УДК 681.3

О.А. Хорозов

Використання реляційної СУБД щодо структурованих *XML* документів

Рассмотрены вопросы проектирования и формирования *XML*-документов для применения в приложениях баз данных. Предложена методология преобразования между реляционной и *XML*-моделями данных без потери семантических ограничений *XML*-схемы. Описана трансляция операции над *XML* в *SQL* для фрагментации *XML*-документов и конвертации реляционных данных в *XML*-документ.

The problems of XML-documents designs for the database applications are considered. The methodology of translation between relation and XML models with capturing semantic constraints of XML-schema are suggested on the example of XBRL statements. In addition, there are discussed the questions how the operations on XML are translated to SQL, then the problems of shredding the XML for relational database and converting the relational data into XML-document.

Розглянуто питання проектування та формування *XML*-документів для застосування у додатках баз даних. Запропоновано методологію перетворення між реляційною і *XML*-моделями даних без втрати семантичних обмежень *XML*-схеми. Описано трансляцію операції над *XML* у *SQL* для фрагментації *XML*-документів та конвертації реляційних даних у *XML*-документ.

Модель даних *XML*-документів

Мова *XML* має безумовні переваги при обміні даними нарівні з можливістю об'єднання даних документа в *XML*-файлі та відокремлення інформаційного наповнення документа від засобів його форматування. Тобто, дані *XML*-документа можуть бути відокремлені від способу представлення (розмітки), а інформація може бути використана базою даних (БД) для подальшої обробки та побудови аналітичних звітів.

Об'єктне моделювання інформаційної системи передбачає використання діаграми відношень сутність–зв'язок та відображення її на реляційну структуру БД. Проблема з використанням файлів або потоку даних *XML*-формату полягає у перетворенні між реляційними і *XML*-моделями даних та застосуванням БД щодо деревовидних *XML*-файлів. Схема *XML*-файла має відображати семантику діаграми сутність–зв'язок об'єктної моделі, тоді фізичні дані реляційної БД будуть адекватно відображати *XML* і навпаки *XML*-дані завантажуватися до БД без втрати семантичних зв'язків.

У загальному вигляді *XML* має структуру дерева, яка описується набором вкладених тегів (вузлів) з атрибутами. Імена вузлів вказують на дані *XML*-документа. Мовою документообігу, клас – це опис шаблону документа, а примірник – заповнений шаблон. Класи поділяються на вихідні (первинні) та аналітичні. Підтримка цілісності даних при видаленні вузла дерева зводиться до вилучення його дочірніх елементів в рамках документа. Природно, що сукупність ієрархічних і горизонтально направлених зв'язків не повинна утворювати циклічних графів.

Проблема полягає в тому, що *XML* підтримує тільки текстовий формат даних, навіть якщо він представляє інші типи даних, такі як дата або число. Якщо програмний додаток потребує іншого формату даних (що ймовірно), він має провести аналіз *XML*-документа, перш ніж використовувати дані. Телекомунікаційне програмне забезпечення, використовуючи *XML* для передачі даних, має нести навантаження, пов'язане з ризиком невизначення зв'язку *ODBC*, незалежно від типу БД. Як правило, програмне забезпечення конвертує дані з тексту (*XML*) до інших типів (БД), і навпаки. Рішенням цієї проблеми є *XML*-обізнана БД, у якій фактично розмітка документа розглядається окремо від сутностей. Оскільки *XML*-документ є інформаційним об'єктом, який не має власних методів, то робота з примірниками документів відбувається за допомогою аналізатора, в який їх завантажують. Як аналізатор можна використати систему управління базами даних (СУБД) із представленням моделі даних відповідно до структури документа.

Загальна архітектура документа передбачає три рівня, які ієрархічно пов'язані між собою: транзакційний, навігаційний та змістовний з типами даних. Використання інформації на прикладному рівні у форматі *XML* передбачає зберігання ієрархічно структурованих даних за допомогою СУБД. Зберігання даних означає використання переваг обробки окремих даних без вибірки всього документа та застосування операцій *SQL*. При обробці даних використовують внутрішні засоби СУБД, а для представлення на зовні – прикладні процедури формування документів у різних форматах, наприклад *HTML*, *PDF*, *XML*.

