조를 나타내는 도면이

다. 도 11은 도 10의 키 필드 구조와 동일한 부분에 다수 포함되어 있어 그 내용에 대한 설명은 생략하겠다.

- [0057] 도 11에 도 10과 다르게 클러스터 키 미니멈 및 키 온 차일드 페이지(Cluster key minimum, Key on child page)와 차일드 페이지 넘버를 저장하고 있다.
- [0058] 클러스터 키 미니멈 및 키 온 차일드 페이지(n bytes)는 n번째 자식페이지 중 가장 작은 키 값이 설정되고, 키 값의 타입에 따라 사이즈 및 형식이 변동되며 차일드 페이지 넘버에서는 해당 차일드 페이지의 넘버를 저장한다.
- [0059] 이상으로 MySQL InnoDB 데이터 베이스의 FRM 파일과 IBD 파일에 대해서 설명하였다. 다음으로 FRM 파일과 IBD 파일을 이용하여 MySQL InnoDB 데이터 베이스에서 삭제된 파일을 복원하는 방법에 대해서 설명하겠다.
- [0060] 도 12는 본 발명의 바람직한 실시 예에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법을 나타내는 순서도이다. 도 12를 참조하면, 데이터 복원 장치는 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하기 위해 먼저 MySQL InnoDB 데이터베이스의 FRM 파일로부터 테이블 정보를 획득한다(S10).
- [0061] 삭제된 데이터를 복원하기 위해서는 IBD 파일 내부의 레코드 해석이 필요한데 이를 해석하기 위해서는 FRM 파일의 분석이 필요하다. 이러한 FRM 파일은 도 1에서 설명한 바와 같이 다양한 테이블 정보를 포함하고 있다.
- [0062] 다음으로 데이터 복원 장치는 MySQL InnoDB 데이터베이스의 IBD 파일 내 페이지 비트맵을 분석하고(S20), 분석된 페이지 비트맵 중 비할당된 페이지 비트맵의 존재 여부를 확인한다(S30).
- [0063] 즉, 데이터 복원 장치는 MySQL InnoDB 데이터베이스에서 삭제된 레코드 추적을 리프 페이지에서 할지 인덱스 페이지에서 할지 결정하기 위해 페이지 비트맵을 분석하여 비할당된 페이지 비트맵의 존재 여부를 확인한다.
- [0064] 비할당된 페이지 비트맵의 존재 여부를 확인한 결과, 비할당된 페이지 비트맵이 존재하는 경우에는 리프 페이지로부터 삭제된 제 1 레코드를 추적한다(S40).
- [0065] 여기서 리프 페이지로부터 삭제된 제 1 레코드를 추적하는 방법에 대해서는 도 13을 참조하여 설명하겠다. 도 13은 리프 페이지로부터 삭제된 제 1 레코드를 추적하는 방법을 나타내는 도면이다.
- [0066] 도 13을 참조하면, 우선 리프 페이지에서 레코드 삭제를 추적하는 방법은 XDES 엔트리 영역을 해석하여 페이지 할당 여부를 확인한다(S131). 이후 할당된 페이지들에만 접근하여 인덱스 페이지 여부를 확인한다(S132).
- [0067] 다음으로 할당된 페이지가 인덱스 페이지인 경우 페이지 레벨을 확인한다(S133). 여기서 페이지 레벨이 0인 리프 페이지일 때 논-키 필드에 일반데이터가 저장되므로, 페이지 레벨을 확인하여 리프 페이지를 찾는다.
- [0068] 페이지 레벨을 확인한 결과 0인 경우 삭제된 레코드의 시작 주소와 삭제된 공간의 크기 영역을 확인한다(S134). 여기서 삭제된 레코드의 시작 주소나 삭제된 공간의 크기 영역이 공백이 아닌 경우에는 삭제된 제 1 레코드로 추정한다.
- [0069] 이렇게 삭제된 제 1 레코드를 추정함으로써, 대용량 데이터베이스에서 데이터베이스 전체를 조사하는 비용을 줄이기 위해 삭제되었을 확률이 높은 레코드 영역을 특정해야 한다. 삭제된 제 1 레코드를 추적하는 방법에서 XDES 엔트리의 상태 정보를 확인하여 복구해야 할 extent를 특정할 수 있다. 이후 페이지 비트맵 정보를 확인하여 비할당 영역을 복구하거나, 인덱스 페이지로부터 삭제된 제 2 레코드를 추적하는 방법을 사용하여 복구해야 할 영역을 복구할 수 있다.
- [0070] 이상으로 리프 페이지로부터 삭제된 제 1 레코드를 추적하는 방법을 설명하였다. 다시 도 12로 돌아가 다음 과정에 대해서 설명하겠다. 삭제된 제 1 레코드 추적이 완료되면, 분석된 FRM 파일을 이용하여 삭제된 제 1 레코드를 복원한다(S50). 즉 FRM 파일에 포함된 정보를 이용하여 삭제된 제 1 레코드의 IBD 파일을 복원한다.
- [0071] 한편, 데이터 복원 장치가 분석된 페이지 비트맵 중 비할당된 페이지 비트맵의 존재 여부를 확인하는 단계(S30)에서 비할당된 페이지 비트맵이 존재하는 않는 경우에는 인덱스 페이지로부터 삭제된 제 2 레코드를 추적한다(S60).
- [0072] 여기서 인덱스 페이지로부터 삭제된 제 2 레코드를 추적하는 방법에 대해서는 도 14 참조하여 설명하겠다. 도 14는 인덱스 페이지로부터 삭제된 제 2 레코드를 추적하는 방법을 나타내는 도면이다.
- [0073] 도 14를 참조하면, 우선 IBD 파일의 스페이스 영역에서 인덱스 페이지 시작위치인 0xC000 주소를 확인한다(S141). 여기서 File space header 영역과 Insert buffer bookkeeping 영역, Index node information 영역은 매 스페이스마다 고정적으로 존재하기 때문에 스페이스 시작지점에서 0xC000 위치로 이동 후 페이지를

