ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ

УДК 004.652.4+004.827

ОСОБЛИВОСТІ ІНТЕГРАЦІЇ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Н.Б. Шаховська, Д.О. Тарасов

Національний університет "Львівська політехніка",
кафедра "Інформаційні системи та мережі"

Вступ

Проблема інтеграції розрізненої інформації з метою її подальшого опрацювання та прийняття рішень на її основі постала разом із появою сховищ даних, ще у 80-х рр. минулого століття. Передумовою її виникнення стало зростання інтересу до розподілених баз даних та масове їх упровадження у бізнесові структури. Значний внесок у вирішення цієї проблеми зробили: Colin White, A. Sheth, J. Larson, К.В. Антипін, А.В. Фомичев, М.Н. Гринев, С.Д. Кузнецов та ін. [1].

Інформатизація ВНЗ також викликає появу такого завдання, як інтеграція, оскільки університетом розроблено ряд інформаційних систем, які повинні обмінюватися між собою інформацією, а також передавати частину інформації в корпоративне сховище даних ВНЗ для подальшого її аналітичного опрацювання з метою:

- пошуку залежностей між отриманими оцінками студентів з предметів та за результатами вступу;
- пошуку дисциплін, у яких показники "Успішність", "Якість" або дуже високі, або дуже низькі;
- пошуку залежностей між результатами наукової діяльності студентів та їх практичними здобутками у вигляді проходження практик, участі в олімпіадах, конкурсах робіт тощо
- та ін.

Метою роботи є аналіз технологій і засобів опрацювання та інтеграції різнорідних джерел даних на прикладі інтеграції даних з інформаційних систем Національного університету "Львівська політехніка".

Постановка завдання

Для автоматизації та супроводу навчального процесу "Львівської політехніки" розроблено такі інформаційні системи:

- Система "Абітурієнт" – облік вступників на перші курси бакалаврату та магістратури, формування наказів на зарахування тощо;
- Система "Навчальні плани" – розроблення та облік навчальних планів за спеціальностями;
- Система "Деканат" – облік студентів, обдік індивідуальних навчальних планів, облік та аналіз успішності студентів;
- Система "Розклад" – облік аудиторного фонду, формування розкладу занять та екзаменів;
- Система "Випускник-працевлаштування" – аналіз якості підготовки випускників "Львівської політехніки", облік практик та дипломних робіт студентів.

Перелічені системи зберігаються на одному сервері під керуванням СУБД SQL Server. Як метод інтеграції використовується тиражування – копіювання визначеної частини даних з одної системи в іншу за певним розкладом.

Схему взаємодії основних БД університету зображено на рис. 1. Використано три методи обміну даними:

- синхронізація – порівняння даних;

- експорт – копіювання даних таблиці за певний період;
- вибірка – копіювання частин таблиці за певний період за параметрами користувача.

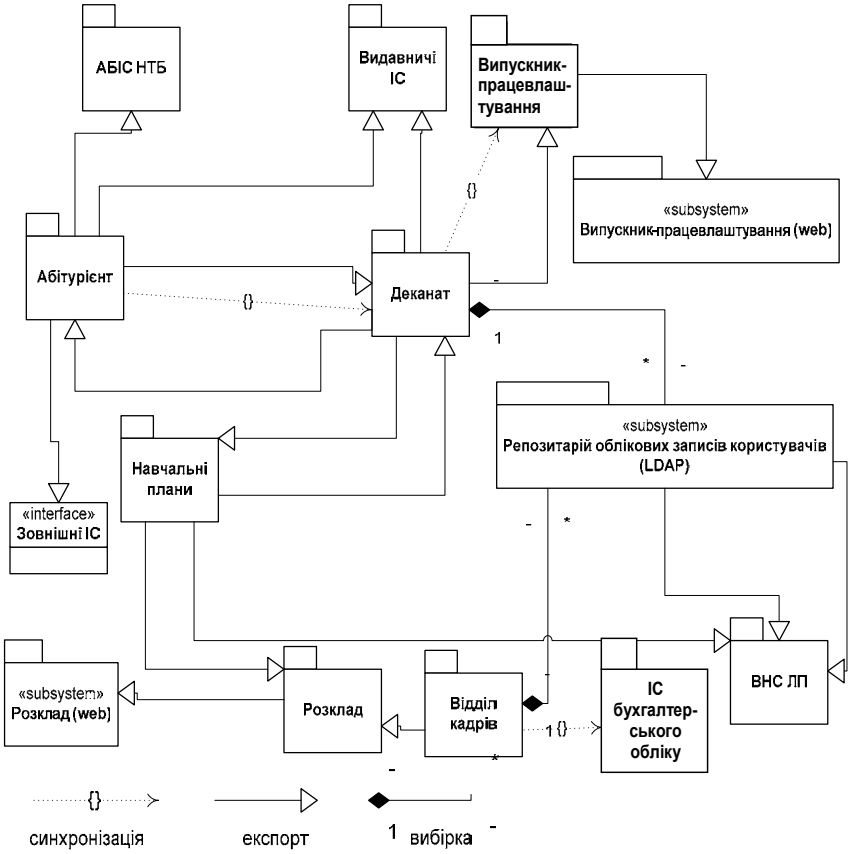

Рис. 1. Схема взаємодії основних БД "Львівської політехніки"

Зупинимося на деяких проблемах реалізації інформаційних систем даних, які приводять до виникнення завдання інтеграції даних. Серед них можна виділити:

- неоднорідність програмного середовища (у нашому випадку неактуальне, оскільки усі системи керуються однією СУБД, усі системи розміщені на одному сервері);
- розподілений характер університету;
- відсутність використання єдиних засобів аутентифікації та авторизації користувачів;
- підвищені вимоги до безпеки даних;
- необхідність наявності багаторівневих довідників метаданих;
- потреба в ефективному зберіганні й опрацюванні дуже великих обсягів інформації.