Відзначимо, що мова *XML* описує дані довільного документа, а реляційна мова представляє окремі його фрагменти. Структура *XML*-документа – це пов'язані між собою елементи, а *XML*-схеми містять багато необов'язкових елементів, які не існують у документі, але повинні зберігатися в БД для збереження цілісності даних. Завдяки цій відмінності доцільно використати механізм трансляції *XSD*-графа на реляційну схему і визначити, як операції над *XML* відображаються у *T-SQL*. Формальне вирішення проблеми полягає в інтеграції *XML*-схеми з БД та використанні технології *XQuery* як засобу обробки *XML*-даних. Якщо дані «розмічати» іменами тегів, а атрибути використовувати лише для посилань, то виходить вельми однорідна структура. Крім того, частину елементів документа можна віднести до метаданих як словника таксономії, що спростить його структури. До цих елементів належать кортежі, які ототожнюють пов'язані між собою елементи і відповідають за ієрархічну структуру.

На теперішній час для зберігання *XML*-даних і селективного пошуку інформації застосовують об'єктно-орієнтовані мови програмування та реляційні БД. У цьому сенсі перспективи використання об'єктно-орієнтованих БД (ООБД) викликають серйозні сумніви, оскільки технологія *XML* розвивається набагато швидше, ніж технологія ООБД. Перетворення інформації у БД, що підтримує мову *XML*, представлено на рис.1.

Для побудови моделі даних використаємо концепцію реляційної БД. Зовнішнім представленням реляційної мови є набір двомірних таблиць з невизначеним набором кортежів та обмеженим набором атрибутів. У контрасті документи *XML* мають ієрархічну структуру.

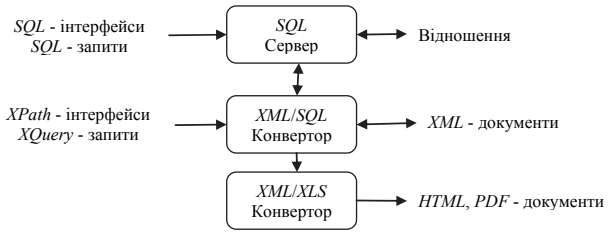

Рис. 1. Реляційна СУБД з *XML*-даними

Для застосування формату *XML* провідними виробниками СУБД доданий тип даних *XML*, що дозволяє розбирати *XML*-документи на сервері БД. Схеми БД підтримують структуру *XML*-документів і забезпечують доступ до конкретного вузла документа за допомогою програмного інтерфейсу. При роботі з *XML*-документами одним із проміжних інтерфейсів до СУБД є транслювання шляхів *XPath* у *SQL*-запити і назад. У цьому випадку вузли *XML*-документа зберігаються в таблицях БД, а при запитах *XQuery* документ *XML* відновлюється з цих таблиць. Для зберігання ієрархічних структур *XML*-документів можна застосувати: тип вузла (елемент, атрибут, простір імен і кореневий вузол); ім'я вузла (ім'я елемента, префікс простору імен); значення вузла (*NULL* для кореневого вузла і елементів).

Для визначення проекції *XML*-документа на реляційні таблиці використовують запити *XQuery* та анотації в *XML*-схемі. Схема визначає типи даних як для елементів, так і атрибутів та включає відомості про первинні ключі об'єктів. Кортежі таблиці БД відображають елементи складного типу схеми, а значення стовпців – атрибути або елементи простого типу. Конвертація *XML*-файла до реляційної структури відбувається за допомогою спеціально створеної прикладної процедури.

Граматичний аналіз (*parsing*) *XML* для зберігання у реляційній СУБД відомий як розщеплення (*shredding*). Для приведення *XML*-даних до реляційних необхідна фрагментація набору даних у відповідності до схеми БД. Механізм відображення (*mapping*) *XML*-файла на

реляційну структуру суттєво спрощується, якщо БД відповідає класу *XML*-файлів з однаковим набором атрибутів елементів тіла документа. Тоді завантаження *XML*-даних із зовнішнього джерела до БД відбувається за допомогою компонент *.NET Framework* і програмного коду прикладного сервера, а потім використовує мову *SQL* для роботи з інформацією.