해석한다.

- [0074] 다음으로 0xC000 위치에서 x2C 위치만큼 추가 이동 후 삭제된 레코드의 시작 주소를 확인하고(S142), 삭제된 공간의 크기 정보를 확인한다(S143). 여기서 삭제된 레코드의 시작 주소와 삭제된 공간의 크기 정보가 공백이 아니라면 삭제된 레코드나 잉여공간이 존재한다. 따라서 이를 확인하여 삭제된 제 2 레코드를 추정한다.
- [0075] 이와 같은 과정을 인덱스 페이지가 끝나는 시점까지 반복 수행하여 삭제된 제 2 레코드를 추적한다(S144).
- [0076] 이상으로 인덱스 페이지로부터 삭제된 제 2 레코드를 추적하는 방법에 대해서 설명하였다. 다음으로는 다시 도 12로 돌아가 이후 과정에 대해서 설명하겠다.
- [0077] 인덱스 페이지로부터 삭제된 제 2 레코드 추적이 완료되면, 데이터 복원 장치는 분석된 FRM 데이터를 이용하여 삭제된 제 2 레코드를 복원한다(S70). 여기서 삭제된 제 2 레코드를 복원하는 과정은 삭제된 제 1 레코드를 복원하는 과정과 동일하게 FRM 파일에 포함된 정보를 이용하여 삭제된 제 2 레코드의 IBD 파일을 복원한다.
- [0078] 상기와 같이 구성된 본 발명에 따른 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원하는 방법에 의하면, 메타데이터 파일의 구조와 데이터베이스 파일의 페이지 정보를 이용함으로써, MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원 가능하며, 트랜잭션 로그를 일정기간만 보관하거나 없는 경우에도 MySQL InnoDB 데이터베이스에서 삭제된 데이터를 복원할 수 있다.
- [0079] 본 명세서에 기재된 본 발명의 실시 예와 도면에 도시된 구성은 본 발명의 가장 바람직한 실시 예에 관한 것이고, 발명의 기술적 사상을 모두 포괄하는 것은 아니므로, 출원시점에 있어서 이들을 대체할 수 있는 다양한 균등물과 변형 예들이 있을 수 있음을 이해하여야 한다. 따라서 본 발명은 상술한 실시 예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변형실시가 가능한 것은 물론이고, 그와 같은 변경은 청구범위 기재의 권리범위 내에 있게 된다.