Неоднорідність програмного середовища

Корпоративна система університету, яка і є засобом підтримки прийняття рішень, не створюється на порожньому місці. Кінцеве рішення буде неоднорідним, оскільки в ньому використовуються автономно розроблені програмні засоби, описані вище. Перш за все, це стосується формування інтегрованого узгодженого набору даних, які знаходяться в різнорідних базах даних, але в подальшому це можуть бути також і електронні архіви, публічні й комерційні електронні каталоги, довідники, статистичні збірки. При розробленні структури єдиного сховища даних університету доводиться вирішувати завдання побудови єдиної інформаційної системи, що узгоджено функціонує на основі неоднорідних програмних засобів і рішень, довідників, має різні системи.

Розподілений характер університету

Концепція корпоративних систем передбачає, що операційне аналітичне опрацювання

може виконуватися в будь-якому вузлі мережі, незалежно від місця розташування основного сховища. Хоча при аналітичному опрацюванні дані тільки читаються, і потреба в синхронізації відсутня, для досягнення ефективності необхідно підтримувати реплікацію даних у різних вузлах мережі.

Відсутність єдиної бази даних користувачів

Оскільки кожна з інформаційних систем має своїх користувачів, і множини користувачів систем перетинаються лише в розрізі користувачів деканату, то це означає, що при тиражуванні даних з однієї системи в іншу необхідно оновлювати значення користувачів, що створили цей запис. Наприклад, користувачами системи "Деканат" є інспектори деканату. Кожен з інспекторів має доступ до студентів усіх спеціальностей свого курсу. Коли відбувається експорт у систему "Випускник-працевлаштування", то основними користувачами стануть працівники кафедр, які мають мати доступ до студентів усіх курсів за спеціальностями. Отже, необхідно здійснити оновлення користувачів, що створили запис, з метою надання прав на записи користувачам кафедр.

При тиражуванні даних в інших системах не виникає потреби в такому перерозподілі прав користувачів (оскільки інформацію про подання оригіналів абітурієнтами вносять працівники деканату, то ці ж працівники у подальшому матимуть доступ до студенів).

Підвищення вимог до безпеки даних

Зібрана узгоджена інформація про діяльність університету загалом дає можливість аналізу минулої і поточної діяльності об’єктів за поведінкою процесів і побудови прогнозів для майбутнього. Ця інформація настільки важлива для університету, що не можна допустити її несанкціонованого розповсюдження. У системах, заснованих на сховищах даних, виявляється недостатнім захист даних у стилі мови SQL, який забезпечують звичні комерційні СУБД. Зокрема, для забезпечення належного рівня захисту доступ до даних повинен контролюватися не тільки на рівні таблиць і їх стовпців, але і на рівні окремих рядків. Доводиться також вирішувати питання аутентифікації користувачів, захисту даних при їх переміщенні в сховищі даних з оперативних баз даних і зовнішніх джерел, захисту даних при їх передаванні по мережі.

Необхідність наявності багаторівневих довідників метаданих

Якщо роль метаданих (що зазвичай містяться в таблицях-каталогах) в оперативних інформаційних системах достатньо обмежена, то для сховищ та просторів даних наявність розвинених метаданих і засобів їх надання кінцевим користувачам є однією з основних умов успішної реалізації. Наприклад, перш ніж користувач задасть системі своє запитання, він повинен зрозуміти, яка це інформація, наскільки вона актуальна, чи можна їй довіряти, скільки часу може зайняти формування відповіді та ін.

Потреба в ефективному зберіганні й опрацюванні дуже великих обсягів інформації

Щоб оцінити обсяги інформації, визначимо кількість записів, які потрапляють в базу даних за одну сесію на рівні інституту комп’ютерних наук та інформаційних технологій (ІКНІ). Приблизна кількість студентів ІКНІ – 2500. Кожен зі студентів у середньому вивчає 7 дисциплін за семестр (включаючи курсові роботи та проекти). Студент отримує 3 обов’язкових оцінки – за модулі та екзамен. В середьому за семестр в ІКНІ видають 650 комісійних талонів та 400 талонів для довідок (достроковe складання, перескладання тощо). Кількість повторних вивчень дисциплін – приблизно 150. Отже, кількість записів становитиме: $2500\times7\times3+650+400+150=53700$. Це означає, що по університету загалом за одну сесію у відношення, яке містить результати успішності студентів, додаватиметься не менше ніж 0,5 млн. записів. Якщо врахувати, що один запис складається з коду студента (int, 4 байти), назви предмета (varchar(100), 100 байт), оцінки (int, 4 байти), типу оцінки (int, 4 байти), дати отримання оцінки (datatime, 8 байт), логіну користувача (varchar(25), 25 байт) та дати внесення інформації (datatime, 8 байт), то за один семестр обсягу даних лише про успішність збільшуватиметься приблизно на 85 Мбайт (не враховуючи обсяг

