Loeng 7. Rakenduse disain. Andmete hoidmise meetmete kavandamine. Andmete uude süsteemi ülekandmise strateegia loomine.

Rakenduse disain

Rakenduste disain annab projekteerijale tehnikate ja vahendite valimisel tunduvalt vabamad käed, kui andmete disain. Seetõttu on ühiseid jooni ja põhimõtteid erinevate rakenduste disainis märksa raskem välja tuua, kuna iga konkreetse rakenduse disain on tugevalt spetsiifiline.
Ka ühe ja sama serveri andmebaasi erinevad rakendused võivad olla realiseeritud erinevate tarkvaravahenditega ning on seetõttu raskesti allutatavad ühele standardile.
Siiski valib ettevõte, firma, organisatsioon tavaliselt oma klienditarkvara platvormiks ühe firma (näiteks Microsoft) tooted (serveri andmebaas võib olla tehtud teise firma, näit. Oracle , vahenditega) ning kehtestab ka andmebaasi kasutajaliideste ülesehitusele ja väljanägemisele nn. “organisatsioonisisese standardi”.

Andmebaasi rakendusmoodulite (ekraanivormide, aruannete) disaini juures võiks järgida järgmisi põhimõtteid:

1. Kasutamismugavuse huvides tuleks saavutada kõigi rakendusmoodulite sarnane väljanägemine ning võimalikult sarnane ülesehitus. Selle saavutamiseks:
    - CASE metoodika kasutamisel luua kõik rakenduse moodulid ühesuguste generaatori määratluste (generator preferences) põhjal. See on parameetrite kogum, mis juhib ekraanivormide ja aruannete generaatorite käitumist. Vastavad parameetrid mõjutavad kodeerimise stiili (generaatorite poolt loodud kood), elementide paigutust ekraanil / aruannetes, genereerimist, rakenduse keskkonda (keskkonna muutujaid), kasutajaliidese väljanägemist (kirja suurused, stiil jne),… Sobivad parameetrite väärtused salvestatakse teadmusbaasi ühise komplektina. Iga mooduli tüübi jaoks (lihtvorm, lisanuppudega vorm, menüü, aruanne, jne.) võib olla salvestatud eraldi parameetrite komplekt.
    - Traditsioonilistes arenduskeskkondades, kus rakendusmooduleid ei genereerita otseselt mudelitest, (näit. Visual Basic, Access, Delphi jpm.) kasutada võimaluse korral

rakendusmoodulite (=kliendiandmebaasi objektide) algvariantide genereerimiseks Wizard tüüpi dialooge ning teha konkreetset tüüpi objekti jaoks alati samased valikud dialoogis.

- Teiste moodulite väljakutsumiseks paigutada nupud ekraanivormidesse selliselt, et kindlustada funktsioonide täitmise loogiline järjestus (vastavalt tegelike protsesside kulgemisele) ning vältida vajadust peamenüüsse pidevalt tagasi pöörduda.
- Menüüde struktuur peab toetama töö tegelikku kulgu. Menüüstruktuuride koostamisel on võimalik lähtuda kahest erinevast loogikast : 1) panna menüüpunktid loogilisse protsessi kulgemise järjekorda, 2) panna menüüpunktid järjekorda täitmissageduse kahanemise alusel, alates sagedamini täidetavatest. Üldjuhul soovitatakse kasutada esimest varianti.

1. Esialgsed moodulid (väljanägemine, kasutatavad ekraanivormide trigerid (Accessis vastavad neile Event Property'd ja Event Procedure'd), PL/SQL utiliidid) luua lähtudes analüüsi etapil paika pandud kirjeldustest (CASE metoodika kasutamise korral tehakse seda automaatselt – esmane rakendusmoodulite disain) ning seejärel alustada nende korrigeerimist vastavalt kasutaja näpunäidetele (väljanägemine, puuduvad väljakutsed jm. täiendused – rakenduste disaini täpsustamine).

2. Ekraanivormides kasutatava koodi jagamisel serveri ja ekraanivormi trigerite vahel lähtuda järgmistest kaalutlustest:
    - Suuremad arvutused, tuletused paigutada reeglina serverisse (ekraanivormis on ainult vastava serveri protseduuri või funktsiooni väljakutse – MS Access'is on serveri protseduuride väljakutsumine võimalik Passthrough Query's, kuhu kirjutatakse vahetult serveri poolt täidetavad SQL käsud).
    - Osad arvutused ja tuletused, millised kasutavad ekraanivormis juba teada olevaid väärtusi (nende teada saamiseks pole vaja uut käsku serveri andmebaasi poole pöördumiseks), paigutada ekraanivormi trigeritesse.

