Replikáció

replica = másolat
cél az adatok automatikus megosztása

modell (előadás a rep.ppt alapján):

- publisher (a forrás)
- distributor (aki a szétosztást felügyeli) A publisher is lehet saját maga distributora
- subscriber (aki kapja) Egy adott publikációs csomagra iratkozik fel
- lehet kétirányú is a kapcsolat, és tovább is publikálhatja más előfizetőknek
- publikáció: cikkekből álló csomag
- cikk (article): tábla, néhány mező, partíció, néhány rekord, tárolt eljárás definíciója vagy végrehajtása, stb.
- subscription (előfizetés, feliratkozás): adatkör, időzítés PUSH/PULL szinkronizáció

Típusok:

- Snapshot: ritkán frissítik, de akkor az egész adatkészletet, nem csak a változásokat. Ritkán változó, kis mennyiségű adat esetén, és ha nem gond a nagy időkésés (elavult adatok lehetnek)
- Transactional: kezdeti snapshot után változások tranzakciónként (az eredeti tranzakció egysége és határai megőrződnek). Ajánlott, ha igény a gyors változáskövetés. Peer-to-peer módszer: a subscriber tranzakciója is lefut a publisheren.
- Merge: több subscriber és a publisher is függetlenül dolgozik, aztán összefésülik az adatokat (adott időzítés szerint)
  Konfliktusfeloldás: default vagy programozott (merge agent)
  ajánlott: autonóm, esetleg gyakran offline előfizetők esetén (pl. utazó kárfelmérő ügynök)
- heterogén: kapcsolat más rendszerekkel, pl. Oracle, DB2

Opciók:

- a publikált adatok szűrése
- adatbázis vagy séma objektumok (pl. külső kulcs kényszerek) publikálása
- update-elhető előfizetések
- a publikáció átalakítható (pl. más tábla/mezőnévvel jön az adat) DTS csomagok illeszthetők az adatútba

Tipikus felhasználások:

- adattárház és reportolás
- offline alkalmazások (merge)
- közel tartani az adatokat a felhasználókhoz (hálózati hatékonyság és autonómia, pl. cég regionális egységei)

Gyakorlat

SQL server agent-et átállítani, hogy a „KARVALY2\karvaly” felhasználónévvel, és a *tudjátok, melyik* jelszóval fusson, újraindítani.
Ha korábban a virtuális gépen volt már replikáció, és a cleanup job nincs meg, akkor a függelékben lévő scripttel újra létre kell hozni. Különben nem lehet új publikációt létrehozni, az ezt a jobot akarja módosítani.

Egyirányú Snapshot-PUSH replikáció SAJÁT GÉPEN BELÜL

1. saját szerveren New publication: database: Northwind, type: snapshot, Immediate updating: no, Transform data: no, Subscriber types: SQL 2000, Tables: orders order details products (3 articles), Publ. name: NW ORDERS, Define filters: **yes**, Filter: nincs, Allow anon pull subscriptions: yes, Schedule: 1 percenként
2. Ha a schedule nincs itt állítva, a Replication Monitor alatt a Snapshot Agents listában Agent properties opcióval átállítható. Az Allow anon. access nem állítható be később (igaz, ennél a feladatnál nincs is rá szükség).
3. Az új publ. felett Push new subs: saját szerver, Database: new NW_repl névvel, Distribution agent schedule: 1 percenként, Initialize schema and data: yes
4. ellenőrizzük: létrejött az új adatbázis, benne a snapshot táblákkal
5. ellenőrizzük: a Northwind.products tábla első rekordjában átírjuk a termék nevét, 2 perc múlva a NW_repl.products táblában is látszik (a snapshot az egész táblát lecseréli drop/create, közben a tábla ÜRES IS LEHET, illetve Invalid object name hibát kaphatunk)
6. eszünkbe jut, hogy a Firsname mezőt hiba volt replikálni: vertikális szűrőt adunk a kész publikációhoz, ellenőrizzük: a subs. automatikus re-inicializációját várjuk. Valóban, 2 perc múlva eltűnik a subs. adatbázisból is
7. Replication monitor: Megnézzük, mely agent-ek jöttek létre, és hogyan futnak (agent history)
8. Töröljük a publikációt. (a replikált adat marad)
9. SAJÁT FELADAT: az employees tábla push replikálása az NW_repl adatbázisba

Egyirányú Snapshot-PUSH replikáció TÁVOLI GÉPRE
A különböző gyáregységeknek csak az őket érintő terméktípusokat replikáljuk

1. első lépés ugyanaz, de most csak a products táblát publikáljuk, horizontális szűrés: categoryid: 1..8
2. A ... (táblára felírt) IP című szervert felvesszük a helyi szerver csoportba (ennél a gépnél a VMware nem NAT, hanem bridged hálózatot használ, ezért kívülről elérhető)
3. PUSH subs. erre a szerverre. Mindenki más adatbázis-nevet válasszon
4. ellenőrzés: minden adatbázis létrejött, a táblák frissülnek
5. Töröljük a publikációt. (a replikált adat marad)