особистої інформації студента, навчальні плани тощо). За даними консалтингової компанії Meta Group, близько половини корпорацій, що використовують або планують використовувати сховища даних, припускають довести їх обсяг до сотень гігабайтів. Проблемою таких великих сховищ є те, що накладні витрати на зовнішню пам'ять зростають нелінійно при зростанні обсягу сховища. Дослідження, проведені на основі тестового набору TPC-D, показали, що для баз даних обсягом 100 гігабайтів потрібна зовнішня пам'ять обсягом ву 4,87 рази більше, ніж потрібно власне для корисних даних. При подальшому зростанні баз даних цей коефіцієнт збільшується.

Окрім потоків від зовнішніх застосувань, усередині сховищ даних, як уже згадувалося відбуваються складні процеси перевірки, очищення і перетворення початкових даних, які також потребують розробки і підтримки.

1. Порівняльна характеристика засобів опрацювання та аналізу розрізнених даних у SQL Server 2005

Уведемо поняття інтеграції даних.

Інтеграція даних – це об'єднання даних, які спочатку вводяться в різні системи. Самі ці системи можуть розташовуватися в одній локальній мережі, але мати різні платформи і внутрішню архітектуру. Розглянемо інтеграцію даних на прикладі таких інформаційних систем, що обмінюються даними: "Абітурієнт", "Деканат", "Навчальні плани", "Випускник-працевлаштування", "Розклад".

Оскільки сервером бази даних, на якому розміщені всі ці бази, є SQL Server, то проаналізуємо засоби інтеграції та обміну даними в SQL Server.

Методи обміну даними, описані на рис. 2, реалізуються такими засобами СУД MS SQL Server: DB Snapshot, перегляд, збережена процедура, розподілена транзакція, репліка.

Database Snapshot (моментальний знімок бази даних) – це "фотографія" бази даних (таблиці, перегляду) на певний момент часу. Вміст моментального знімка бази даних доступний тільки для читання і не змінюється в часі. Файлова структура і набір об'єктів у моментальному знімку бази даних повністю відповідають початковій базі. Кожна початкова база даних може мати необмежену кількість моментальних знімків. Їх зручно використовувати для формування звітів про стан даних на певний момент часу, відновлення даних.

У файлах моментальних знімків баз даних не зберігається повна копія бази даних на момент створення знімка, а містяться тільки копії змінених, з моменту створення знімка, сторінок бази. У зв'язку із цим існує низка обмежень на використання моментальних знімків баз даних:

- моментальний знімок бази даних не є її резервною копією;
- не можна створювати моментальні знімки системних баз даних;
- не можна робити резервну копію моментального знімка бази даних;
- не можна відключати і підключати моментальні знімки, як звичайні бази даних за допомогою операторів attach і detach database;
- всі моментальні знімки бази даних мають бути знищені до знищення початкової бази даних, оскільки не можуть без неї функціонувати.

Snapshot файл заповнюється даними поступово. Відразу після створення він порожній. Але як тільки в початковій базі відбувається модифікація даних, копія зміненої таблиці переноситься в Snapshot файл.

При зверненні користувача до Snapshot файла на вибірку перевіряється, чи є сторінки із запрошуваною інформацією увсередин snapshot-і. Якщо є, їх повертають користувачеві, якщо немає – запит перенаправляють в початкову базу і здійснюють вибірку даних з оригінального сховища.