Selline lähenemine on mõistlik, kuna:

- Andmebaasi objektide ja nende objektidega opereeriva koodi paiknemine samas kohas (serveris) vähendab võrgu koormust (vt. Teema "Serveris hoitavad protseduurid").

- Ainult ühes kohas paikneva koodi hooldamine (täienduste, muudatuste sisseviimine ainult ühes kohas) on lihtsam ja mugavam.

1. Ekraanivormis või aruandes kasutatavad / uuendatavad serveri andmebaasi andmed peaksid üldjuhul olema kirjeldatud vaatena (View) serveri andmebaasi tabelitele (rakendusmoodul pöördub View kui liidesobjekti poole, mitte otse tabelite poole; rakendus näeb View'd nagu iseseisvat tabelit).

Andmete hoidmise (säilitamise) meetmete kavandamine

Teemaks on meetmed, mida saab (tuleb) rakendada produktsioonilise andmebaasi pideva ohutu funktsioneerimise tagamiseks. See hõlmab andmete hoidmise, taastamise, dubleerimisega (peegeldamisega) seotud probleeme.

Disaini etapi ülesandeks ei ole kasutatavate meetmete üksikasjaline kindlaksmääramine. Rohkem on siin tegemist esialgsete soovitustega lähtudes andmete ja rakenduse iseloomu tundmisest (vt. Teema "Disaini etapi ülesanded ja tegevused"). Lõpliku otsuse valitavate andmete hoidmise strateegiate ja kasutatavate vahendite kohta teeb ikkagi süsteemi administraator edasise süsteemiarenduse protsessi käigus.

Andmete hoidmise (säilitamise) strateegiad

Järgneb võimalike strateegiate lühike iseloomustus.

1. Koopiate tegemine operatsioonisüsteemi tasandil.

1.1. Koopiate tegemine ilma logfailide arhiveerimiseta:
- Nõuab kogu andmebaasi täielikku sulgemist ja koopia tegemist kõigist andmebaasi failidest (kopeeritakse kõik andmeruumid sama hetke seisuga, et garanteerida kooskõlaline pilt andmete taastamisel),
- Taastamine on võimalik koopia tegemise hetke seisuga,
- Protsessi haldamine on lihtsam

1.2. Koopiate tegemine koos logfailide arhiveerimisega:
- Peab olema käivitatud arhiveerimise protsess (ARCH), andmebaas peab töötama ARCHIVELOG rezhiimis,

- Koopiate tegemine on põhimõtteliselt võimalik nii suletud kui ka töös oleva andmebaasiga (üksikute andmeruumide kaupa),
- Täieliku ja kooskõlalise seisu taastamine on võimalik kuni viimase hetke seisuni (kahjustuse tekkimise hetkeni – kaduma lähevad ainult veel salvestamata tegevused, millised võetakse tagasi),
- Protsessi haldamine on keerulisem (võivad tekkida probleemid (andmebaas jääb rippuma), kui arhiveerija protsess ei saa kirjutada näidatud teeki (ketas täis, …), ei jõua piisava kiirusega arhiveerida, …).

2. Andmete eksportimine (arhiveerimiseks) ja importimine (taastamisel). See meetod ei peaks olema põhiline andmete kaitsmise vahend üheski andmebaasis.

Andmete hoidmise meetmete rakendamine

Tüüpilises andmebaasi rakenduses on süsteemi ohutu funktsioneerimise tagamiseks soovitav lähtuda järgnevast:

1. Kasutada andmetest täiskoopiate tegemist koos logfailide arhiveerimisega.
   Taoline lähenemine on soovitatav kõigi süsteemide puhul, kus andmeid pidevalt uuendatakse (, et tagada võimalus viimase hetke seisu taastamiseks vajaduse tekkimisel).
2. Koopiate tegemine peaks toimuma keskmiselt 1 kord ööpäevas (tööaja väliselt) kõigist andmeruumidest (kõigist andmebaasi failidest).
3. Dubleerida andmebaasi juhtfail ja logfailid ning parameetrite fail. Duplikaadid paigutada erinevatele kettaseadmetele (vt. Teema “Andmebaasi objektide paigutuse planeerimine”).
4. Hoida samast seisust mitut koopiat erinevatel seadmetel – ühte koopiat kiireks taastamiseks vabal kõvakettal, teist mõnel aeglasemal arhiveerimisseadmel (ohutuse garanteerimiseks varukoopia).