도면

도면1

구성요소	설명
Storage Engine Information	MySQL 버전 정보, 테이블을 구성한 스토리지 엔진의 종류
Table Type	Static, Dynamic 테이블 정보
Table Schema	컬럼의 개수, 이름, 데이터 타입, Charset 타입 정보

도면2

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FE	01	09	09	03	00	00	10	01	00	00	30	00	00	10	00	b 0
00000010	2F	00	00	00	00	00	00	00	00	00	00	02	08	00	08	00	/
00000020	00	05	00	00	00	00	00	21	00	00	00	00	00	00	00	10	!
00000030	00	00	00	B5	CS	00	00	23	00	00	00	00	00	00	00	00	#
00000040	2F	2F	00	00	20	00	00	00	00	00	00	00	00	00	00	00	//
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

도면3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	FE	01	0A	0C	03	00	00	10	01	00	00	30	00	00	10	00	b.....0...
0010h:	CE	0B	00	00	00	00	00	00	00	00	00	02	08	00	09 00		f......
0020h:	00	05	00	00	00	00	21	00	00	00	00	00	00	00	00	10	!

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	FE	01	09	0C	03	00	00	10	01	00	00	30	00	00	10	00	b.....0...
0010h:	0D	00	00	00	00	00	00	00	00	00	00	02	08	00	08 00		
0020h:	00	05	00	00	00	00	21	00	00	00	00	00	00	00	00	10	!

도면4

000020E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000020F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00002100	01	00	02 00	3C	00	47	00	00	00	42	00	08	00	00	00	00	< G B
00002110	00	00	00	00	00	00	50	00	16	00	02	00	00	00	00	00	P
00002120	00	00	00	00	00	14	00	00	00	00	00	00	00	00	00	00	

