Normalization

데이터베이스 정규화란?

- 관계형 데이터베이스의 설계 단계에서 데이터의 중복을 최소화하기 위해 데이터의 구조를 결정하는 작업을 정규화(Normalization)라고 한다.
- 일반적으로 데이터베이스 정규화 작업에서는 크고 제대로 조직되지 않은 테이블들과 테이블 간의 관계를 작고 잘 조직된 것으로 변경한다.
- 이렇게 중복된 데이터를 허용하지 않음으로써 무결성을 유지할 수 있다.

제 1 정규화 (1NF)

- 테이블에 존재하는 필드가 모두 scalar value만을 가지고, 필드의 값이 모두 atomic할 때, 1NF라고 한다.
  - atomic하다는 것은 테이블에 중복되는 항목이 존재하지 않는 것이다.
- 1NF에서 “중복되는 항목이 없다”에 대한 정의 는 명확한 것이 아니기 때문에 1NF에 대한 정의 또한 여러 개가 존재할 수 있다.

Table: SUPPLIERS

S#	STATUS	CITY	P#	QTY
S1	10	Seoul	P1	300
S1	10	Seoul	P2	200
S1	10	Seoul	P4	300
S2	20	Busan	P1	100
S2	20	Busan	P2	100
S3	20	Busan	P2	300
S4	10	Seoul	P2	200
S4	10	Seoul	P4	400

[그림 1] 예제 테이블 SUPPLIERS

위의 [그림 1]에 있는 테이블 SUPPLIERS는 {S#, P#}을 primary key로 가지며, 1NF의 정의를 만족하지 않는 테이블을 보여준다. 1NF의 정의를 만족하지 않기 때문에 아래의 그[그림 2]에 표시된 것과 같이 서로 다른 primary key를 갖는 두 레코드가 같은 값을 가지고 있다.

Table: SUPPLIERS

S#	STATUS	CITY	P#	QTY
S1	10	Seoul	P1	300
S1	10	Seoul	P2	200
S1	10	Seoul	P4	300
S2	20	Busan	P1	100
S2	20	Busan	P2	100
S3	20	Busan	P2	300
S4	10	Seoul	P2	200
S4	10	Seoul	P4	400

[그림 2] 중복된 레코드를 포함하는 테이블

위의 [그림 2]와 같은 테이블에서는 INSERT, DELETE, UPDATE 시에 이상 현상이 나타날 수 있다.

1) INSERT anomaly

만약, [그림 2]의 테이블에 S5라는 공급자가 London에 있다는 정보를 저장하고 싶다면, primary key의 구성 요소인 P#에 해당하는 필드에 dirty data를 입력해야 한다.

- dirty date : 의미 없는 데이터를 의미한다.

2) DELETE anomaly

테이블에서 S3라는 공급자가 더는 P2를 공급하지 않게 되어, S#이 S3에 해당되는 레코드를 삭제하면 S3라는 공급자가 Busan에 있다는 정보까지 손실된다.

3) UPDATE anomaly

테이블에서 S1이라는 공급자가 있는 도시를 Seoul에서 Paris로 변경하려면 총 3번의 update 연산이 필요하다. 즉, 하나의 정보를 변경하기 위해 불필요한 update 연산이 2번 더 실행되어야 한다.

[그림 1]의 테이블을 {S#, STATUS, CITY}, {S#, P#, QTY}로 이루어진 두 개의 테이블로 분해하여 1NF의 정의를 만족하는 새로운 테이블로 변경할 수 있다. 1NF의 정의를 만족하도록 분해한 새로운 테이블에서는 위에서 예시로 들은 anomaly가 발생하지 않는다.

제 2 정규화 (2NF)

- 2NF는 1NF의 속성을 만족하면서, 테이블에 존재하는 모든 함수 종속 관계가 **완전 함수 종속(Full Functional Dependency)**이어야 한다. 2NF를 정의할 때 이용되는 완전 함수 종속의 정의는 다음과 같다.

