12. Andmete lukustamine ja andmete ühiskasutusest tulenevate probleemide lahendamine

andmed pole sugugi ainus ressurss

Selleks et rääkida andmete ühisest kasutamisest ja andmete lukustamisest vaatleme kõigepealt ühiskasutuses olevate ressursside jagamise üldist loogikat. Alles seejärel asume lahkama andmebaasides talletatud andmete ühiskasutuse probleeme.

12.1. Ühiskasutatav piiratud ressurss

ressursi jagamine ja ümber jagamine on üks maailma vanimaid probleeme

Ühiskasutatava piiratud ressursi kasutamise probleemid on eksisteerinud nii kaua kui on eksisteerinud elu maa peal. Esmaseks tähtsaks ühiskasutatavaks ressursiks oli söök. Selle pärast võideldi ja tapeti esialgu suhteliselt instinktiivselt - juba saurused kaklesid õiguse pärast mõni endast väiksem nahka panna. Hiljem kui tekkis inimene hakati kaklust ja tapmist kasutama süstemaatiliselt – ühiskasutatava piiratud ressursi ümber jagamiseks. Alguses oli selliseks ressursiks loomulikult ikkagi toit ja soodsam elukoht (“parem koobas”). Ühiskonna edasises arenemises said tähtsaks ressursiks toidu hankimise limiteeritud vahendid (metsad, maad, tööriistad ja inimesed - orjad). Seejärel lisandusi maavarad (vask, raud jne.), seejärel soodsad geograafilised piirkonnad (kaubandus, sõjaline strateegia jms.).

ressursse jagatakse majanduslikult, poliitiliselt ja sõjaliselt

Ka tänapäeval jagatakse ressurssi pidevalt ümber. Jagamise vahendiks on peamiselt majanduslikud ja poliitilised meetodid. Siiski ei põlata ära ka sõda - igal hetkel peetakse mingis maailma piirkonnas mõnda sõda, mille peamiseks eesmärgiks on mingeid ressursse enda valdusesse haaramine. Kõigile on vast kõrva jäänud Iisraeli ja Palestiina vahele jääv Kolani kõrgustik, mille Iisrael haaras palestiinlastelt 1967 aastal. Peamise põhjusena, miks juudid kõrgustiku enda kätte haarasid, tuuakse pidevalt hea strateegiline asukoht, mis annab hea sõjalise positsiooni ja ülevaate vaenlase tegevusest. Hoopis vähem on räägitud sellest, et antud piirkonna peamised veesooned on koondunud Kolaani kõrgustikule. Vesi on aga seal kandis oluliselt väärtuslikum ressurss kui seda on sõjalis-strateegiliselt hea positsioon. Pole vett - pole strateegiat.

12.1.1. Konkureerimine piiratud ressursile

piiratud ressurss

Piiratud ressurss tähendab seda, et ressurssi:

1. on antud hetkel jääv, täpselt mõõdetav hulk või on seda ajaühiku kohta võimalik kätte saada teatav piiratud hulk
2. nõudlus sellele ressursile on suurem, kui seda ressurssi on olemas või seda ressurssi õnnestub kätte saada vähem kui selle kasutajad seda sooviksid või saab seda ressurssi korraga kasutada (pääseb ligi) piiratud hulk kasutajaid

konkurents ressursile

Et, tekkiks konkurents ressursile peab olema täidetud kolm tingimust:

1. peab olema mingi ressurss
2. olemas peab olema selle ressursi potentsiaalne tarbija
3. ressurssi peab olema vähemalt aeg-ajalt vähem, kui on selle kasutada soovijaid või peab ressursile ligipääs olema piiratud s.t. ressurssi ei jõuta "jagada" sellise kiirusega, et kõik seda saaksid kohe kui seda vajatakse

Kui teisiti öelda, siis eksisteerib teatav ressursi defitsiit, ning seda defitsiiti tunnevad (omal nahal) selle ressursi potentsiaalsed tarbijad.

Mõista tuleb ka seda, et ressursi puudus ei pruugi olla pidev – st. et ressursi puudus võib tekkida ainult teatavatel, kas või väga väikestel, ajahetkedel – ülejäänud ajal võib olla ressurssi piisavalt kõikide jaoks. Aga just need väikesed ressursipuuduse ajahetked võivad tekitada väga tugeva konkurentsi sellele ressursile. Ajutine puudus võib tekitada ressursi varumise ja selle ressursi pideva puuduse.