Tranzakcionális, azonnali PUSH replikáció saját gépről

1. a pubs.titles táblát replikáljuk a pubs_repl adatbázisba: a tr. publikáció létrehozása, enable anonymous pull subs. után PUSH new a publikációra. Időzítés:continuously
2. publication properties/Status: MSDTC-t elindítani, ha nem fut
3. ellenőrzés: a változás azonnal látszik (kb. 30 s). Pl.:

```
update pubs..employee set minit = 'W' where emp_id='PMA42628M'
select * from pubs_repl..employee--kb. 30 s után látszik a W
```

PULL

4. készítünk egy pull subscription-t (New pull subs.) is, ugyanerre a publication-ra, másik (új) adatbázisba, erre engedélyezzük az updateable opciót. (A varázsló indulásakor kérjük az 'advanced options' megjelenítését, ekkor megjelenik az Updatable Subs panel, ahol válasszuk az MSDTC által vezérelt immediate opciót)
5. ellenőrzés: az előfizetőn végzett változás azonnal átmegy (a tranzakció kicsit lassabb)

Merge replikáció saját gépről

1. a pubs.titles táblát replikáljuk két másik adatbázisba (a valóságban ezek vannak a két távoli gépen az utazó ügynökeinknél)
2. az egyik replikált táblán a módosítás megjelenik az eredeti táblában és a másik replikált táblában is (merge művelet). Biztonságos megoldás akkor is, ha a kliensek nincsenek folyamatosan kapcsolatban a központtal (pl. mert az ügynök eldugott vidékeket látogat), a következő sikeres kapcsolódáskor megtörténik az összefésülés.

ÖNÁLLÓ FELADAT (saját gépen)
Az első órai laza csatolás replikációs megoldása

1. nincs trigger az orders táblán, viszont beteszünk egy státusz mezőt, és készítünk belőle egy publikációt.
2. erre egy tr. push (vagy pull) subs.
3. a replikált táblát a státusz mező alapján feldolgozza a job. Mivel a már replikált rekordok a tr. replikáció esetén nem íródnak felül, ezért a státuszt a job átírhatja a feldolgozás után.
4. státusz mező helyett alternatív megoldás (például ha a központ nem akar adatmodellt módosítani, vagy valami miatt csak egy snapshot push replikációt hajlandó megvalósítani): a job egy helyi log táblában feljegyzi, mely rendeléseket dolgozta már fel.

Függelék

Cleanup job helyreállítása

```
--ez a script helyreállítja a cleanup jobot, ami nélkül nem lehet új publikációt készíteni
Use master
BEGIN TRANSACTION
DECLARE @JobID BINARY(16)
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
IF (SELECT COUNT(*) FROM msdb.dbo.syscategories WHERE name = N'REPL-Distribution Cleanup') < 1
EXECUTE msdb.dbo.sp_add_category @name = N'REPL-Distribution Cleanup'

-- Delete the job with the same name (if it exists)
SELECT @JobID = job_id
FROM msdb.dbo.sysjobs
WHERE (name = N'Distribution clean up: distribution')
IF (@JobID IS NOT NULL)
BEGIN
-- Check if the job is a multi-server job
IF (EXISTS (SELECT *
FROM msdb.dbo.sysjobservers
WHERE (job_id = @JobID) AND (server_id <> 0)))
BEGIN
-- There is, so abort the script
RAISERROR (N'Unable to import job ''Distribution clean up: distribution'' since there is
already a multi-server job with this name.', 16, 1)
GOTO QuitWithRollback
END
ELSE
-- Delete the [local] job
EXECUTE msdb.dbo.sp_delete_job @job_name = N'Distribution clean up: distribution'
SELECT @JobID = NULL
END

BEGIN

-- Add the job
EXECUTE @ReturnCode = msdb.dbo.sp_add_job @job_id = @JobID OUTPUT , @job_name = N'Distribution
clean up: distribution', @owner_login_name = N'sa', @description = N'Removes replicated
```

```
transactions from the distribution database.', @category_name = N'REPL-Distribution Cleanup',
@enabled = 1, @notify_level_email = 0, @notify_level_page = 0, @notify_level_netsend = 0,
@notify_level_eventlog = 0, @delete_level= 0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

-- Add the job steps
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID, @step_id = 1, @step_name =
N'Run agent.', @command = N'EXEC dbo.sp_MSdistribution_cleanup @min_distretention = 0,
@max_distretention = 72', @database_name = N'distribution', @server = N'KARVALY2',
@database_user_name = N'', @subsystem = N'TSQL', @cmdexec_success_code = 0, @flags = 0,
@retry_attempts = 0, @retry_interval = 0, @output_file_name = N'', @on_success_step_id = 0,
@on_success_action = 1, @on_fail_step_id = 0, @on_fail_action = 2
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXECUTE @ReturnCode = msdb.dbo.sp_update_job @job_id = @JobID, @start_step_id = 1

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

-- Add the job schedules
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name = N'Replication
agent schedule.', @enabled = 1, @freq_type = 4, @active_start_date = 20030515,
@active_start_time = 500, @freq_interval = 1, @freq_subday_type = 4, @freq_subday_interval =
10, @freq_relative_interval = 1, @freq_recurrence_factor = 0, @active_end_date = 99991231,
@active_end_time = 459
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

-- Add the Target Servers
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @JobID, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
```