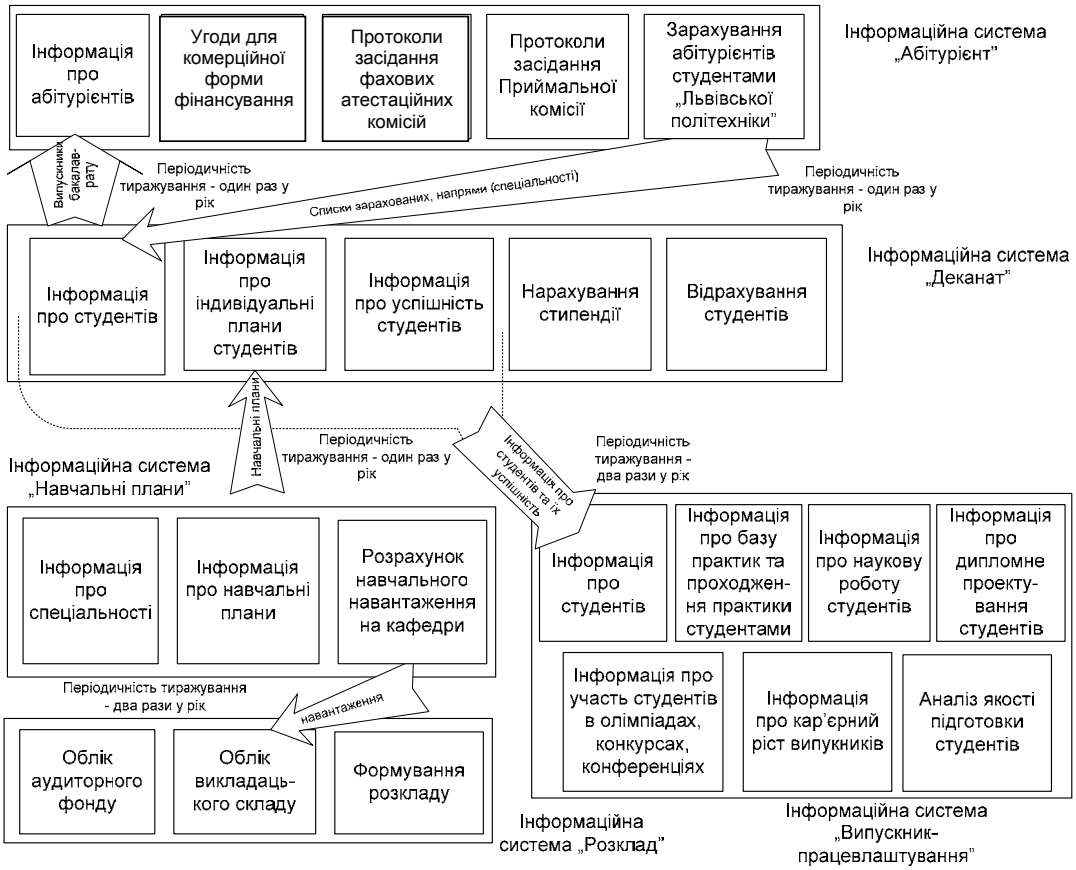

Рис. 2. Тиражування даних в інформаційних системах "Львівської політехніки"

Наведемо приклад вибірки даних зі знімка бази даних "Деканат" за літній семестр.

```
select *
from AW_Snapshot.student_group
where (year(evdate)=year(now())) and (mod(semestr,2)=0)
```

Переміщення копії даних з початкової бази в Snapshot відбувається для кожного Snapshot-а тільки один раз, при першій модифікації даних, Snapshot-а, що сталася після створення. Тому в Snapshot-і завжди міститиметься одна і та ж версія даних, або "фотографія" даних на момент створення Snapshot-у.

За допомогою Snapshot-у можна відновити знищені та змінені рядки.

Приклад 1: якщо зміна прізвища студента виявилося помилковим, можна відновити попереднє значення поля за допомогою Snapshot:

```
update AdventureWorks.ob_student
set LastName=(select LastName from AW_Snapshot.ob_student where curdate=#12.09.09#)
```

Перегляд (view) – це запит на вибірку, що зберігається на сервері як окремий об'єкт.

Оскільки результат цього запиту можна розглядати як таблицю, перегляд допускається використовувати в інших запитах, як будь-яку звичайну таблицю. Дані з перегляду не зберігаються в базі даних.

Матеріалізований перегляд зберігається на сервері у вигляді таблиці, яка автоматично оновлюється при зміні даних, що мають відношення до цього перегляду.

Перегляди використовуються в таких випадках [2]:

- для обмеження доступу користувачів до певних рядків таблиці;
- для обмеження доступу користувачів до певних стовпців таблиці;

- для представлення даних стовпців різних таблиць у вигляді одного об'єкта;
- для проглядання інформації, що виходить у результаті перетворення даних стовпців.

Перегляд доцільно використовувати для реалізації обмежень доступу користувачів до бази даних [5; 6]. Це зроблено таким чином.

Заповнюється відношення реєстру користувачів – вказується логін користувача, його роль і належність до підрозділу університету. Для прикладу:

op_user				
note	crtuser	dept_id	crtprior	crtdate
	Shahovska_N_B		admin	31.10.2009 14:32:22
	Logush_O_I		viddil	04.11.2009 0:15:22
	Marcovets	IKHI	cafedra	04.11.2009 0:15:42
	Fedasyuk_D_V		prorector	04.11.2009 0:16:07
	Medykovsky_M_O	IKHI	institute	04.11.2009

Прописується доступ користувачів до відношень та до рядків відношень. Так, відношення rf_practice можуть читати та редагувати всі користувачі групи institute, а відношення ob_practice можуть переглядати користувачі групи institute по усіх кортежах, групи cafedra – по створених користувачами групи cafedra.

op_userrole					
tablename	tabletype	username	note	crtuser	crtdate
ob_contract		admin			31.10.2009 14:35:37
ob_document		admin			31.10.2009 14:35:51
……	..	….			…
rf_stazuv_type		admin			02.11.2009 11:23:17
ob_student		institute		cafedra	31.10.2009 14:38:30
ob_StudyGroupDefinition		institute		cafedra	31.10.2009 14:38:30
rf_practice		institute			31.10.2009 14:38:30
ob_visn		institute			31.10.2009 14:38:30
ob_practice		institute		cafedra	31.10.2009 14:38:30

Далі по кожному з відношень формується перегляд. Умова, що задається у перегляді, власне, і розмежовує доступ користувачів до відношення загалом та кортежів у ньому. Наприклад:

```
SELECT *
FROM dbo.ob_practice AS q
WHERE  EXISTS
        (SELECT  tablename
        FROM dbo.op_userrole AS op
       WHERE (tablename = 'ob_practice') AND (username = USER) AND (q.crtuser = crtuser) OR (tablename
        = 'ob_practice') AND (username = USER) AND (crtuser IS NULL))
```

Перевага переглядів [4]:

- фільтрування інформації за правами користувача здійснюється на стороні сервера, що дає змогу скоротити час виконання запиту;
- розвантаження мережі (передається менший обсяг даних);
- незалежність реалізації процесу розмежування прав доступу від платформи та програмних засобів (легкий перехід на іншу версію або й на іншу СУБД).