12.1.2. Ressursi jagamine

erinevad ressursi jagamise viisid

Vältimaks vapustusi, mida võivad põhjustada erinevate ressursside puudused on ühiskonnas välja mõeldud erinevaid ressursi jagamise meetodeid. Jagamise põhialuseid on peamiselt viis:

1. vastavalt vajadusele
2. sihtrühmale
3. vastavalt traditsioonile

4. vastavalt jõule
5. vastavalt seadusele

Vastavalt vajadusele jaotatavaid ressursse on maailmas väga-väga vähe. Vastavalt vajadusele jaotatakse näiteks ... õhku ja varbaville. Minul midagi muud meelde ei tule. Õhkugi ei saa tegelikult alati vastavalt vajadusele

vastavalt vajadusele, kas midagi sellist üldse eksisteerib

Sihtrühmale jaotatakse näiteks ühekordseid süstlaid ja süstlanõelu, kasutaud riideid, toiduabi jms.

sihtrühmale

Vastavalt traditsioonidele jaotatakse näiteks laevahuku korral päästepaadi kohti:

vastavalt traditsioonidele

1. kõige pealt naised ja lapsed (noored naised lastega ?)
2. (seejärel haiged ja invaliidid)
3. seejärel mehed (nooremad enne ?)
4. seejärel meeskonna liikmed (laeva reisijate päästmise seisukohalt mittevajalikuks muutumise järjekorras)
5. kõige lõpuks kapten (see kes selle jama korraldas)

Jõuga jaotatakse näiteks kõiki majanduslikke ja poliitilisi ressursse. Erinev võib olla ainult selle jaotava jõu liik (raha, relvad, ajupesu jne.)

suurem jõud - rohkem ressurssi

Seadusega jaotamine on tegelikult "jõuga jaotamise" alaliik. Selle jaotuse kehtestamiseks on keegi kunagi kusagil rakendanud mingit poliitilist või majanduslikku jõudu. Seadusega jaotatakse näiteks puhkuse päevi, riikliku pensioni suurust, tööaega jne.

jaotamine seadusega

Enamikel juhtudel ei toimu ressursi jagamine vahetult, kuna vajadus on jaotada samaaegselt mitmeid alternatiivseid ressursse. Sellisel juhul kasutatakse ressursside ümber jagamiseks mingit jaotusväärtust. Universaalseimaks ressursi jagamise jaotusväärtuseks on raha.

jaotamine jaotusväärtuse abil

Sellise jaotussüsteemi toimimiseks peab olema täidetud kas nõuet:

nõuded ressursi jaotamisele

1. jaotatavate ressursside iga komponent peab olema hinnatud jaotusväärtuse ühikutes
2. jaotusväärtuste ressursi hulk peab olema piiratud – piiratud jaotatava ressursi koguväärtusega arvestatuna jaotusväärtuses

Mõista tuleb seda, et jaotuse mehhanism peab kogu aeg töötama – olenemata sellest, kas ressursi hulk või kasutus on antud hetkel piiratud või mitte. Jaotusmehhanismi puudumine võib tekitada mingi ressursi puudujäägi kõige

jaotusmehhanism peab töötama kogu aeg

absurdsematel põhjustel ning viia süsteemi lõplikult tasakaalust välja. Jaotusmehhanismi olemasolu on tihti just see, mis tagab mulje, et antud jaotatavat ressurssi on piisavalt ja et selle puudust ei saa kunagi tekkida. Näiteks kui autotööstuse toodangut lasta kõigil "vabalt ilma piiranguteta võtta", siis ilmselt autotööstuse võimsusest enam nõudluse rahuldamiseks ei piisa. Samas on ka ilmne et osa automudeleid jääb seisma – neid lihtsalt ei taheta.

On mõttetu loota, et kui mingit ressurssi on palju, siis võime seda "vabalt jaotada" ja et jaotusmehhanismi võime rakendada alles siis, kui mingil põhjusel tekkib häireid "vabas jaotuses". Selleks hetkeks kui tekkib ressursi puudus oleme me kaotanud igasuguse lootuse protsessi oma kontrolli alla saada.

12.1.3. Monopoolselt kasutatav ressurss - ressursi lukustamine

osasid ressursse ei saa jagada välja "tükk-haaval" - ressurss tuleb kasutamiseks reserveerida

Mõningaid ressursse ei saa "tükk haaval" välja jagada vaid neid saab ainult teatud ajaks anda monopoolseks kasutamiseks Siin on tavaliselt tegemist duaalse ressursiga. Esmaseks ressursiks on ressursi töö võime. Teiseseks ressursiks aga aeg, mille jooksul antud ressurssi kasutada saab. Sisulisest küljest on oluline, mida ressursiga aja ühikus "ära teha saab", sellest aga tuleneb see, milliseks ajaks on vaja ressurss reserveerida.

takso üürimine

Selliseks ressursiks on näiteks takso. Selleks ajaks, kui sõita ühest kohast teise reserveerite te takso. Keegi teine ei saa selle taksoga samal ajal kusagile mujale sõita. Te võite küll võtta takso mitme peale ja sõita siis kordamööda, kuhu kellelegi vaja aga sisuliselt kasutab taksot siiski see, kelle sõitu parasjagu tehakse – teised lihtsalt viibivad taksos.