Toodud soovitused on üldised, üksikasjadeni täpsustamata. Lõpliku süsteemi ohutu funktsioneerimise tagamise meetmestiku paneb kokku süsteemi administraator edaspidise süsteemiarenduse käigus.

Andmete uude süsteemi ülekandmise strateegiast

Andmete uude süsteemi ülekandmise probleemid kuuluvad süsteemiarenduse ülemineku etapi juurde (vt. Teema “Disaini etapi ülesanded ja tegevused”). Disaini etapi raames käsitletakse antud teemat väga üldiselt (üldine strateegia), kuna siin (disaini etapi käigus) toimub ainult ülemineku etapi ettevalmistamine. Konkreetsed lahendused töötatakse välja reaalse ülemineku käigus uuele süsteemile ülemineku etapil.
Andmete ülekandmine uude süsteemi on tõenäoliselt ülemineku etapi kõige keerulisem ja probleeme tekitavaim osa, kuna siin on väga täpse koodi kirjutamise vajadus, milline garanteeriks iga tabeli kõigi kirjete õige konverteerimise uue süsteemi tabelitesse, kaasa arvatud erandlikud read.

Järgnevalt mõned lähtekohad andmete uude süsteemi ülekandmise ettevalmistamiseks:

1. Panna paika andmete ülekandmise järjekord (tabelite täitmise järjekord uues loodavas süsteemis).

Sisuliselt panevad selle järjekorra paika serveri poolt peale sunnitavad piirangud – välisvõtmetega seotud kitsendused. Kõigepealt tuleb tekitada read peremeestabelis, alles seejärel saab hakata looma neile viitavaid ridu sõltuvas tabelis, kuna sõltuva tabeli rea välisvõtme väärtus vastab peremeestabeli rea primaarse võtme väärtusele. Seega on tabelite täitmise järjekord järgmine:

- Kõigepealt ilma välisvõtmeteta tabelid,
- Seejärel (tsüklis) ainult täidetud tabelitele viitavate välisvõtmetega tabelid seni, kuni kõik tabelid on täidetud.

Konkreetne järjekord reaalsete tabelite kaupa tuleb paika panna uuele süsteemile ülemineku etapi alguses.

1. Panna paika andmete ülekandmise ajalised parameetrid.

Need sõltuvad andmete iseloomust, sõltuvalt sellest, kas on tegemist:
1) küllalt staatiliste andmetega,
2) sageli muudetavate (dünaamiliste) andmetega.

Staatilisi andmeid võib hakata üle kandma juba paar nädalat enne tegelikku planeeritavat üleminekut uuele süsteemile, vabastades sel teel "kiire aja" ainult dünaamiliste andmete ülekandmise jaoks. (Reeglina on staatilist tüüpi just ilma välisvõtmeteta või väheste välisvõtmetega tabelites sisalduvad andmed, kuid mõnikord leidub selliseid ka mitmete kihtide sügavuses.)
Dünaamilised andmed tuleks üle kanda vahetult enne uue süsteemi tegelikku töölepanemist, kasutades selleks aega, mil andmeid ei uuendata (nädalavahetus enne uuele süsteemile üleminekut). Dünaamilisi andmeid on mõtet üle kanda paari viimase kuu või siis jooksva aasta kohta, et mitte koormata uut süsteemi ajalooga.

1. Pöörata tähelepanu teistele serveri poolt kasutatavatele kitsendustele (lisaks välisvõtmetega seotud kitsendustele).

Siin on võimalik:
1) Ridade väljapraakimine serveri poolt seoses unikaalsuse nõude rikkumisega (primaarsed ja unikaalsed võtmed).See võib väga kergesti juhtuda, kui vanas süsteemis automaatne ridade unikaalsuse kontroll puudub.
2) Mittesoovitavate väärtuste genereerimine tabeli veergudesse seoses andmebaasi trigerite käivitumisega ridade lisamisel. (Varem vaadeldud näites läheb tööle arve numbrite genereerimise triger andmete lisamisel tabelisse ARVED. See pole aga soovitav, kuna olemasolevaid arveid tahetakse näha nende tegelike numbritega.) Sellise efekti ära hoidmiseks tuleks enne tegelikku andmete ülekandmise alustamist üle vaadata kõik andmetabelid ning ära keelata trigerite käivitamine (DISABLE) tabelitel, kus trigerid genereerivad mittesobilikke väärtusi (ülejäänud tabelitel, kus trigerite poolt genereeritavad väärtused on sobivad, tuleks nende käivitamine lubada),

