Loeng2. Korrastatud andmestruktuurid. Normaliseerimine.

Mõisted eelmisest korrast - ***infoobjekt, tabel, veerg(column), rida (row), kirje(record), primaarvõti(primary key), välisvõti (foreign key), relatsioon (relation), üks-mitu suhe (one-to-many relationship)***

- Relatsioonilises andmebaasis kujutab tabeli iga lahtri sisu (*row - column entry*) endast

atomaarset andmeelementi, so. igasse lahtrisse saab salvestada ainult ühe andmeelemendi ning mitut andmeelementi(nn. korduvaid v. lisaandmeelemente) ei ole võimalik salvestada ühte lahtrisse.

Andmestruktuure, millest on kõrvaldatud korduvad andmeelemendid , nimetatakse *normaliseeritud andmestruktuurideks.* Relatsioonilistes andmebaasides , selleks et andmeid organiseerida tabelite vormis, peavad andmestruktuurid olema normaliseeritud .

Tootaja _nr [pr. key]	**isikukood**	**nimi**	**aadress**	**ameti _nr [pr. key]**	**ameti_nimetus**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34	1	programmeerija
2	45586790	K.Jõgi	Paide, Roheline 7	2	projekti juht
3	47656896	N.Mets	Tallinn, Pirita tee 56	3	süsteemianalüütik

1	programmeerija
6	andmebaasi administraator

Igale töötajale võib vastata mitu ametit.

joonis 3.1. Korduvad andmeelemendid , normaliseerimata andmestruktuur, pole võimalik paigutada relatsioonilisse andmebaasi.

- Normaliseerimine, esimene normaalkuju (*first normal form*)

Andmestruktuuride korrastamist nii, et puuduksid korduvad andmeelemendid, nimetatakse andmestruktuuride viimiseks esimesele normaalkujule e. normaliseerimiseks. Võimalikud on ka edasised , kõrgemad andmete normaliseerimistasemed - teine, kolmas, neljas ja viies normaalkuju. Antud kursuses on käsitletud kolme esimest.

Andete normaliseerimisprotsess algab andmete viimisega esimesele normaalkujule.Kui andmeid on võimalik esitada tabeli kujul, siis on need andmed vähemalt esimesel normaalkujul, iga relatsioonilise andmebaasi tabel on automaatselt esimesel normaalkujul.

Üldiselt on soovitav organieerida andmed relatsioonilises baasis vähemalt kolme esimese normaalkuju nõuetele vastavalt. Relatsiooniline andmemudel ise tingib küll vaid esimese normaalkuju kasutamise (st. selleks, et andmeid kande relatsioonilisse baasi piisab sellest, et andmed oleksid esimesel normaalkujul).

Korduvaid andmelemente saab likvideerida, luues iga sellise korduva andmeelemendi jaoks eraldi kirje . Joonisel 3.2 on sellisteks lisakirjeteks kaks viimast. Viies andmetabeli sellisele kujule tekib aga andmete liiasus (*data redundancy*). Andmete liiasus on üheks põhjuseks, miks järgmised normaliseerimised on vajalikud. Andmete liiasus põhjustab andmebaasi ruumi raiskamist ja suurendab võimalust, et andmete terviklikkus baasis võib saada rikutud.

Tootaja _nr [pr. key]	**isikukood**	**nimi**	**aadress**	**Ameti _nr [pr.key]**	**ameti_nimetus**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34	1	programmeerija
2	45586790	K.Jõgi	Paide, Roheline 7	2	projekti juht
3	47656896	N.Mets	Tallinn, Pirita tee 56	3	süsteemianalüütik
3	47656896	N.Mets	Tallinn, Pirita tee 56	1	programmeerija
3	47656896	N.Mets	Tallinn, Pirita tee 56	6	andmebaasi administraator

Joonis 3.2 Normaliseeritud andmestruktuur, esimene normaalkuju

- **Teine normaalkuju, funktsionaalne sõltuvus (*functional dependency*).**

Andmetabelites on tihti veerg A funktsionaalses sõltuvuses veerust B, st. veeru B igale väärtusele vastab mingi väärtus veerus A. Veerg A identifitseerib veeru B. Joonisel 3.2 on näiteks kolm ride, milles tootaja_nr on 3, kõigis nendes ridades on isikukood, nimi ja aadres võrsed. **tootaja_nr** identifitseerib siin kolme ülejäänud veergu. Veerg võib funktsionaalselt sõltuda ka korraga mitmest veerust.
Andmete paigutamine teise normaalkuju tähendab kõigi veergude omavahelise funktsionaalse sõltuvuse väljaselgitamist ning eraldi tabelit loomist nende sõltuvuste jaoks, st. ühe tabeli “lõhkumist” mitmeks väiksemaks.

