**2.5. Normalizálás**

iDevice ikon

Ebben a fejezetben megfogalmazzuk az első 3 normálformát, mint a redundancia megszüntetésének algoritmizálható lépéseit. Az eljárást egy mintapéldán keresztül mutatjuk be, de a normalizálás készségszintű művelését a képzésben résztvevő hallgatóknak nem kell elérni. Itt azért kell az eljárással megismerkedni, hogy a már - esetleg mások által -megtervezett adatbázist is felismerjük, és képesek legyünk feldolgozni.

iDevice ikon

**Normálforma: az egyed szerkezeti állapota.**

A normálformák egymásra épülnek, azaz normálformában lenni annyit jelent, hogy már az (n-1). formában benne van.

Ha relációs modellben az emberekről a nevet, születési dátumot és a szakképzettségeket kellene tárolni, akkor egyelőre 3-oszlopos táblát képzelünk el, melyben a szakképzettség bizony többértékű tulajdonság lenne.

|  |  |  |
| --- | --- | --- |
| NÉV | SZAKKÉPZETTSÉG | SZÜLETÉSI DÁTUM |
| Nagy Zsolt | gépészmérnök | 52.02.16 |
|  | közgazdász |  |
| Kiss Pál | lakatos | 58.08.08 |

**R 0. normálformában van (0NF, vagy N1NF típusú), ha létezik olyan másodlagos attribútum, amely a kulcstól funkcionálisan független.**

Magyarul a táblázat ismétlődő ismeretet tartalmaz (esetünkben a szakképzettség, mely a név és születési dátum kettőstől nem függ, azaz nem egyértelmű a kulcs alapján).

Gondolhatunk először olyan ostoba megoldásra, mely szerint a szakképzettség tulajdonságot olyan hosszúra (vagy változó hosszúságúra) definiáljuk a tábla létrehozásakor, hogy oda beférjen több szakképzettség is (valamilyen elválasztójelek között). Nos, ilyen esetben nemcsak az a gond, hogy esetleg mégis rövidnek bizonyul, hanem az, hogy nem tudunk benne majd gyorsan keresni. Például, a „Kik a közgazdászok?" kérdésre minden sorban hosszasan kell majd rész-karakterláncokat ellenőrizni, hátha előfordul benne a keresett szó, pedig ha minden sorban csak egy szerepelne, akkor arra érdemes lenne lerendezni, hogy felező módszerrel lehessen benne keresni.

Gondolhatunk még rettenetesebb megoldásra is: mintegy soremeléssel az új sorban csak a következő szakképzettséget töltjük ki az előző sor személyéről. De a személyekről nem egyszerre viszik be az összes szakképzettséget, mert esetleg később szereznek újabbakat, vagyis nem működhet a *lásd előző* módszere. Arról nem beszélve, hogy adatainkat több szempont szerint szeretnénk rendezni, hogy majd gyorskereshessünk. Milyen szörnyű lenne egy név szerinti rendezésben egymás alatt a sok-sok csupa szóköz név, melyek mellett a nem első szakképzettségekről már nem lehetne tudni, hogy kihez tartoznak.

**R 1. normálformájú (1NF típusú), ha minden másodlagos tulajdonság funkcionálisan függ a kulcstól.**

Magyarul a táblázat minden sorában pontosan egy attribútumérték lehet minden oszlopban. (Ez nem azt jelenti, hogy kell: vagyis azokat a tulajdonságokat, amelyek nem kötelezők - mint a hajszín -, nem kell kitölteni.)

**Az előző példa 1NF-ben:**

|  |  |  |
| --- | --- | --- |
| **NÉV** | **SZAKKÉPZETTSÉG** | **SZÜLETÉSI DÁTUM** |
| Nagy Zsolt | gépészmérnök | 52.02.16 |
| Nagy Zsolt | közgazdász | 52.02.16 |
| Kiss Pál | lakatos | 58.08.08 |

**R 2. normálformájú (2NF típusú), ha 1-es normálformában van, és minden másodlagos attribútuma a reláció bármely kulcsától teljesen függ.**

Tehát nincs benne részleges függés.