WC lukustamine

Veelgi konkreetsema näite saate, kui mõtlete WC kabiinide peale mõnes ühiskondliku läbikäimise asutuses - näiteks teatri- või kontserdimajas aga ka kodus.

12.2. Andmed – ühiskasutatav ressurss

andmed kui ühiskasutatav ressurss tekkisid väga ammu

Ühiskasutatava ressursi tekitatud probleemid on infotehnoloogias püsinud selle tekkest alates. Seejuures ei tule "tekke ajana" käsitleda mitte arvutite "tekkimist" vaid tunduvalt varasemat aega – seda aega, kui inimesed oma esimesi piltkirja märke kiviseinale vedasid. Aga võib olla veel varasemasse aega – sellesse aega, kui inimene suhtlema ja oma teadmisi teistele inimestele edasi andma õppis. Toonane "info ühiskasutus" piirdus ühise kogemusliku teabe haldamises ja selle vahendamises kõigile kogukonna liikmetele. Mingi info (ühisressursi) muutumisel tuli seega "välja vahetada" info kõigi kogukonna liikmete peades – see nõudis aega

ja olulise teabe puudumise tõttu võis mõni kogukonna liige vahe peal oma elunatukesest ilma jääda.

Andmete, kui ühiskasutatava ressursina, ei tohi mõista ainult kusagil arvuti ketastel olevates failides või andmebaasides sisalduvaid andmeid. Andmete mõiste on tunduvalt laiem. Andmed moodustuvad kogu meie käsutuses olevast teabest, olenemata sellest, millisel kujul me neid talletame või kus me neid hoiame (kartoteegid, filmid, arhiivid filmidel ja paberitel jms.)

andmed ei asu ainult arvuti ketastel

Andmete korral ei ole piiratud ressursiks mitte andmed ise, vaid see aeg, mille jooksul on võimalik mingeid konkreetseid andmeid muuta või kasutada. Mõelge näiteks raamatukogule, kus olevat iga raamatut saab korraga lugeda ainult üks inimene.

andmete puhul on kriitiliseks ressursiks andmete kasutamise aeg

Andmeid kui ressurssi saab vaadelda ainult koos nende andmete talletamise viiside (paber, papp, kivi, mikrofilm, film, magnetlint, CD-plaat, kõvaketas jms.) ja andmete kasutamise meetodite (N: SQL, RLA, "käsitsi" vms.), metoodikate (N: Andmebaas, jadafail, paberkartoteek vms.) ning vahenditega (riistvara tarkvara). Ainult see terviklik komplekt määratleb antud konkreetse ressursi olulise komponendi - aja mille jooksul andmeid saab kasutada.

andmed on väga tihedalt seotud nende talletamise viisi, formaadi ja käsitlemise viisiga

12.2.1. Elektrooniliselt talletatud lihtsate andmestruktuuride ühiskasutus

Elektrooniliselt talletatud andmete ajalugu algab lihtsatest jadastruktuuridest – jadafailidest. Siin kohal pole mingit mõtet vaadata, kas tegemist oli binaarfailidega, tekstifailidega või andmefailidega – nende käsitlus ei erinenud üksteisest millegi poolest. Olulise "pitseri" vajutas kogu tehnoloogiale see, et sama andmekogumi kasutajaid sai korraga olla ainult üks. Kui hakata lähemalt mõtlema, miks see nii oli, siis tekkib küsimus, kas üldse ongi võimalik teisiti?

jadafail ja üks kasutaja

Kujutleme ette, et kaks kasutajat üritavad kirjutada samasse andmefaili. Uusi kirjeid saab lisada ainult jada faili lõppu. Kui nüüd kaks kasutajat avavad korraga ühe ja sama faili, siis saavad nad teada ühe ja sama faili pikkuse – kirjutamise algusaadressi. Kui nad nüüd mõlemad positsioneerivad ennast sellel aadressil ja kirjutavad sinna uue kirje, siis selle andmed, kes enne kirjutab (paraku on see ikka nii et paralleelne kirjutamine kettale on ainult näiline – ketta kirjutuspead saavad ju korraga olla ainult ühes kohas) kirjutatakse viimasena kirjutanu poolt üle. Ja nii ka kõigi järgnevate kirjetega, kuna kumbki hoiab oma kirjutusaadressi ja ei arvesta millegagi, mis toimub väljaspool tema enda protsessi.

mitu kasutajat ja üks jadafail

Asi ei parane ka siis, kui teine protsess avaks faili pärast seda kui esimene protsess sinna midagi juba kirjutanud on – sellisel juhul oleks tema kirjutamise algusaadress võrdne avamise hetkel olnud faili pikkusega ja kõik kordub taas...