> 💡 *어떤 테이블 R의 필드 Y가 필드의 집합 X에 함수 종속이면서, X 자신을 제외한 X의 어떤 부분 집합에도 함수 종속이 아니면, Y는 X에 완전 함수 종속이라고 한다.*

- 예를 들어, [그림 1]의 테이블에서 primary key는 {S#, P#}이지만, CITY라는 필드는 {S#, P#}이 아니라 S#에 대해 함수 종속적이다. 따라서, [그림 1]의 테이블에는 완전 함수 종속이 아닌 함수 종속 관계가 존재하므로 2NF가 아니다. 위의 [그림 1]의 테이블을 2NF의 정의를 만족하도록 변형하면, 아래의 [그림 3]과 같이 SSC와 SPQ로 표현되는 두 개의 테이블로 분해할 수 있다.

Table: SSC

S#	STATUS	CITY
S1	10	Seoul
S2	20	Busan
S3	20	Busan
S4	10	Seoul

Table: SPQ

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P4	300
S2	P1	100
S2	P2	100
S3	P2	300
S4	P2	200
S4	P4	400

[그림 3] 2NF의 정의를 만족하는 테이블

그러나 2NF의 정의를 만족하는 SSC와 SPQ 테이블에서도 다음과 같은 이상 현상이 발생한다.

1) INSERT anomaly

SSC 테이블에는 “Paris라는 CITY의 STATUS는 40이다”라는 정보를 저장할 수 없다. SSC 테이블의 primary key는 S#이기 때문에 어떠한 도시의 상태에 대한 정보를 SSC 테이블에 저장할 수 없다. 또는, 해당 정보를 저장하기 위해서 S#에 dirty data를 추가해야 한다.

2) DELETE anomaly

만약, SSC 테이블에서 두 공급자 S2와 S3가 제거되면, “Busan이라는 CITY의 STATUS는 20이다”라는 정보도 같이 손실된다.

3) UPDATE anomaly

만약, Seoul이라는 CITY의 STATUS가 10에서 30으로 변경되면, SSC 테이블에서는 Seoul의 STATUS를 30으로 변경하는 작업을 1번이 아니라, 총 2번 수행해야 한다.

제 3 정규화 (3NF)

- 2NF의 정의를 만족하면서, 어떠한 테이블에 존재하는 key가 아닌 필드들이 서로 독립적일 때 3NF라고 한다.
- 예를 들어, [그림 3]의 SSC 테이블에서 STATUS 필드는 테이블의 key가 아닌 CITY 필드에 함수 종속적이기 때문에 SSC 테이블은 3NF의 정의를 만족하지 못한다. 3NF의 정의를 만족하도록 [그림 3]의 SSC 테이블을 분해하면, 아래의 [그림 4]와 같다.

Table: SC

S#	CITY
S1	Seoul
S2	Busan
S3	Busan
S4	Seoul

Table: CS

CITY	STATUS
Seoul	10
Busan	20

[그림 4] 3NF의 정의를 만족하도록 분해된 SSC 테이블

위의 [그림 3]의 SSC 테이블에서는 key가 아닌 CITY에 STATUS가 종속적이었지만, [그림 4]의 SC와 CS 테이블에서는 key가 아닌 필드 사이에 어떠한 종속 관계도 존재하지 않는 것을 볼 수 있다.

Boyce-Codd Normal Form (BCNF)

어떠한 테이블에 대해 테이블에 존재하는 모든 함수 종속 관계의 determinant가 candidate key이면, BCNF라고 한다.

Table: SCT

STD#	COURSE	TEACHER
S1	D100	P1
S1	C101	P2
S2	D100	P1
S2	A205	P3
S3	A205	P3
S4	B401	P4

[그림 5] 예제 테이블 SCT

위의 [그림 5]는 BCNF를 만족하지 않는 테이블을 보여준다. [그림 5]의 테이블에서 candidate key는 {STD#, COURSE}이다. 따라서, {STD#, COURSE}를 통해 하나의 레코드를 유일하게 구별할 수 있다.

그러나 테이블 SCT에는 TEACHER에 의해 COURSE가 결정되는 함수 종속 관계가 존재한다. 즉, determinant가 candidate key에 해당하지 않는 함수 종속 관계가 존재하기 때문에 테이블 SCT는 BCNF의 정의를 만족하지 않는다. 이러한 SCT를 BCNF의 정의를 만족하도록 변경하면, 테이블 SCT는 아래의 [그림6]과 같은 두 개의 테이블로 변환된다.

