Database:

191103

1. advantages of DBMS over traditional file-based systems

DMBS는 기존 파일 프로세싱 시스템의 다음과 같은 어려움을 해결하기 위해 개발되었다.

1) 데이터 중복과 데이터 불일치 (data redundancy & inconsistency)

2) 데이터 접근의 어려움

3) 데이터 고립화 (isolation)

4) 무결성 문제 (integrity)

5) 업데이트의 원자성 (atomicity)

6) 다수 사용자의 동시 접근

7) 보안 문제

* file system vs DBMS: https://yaboong.github.io/database/2017/09/01/database-1/

* 데이터 중복과 불일치: https://pediaa.com/what-is-the-difference-between-data-redundancy-and-data-inconsistency/

* 데이터 무결성: http://www.devholic.net/1000290

2. super key, candidate key, primary key, alternate key, foreign key

슈퍼키는 관계형 DB 모델에서 릴레이션의 각 튜플을 고유하게 식별하는 속성 또는 속성의 집합이다.

후보키는 최소의 슈퍼키다. 키를 구성하는 속성 중 하나라도 제외하는 경우 유일성이 깨진다.

기본키는 가장 중요한 것으로 선택된 후보키다. 하나의 테이블에서는 하나의 기본키만을 가져야 한다. null 값을 가질 수 없다.

대체키는 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키다.

외래키는 한 테이블 내에서, 참조하고 있는 다른 테이블의 기본키와 같은 해당 테이블의 속성이다.

* 키 개념 및 종류: https://limkydev.tistory.com/108

* 카카오 후보키 문제: https://programmers.co.kr/learn/courses/30/lessons/42890

2.1. primary key vs unique constraints

고유 제약조건은 null 값을 가질 수 있지만, 기본키는 null 값을 가질 수 없다. 한 릴레이션 내 고유 제약조건은 여러 개가 있을 수 있지만, 기본키는 오직 하나다.

3. relation vs table

테이블은 다음의 조건을 만족하면서 릴레이션과 같아진다.

1) 행과 열의 순서를 고려하지 않는다.

2) 중복되는 튜플이 존재하지 않는다.

3) 모든 행과 열의 교차점은 값을 가진다. 즉, null 값을 가지지 않는다.

4) 각 행마다 한 열에 하나의 값을 가져야 한다. 즉, 원자성을 만족해야 한다.

4. database normalization

데이터베이스 정규화는 다음의 특성을 얻기 위해 함수 종속성과 기본키를 바탕으로 주어진 릴레이션 스키마를 분석하는 과정이다.

1) 중복 최소화

2) 삽입, 삭제, 갱신에 의한 이상(anomaly) 최소화

상기한 특성을 만족시키지 못하는 릴레이션 스키마는 그것들을 만족하는 더 작은 릴레이션 스키마로 나뉜다.

* 이상과 함수 종속성: https://yaboong.github.io/database/2018/03/09/database-anomaly-and-functional-dependency/

4.1. normal form

정규화된 정도를 정규형으로 표현한다. 높은 단계로 갈수록 중복과 이상이 적은 릴레이션으로 설계된다. 높은 단계의 정규형은 이전 단계 정규형의 조건들을 만족한다. 비공식적으로 3NF가 되었으면 정규화 되었다고 말한다.

1) 1NF: 릴레이션이다. 부분 함수 종속성을 가지게 되는 경우 삽입, 삭제, 갱신 이상이 모두 나타난다.

2) 2NF: 1NF + 기본키에 속하지 않는 모든 속성이 기본키에 완전 함수 종속이다. (=기본키의 진부분집합에 대해 함수 종속이 아니다.) 이행 함수 종속성을 가지는 경우 삽입, 삭제, 갱신 이상이 모두 나타난다.

3) 3NF: 2NF + 기본키에 속하지 않는 모든 속성이 기본키에 이행 함수 종속이 아니다. 후보키가 여러 개인 경우 이상이 발생할 수 있다.

4) BCNF: 3NF + 모든 결정자가 슈퍼키다.

* 1-3NF: https://yaboong.github.io/database/2018/03/09/database-normalization-1/

* BCNF: https://yaboong.github.io/database/2018/03/10/database-normalization-2/

* 그 외 정규형: https://victorydntmd.tistory.com/132,

5. SQL

Structured Query Language의 약자. 관계형 DBMS의 데이터를 관리하기 위해 설계된 프로그래밍 언어이다. SQL

로 할 수 있는 것들은 다음과 같다.

1) DB에 질의를 보낼 수 있다.

2) DB로부터 데이터를 가져올 수 있다.

3) DB에 새로운 레코드를 삽입할 수 있다.

4) 이미 존재하는 레코드를 갱신할 수 있다.

5) 레코드를 삭제할 수 있다.

6) 새로운 DB를 생성할 수 있다.

7) DB에 새로운 테이블을 생성할 수 있다.

8) DB에 저장 프로시저(stored procedure)를 생성할 수 있다.

9) DB에 뷰를 생성할 수 있다.

10) 테이블, 프로시저, 뷰에 권한을 설정할 수 있다.

* SQL이란: https://nachwon.github.io/sql-1-intro/

* Wikipedia SQL: https://ko.wikipedia.org/wiki/SQL

6. DDL, DML, DCL

1) 데이터 정의 언어(data definition language): RDB의 구조를 생성, 삭제, 변경하는 데 사용한다. 구조에는 행렬, 테이블, 인덱스, 저장 프로시저 등이 포함된다.