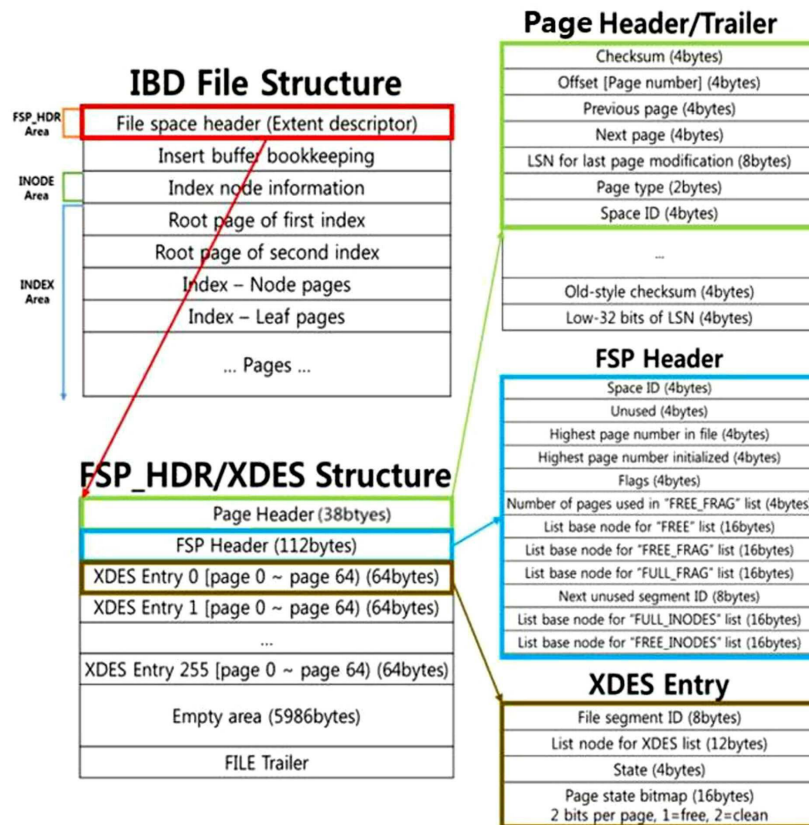
도면5

2 byte	1 byte	n byte	2 byte	1 byte	n byte	...
Seq number	Column name length	Column Name	Seq number	Column name length	Column Name	
00002150	04 00 03	76 31 00	05 00 03	76 32 00	04 03 0B 0B	v1 v2
00002160	00 02 00 00	1B 80 00 00	00 03 21 00	00 05 03 3C		! ! <
00002170	3C 00 06 00	00 00 80 00	00 00 0F 21	00 00 FF 76		< ! ! yv
00002180	31 FF 76 32	FF 00				1yv2y

도면6

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Seq Number	Column name Length	N/A	Column Allocation Length										Data Type	Charset Type	End Signature 0x00
End Signature 0x00															

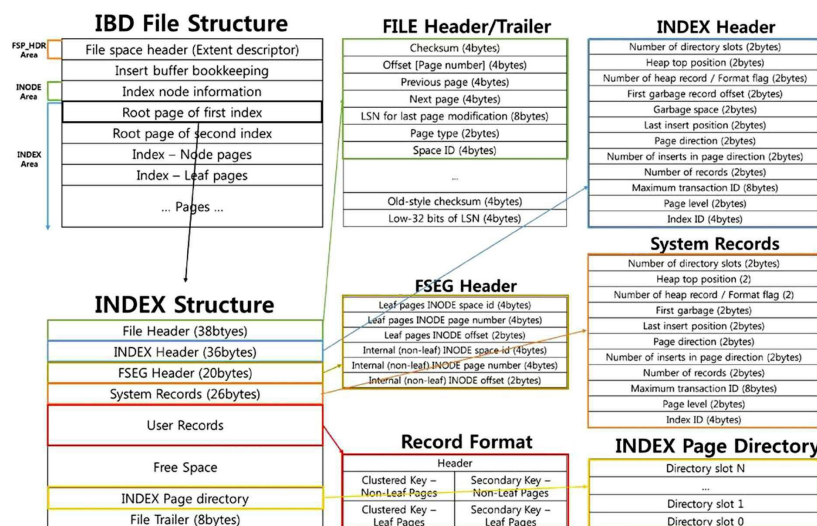
도면7



도면8

State Information	Value
XDES_FREE : Extent in Free List of Space.	1
XDES_FREE_FRAG : Extent in Free Fragment List of Space.	2
XDES_FULL_FRAG : Extent in Full Fragment List of Space.	3
XDES_FSEG : Extent existing in Segment.	4