Програмні засоби реляційних СУБД щодо обробки *XML*-даних

На сервері БД (наприклад, *MS SQL* 2005/2008) можна писати логіку, використовуючи інтеграцію з *CLR* (*Common Language Runtime* – компонент *.NET Framework*) для підтримки бізнес-правил. Бізнес-логіка задається за допомогою керуючого коду *C#.NET* (*VB.NET*) або процедур (*storing procedures*) та функцій *T-SQL*, які виконують обробку даних.

Розглянемо переваги та недоліки використання *T-SQL* або *CLR*:

- розщеплення *XML*-файлів на *SQL*-сервері з використанням *T-SQL*:

  плюси: *T-SQL* є рідним для сервера, тому виконання процедури буде швидше, ніж *CLR*-рішення; мінуси: функціональність мови *XML* є бідною порівняно з мовами програмування; структура *XML*-файла має бути передана до процедури; *T-SQL* не може працювати з дисковим простором операційної системи;

- розщеплення *XML*-файлів на *SQL*-сервері з використанням *CLR*:

  плюси: універсальність мови *CLR* (*C#* або *VB.NET*); розширення можливостей для обробки складних *XML*-файлів; мінуси: необхідно розгорнути і підтримати збірку на сервері; відведення більшого ресурсу оперативної пам'яті для виконання процедур.

Сервер *SQL*, використовуючи *CLR*, запускає інтерпретатор коду. Код «*Just-in-time*» компілюється в машинний код і виконується у власному просторі, а не інтерпретується. Ідея використання *CLR* полягає в тому, що процедура пишеться мовою *C#* або *VB .NET* для вибірки необхідного набору даних. Потім набір даних записується у таблиці БД. Методологія *CLR*, як правило, використовується лише для читання даних. Причина полягає в тому, що *T-SQL* краще, ніж *CLR* обладнано для роботи із записом даних у таблиці і тому не слід наповнювати пам'ять за межами буферної зони сервера.

Платформа *.Net Framework* надає великі можливості у роботі з *XML*. Серед них: читання, запис і серіалізація *XML*, підтримка *XPath*, *XQuery*, *XSLT*, а також *DataSet*. Наприклад, за допомогою запитів *XQuery* можна отримати дані для подальшого формування *XML*-файлу з відповідною схемою *XSD*. Остання формується за допомогою об'єктів *DataSet*, які дозволяють подавати реляційні дані в ієрархічній формі.

Коли використовується клас *XmlSerializer* для отримання *XML*-даних, він фактично проеціюється на систему *CLR*. Щоб перенести дані між об'єктами і *XML*, потрібне відображення конструкцій мови програмування до схеми *XML* і навпаки. Клас *XmlSerializer* та інструменти типу *Xsd.exe* забезпечують зв'язок між двома технологіями як на етапі розробки, так і при виконанні процедур.

Для розробки прикладних додатків пропонується використати бібліотеку *ADO.NET*, яка містить класи простору імен *System.XML*. Бібліотека *ADO.NET* надає можливість переводити *XML*-документи в таблиці і навпаки, відображати дані із реляційних таблиць в *XML*-документ. Зокрема, об’єкти класу *DataSet* надають можливість: читати дані у *XML*-форматі; заповнювати *DataSet* даними із *XML*-файлів; маніпулювати даними. Фактично об'єкт *Dataset* зберігає дані в локальній пам'яті та заповнюється через провайдера даних. Провайдери даних *.NET* встановлюють з'єднання та витягують дані джерела.

Принципи роботи класів *ADO.NET* с БД наступні: встановлюють з'єднання з джерелом даних, використовуючи *SQLConnection*; створюють об'єкти *SQLCommand*, які посилають на виконання; використовують *SQLDatareader* для вибірки даних; закривають *SQLDatareader* і *SQLConnection*. Клас *SqlDataAdapter* використовується спільно з класами *SQLConnection* і *SQLCommand* при підключенні до БД і слугує містком між *DateSet* і *SQL Server* для отримання даних за допомогою інструкцій *T-SQL*. Функції *SqlDataAdapter* полягають у заповненні, перетворенні та оновленні даних.