- CREATE: 새로운 테이블, 뷰, 인덱스, 저장 프로시저를 만든다.
- DROP: 기존의 테이블, 뷰, 인덱스, 저장 프로시저를 제거한다.
- ALTER: 기존의 DB 개체를 변경한다. 열의 삭제 및 추가, 테이블 이름 변경 등이 예이다.
- TRUNCATE: 테이블의 모든 행을 삭제한다.

2) 데이터 조작 언어(data manipulation language): 데이터를 검색, 삽입, 삭제, 갱신하는 데 사용한다.

- SELECT: 데이터를 검색한다.
- INSERT: 새로운 데이터를 삽입한다.
- UPDATE: 기존의 데이터를 갱신한다.
- DELETE: 기존의 데이터를 삭제한다.

3) 데이터 제어 언어(data control language): 데이터에 대한 접근을 제어하는 데 사용한다.

- GRANT: 특정 DB 사용자에게 특정 작업을 수행할 권한을 부여한다.
- REVOKE: 특정 DB 사용자에게 부여한 특정 권한을 박탈한다.

6.1. WHERE vs HAVING

WHERE 절과 HAVING 절 모두 조건을 걸 때 사용한다. HAVING 절은 그룹화된 결과 집합의 행에 대해서만 적용된다. 반면, WHERE 절은 개별 행에 적용된다. WHERE 절은 SUM, AVG 등의 집계 함수(aggregate function) 보다 먼저 실행된다. 따라서 집계 함수를 사용하려면 HAVING 절을 사용하면 된다.

* GROUP BY, HAVING 절: http://wiki.gurubee.net/pages/viewpage.action?pageId=26743892

6.2. JOIN

JOIN은 공통 필드를 가지고 두 개 이상의 테이블을 결합하는 데 사용한다.

*What is join? 항목 참조: https://www.geeksforgeeks.org/commonly-asked-dbms-interview-questions/

7. identity column

ID열은 DB에 의해 자동으로 생성되는 값으로 구성되어 있다. 시작 값과 증가 값을 설정할 수 있다. 대부분의 DB 애플리케이션은 각각을 1로 설정한다. GUID열 역시 자동으로 숫자가 생성된다. GUID열의 값은 제어할 수 없다. 많은 경우 ID와 GUID는 기본키로 사용된다.

* GUID vs ID: https://stackoverflow.com/questions/829284/guid-vs-int-identity

8. view

뷰는 하나 이상의 원본 테이블에 대한 쿼리의 결과로 가상 테이블이다. 외부 스키마와 개념 스키마의 독립성을 보장하기 위해 사용한다. 뷰의 용도는 다음과 같다.

1) 원본 테이블 내용의 노출을 제한할 수 있다.

2) 여러 원본 테이블의 내용을 하나의 가상 테이블로 결합/단순화할 수 있다.

3) SUM, AVG 등의 집계 연산의 결과 테이블로 기능한다.

4) 외부 스키마에 맞춘 결과를 보여줄 수 있다. (=실제 데이터의 복잡성을 숨길 수 있다.)

5) 원본 데이터의 일부만 포함하고 있기 때문에 매우 작은 저장공간만을 차지한다.

* 뷰: https://victorydntmd.tistory.com/131?category=687930

9. trigger

트리거는 DB의 특정한 테이블이나 뷰의 이벤트에 반응하여 자동으로 실행되는 프로시저 코드다. 데이터의 변경이 있을 때 관련 값들을 재조정하는 등 데이터 무결성을 유지하는 데 유용하다.

* Wikipedia trigger: https://en.wikipedia.org/wiki/Database_trigger#Microsoft_SQL_Server

10. stored procedure

저장 프로시저는 일련의 쿼리를 하나의 함수처럼 실행하기 위한 쿼리 집합이다. 트리거와는 다르게 직접 호출이 가능하다.

11. transaction

트랜잭션은 DB의 상태를 변화시키는 논리적 작업의 단위다. 여러 연산의 결합일 수 있지만 하나의 작업으로 간주되어야 한다. 하나의 트랜잭션은 COMMIT 되거나 ROLLBACK 된다.

12. ACID

ACID는 트랜잭션이 안전한 수행을 보장하기 위한 성질이다.

1) 원자성(Atomicity): 트랜잭션의 연산은 DB에 모두 반영되어야 한다. 트랜잭션 내의 모든 명령은 반드시 완벽하게 수행되어야 한다. 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 한다. ex) 자금이체 시 출금과 입금이 모두 수행되어야 한다.

2) 일관성(Consistency): 트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성 있는 DB상태를 유지한다. 성공적으로 수행된 트랜잭션은 정당한 데이터만을 DB에 반영해야 한다. 시스템이 가지고 있는 고정요소는 트랜잭션 수행 전후의 상태가 같아야 한다. ex) 기본키, 외래키 제약. 자금이체 시 두 계좌 잔고의 합이 이체 전후로 같아야 한다.

3) 독립성(Isolation): 둘 이상의 트랜잭션이 동시에 실행되는 경우, 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없다. 수행 중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 결과를 참조할 수 없다.

4) 지속성(Durability): 성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다.

* 트랜잭션의 정의: https://coding-factory.tistory.com/226

* DBMS는 어떻게 트랜잭션을 관리할까: https://d2.naver.com/helloworld/407507

13. index

인덱싱은 RDBMS에서 검색 속도를 높이기 위해 사용하는 하나의 기술이다. 해당 테이블의 열을 색인화(따로 파일로 저장)하여 검색 시 해당 테이블의 레코드 전체를 스캔하는 것이 아니라 색인화된 인덱스 파일을 검색해 검색속도를 빠르게 한다. 인덱스의 구조로는 트리를 사용한다.

13.1. clustered vs non-clustered index

* 클러스터드 인덱스와 넌클러스터드 인덱스: https://m.blog.naver.com/islove8587/220431192221