Az előző fejezetben tisztáztuk, hogy csak azok a tulajdonságok maradhatnak egy táblában, amelyek a kulcstól teljesen függnek, ugyanis a részleges függés redundanciát okoz. Cél, hogy minden tulajdonság olyan táblába kerüljön, amelynek kulcsától teljesen függ.

**Megjegyzések**

* Ha az R kulcsa egyetlen attribútumból áll, akkor máris 2NF típusú. Ugyanis a kulcsnak nincs valódi részhalmaza, amitől függni lehetne.
* Ha nincsen R-ben másodlagos attribútum, akkor úgyis 2NF típusú. Ugyanis nincsenek leírók, amik valamelyike a kulcstól részlegesen függhetne.

**Példa**: a BIZONYLATFEJ és BIZONYLATTÉTEL esete

Általában mindenféle bizonylat, ami több tételt is tartalmazhat, mindig fej- és tétel-adatokra fog bomlani, ugyanis vannak olyan tulajdonságok, amik csak a bizonylat számától függnek teljesen, és vannak azok, amelyek egyértelmű meghatározásához a bizonylatszám mellett legalább a tételsorszámra szükség van. (Rendelés-mintánkban a tételt a rendszám és cikkszám azonosította.)

**R 3. normálformájú (3NF típusú), ha 2-es normálformában van, és egyetlen másodlagos attribútuma sem függ tranzitíven valamely kulcstól.**

Tehát nincs benne tranzitív függés.

Az előző fejezetben tisztáztuk, hogy csak azok a tulajdonságok maradhatnak egy táblában, amelyek egy másik leíró tulajdonságtól sem függnek, ugyanis a tranzitív függés redundanciát okoz. Cél, hogy minden tulajdonság olyan táblába kerüljön, amelynek csak a kulcsától függ.

**Példa**: a BIZONYLATFEJ és PARTNER esete

Általában mindenféle bizonylat pontosan egy partnertől érkezik, és az a partner több bizonylatot is küldhet. A partner sok törzsadata mellett bizonyára kap egy kódot nálunk, aminek valóban be kell kerülni a bizonylatfejbe, de a többi törzsadatnak nem. A partner kódja a partner táblában lesz kulcs, ami mellett a többi törzsadatot tároljuk helyesen.

**Megjegyzések**

1. Nézzünk példát a csupakulcs-esetre:

ÜGYELET {ki, mikor}

Itt a tábla csak kulcsszerepű tulajdonságokat tartalmaz (ki mikor jött be ügyeletet tartani), tehát 3NF-ben van. Valószínűleg a *ki* egy dolgozókódra mutat, a *mikor* pedig egy dátum. Ha naponta több műszak lenne, akkor a műszak sorszámát is tárolni kellene, de kérdés, hogy a kulcs részeként, vagy leíróként. Ha a feladat olyan helyre készül, ahol egy dolgozó egy napon csak egy műszakban ügyelhet, akkor leíróként tároljuk, hányadik műszakban jelent meg, egyébként pedig kulcsszerepben.

2. Nézzünk példát a több kulcsjelölt esetére:

SZÁMLATÉTEL {*szlaszám*, sorszám, *cikkszám*, mennyi}

Itt a táblának több kulcsa is van: az aláhúzással jelölt számlaszám és sorszám (ez az általános), de sokszor előfordul, hogy a dőlt betűvel jelzett számlaszám és cikkszám (miután egy számlán egy cikk csak egyszer szerepelhet). Igazság szerint (ha poroszosan letároljuk a sorszámokat is), dönteni kell, melyik legyen az elsődleges kulcs, amit automatikus megszorításként kezel majd egy adatbázis-kezelő rendszer, majd gondoskodni kell a másik egyediségéről is egy újabb megszorításban.

**Normalizálás (normálforma dekompozíció):**

Az eljárás, amelyben a kedvezőtlen normálformájú egyedet lebontjuk több kívánt normálformájú egyedre.

Célja: a tárolási és karbantartási káosz megszüntetése veszteségmentesen.

Eszerint a redundancia csökkentése érdekében normalizálunk, de úgy, hogy a funkcionális függőségeket ne veszítsük el. Az eljárást az első példán (Rendelés) mutatjuk be, és bebizonyítjuk, hogy a redundancia meghagyása anomáliákat okozott volna az adatbázis feldolgozásában.