Table: ST

STD#	TEACHER
S1	P1
S1	P2
S2	P1
S2	P3
S3	P3
S4	P4

Table: TC

TEACHER	COURSE
P1	D100
P2	C101
P3	A205
P4	B401

[그림 6] BCNF의 정의를 만족하도록 변경된 테이블 ST와 TC

위의 [그림 6]에서 함수 종속 관계는 테이블 TC에서만 존재하며, deteminant에 해당하는 것은 TEACHER로써 테이블 TC의 candidate key에 해당한다. 따라서, [그림 6]의 두 테이블은 BCNF의 정의를 만족한다.

Fourth Normal Form (4NF)

4NF는 MVD라는 개념을 통해 정의된다. MVD는 functional dependency의 일반화된 개념으로써, functional dependency에서는 determinant X에 의해 Y의 값이 하나만 결정되었다면, MVD에서는 determinant X에 의해 다수의 Y값이 결정된다. 위와 같은 MVD 관계는 X→→Y로 표기한다.

MVD는 trivial MVD와 nontrivial MVD로 구분된다. 각각의 정의는 아래와 같다. 또한, 이 글에서 4NF를 정의하기 위해 언급되는 MVD는 모두 아래의 nontrivial MVD를 의미한다.

만약, 어떠한 테이블 R에 존재하는 필드 A와 B가 A→→B이면, R에 존재하는 다른 모든 필드가 A에 의해 functional dependent하게 값이 결정될 때, 4NF라고 한다. 예를 들어, 테이블 R(A, B, C, D)에 대해 A→→B일 때, A→ C이고 A→D이면 R은 4NF의 정의를 만족한다고 할 수 있다.

아래의 [그림 7]과 같이 COURSE와 MVD 관계를 갖는 두 개의 필드를 포함하는 정보가 있다. 두 개의 필드는 각각 COURSE 필드의 값에 의해 값이 결정되며, 하나의 COURSE 값에 대해 다수의 TEACHER와 TEXT 값이 존재할 수 있다.

COURSE	TEACHER	TEXT
File Structure	Prof. Kim	File Structure Concepts
	Prof. Lee	Basic Data Structure
Data Structure	Prof. Kim	Basic Data Structure
		Data Structure & Algorithms
		Data Structure Concepts

[그림 7] MVD 관계를 포함하는 데이터

위의 [그림 7]의 데이터를 정규화하여 나타내면 아래의 [그림 8]과 같으며, [그림 8]의 테이블 CTT는 BCNF의 정의를 만족한다. 그러나 COURSE와 TEACHER가 MVD 관계이면서, COURSE와 TEXT 또한 MVD 관계이기 때문에 4NF로 정규화된 테이블은 아니다.

Table: CTT

COURSE	TEACHER	TEXT
File Structure	Prof. Kim	File Structure concepts
File Structure	Prof. Kim	Basic Data Structure
File Structure	Prof. Lee	File Structure concepts
File Structure	Prof. Lee	Basic Data Structure
Data Structure	Prof. Kim	Basic Data Structure
Data Structure	Prof. Kim	Data Structure & Algorithms
Data Structure	Prof. Kim	Data Structure Concepts

[그림 8] BCNF로 정규화된 테이블

위의 [그림 8]의 테이블 CTT에서 Data Structure를 가르치는 사람이 변경되면, 하나가 아니라, 총 세 개의 레코드를 변경해야 하는 UPDATE anomaly가 발생한다. 이러한 UPDATE anomaly를 해결하기 위해서는 [그림 8]의 테이블을 4NF의 정의를 만족하도록 변경하는 것이 필요하다. 테이블 CTT를 4NF의 정의에 맞도록 분할하면, [그림 9]의 두 테이블, CT1과 CT2로 변경된다.

Table: CT1

COURSE	TEACHER
File Structure	Prof. Kim
File Structure	Prof. Lee
Data Structure	Prof. Kim

Table: CT2

COURSE	TEXT
File Structure	File Structure concepts
File Structure	Basic Data Structure
Data Structure	Basic Data Structure
Data Structure	Data Structure & Algorithms
Data Structure	Data Structure Concepts

[그림 9] 4NF의 정의를 만족하도록 분할된 테이블

위의 [그림 8]의 테이블 CTT를 [그림 9]의 두 테이블 CT1과 CT2로 분할하면, Data Structure를 가르치는 사람을 변경할 때, 더는 UPDATE anomaly가 발생하지 않는다.