Збережені процедури (SP — Stored Procedure є послідовністю команд на розширеннях SQL, або іншими мовами, що підтримуються сервером. Можуть приймати параметри і повертати значення заданого типу.

Процедури доцільно використовувати для операцій імпорту/експорту, синхронізації

даних.

Збережені процедури також можуть використовуватись для розмежування доступу користувачів до відношень і кортежів у них. Збережені процедури доцільно використовувати разом з переглядами для забезпечення безпеки БД (всі зміни через SP, всі вибірки через view).

Наведемо приклад використання збереженої процедури для перерозподілу прав користувачів при експорті даних про студентів із системи "Деканат" у систему "Випускник працевлаштування".

Ця збережена процедура оновлює атрибут "Користувач" з метою забезпечення доступу користувачів кафедри до записів про своїх студентів.

```
USE [pracevlasht]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[upd_ob_student]
(
   @username,
   @cafedra
  )
AS
BEGIN
  UPDATE ob_student SET crtuser = @username
      WHERE student_id in
        (SELECT student_group.StudentID
          FROM student_group
          WHERE student_group.Cafedra=@cafedra);
END
```

Перерахуємо *переваги використання збережених процедур* [2]:

- використання збережених процедур підвищує швидкість виконання операцій, оскільки процедура заздалегідь компілюється на сервері, і при повторному виклику процедура вже завантажена в пам'ять (кеш), де знайти її можна значно швидше, ніж на диску, до того ж не потрібна повторна компіляція й оптимізація;
- збережені процедури можуть складатися з десятків і сотень команд, але для їх запуску достатньо вказати лише ім'я потрібної процедури та параметри запуску. Це дає змогу зменшити розмір запиту, що посилається по мережі від клієнта на сервер, оскільки весь набір команд знаходиться в тому місці, де він має бути виконаний. Таким чином, при використанні процедур, що зберігаються, можливе зменшення навантаження на мережу;
- використання збережених процедур реалізує принцип модульного проектування, оскільки процедури дають змогу розбивати великі завдання на самостійні, дрібніші і зручніші в управлінні частини.

Проте використання збережених процедур може мати й негативні наслідки. Йде мова про розширені SP. Наприклад, використання розширеної процедури (extended stored procedure) xp_cmdshell дає змогу виконувати функції операційної системи з командного рядка так, як ніби віддалений користувач СУБД працює за консоллю сервера баз даних. При цьому функції, що викликаються за допомогою процедури xp_cmdshell, виконуються з привілеями того облікового запису, під управлінням якого завантажений SQL-Server. За замовчуванням це обліковий запис System. У цьому випадку зловмисник, що дістав доступ до СУБД, зможе створити користувача із заданими паролем і правами адміністратора [4].

Розподілені транзакції використовуються, коли користувач звертається до різних баз даних, навіть якщо цими базами даних фізично керує одна й та сама СУБД. Усі такі бази

даних, до яких звертаються користувачі при розподіленій транзакції, називаються менеджерами ресурсів.

СУБД для роботи з менеджерами ресурсів використовують transaction manager (TM), який у SQL Server 200x називається MS Distributed Transaction Coordinator (MS DTS), що задовольняє специфікації X/OPEN XA.

Будь-який менеджер транзакцій використовує протокол двофазної фіксації змін.

При виконанні розподілених транзакцій основна проблема – їх завершення, оскільки для різних менеджерів ресурсів швидкості виконання частин розподіленої транзакції протермінованого з'єднання можуть істотно відрізнятися.

Розподілені транзакції стартують, якщо:

1) у запиті явно використовується декілька баз даних;

2) застосування починає локальну транзакцію, з якої викликається віддалена збережена процедура. Якщо параметр бази даних REMOTE_PROC_TRANSACTION встановлений в ON, то локальна транзакція автоматично розширюється до розподіленої;

3) застосування починає розподілену транзакцію, використовуючи методи OLE DB або ODBC;

4) сервер зустрічає команду BEGIN DISTRIBUTED TRANSACTION.

Механізм розподілених транзакції SQL Server дає змогу працювати з даними будь-якого джерела через провайдери OLE DB.