Faili käsitlus sõltus ka sellest, kuidas mingis konkreetses faili käsitlevas programmis oli kirjeldatud faili käsitlusprotsessi. Erinevad programmid võisid isegi selle faili kirje struktuuri tõlgendada erinevalt.

faili kirjutamise puhverdamine

Lisaks selle toimus veel ka kirjutamise puhverdamine selliselt, et igat faili kirjutatud kirjet ei kirjutatudki füüsiliselt andmekandjale kohe, vaid neid koguti teatav hulk (puhvritäis) puhvisse ja seejärel kirjutati kogu puhver korraga andmekandjale. Tehnoloogilistel põhjuste nimetati failide töötluse käigus andmekogumeid ümber ja kustutati terveid faile korraga, et neid seejärel uuesti tekitad. Näiteks faili sorteerimisel kirjutati failis siu sorteerimise järjekorras uude faili, siis kustutati vana fail ja nimetati uus fail ringi selliselt, et tal oleks oma nimi, mis vanal (sorteerimata) failil gi.

See kõik tingis failide monopoolse kasutamise. Teatud juhtudel, kui ükski faili kasutavatest protsessid failid ei muutnud, oli võimalik mitmel eri protsessil lugeda korraga samast failist.

faili muuta saab ainult üks kasutaja korraga

faili lugeda saab korraga kuitahes palju kasutajaid

Seega taandus kogu ühiskasutus reeglile – kui loed, võivad seda teha ka teised, kui kirjutad või muudad, siis võid seda teha ainult monopoolselt. Kui see kes esimesena andmekogumi kasutusele võttis avas selle lugemiseks ei saanud keegi teine seda avada muutmise režiimis – ainult lugemise režiimis. Kui keegi avas andmekogumi aga kirjutamise režiimis, ei saanud seda enam keegi teine nii kaua avada, kuni esimene avaja selle sulgenud oli.

Ka tänapäeval käib failikäsitlus sama metoodika kohaselt – miski ei ole selles osas muutunud.

12.2.2. Andmebaaside ühiskasutus ajaloolises kontekstis

andmebaaside tulekuga asendus faili lukustamine kirje lukustamisega

Asi läks paremaks siis, kui tekkisid andmebaasid. Sellel hetkel hakati koos andmetega salvestama ka andmete kasutamise juhtandmeid – andmete juurde kirjutati ka andmete struktuuri kirjeldus ja iga kirje juurde hakati kirjutama ka kirje päist, kuhu sai kirjutada seda konkreetset kirjet iseloomustavaid juhtandmeid. Näiteks, et kirje on (loogiliselt) kustutatud või et kirje on mingi protsessi poolt kasutusele võetud ja seega praegu lukus. Mõiste lukus fail asemele tekkis uus

mõiste - lukkus kirje. Mõiste fail aga asendati mõistega andmetabel, mis esialgu siiski langes kokku ka mõistega fail – igat andmetabelit lihtsalt hoiti erinevas failis.

alles jäi ka failide lukustamine

Samas jäi loomulikult alles ka terve tabeli (faili) lukku panemise võimalus – seda juhuks, kui andmebaasi tabelis tuli teha massiliselt selliseid operatsioone, ja teiste kasutajate ligipääs oli praktilisem piirata tervele tabelile kui et hakata igat muudetavat kirjet eraldi lukustama.

DBMS tekkimine

Koos andmete säilitamise struktuuride muutusega tekkis ka spetsiaalne tarkvara, mis häälestus andmebaasides kirjeldatud andmestruktuuridele ja tekitas võimalused samade andmete "rist-kasutuseks" erinevate kasutajate poolt. Seda tarkvara hakati kutsuma andmebaasisüsteemideks (ka andmebaasi juhtimissüsteemideks, *DBMS - database management system*).

kirjete füüsiline lukustamine

Nüüd tekkis võimalus lisada samasse andmebaasi tabelisse uusi kirjeid mitmest erinevast protsessist ja neid kirjeid ka mitmetest erinevatest protsessidest kustutada ja uuendada . Kui tekkis vajadus mõningaid kirjeid mitte teistele kasutada anda, siis kirjutati nendele kirjetele lukud. Kuna tegemist oli füüsiliste lukkudega, siis oli väga oluline need "kirjutatud" lukud ka hiljem maha võtta, kuna muidu jäidki kirjed "igaveseks lukku". Kui see protsess, mis kirjed lukku pani õnnestus nii ära lõpetada, et lukke maha ei võetud, siis tekkisid baasi sellised lukustatud kirjed, mis "ei olnud kellegi omad". Selliste lukkude maha võtmiseks kirjutati baasidele spetsiaalseid protseduure. Halvemal juhul tuli kõikidel kasutajatel baasist välja logida. Siis kustutati baasist kõik veel sinna jäänud lukud ja seejärel jätkati tööd.