Esialgu tuleks identifitseerida võimalikud võtmed ja vaadata, millised veerud nn.”suurest tabelist” sellest võtmest sõltuvad ning eraldada välja eraldi tabelid. Joonisel 3.2 on üks võimalikke võtmeid ilmselt **tootaja_nr**, millest sõltuvad **isikukood**, **nimi** ja **aadress**

Tootaja _nr [pr. key]	**isikukood**	**nimi**	**aadress**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34
2	45586790	K.Jõgi	Paide, Roheline 7
3	47656896	N.Mets	Tallinn, Pirita tee 56

joonis 3.3. Tabel “TOOTAJA”

Pärast esimese tabeli eraldamist järale jäänud veerg **ameti_nr** sõltub funktsionaalselt **tootaja_nr**-st (igal töötajal on oma ametid). **ameti_nimetus** sõltub omakorda **ameti_nr**-st.Nii tekib eraldi kolmeveeruline tabel, milles hoitakse töötajatele vastavaid ameteid.

tootaja_nr [pr. key]	**ameti_ nr[pr. key]**	**ameti_nimetus**
1	1	programmeerija
2	2	projekti juht
3	3	süsteemianalüütik
3	1	programmeerija
3	6	andmebaasi administraator

joonis 3.4. Tabel “AMET”

Korduvad veerud (*duplicated columns*) **tootaja_nr** on salvestatud nüüd korraga kahte tabelisse. Selline andmeveergude kordamine on vajalik, et säilitada tabelites sisalduvate andmete õigsust peale teisele normaalkujule viimist. **tootaja_nr** “AMETI” tabelis loob informatsiooni selle kohta , millised ametid vastavad igale töötajale, nii säilib esimeses tabelis (joonis 3.2) toodud informatsioon. Tekib relatsioon kahe tabeli vahel, relatsiooniliste andmebaaside üks põhimõisteid.

tootaja_nr [pr. key]	**isikukood**	**nimi**	**aadress**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34
2	45586790	K.Jõgi	Paide, Roheline 7
3	47656896	N.Mets	Tallinn, Pirita tee 56

relatsioon (loodud duplikaatveeru abil "AMETI" tabelis)

tootaja_nr [pr. key]	**ameti_nr[pr. key]**	**ameti_nimetus**
1	1	programmeerija
2	2	projekti juht
3	3	süsteemianalüütik
3	1	programmeerija
3	6	andmebaasi administraator

Joonis 3.5 Andmed teisel normaalkujul.

NB! Relatsioonide määramine võimaldab hiljem kasutada relatsioonilisi operaatoreid, et kombineerida tabelite andmeid kokku erinevates päringutes.

Teise normaalkuju plussid:

- andmete liiasuse vähendamine
- loodud relatsioonid andmetabelite vahel võimaldavad säilitada andmete terviklikkust , n. pole võimalik lisada uut ametit töötajale, mida ei eksisteeri "TOOTAJA" tabelis. (sõltuvalt andmebaasisüsteemist võidakse see kindlustada programselt rakendusprogrammide poolt või uuemate süsteemide korral automaatselt andmebaasisüsteemi mehhanismidega)

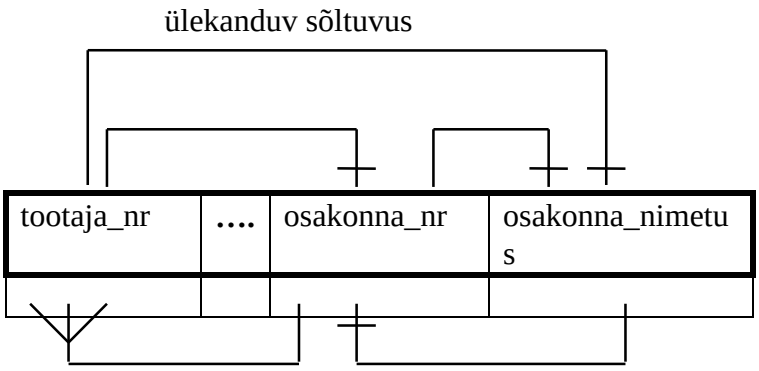
- võime lisada tabelitesse andmeid, ilma teisi tabeleid muutmata, andmeuuendus on suunatud konkreetsematele objektidele. (n. uue töötaja korral uuendatakse andmeid ainult TOOTAJA tabelis.

