Normalne forme

Normalizacija je struktuiranje baze podataka kroz koje se ponavljanje podataka i anomalije kod pisanja/azuriranja/brisanja podatka svode na minimum. U teoriji postoji 6+ normalnih formi, a u praksi se najcesce primjenjuju prve 3 (ili kada se dostigne Boyce-Codd-ova normalna forma, kazemo da je baza normalizovana, tj. da nije neophodno ici sa ostalim normalnim formama).

Postoji vise razlicitih definicija normalnih formi, a mi cemo ici sa onim koje su najlakse za razumjeti:

1NF - ako svako polje u sebi sadrzi samo jednu vrijednost i ako je svaki red jedinstven.
Na primjer:

PRIJE 1NF:

ID	Name	Address	ZipCode	Phone	FilmsRented
1	Milos Nikolic	Nikole Tesle 1a	76300	066012347	Pirati sa Kariba 1
2	Stefan Rakic	Vuka Karadizca 17/3b	76300	065123123	Avatar, Odiseja u svemiru 2001
3	Slavica Popovic	Mihaila Pupina 7	76100	055001056	Sam u kuci 2

NAKON 1NF:

ID	Name	Address	ZipCode	Phone	FilmsRented
1	Milos Nikolic	Nikole Tesle 1a	76300	066012347	Pirati sa Kariba 1
2	Stefan Rakic	Vuka Karadizca 17/3b	76300	065123123	Avatar
3	Stefan Rakic	Vuka Karadizca 17/3b	76300	065123123	Odiseja u svemiru 2001
4	Slavica Popovic	Mihaila Pupina 7	76100	055001056	Sam u kuci 2

2NF: **Tabela je u 2NF ako je u 1NF i ako su svi podaci u njoj potpuno zavisni od primarnog kljuca, a ne od drugih parametara. Prethodna tabela (1NF) ima ponavljanja koja zavise od FilmsRented. Da bismo ovu tablu doveli u 2NF moramo je "razbiti" na vise tabela:**

NAKON 2NF

ID	Name	Address	ZipCode	Phone
1	Milos Nikolic	Nikole Tesle 1a	76300	066012347
2	Stefan Rakic	Vuka Karadizca 17/3b	76300	065123123
3	Slavica Popovic	Mihaila Pupina 7	76100	055001056

MemberID	Film
1	Pirati sa Kariba 1
2	Avatar
2	Odiseja u svemiru 2001
3	Sam u kuci 2

3NF: Tabela mora biti u 1NF i 2NF i ne smije sadrzati "tranzitivne" duplikate. Koristi se da bi se eliminisala redundantnost i sprijecile anomalije (narocito kod brisanja i azuriranja). Na primjeru prethodne tabele mozemo vidjeti da se postanski broj ponavlja vise puta. Ovo bi trebalo eliminisati kako bismo nase tabele doveli u 3NF:

NAKON 3NF:

ID	Name	Address	ZipCodeID	Phone
1	Milos Nikolic	Nikole Tesle 1a	1	066012347
2	Stefan Rakic	Vuka Karadizca 17/3b	1	065123123
3	Slavica Popovic	Mihaila Pupina 7	2	055001056

ID	ZipCode
1	76300
2	76100

Boyce-Codd-va NF je jos striktnija varijanta 3NF.

Primarni kljucevi (prirodni i surogat)

O primarnim kljucevima smo vec pricali. Tipican primjer je sekvencijalni kljuc numerickog tipa koji se automatski generise - ovo je surogat kljuc. Pored ovakvog nacina kreiranja, kljucevi mogu biti UUID, na primjer: **123e4567-e89b-12d3-a456-426614174000.** Ovakvi kljucevi se takodje automatski generisu, ali nisu sekvencijalni, sto ih cini donekle sigurnijim u odnosu na sekvencijalne.

Prirodni kljucevi su oni koji po svojoj prirodi su kandidati (jedinstveni su za svaki zapis). To su, na primjer: JMBG, ime i prezime + adresa i slicno. Za poslednih primjer takodje kazemo da je kompozitni, tj. sastavljen je iz vise vrijednosti iz odredjene tabele.

SQL Views

SQL views su virtueleni setovi podataka koji ne sadrze starne vrijednosti, nego reference ka postojecim vrijednostima u postojecim tabelama. Na primjer:

```
CREATE VIEW MyView AS
SELECT co1, col2, col3
FROM table1;
```

Ovo ce kreirati view koji sadrzi samo 3 kolone iz neke tabele. Medjutim, ovo nije prava moc view-a. Prava prakticna vrijednost dolazi iz mogucnosti da se uzmu podaci iz vise tabela i spoje u jedan view - ovo ce ustedjeti resurse i doprinijeti performansama u poredjenju sa konstantnim upitima sa join-ovima - dakle, sa view-ovima mozemo napraviti denormalizaciju, koja je korisna kada se citaju podaci iz vise tabela. Vazno je napomenuti da se view automatski regenerise kada se dodaju novi podaci u povezane tabele - ovo ne moramo rucno raditi.