exclusive* ja *share lock's

Lukud, mida kirjetele pandi oli kahe sugused: eksklusiivsed *(exclusive)* ja mitte-eksklusiivsed *(shared)* . Esimeste korral oli kogu lukustatud kirjete kasutamise õigus *(Read/Write/Delete/Update)* ainult kirje lukustajal. Teisel puhul said ülejäänud kasutajad küll kirjete sisu lugeda, mitte aga seda muuta või kirjeid kustutada. Lisaks sellele sai kehtestada režiimi, kus üks kasutaja sai korraga lukustada ainult ühe kirje või siis mitu kirjet korraga. Ühe kirje lukustamise režiimis vabastati eelmine lukk uue kirje luku panemisel automaatselt. Viimati pandud lukk tuli siiski ise "käsitsi" vabastada.

12.2.3. Konkureerimine samade andmete kasutamisele - tänapäev

klient / teenindaja tekkimine tõi kaasa automaatsed

Uued lukustusmehhanismid saabusid koos klient/teenindaja tüüpi arhitektuuril töötavate andmebaaside kasutusele võtmisega. Peamise "ideoloogilise"

edasimineku tingis siin uue tarkvaralise arhitektuuri rakendamine. See, et andmebaasi käsitlus tõsteti protsesside juhtimisest eraldi, võimaldas "protsessi poolt paigaldatavate ja vabastatavate" lukkude asendamise "automaatselt paigaldatavate ja vabastatavate" lukkudega.

lukustusstrateegiad

Kuna kogu andmebaasi käsitlus (andmete pärimine, lisamine uuendamine ja kustutamine) teostatakse ühe programmi (andmebaasi mootori) poolt, millele teised protsessid vaid "ütlevad", mida peab andmetega tegema, siis selles protsessis on võimalik kirjeldata teatav metoodika ja meetodid, mille järgi andmeid lukustatakse. Iga baasi mootori jaoks on neid metoodikaid mitmeid (tavaliselt kuni 4 erinevat koos variatsioonidega). See kuidas andmeid konkreetse protsessi töö käigus kirjeid lukustatakse sõltub sellest, millise konkreetse lukustamise metoodika antud protsess enda jaoks välja on valinud.

andmebaasi mootori poolt juhitavad lukud

Järgnevalt vaatlemegi millised metoodikaid kasutatakse andmete lukustamisel kaasaaegsetes klient/teenindaja arhitektuuriga andmebaasides.

12.2.4. Klient / teenindaja arhitektuuriga andmebaasides kasutatavad lukustusviisid

Erinevad lukustamise meetodid erinevad üksteisest mitmete erinevate parameetrite poolest. Üheks tähtsamask erinevuseks on see, kui palju kirjeid korraga lukustatakse.

lukustusviisid määravad korraga lukustatavate kirjete hulga

12.2.4.1. Lukkudeta andmekäsitlus

Lukkudeta andmekäsitluse puhul ei lukustata ühtegi kirjet. Lukkudeta andmekäsitlust kasutatakse siis, kui andmebaasi ei kirjutata uusi andmeid, ei uuendata ega kustutata ka olemasolevaid andmeid, vaid ainult loetakse neid sealt. Sellise "lukustamise" tähistuseks kasutatakse tavaliselt mõistet "Read Only" (RO). Kui protsess on kehtestanud omale sellise lukustamise meetodi, siis saab ta andmebaasist andmeid ainult pärida (SELECT), mingeid muudatusi (INSERT, UPDATE, DELETE) tal andmebaasis teha ei õnnestu. Kui ta peaks siiski baasi poole saatma mõne muudatus-lause, siis on selle täitmise tulemuseks veateade. Samuti ei õnnestu tal teha ka muudatusi andmebaasi mistahes kirjeldustes (CREATE, DROP, ALTER jms.)

lukkudeta andmekäsitlus

12.2.4.2. Lehekülje lukustamine

andmebaasi lehekülje lukustamine

Lehekülje lukustamise režiimis ei lukustata kunagi ainult ühte kirjet korraga. Alati kui midagi lukustatakse, lukustatakse terve andmebaasi lehekülg korraga. Andmebaasi lehekülg on teatavat kirjete arvu sisaldav loogiline mälu üksus, mis on erinevates andmebaasi mootorites erineva suurusega ja tihti ka määratav andmebaasi installeerimisel.