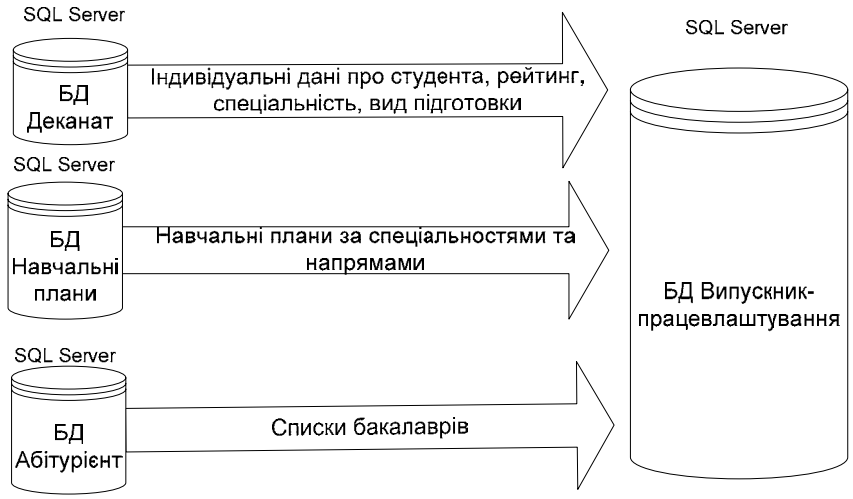

Рис. 3. Схема взаємодії систем через механізм розподілених запитів

Наведемо приклад розподіленої транзакції, результатом якої є отримання інформації про оцінку випускника Student1 з предмета Subject1 та тему його магістерської роботи. Інформацію для реалізації цього запиту необхідно отримати з баз даних "Деканат" та "Випускник-працевлаштування". Отримання інформації проходитиме на стороні системи "Випускник-працевлаштування".

```
SELECT a.student_id, a.thema, b.subject, b.curmark
FROM dbo.ob_magisterWork a, Student.dbo.student_Uspishnist b
WHERE (a.student_id=”Student1”) and (b.student_id=”Student1”) and (b.subject_id=”Subject1”)’
```

Схема взаємодії систем через розподілені запити подана на рис. 3.

Реплікація є набором технологій копіювання й розповсюдження (тиражування) даних і об'єктів баз даних між базами даних, а також синхронізації баз даних для підтримки узгодженості. Використовуючи реплікацію, можна розповсюджувати дані в різні застосування, а також віддаленим або мобільним користувачам по локальних або

глобальних мережах, за допомогою віддаленого доступу, по бездротових з'єднаннях і через Інтернет [3].

Реплікацію зазвичай використовують на регулярній основі у сценаріях "сервер-сервер", для яких необхідна висока пропускна здатність, у тому числі покращання масштабованості й доступності, зберігання та протоколювання даних, інтеграція даних з декількох вузлів, об'єднання різнорідних даних, автономна обробка пакетів. Реплікація моментальних знімків використовується для забезпечення початкового набору даних для реплікації і реплікації злиттям; вона також може застосовуватися при необхідності виконання повного оновлення даних. Прикладом застосування реплікації може бути оновлення інформації про студента у БД "Деканат" та "Працевлаштування" в межах однієї розподіленої транзакції. Використання розподіленої транзакції дасть змогу позбутися необхідності синхронізації даних.

Окрім реплікації в SQL Server 2005, можна синхронізувати бази даних за допомогою служб Microsoft Sync Framework і Sync Services for ADO.NET. Служби Sync Services for ADO.NET надають гнучкий API, який можна використовувати для побудови додатків, призначених для автономних і спільних сценаріїв.

Приклад 2: скрипт синхронізації

```
DECLARE @publication AS sysname;
DECLARE @subscriber AS sysname;
DECLARE @subscriptionDB AS sysname;
DECLARE @hostname AS sysname;
SET @publication = N'StudentPubl';
SET @subscriber = $(SubServer);
SET @subscriptionDB = N'StudentReplice';
SET @hostname = N'Student\david8'
USE [Student]
EXEC sp_addmergesubscription
  @publication = @publication,
  @subscriber = @subscriber,
  @subscriber_db = @subscriptionDB,
  @subscription_type = N'push',
  @hostname = @hostname;
EXEC sp_addmergepushsubscription_agent
  @publication = @publication,
  @subscriber = @subscriber,
  @subscriber_db = @subscriptionDB,
  @job_login = $(Login),
  @job_password = $(Password);
GO
```

2. Алгоритм обміну даними (тиражування) між інформаційними системами університету на прикладі систем "Деканат" та "Випускник-працевлаштування"

Завдання завантаження інформації про студентів у систему "Випускник-працевлаштування" виникає після закінчення семестру, тобто 2 рази на рік. Здійснюється за допомогою збережених процедур.

Її реалізація ділиться на два випадки:

1) навчальні плани не мінялися, нові групи не додавалися (міняється рік навчання студента та додається історія) (рис. 4.а);

2) навчальні плани мінялися, а, отже, утворились нові спеціальності, групи і т.д. (рис. 4.б).

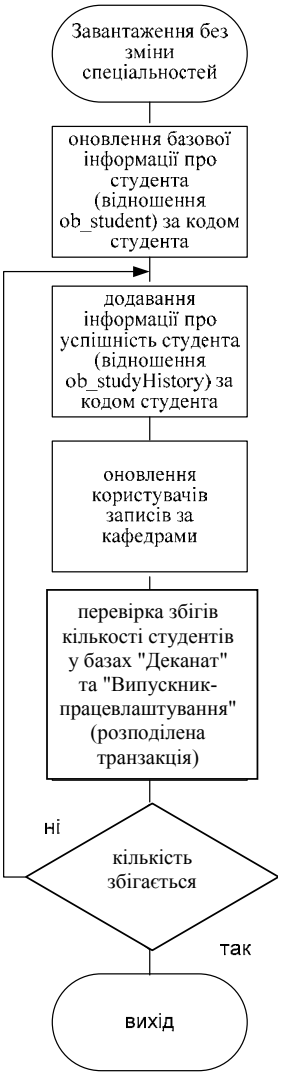

Рис. 4.а. Завантаження даних за умови відсутності появи нових спеціальностей та груп