Stored procedures

Procedure su pripremljeni SQL kod koji se moze snimiti u SQL endzinu i pokrenuti kada nam je potreban. Na primjer:

```
CREATE PROCEDURE MyProcedure
BEGIN
SELECT * FROM users;
END;
```

Proceduru izvrsavamo koristeci kljucnu rijec EXEC:

```
CALL MyProcedure;
```

Procedure mogu imati parametre. Na primjer:

```
CREATE PROCEDURE getUsersByType (IN userType varchar(30))
BEGIN
SELECT * FROM user WHERE role = userType;
END

CALL getUsersByType('admin');
```

Triggers

Trigeri su koji se moze koristiti kako bi izvrsio neku akciju prije ili posle neke operacije u bazi/tabeli. Na primjer:

```
CREATE TRIGGER logUserChange
    AFTER UPDATE ON user
    FOR EACH ROW
BEGIN
 INSERT INTO userLog
 SET action = 'update',
     userId = OLD.id,
     type = OLD.role,
     date = NOW();
END
```

Trigere mozemo podesiti da se okidaju na:

- BEFORE | AFTER, praceni sa
- INSERT | UPDATE | DELETE

BEFORE i AFTER se mogu naci u bilo kojoj kombinaciji sa INSERT, UPDATE i DELETE.

Kompletna mysql dokumentacija: https://dev.mysql.com/doc

SQL cheatsheet: https://www.sqltutorial.org/wp-content/uploads/2016/04/SQL-cheat-sheet.pdf

SQL reference sa mogucnoscu testiranja: https://www.sqltutorial.org

Zadatak 1:

Napisati SQL upit kodi vraca sve zapise iz tabele warrants gdje je user_id = 4, a koji nisu izbrisani.

SELECT * FROM warrants WHERE user_id=4 AND deleted_at IS NULL

Zadatak 2:

Nadograditi prethodni upit tako da dodaje (join) zapise iz warrant_containers gdje je status razlicit od draft.

```
SELECT * FROM warrants AS war
LEFT JOIN warrant_containers AS wc ON wc.warrant_id = war.id
WHERE war.user_id = 4 AND  war.deleted_at IS NULL
AND wc.status <> 'draft';
```

Zadatak 3:

Napisati upit koji vraca broj utrosenih sati po korisniku iz tabele warrant_containters.

SELECT CONCAT(users.first_name, " " ,users.last_name) AS user, SUM(time_spent) FROM warrant_containers LEFT JOIN users ON warrant_containers.user_id = users.id GROUP BY user_id

Zadatak 4:

Napisati upit koji racuna ukupan promet za received_at_location = 2 i gdje je is_paid true ~~i grupisati po godina-mesec (iskljuciti izbrisane naloge).~~

SELECT SUM(wc.price * wc.quantity) AS total FROM warrant_containers AS wc LEFT JOIN warrants w ON wc.warrant_id = w.id WHERE w.is_paid = 1 AND w.received_at_location = 2

Zadatak 5:

Napisati upit koji racuna koji je korisnik (user_id) imao najvise zavrsenih proizvoda (tabela warrant_containers) i sloziti ih od najveceg ka najmanjem.

SELECT users.id, COUNT (warrant_containers.id) FROM users LEFT JOIN warrant_containers ON users.id = warrant_containers.user_id GROUP BY users.id ORDER BY COUNT (warrant_containers.id) DESC;

Zadatak 6:

Napisati SQL upit koji kreira tabelu customers (id, name, address, city, phone, created_at, deleted_at)
CREATE TABLE customers (id INT(11) PRIMARY KEY, name VARCHAR(255) NOT NULL, address VARCHAR(255) NOT NULL, city VARCHAR(255) NOT NULL, phone VARCHAR(20) NOT NULL, created_at DATETIME NOT NULL, deleted_at DATETIME DEFAULT NULL)

Zadatak 7:

Napisati upit koji kreira tabelu todos_backup, zatim upit koji kopira sadrzaj iz todos u todos_backup

CREATE TABLE todos_backup SELECT * FROM todos;

Zadatak 8:

Napisati funkciju koja asinhrono konzumira https://api.mathjs.org i prosledjuje parametar expression navedenom API-ju, te ispisuje:

Status: SUCCESS
Value: [value]
Expression: [input_string]

ako je sve proslo uredno.

Ispisati:

Status: FAILURE
Value: N\A
Expression: [input_string]
Reason: [error]

ako je API vratio gresku ili ukoliko interna validacija izraza ne prodje uredno. Validacija bi trebala da pokrije bar sledece (pod pretpostavkom da se rade samo osnovne matematicke operacije +, -, * i /):

- izraz ne smije sadrzati nedozvoljene karaktere,
- zagrade moraju biti uredno otvorene i zatvorene.

```
Async function mathSolver(input) {
        const obj = {
                Status: “FAILURE”,
                Value: Nan,
                Expresion: input.expr
        };
        const mathjs = "http://api.mathjs.org/v4/";

Let invalid = input.expr.filter((element) => {
        if(element.match(/[0-9.*/+\-\(\)]/g).length !== element.length){
        obj.Reason = "Izraz ne smije sadrzati nedozvoljene karaktere.";
        return true;
        }

        if(count(“(”, element) !== count(“)”,element)){
        obj.Reason = "Zagrade moraju biti uredno otvorene i zatvorene";
        return true;
        }
})
If (invalid[0] === true){
        return obj;
}
obj.Value = await getResult(mathjs, input, obj);
obj.Value = obj.Value === undefined ? NaN : obj.Value;

return obj;
}
// “()(()((()))))(”
```

```
function count(char,str){
	let count = 0;
	for(let i = 0; i < str.length; i++){
		if(str[i] === char){
			count++;
		}
	}
	return count;
}
async function getResult(url, data , obj) {
	try{
	const response = axios.post(url,data,{
	headers: {
	"Accept-Encoding": "gzip,deflate,compress",
}
});
	obj.Status = “SUCCESS”;
	return response.data.result;
} catch(err){
	obj.Reason = err.response.data;
}
}
```

Za one koji bi vise:

- https://leetcode.com
- https://www.codewars.com

Podijelite koje zadatke ste rijesili i ako/gdje ste naisli na prepreku. Srecni praznici :)