Kuna igal leheküljel on juhuslik komplekt kirjeid, mis sõltub sellest, milliseid andmeid üldse ja millises järjekorras on baasi lisatud või kustutatud, siis ühe kirje lukustamisel lukustub juhuslik komplekt teisi kirjeid. Lukustatavaid "lisakirjeid" pole kuidagi võimalik määrata.

12.2.4.3. Kirje lukustamine

tabeli kirje lukustamine

Kirjete lukustamise korral toimub tõe poolest ainult selle kirje lukustamine, mida lukustada on vaja. Mingeid "juhuslikult kaasnevaid lukustusi" ei toimu. Lukkude hulk on küll mitmeid kordi suurem kui lehekülje lukustamise korral ja töötlus palju keerulisem, kuid võit "juhuslike lukkude" kadumisest on mitmeid kordi suurem kui keerukuse ja andmebaasi mootori töömahukuse kasv.

12.2.4.3. Tabeli lukustamine

andmebaasi tabeli lukustamine

Tabeli lukustamisel lukustatakse kõik tabelis olevad kirjed. Eesmärgiks on tavaliselt kaitsta kirjeid muutuste eest mingi suure, paljusid kirjeid töötleva protsessi jaoks.

12.2.5. Erinevad lukustusmeetodid

lukustusmeetodid määravad luku tugevuse

Eelmises jaotises vaatasime erinevaid viise, kuidas komplekteeritakse lukustamiseks kirjeid. Käesolevas jaotises vaatleme erinevaid lukke, mida on võimalik andmebaasi panna (kas siis lehekülgede või kirjete lukustamiseks).

Lukustusmeetodid määravad selle, millise "tugevusega" kirjed lukustatakse - kas kirjete kasutamise õigus on ainult (ühel) lukustajal või on ka teistel kasutajatel õigus andmeid töödelda (lugeda andmeid).

12.2.5.1. S-Locks *(Shared locks)*

S-lock tüüpi lukk

“S-lock” tüüpi lukku saavad samale kirjele (või leheküljele, või tabelile) panna mitu protsessi. “S-lock” tüüpi lukk lubab kõigil luku pannud protsessidel ja ka teistel protsessidel, mis lukku nendele kirjetele pole üldse pannud, lugeda kõiki “S-lock” lukuga kirjeid. Uuendada või kustutada ei saa “S-lock” lukuga lukustatud kirjeid keegi.

S-lock tüüpi lukk "ei salli" _X-lock_ tüüpi lukku

Kui kirjel (või leheküljel. või tabelil) on peal kas või ühe protsessi poolt pandud “S-lock” lukk, siis ei saa talle samaaegselt peale panna ühtegi “X-lock” tüüpi lukku (sellest räägime hiljem)

S-tüüpi lukku saab muuta kas _X-lock_ või _U-lock_ tüüpi lukuks

Protsess saab enda poolt pandud “S-lock” tüüpi luku muuta “X-lock” või “U-lock” tüüpi lukuks ainult siis, kui ükski teine protsess ei oma antud kirjele “S-lock” tüüpi lukku.

ROLLBACK ja COMMIT korraldused vabastavad kõik lukud

Andmete SQL-keelse käsitluse korral vabastavad nii COMMIT, kui ROLLBACK-korraldus kõik lukud. “S-lock” luku korral pole oluline kumba nendest kasutati, kuna andmeid nii kui nii ei muudetud. Alati tuleb aga arvestada et samal ajal võivad kusagil (mõnes teises tabelis) mujal kehtida mõned teised, antud protsessi poolt pandud lukud, mis ei ole “S-lock” tüüpi lukud. Sellisel juhul tuleb valida kas COMMIT või ROLLBACK korralduse kasutamine just nende teiste lukkudega lukustatud kirjete järgi – mida nendega teiste kirjetega soovitakse teha – kas loobuda muudatustest (ROLLBACK) või kinnitada need (COMMIT).

12.2.5.2. X-Locks *(Exclusive locks)*

X-lock tüüpi lukk

“X-lock” lukuga saab ainult üks protsess korraga lukustada kirje (või lehekülje, või tabeli). Kui üks protsess on kirjele pannud “X-lock” luku, siis ükski teine protsess ei saa panna samale kohale ühtki teist mistahes tüüpi lukku.

Protsess, mis on pannud “X-lock” tüüpi luku võib teostada nii lisamise kustutamise kui ka uuendamise operatsioone.

