Normalizacija

Normalizacija se vrši radi eliminisanja redundantnih(ponavljajućih) podataka, pojednostavljivanja upita(query), i da se izbjegnu anomalije pri manipulisanju podacima. Normalizacija se sastoji od refaktorisanja tabela u manje tabele bez gubitka informacija, definisanja stranih ključeva(FK) u pogodnim tabelama čime referenciramo primarni ključ(PK) u drugim tabelama. Cilj je olakšati dodavanje, brisanje i mijenjanje atributa tako da se to radi na samo jednom mjestu.

Sve navedeno postižemo dovođenjem strukture baze podataka u oblik koji podliježe prvoj, drugoj i trećoj normalnoj formi.

Prva normalna forma(1NF)

1. Moraju se definisati podaci, tj. **kolone** tabela u koje će biti smještani podaci
2. Ne smije biti **ponavljanja podataka**

Razmotrimo sljedeću tabelu, CUSTOMERS:

ID	NAME	AGE	ADDRESS	ORDERS
100	Sachin	36	Lower West Side	Cannon XL-200
100	Sachin	36	Lower West Side	Battery XL-200
100	Sachin	36	Lower West Side	Tripod Large

Vidimo da u tabeli imamo 3 puta iste podatke, samo zato što je isti korisnik naručio više artikala. Naravno, logično je da trebamo napraviti novu tabelu, ORDERS u koju ćemo smještati narudžbe i referencirati customer-a preko njegovog ID primarnog ključa.

Prethodna tabela bi izgledala ovako:

ID	NAME	AGE	ADDRESS
100	Sachin	36	Lower West Side

a nova tabela, CUSTOMERS ovako:

ID	CUSTOMER_ID	ORDERS
10	100	Cannon XL-200
11	100	Battery XL-200
12	100	Tripod Large

3. Mora se izabrati **primarni ključ** za svaku tabelu koju smo definisali

Druga normalna forma(2NF)

Druga normalna forma mora prvo ispoštovati pravila prve normalne forme.
U 2NF sve kolone u tabeli moraju biti **funkcionalno zavisne od PRIMARNOG KLJUČA**.
2NF se najčešće odnosi na tabele sa **kompozitnim primarnim ključem**.

Razmotrimo relaciju između tabela customers i orders:

```
CREATE TABLE CUSTOMERS(
       CUST_ID    INT              NOT NULL,
       CUST_NAME VARCHAR (20)      NOT NULL,
       ORDER_ID   INT              NOT NULL,
       ORDER_DETAIL VARCHAR (20)  NOT NULL,
       SALE_DATE  DATETIME,
       PRIMARY KEY (CUST_ID, ORDER_ID)
);
```

Ova tabela je u 1NF, a njen primarni ključ se sastoji od CUST_ID i ORDER_ID.
Kombinovani, oni su jedinstveni, pretpostavljajući da isti kupac neće naručiti istu stvar više puta.

Međutim, ova tabela nije u 2NF jer postoje parcijalne zavisnosti između primarnih ključeva i kolona.

- CUST_NAME zavisi od CUST_ID, ali oni nemaju veze sa narudžbom
- ORDER_DETAIL zavisi od ORDER_ID, ali ne i od CUST_ID
- SALE_DATE ne zavisi ni od same narudžbe ni od naručioca

Da bi ova tabela bila u 2NF, moraćemo razdvojiti njene kolone u 3 tabele.
Time ćemo ustvari napraviti vezu n naprema m.

```
CREATE TABLE CUSTOMERS(
       CUST_ID    INT              NOT NULL,
       CUST_NAME VARCHAR (20)      NOT NULL,
       PRIMARY KEY (CUST_ID)
);

CREATE TABLE ORDERS(
       ORDER_ID   INT              NOT NULL,
       ORDER_DETAIL VARCHAR (20)  NOT NULL,
       PRIMARY KEY (ORDER_ID)
);

CREATE TABLE CUSTMERSORDERS(
       CUST_ID    INT              NOT NULL,
       ORDER_ID   INT              NOT NULL,
       SALE_DATE  DATETIME,
       PRIMARY KEY (CUST_ID, ORDER_ID)
);
```

Ovako smo napravili nezavisne naručioce i nezavisne narudžbe.
Oni su povezani samo u customersorders tabeli, koja predstavlja spregu.
Ona također sadrži i dodatni atribut, SALE_DATE koji zavisi samo od date narudžbe.

Treća normalna forma(3NF)

Treća normalna forma mora prvo ispoštovati pravila druge normalne forme.
Sva polja moraju biti zavisna **direktno od primarnog ključa**.

Primjer:

```
CREATE TABLE CUSTOMERS(
       CUST_ID        INT              NOT NULL,
       CUST_NAME      VARCHAR (20)      NOT NULL,
       DOB            DATE,
       STREET         VARCHAR(200),
       CITY           VARCHAR(100),
       STATE          VARCHAR(100),
       ZIP            VARCHAR(12),
       EMAIL_ID       VARCHAR(256),
       PRIMARY KEY (CUST_ID)
);
```

U prethodno deklarisanoj tabeli polja STREET, CITY, STATE su direktno zavisna od polja ZIP(pošt. broj).

Da bi ispunili zahtjeve 3NF potrebno je polja STREET, CITY, STATE, ZIP prebaciti u novu tabelu:

```
CREATE TABLE ADDRESS(
       ZIP            VARCHAR(12),
       STREET         VARCHAR(200),
       CITY           VARCHAR(100),
       STATE          VARCHAR(100),
       PRIMARY KEY (ZIP)
);
```

Tabela CUSTOMERS se transformiše u sljedeću:

```
CREATE TABLE CUSTOMERS(
       CUST_ID        INT              NOT NULL,
       CUST_NAME      VARCHAR (20)      NOT NULL,
       DOB            DATE,
       ZIP            VARCHAR(12),
       EMAIL_ID       VARCHAR(256),
       PRIMARY KEY (CUST_ID)
);
```

Ovako je smanjena redundancija podataka i poboljšan je integritet podataka.
Kada imamo duplicirane podatke u tabeli, npr. adrese u prvoj tabeli, svaka izmjena je vrlo teška i mukotrpna za razliku od ove nove tabele gdje izmjenu vršimo samo na jednom mjestu.