1. Panna paika uue süsteemi andmetabelite täitmiseks vajalikud tegevused, kirjutada neid tegevusi toetavad programmikoodid.

Põhimõtteliselt võiks taoline tegevuste skeem olla järgmine:

1) Luua olemasoleva rakenduse baasil (olemasoleva rakenduse keskkonnas) tabelid, mille struktuur vastaks uue süsteemi tabelite struktuurile.
2) Luua konverteerimisprogrammid uue süsteemi tabelite täitmiseks andmetega olemasoleva rakenduse tabelitest.

Konverteerimisprogrammid:

- Peavad tagama ka erandite käsitlemise, (põhimõtteliselt on võimalik see töö teha kahes osas: 1) kanda üle põhimass "normaalseid" andmeid, 2) seejärel käsitleda ülejäänud, erandlikke andmeid rida realt, püüdes leida nende ridade ühisosa, mille alusel oleks võimalik kirjutada kood ka nende andmete konverteerimiseks).
- Peavad sageli tagama unikaalsete identifikaatorite loomise,
- Võivad tagada dubleerivate ridade eemaldamise (kui konverteerimisprogrammid seda ei tee, viskab Oracle server taolised read ikkagi välja).
- Peavad tagama uute kohustuslikku tüüpi veergude (vanas rakenduses neid veerge polnud, väärtust pole kusagilt võtta) täitmise vaikimisi väärtustega (vastasel juhul viskab server need read välja – kohustuslikku tüüpi veerg määramata).

1) Viia uue struktuuriga tabelid tekstifailidesse. Sellisel kujul olevaid andmeid suudavad eksportida ja importida kõik andmebaasisüsteemid.
2) Operatsioonisüsteemi MS-DOS all töötavast rakendusest viia andmete ülekandmisel täppidega tähed (ä, ö, ü, õ) õigele kujule. Selleks sobib Windows'I keskkonnas kasutatavate tekstiredaktorite "Search and Replace" funktsioon.
3) Lugeda tekstifailides sisalduvad kirjed uude süsteemi selleks kirjutatud käsufailide abil (SQL Loader'i juhtfailid).

Järgnevalt toome ära andmete ülekandmisel tekkivate probleemide allikad:

- Tabelite struktuuride erinevus uues ja vanas süsteemis,
- Andmete denormaliseeritus vanas süsteemis (väga palju on tuletatud informatsiooni),
- Uue ja vana süsteemi ulatuse erinevus, mistõttu osad tabelid, veerud vanas versioonis täielikult puuduvad ning nende sisu tuleb täita mingite vaikimisi väärtustega (viimased tuleb aga uue süsteemi töötamise käigus asendada mingite mõtet omavate väärtustega),

- Uude süsteemi ülekantavad andmed hõlmavad praeguses infosüsteemis erinevates osakondades laiali paiknevaid erinevaid rakendusi (erinev loogika, erinev kood, erinevate tegijate poolt erinevatel aegadel loodud, sisult sarnased andmed,mis on esitatud natuke erinevalt, jne.). Neis sisalduvad andmed ei pruugi olla omavahel kooskõlas ning pole selge, milliseid andmeid tuleks lugeda tegelikkusele vastavateks või tegelikkust kõige täpsemini peegeldavateks.
- Puudulikud või infosüsteemis kajastamata andmed (olemasolevad rakendused ei ole täielikult vastavuses reaalse olukorraga),
- Oracle'i mõistes vigaste andmete lubamine vana rakenduse keskkonnas:
    - Sama identifikaatoriga topelt read (unikaalsete identifikaatorite kontrolli puudumine),
    - On võimalikud Oracle jaoks poolikud andmed (tegevuse tagasivõtmise mehhanism Oracle'is väldib pooleli oleva tegevuse salvestamise),
    - Muude kitsenduskontrollide puudumine vana rakenduse keskkonnas.