***X-lock* tüüpi lukk võimaldab andmebaasi tabelisse andmeid lisada, uuendada ja kustutada**

Ükski teine protsess ei saa mingil tingimusel pöörduda mistahes operatsiooni (kaasaarvatud lugemine) teostamiseks “X-lock” lukku omavate kirjete poole. Olgu see lukk nendele kirjetele pandud siis tabelile kui tervikule, leheküljele või igale kirjele eraldi.

***X-lock* tüüpi lukk sulgeb lukustatud kirjedte kasutamise teistele protsessidele**

Enne kirje uuendamist või kustutamist peab protsess sellele panema “X-lock” tüüpi luku.

***X-lock* tüüpi luk paigaldatakse kirjele enne selle muutmist**

Kui kirjet on muudetud või kustutatud, siis ei saa sellelt “X-lock” lukku eemaldada muidu, kui võttes muudatused tagasi (ROLLBACK) või kinnitades muudatused (COMMIT).

Kui tabelisse lisada uus kirje, siis jääb sellele kirjele, kuni ROLLBACK- või COMMIT-korralduse tegemiseni, X-lock tüüpi lukk.

kirje lisamise järel paigaldatakse sellele *X-lock* tüüpi lukk

ROLLBACK ja COMMIT eemaldavad iga luku.

ROLLBACK ja COMMIT korraldused vabastavad kõik lukud

12.2.5.3. U-Locks *(Update locks)*

“U-lock” lukkusid paigaldatakse siis, kui andmed loetakse baasist kindla sooviga neid uuendada (UPDATE) või kustutada (DELETE). Kui üks protsess on kirjele pannud “U-lock” luku, siis mingi teine protsess võib panna samale kohale teise “U-lock” tüüpi lukku. Samuti võib see “U-lock” tüüpi lukk eksisteerida koos “S-lock” tüüpi lukkudega.

***U-lock* ltüüpi lukk paigaldatakse vahetult enne kirje uuendamist või kustutamist**

Kui andmete uuendamine toimub, siis peab “U-lock” tüüpi lukk enne muudetama “X-lock” tüüpi lukuks. Seda ei saa teha, kui samale kohale on tehtud ka mõni “S-lock” tüüpi lukk. Kui samale kohale on tehtud mõne teise protsessi poolt “U-lock” lukk, siis see vabastatakse automaatselt.

enne andmete muutmist muudetakse *U-lock* tüüpi lukk *X-lock* tüüpi lukuks

Kui “U-lock” tüüpi lukk on muudetud “X-lock” tüüpi lukuks, siis hakkavad sellele kehtima eranditult kõik “X-lock” tüüpi luku omadused

12.2.5.4. Seosed erinevate luku-tüüpide vahel.

iga luku tüüp omab spetsiifilist järgnevus-seost iga teist tüüpi lukuga

Järgnev tabel võtab kokku erinevat tüüpi lukkude vahelised seosed. Tabeli ridades (vasakul) on kirjeldatud kirjele juba paigaldatud lukkude tüübid Tabeli veergudes (ülal) on kirjeldatud need lukkude tüübid, mida soovitakse kirjele paigaldada. Tabeli lahtrites on kirjeldatud see, kuidas olemasoleva luku tüüp ja paigaldatava luku tüüp omavahel suhtuvad - mis juhtub kui olemasoleva luku tüübiga kirjele püüab mingi teine protsess paigaldada teist lukku.

Eksisteeriv lukk	Lukk mida soovitakse paigaldada		
	S-lock	U-lock	X-lock
Lukkudeta	Lubatud	Lubatud	Lubatud
S-lock	Lubatud	Lubatud *)	Jääb ootele
U-Lock	Lubatud *)	Lubatud	Jääb ootele **)
X-Lock	Jääb ootele	Jääb ootele	Jääb ootele

S-lock ,a U-lock koos X-lockiga

*) “S-lock” ja “U-lock” tüüpi lukud võivad koos eksisteerida. Kui andmete muutmiseks on vaja “U-lock” tüüpi lukk muuta “X-lock” tüüpi lukuks, siis kõik “S-lock” tüüpi lukud peavad eelnevalt olema vabastatud.

X-lock koos U-lockiga

**) Kunagi ei õnnestu X-locki panna olemasoleva U-locki kõrvale. Õnnestub vaid paralleelselt eksisteeriva U-locki muutmine X-lock’ks. Sellisel juhul kõikide teiste protsesside U-lock’d vabastatakse.