- **Kolmas normaalkuju .**

- eraldades tabelid teise normaalkuju reeglite järgi võib tekkida võimalus sellest tabelist eraldada veel eraldi tabeleid, juhul kui on tegemist nn. ülekantud sõltuvusega (*transitive dependencies*) St. tekivad järjestikused üks-ühesed sõltuvused veergude vahel - veerg B on üks-üheselt sõltuvuses veerust A, veerg C on omakorda üks-üheses sõltuvuses veerust B. Näiteks

Tootaja_nr [pr. key]	**isikukood**	**nimi**	**aadress**	**osakonna_nr**	**osakonna_nimetus**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34	1	plaaniosakond
2	45586790	K.Jõgi	Paide, Roheline 7	2	arendusosakond
3	47656896	N.Mets	Tallinn, Pirita tee 56	2	arendusosakond

osakonna_nr on sõltuvuses **tootaja_nr**-st, **osakonna_nimetus** on sõltuvuses **osakonna_nr**-st, aga samas ka üks-üheselt sõltuvuses **tootaja_nr**-st.



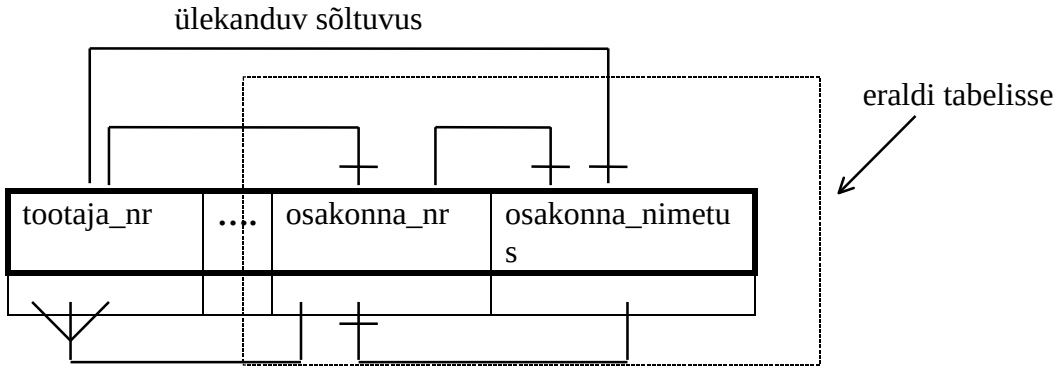
joonis 3.6 Ülekanduvad sõltuvused

Andmete organiseerimine kolmandale normaalkujule tähendab ülekanduvate sõltuvuste likvideerimist ühe tabeli mitmeks tabeliks eraldamise teel, üks-üheselt teineteisest sõltuvad veerud on paigutatud eraldi tabelitesse.

tootaja_nr [pr. key]	**isikukood**	**nimi**	**aadress**	**osakonna_nr**
1	12456789	T.Mägi	Tallinn, Pärnu mnt 34	1
2	45586790	K.Jõgi	Paide, Roheline 7	2
3	47656896	N.Mets	Tallinn, Pirita tee 56	2

osakonna_nr	**osakonna_nimetus**
1	plaaniosakond

2	arendusosakond



Joonis 3.7 Andmed kolmandal normaalkujul

NB! Kolmandal normaalkujul olevas andmetabelis sõltuvad kõik veerud võtmeveerust (primary key) ja ainult võtmeveerust(e. võtmeatribuudist)

Kolmanda normaalkuju plussid:

- paremini kindlustatud andmete õigsus ja terviklikkus kuna väheneb andmete dubleerimine (n. osakonna andmed on nüüd salvestatd ainul ühes tabelis, aga mitte igale töötajale vastavas kirjes)

- **<u>Kokkuvõtteks:</u>**

Andmete organiseerimine esimesele normaalkujule tähendab nende korrastamist nii, et neid oleks võimalik salvestada relatsioonilise andmebaasi tabelitesse , iga andmeelemendi jaoks leidub mingi lahter mingis tabelis.

Andmete viimisel teise normaalkujusse leitakse tabelis veergudevahelised funktsionaalsed sõltuvused ja eraldatekse nende sõltuvuste alusel uued tabelid.

Andmete viimine kolmandale normaalkujule tähendab ülekanduvate sõltuvuste likvideerimist uute tabelite eraldamise teel.