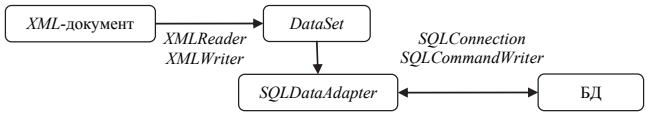

Рис. 2. Функціональне навантаження *SqlDataAdapter*

Для виконання сценарію обробки *XML*-даних необхідно: сформувати *XML*-файл, заснований на схемі; імпортувати *XML*-дані до сервера та зберігати первинний *XML*-файл в БД; отримати набір даних (після їх обробки на сервері) для формування *XML*-документа. Цей клас використовують у випадку відсутності необхідності дотримуватися принципу цілісності даних.

Формування інформаційного ресурсу (сховища) *XML*-документів

Розглянемо технологію формування сховища фінансової статистичної звітності з використанням СУБД. Оскільки мова *XML* використовується як засіб передачі інформації, природно застосувати її для формування та подання звітності суб'єктами господарювання. Мова ділової звітності *XBRL*, що є одним з розширень *XML*, дозволяє здійснювати обмін інформацією між різними програмними засобами та операційними системами. При обміні інформацією між учасниками документообігу *XBRL* забезпечує трансляцію бухгалтерських концептів у набір даних та зберігає семантику значень фінансових показників. Таксономія *XBRL* являє собою колекцію метаданих, які визначають модель даних, включаючи відношення між елементами. Існують різні розширення таксономій відповідно до національних стандартів. Повний опис специфікації *XBRL* версії 2.1 є на сайті *www.xbrl.org* [1].

Кожен документ *XBRL*-формату містить об'єкти, що відображують його контекст і дані. Наявність різноманітних ієрархічних структур документів веде до проектування цілого набору конверторів щодо завантаження даних у БД. З урахуванням даних для подальшого аналізу, адаптер завантаження *XML*-файлів у БД має передбачити перевірку узгодженості елементів документа з класифікатором, оскільки примірник документа може мати розширення таксономії.

Підхід, який дає можливість уникнути подання даних через складні ієрархічні структури, є залучення системи зв'язків *XLink*. Відповідно до технічної специфікації *XBRL*, таксономія складається зі схеми (*Schema*) і системи зв'язків (*Linkbases*). Специфікація *XBRL* визначає елементи та атрибути *XML*, які застосовуються для опису інформації. Базовий синтаксис документів спирається на схему, яка містить визначення елементів, тоді як система зв'язків визначає стосунки між елементами. При оголошенні кореневого елемента з іменем *document* вказується посилання на схему і бази зв'язків, а саме:

```
<?xml version="1.0" encoding="utf-8"?>
<ua-pfs:document xmlns:ua-pfs='http://xbrl.org.ua/ua-pfs/2009-05-15'
    xmlns:iso4217='http://www.xbrl.org/2003/iso4217'
    xmlns:xbrli='http://www.xbrl.org/2003/instance'
    xmlns:xbrll = 'http://www.xbrl.org/2003/linkbase'
    xmlns:xlink = 'http://www.w3.org/1999/xlink'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:samp='http://www.ABC.com/Taxonomy'>
<xlink:schemaRef xlink:type='simple' xlink:href='ABC-Taxonomy.xsd'/>.
```

Елементи документа мають атрибути: ім'я, тип даних, код, контекст. Останній необхідний для розуміння фінансового факту, що міститься в елементі.

Специфікація *XBRL* не містить рекомендацій з використання методів передачі інформації. Питання цілісності і конфіденційності лежать поза сферою *XBRL*, основне завдання якої – відображення вмісту звітності в погодженому форматі. Тому система консолідації звітності мусить враховувати особливості протоколу обміну даними *XML*-формату [2].