TIME OUT

Kui kirjel (või leheküljel, või tabelil) on mingi protsessi poolt pandud mingit tüüpi lukk, siis vastavalt ülal toodud tabelile kas õnnestub või ei õnnestu teisel protsessil sinna lisada veel üks lukk. Kui luku lisamine ei õnnestu jääb lukku

lisav protsess ootama situatsiooni muutust. See, kui kaua oodatakse, on paika pandud kas andmebaasi mootori häälestuses või protsessi enda poolt. Oodata võib kas “igavesti” s.t. seni kuni lukustussituatsioon tõepoolest muutub või ette antud sekundite arvu jagu. Viimasel juhul, kui etteantud sekundite jooksul lukustussituatsioon ei muutu selliseks, et soovitud lukku õnnestuks paigutada või soovitud lukku muuta teistsuguseks (U-lock -> X-lock), siis andmebaasi poole pöördumine katkestatakse ja antakse tagasi TIME OUT viga.

12.2.6. Lukkude paigutamise metoodikad. Räpased lugemised *(dirty reads)*

lukustusmetoodikad - X-, S- ja U-lock tüüpi lukkude kombineerimine

Eelnevastes jaotistes vaadatud metoodikad kirjeldavad üksikute, erinevat tüüpi lukkude paigaldamist. Andmebaasisüsteemides on tegelikult sisse ehitatud metoodikad, kuidas paigaldada lukkusid automaatselt. Seejuures kombineeritakse erinevaid lukustustüüpe (X-, U- ja S-lock).

vaatame ORACLE vaikimisi lukustus-metoodikat

Vaatleme siinkohal Oracle nn. vaikimisi lukustusmehhanismi, mis on päris hästi välja mõeldud.

READ ONLY päring lukke ei paigalda

Kui protsess teeb SELECT-päringu, mis on “READ ONLY”, siis ei panda andmebaasile ühtegi lukku. Samas loetakse ainult commit’itud kirjeid. See tähendab, et kui mõni teine protsess on kirjeid lisanud või uuendanud (neil on “X-lock” tüüpi lukud), siis neid uuendatud kirjeid justkui ei märgatakski. Neid pole teiste kasutajate jaoks nii kaua olemas, kui kirjete looja on COMMITinud kirjed.

andmete uuendus paneb uuendatud kirjetele X-lock tüüpi luku

Kui protsess teeb andmebaasi uuenduse (UPDATE), siis pannakse kõikidele uuendatud kirjetele “X-lock” lukud. See takistab teistel protsessidel lugemast neid kirjeid. Samas on teistele protsessidele kättesaadavad need samad lukustatud kirjed sellistena nagu need olid enne uuendamist

andmete muutmiseks päritud kirjetele paigaldatakse X-lock kirjetel takistab andmete lugemist

Kui protsess teeb SELECT-päringu, milles on fraas “FOR UPDATE”, siis pannake kõikidele valitud kirjetele “U-lock” tüüpi lukk. Kui valimisse sattub kirjeid, mis hetkel omavad “X-lock” lukku, siis ei saa valimit luua ja oodatakse kuni “X-lock” lukkude vabanemiseni või TIME OUT situatsiooni tekkimiseni.

U-lock tüüpi lukud ei takista andmete

Kui mõnel kirjel on “U-lock” tüüpi lukk siis loetakse kirje ikkagi sisse. Kui nüüd mõni protsessidest, mis on paigaldanud kirjele “U-lock” tüüpi luku, uuendab

seda kirjet (seda saab teha, kui lisaks “U-lock” tüüpi lukule ei ole antud kirjele paigaldatud “S-lock” tüüpi lukku) siis andmeid uuendanud protsessi lukk muudetakse (enne uuendamist) “X-lock” tüüpi lukuks ja kõikide teiste protsesside “U-lock” tüüpi lukud vabastatakse.

uuendamiseks pärimist

Kirje lisamisel pannakse kirjele “X-lock” tüüpi lukk. Kuna teda varem polnud, siis ükski teine protsess seda kirjet lugeda ei saa.

kirje lisamisel pannakse kirjele X-lock tüüpi lukk

Kõik lukud vabastatakse kas ROLLBACK või COMMIT lausega.

ROLLBACK ja COMMIT hävitavad kõik lukud

Koos ülal kirjeldatud metoodikaga rakendatakse veel ühet lisa-võimalust, mida nimetatakse “räpseks lugemiseks” *(dirty reads). "Räpaseks lugemiseks"* nimetatakse sellist lugemist, kus, juhul kui kirje on selliselt lukus, et tema andmeid ei saa lugeda (kirjele on paigaldatud “X-lock”), siis kui kirjest on olemas mingi varasem, lukustamisele ja muutmisele eelnev koopia, siis loetakse sisse see. Seda režiimi saab nii sisse kui välja lülitada. Selline lähenemine väldib ootamist lukustatud kirjete taga ja lihtsustab oluliselt andmebaasi mootori tööd.

räpane lugemine - X-lock lukku omavatel kirjetel loetakse kirjest tema lukustamise eelne koopia (kuna kirje ise on lukus)