도면9



도면10

Clustered Key : Leaf Pages

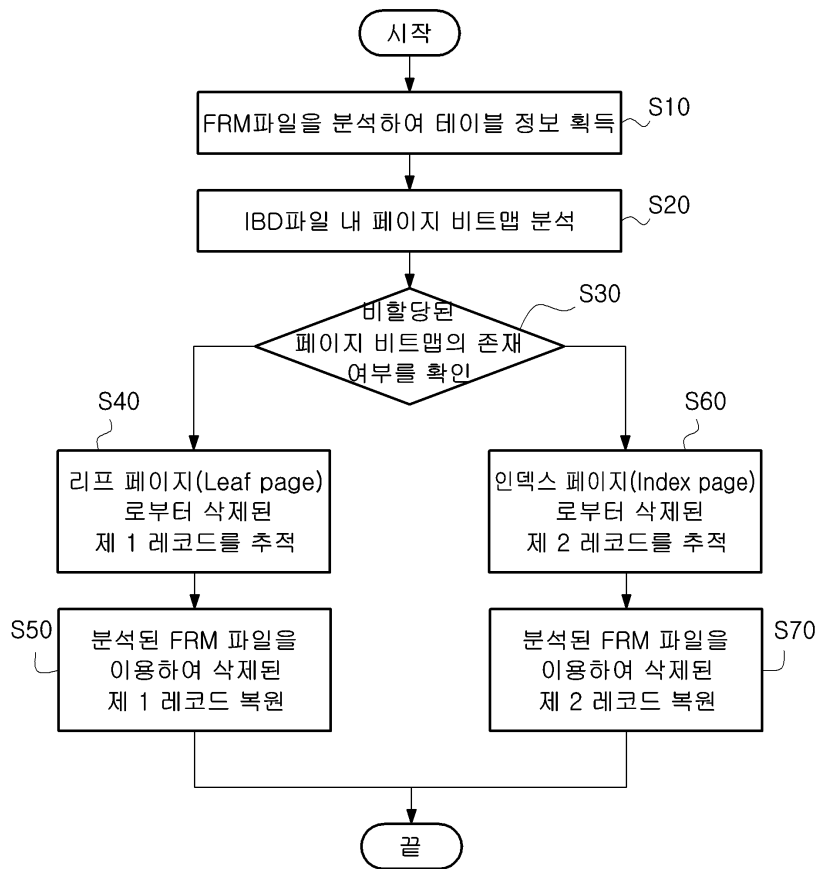
Variable field lengths (1-2 bytes per var. field)
Info flags (4 bits)
Number of records owned (4 bits)
Order (13bits)
Record type (3 bits)
Next record offset (2 bytes)
Cluster Key Fields (n bytes)
Transaction ID (6bytes)
Roll Pointer (7bytes)
Non-Key Fields (m bytes)

도면11

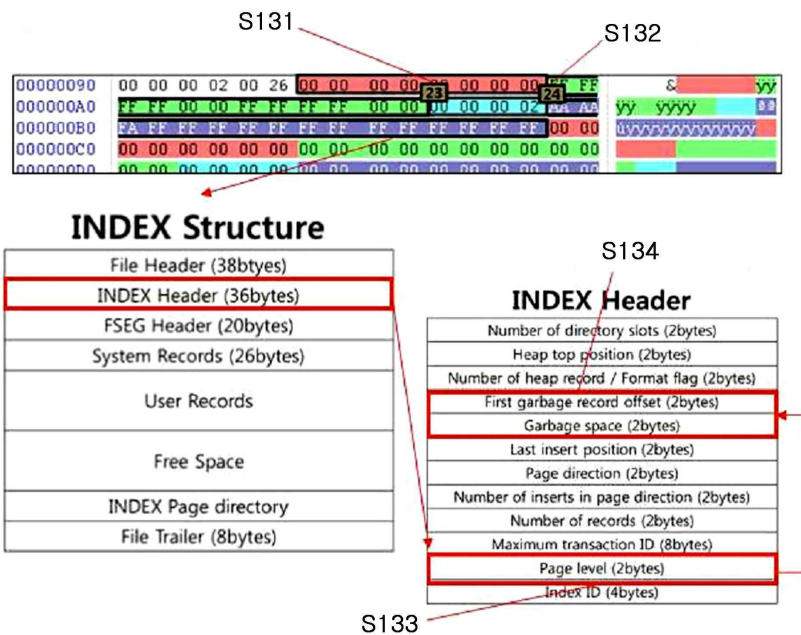
Clustered Key : Non-Leaf Pages

Variable field lengths (1-2 bytes per var. field)
Info flags (4 bits)
Number of records owned (4 bits)
Order (13bits)
Record type (3 bits)
Next record offset (2 bytes)
Cluster key minimum, Key on child page (n)
Child page number (4)

도면12



도면13



도면14