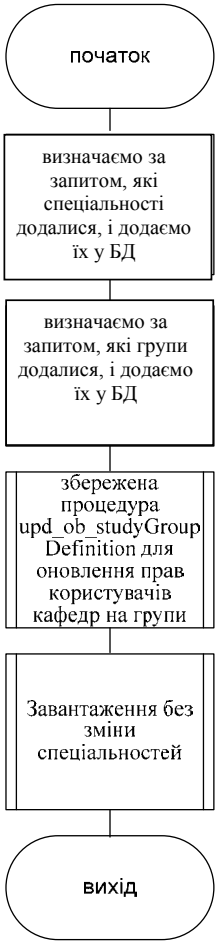

Рис. 4.б. Завантаження даних за умови появи нових спеціальностей та груп

3. Схема взаємодії систем університету за архітектурою Інмона

Оскільки дані із систем університету використовуватимуться для підтримки прийняття рішень та комплексного аналізу, то пропонується спроектувати корпоративне сховище даних університету за архітектурою "корпоративна фабрика".

Уведемо ряд означень.

Сховище даних – це агрегований інформаційний ресурс, що містить консолідовану інформацію з усієї проблемної області та використовується для підтримки прийняття рішень.

Консолідована інформація – це одержані з декількох джерел та системно інтегровані різнотипні інформаційні ресурси, які сукупно мають ознаки повноти, цілісності, несуперечності та складають адекватну інформаційну модель проблемної області, з метою її аналізу, опрацювання та ефективного використання в процесах підтримки прийняття рішень [1].

Парадигма для реляційних даних у сховищі даних (парадигма корпоративної інформаційної фабрики КІФ – Corporate Information Factory, CIF) розроблена Інмоном і передбачає, що дані повинні перебувати на низькому ступені деталізації і в третій нормальній формі (3НФ, 3NF).

Як відмітні характеристики підходу Білла Інмона до архітектури сховищ даних можна назвати такі.

1. Використання реляційної моделі організації атомарних даних і просторової – для організації сумарних даних.

2. Використання ітеративного або "спірального" підходу при створенні більших сховищ даних, тобто "будівництво" сховища даних не відразу, а по частинах (рис. 4). Оскільки інформаційні системи університету вже створені й функціонують, то такий підхід є найефективнішим і вимагатиме найменших витрат для налаштування систем до взаємодії. Так, на рис. 5 показано фази проектування сховища даних університету. На початковому етапі існує множина систем, деякі з них перебувають на стадії розроблення. Наступна фаза настає тоді, коли всі системи розроблені та активно експлуатуються, а основне завдання, що виникає при цьому – синхронізація даних у цих системах. Завершальна фаза проектування сховища даних настає тоді, коли будується корпоративне сховище даних, яке отримує детальну та агреговану інформацію із систем, а також використовується для аналізу даних підтримки прийняття керівних рішень.

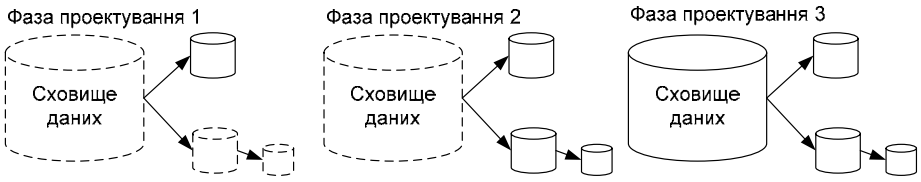

Рис. 5. Ітераційне розроблення сховища даних за методом знизу вгору

3. Використання третьої нормальної форми для організації атомарних даних, що забезпечує високий ступінь деталізації інтегрованих даних і, відповідно, надає корпораціям широкі можливості для маніпулювання ними і зміни формату і способу подання даних у міру необхідності.

4. Сховище даних – це проект корпоративного масштабу, що охоплює всі підрозділи й обслуговує потреби всіх користувачів університету.

Отже, згідно з вимогами Білла Інмона, сховище даних університету повинно виглядати так (рис. 6):

a) вітрини даних інститутів, які містять локальні копії систем "Деканат", "Випускник-працевлаштування" з первинними схемами даних (висока нормалізація) – інститути вносять детальну інформацію про свої потоки даних та використовують вітрини для оперативного аналізу даних;
b) центральне сховище університету, що містить інформацію із систем "Навчальні плани", "Розклад", "Абітурієнт", а також денормалізовану агреговану інформацію вітрин інститутів – інформація про навчальні плани та розклад тиражуються у вітрини даних, а агреговна інформація про успішність студентів консолідується з вітрин у центральне сховище.

Переваги підходу з використанням агрегованого сховища:

1. Мінімізація ризику втрати даних:
– фізичне знищення вітрини одного інституту не вплине на знищення інших, а знищену вітрину можна відновити з центрального сховища;
– знищене центральне сховище можна відновити з вітрин.
2. Збільшення швидкості виконання операцій аналітичного аналізу даних – такий аналіз проводитиметься у центральному сховищі, де дані вже частково агреговані та не сильно нормалізовані, що значно підвищить швидкість виконання запитів (виконано проміжний GROUP BY та зменшено кількість JOIN).
3. Збільшення швидкості виконання оперативних операцій – на порядок зменшений обсяг даних (адже у локальній вітрині зберагаються дані лише одного інституту), кількість користувачів, а отже, і зменшено навантаження на мережу.

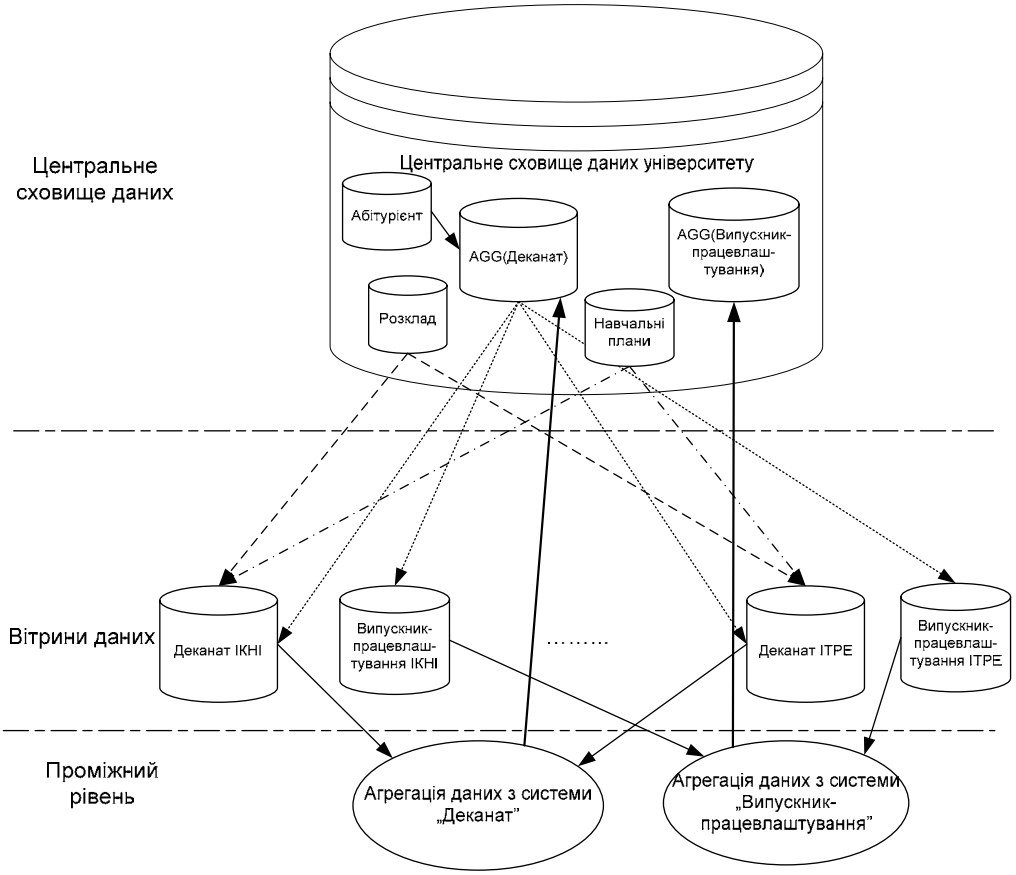

Рис. 6. Схема корпоративної фабрики сховища даних університету

Архітектуру "корпоративна фабрика" використовують також інші організації з розподіленою структурою та великим обсягом даних: "Інтермаркет", "Укрнафта" та інші.

Висновки

У статті описано методи обміну інформацією між інформаційними системами університету "Львівська політехніка". Розроблено алгоритм тиражування даних з однієї системи в іншу. Спроектовано схему корпоративної фабрики сховища даних університету.

Література

1. Шаховська Н.Б. Сховища та простори даних : монографія / Н.Б. Шаховська, В.В. Пасічник. – Л. : Вид-во Львівської політехніки, 2009. – 244 с.

2. Документация по SQL Server 2005 [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/ru-ru/library/ms203721(SQL.90).aspx.

3. Репликация данных. Виды и свойства репликации. Сравнение механизмов репликации в MS SQL Server 2005 и ORACLE Server 10g. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/department/database/olap/8/.

4. Speeding Up the Query [Електронний ресурс]. – Режим доступу: http://www.sqlmag.com/Articles/Index.cfm?ArticleID=22084.

5. Тарасов Д.О. Обмежений набір операцій для роботи з базами даних / Д.О. Тарасов, А.М. Пелещишин, П.І. Жежнич // Інформаційні системи та мережі. Вісник НУ “Львівська політехніка”. – 2001. – № 438. – С. 125–131.

6. Глобальні інформаційні системи та технології: моделі ефективного аналізу, опрацювання та захисту даних : монографія / В.В. Пасічник, П.І. Жежнич, Р.Б. Кравець, А.М. Пелещишин, Д.О. Тарасов. – Л. : Вид-во Нац. ун-ту “Львів. політехніка”, 2006. – 348 с.