Під час обміну інформацією між абонентами, основна проблема полягає в тому, щоб дані могли бути визначені незалежно від платформи програмного забезпечення. Зазвичай, для створення звітних документів використовують програмні засоби бухгалтерського обліку або формат електронних таблиць типу *XLS*. Формування звітності мовою *XBRL* потребує відповідного словника таксономії. Якщо бухгалтерський облік ведеться на базі сучасної СУБД, то не складає труднощів отримати звіт у *XBRL*- форматі або формуванні *XML*-файла із *XLS*- таблиць з кодуванням макросу для даних формату «*utf*-8».

Клієнтська частина системи виконує операцію формування файлу *XML* та криптографічного перетворення (накладення/перевірка ци-

фрового підпису, шифрування/дешифрування). Абонент взаємодіє з серверною частиною системи через *Web*-службу синхронного обміну повідомленнями, яка підтримує *XML*-формат даних за допомогою *ASP*-сторінок. Суб'єкт ініціює виконання функції звернення до *Web*-сервісу та передачі даних. Після завершення процесу ініціаторові запиту передається відповідне повідомлення.

Інформаційна система моніторингу фінансової звітності передбачає прийом даних при взаємодії з абонентами та агрегацію показників діяльності суб'єктів господарювання в єдине сховище для порівняльного аналізу і створення аналітичних матеріалів. Програмне забезпечення блоку збору даних суб'єктів обліку виконує наступні функції: підтримку протоколу обміну даними та перевірку формату введення даних; прийом і завантаження даних з *XML*-файлів у СУБД; введення довідників, класифікаторів та журналу змін даних.

Структура БД, крім зберігання первинних документів, має забезпечити введення реєстру показників звітності, статистичних кодів підприємств та враховувати необхідність розрахунку індикативних показників. Фрагмент структури БД представлено на рис. 3.

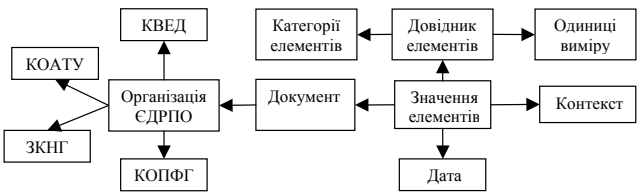

Рис. 3. Фрагмент діаграми БД

Сховище метаданих документів передбачає класифікацію за регіонами, галузями, видами діяльності, організаціями, періодами, звітами та їх розділами. Надана структура БД відображає сутність документа через довідники елементів звітності, таблиці контексту і одиниці виміру, значення яких визначаються атрибутами елементів. Для обліку господарюючих суб'єктів використано наступні статистичні коди підприємств (організацій): ЄДПРОУ (реєстровий номер); КОАТУУ (територія); КОПФГ (форма господарювання); СПОДУ (орган управління); ЗКНГ (галузь); КВЕД (вид економічної діяльності); ДКУД (управлінська документація). Ці класифікатори дозволять групувати підприємства за регіонами, галузями та видами діяльності при порівняльному аналізі. Для аналізу суб'єктів обліку введено додаткові виміри як комбінації первинних показників звітності та розраховано індикатори діяльності підприємств за відповідними функціями. Збереження ключових індикаторів ефективності ведеться з урахуванням рекомендованих значень індикаторів.

В процесі формування статистичного ресурсу програмний блок обробки даних виконує наступні функції: розрахунок аналітичних показників та виявлення розбіжностей з рекомендованими значеннями; групування аналітичних показників за різними аспектами діяльності підприємств та групування підприємств для порівняння їх показників за регіонами, галузями та видами діяльності; формування аналітичних звітів з урахуванням динаміки показників у часі.

Система ідентифікаторів і класифікаторів інформаційного ресурсу

Інформаційний ресурс (ІР) фінансової звітності складається з сукупності об'єктів з атрибутами (наприклад, інформаційний об'єкт *документ*). Кожному об'єкту необхідно присвоїти ідентифікатор, який кодує документи і класифікатори. Об'єктний ідентифікатор (*OID*) однозначно ідентифікує об'єкт в адресному просторі об'єктних ідентифікаторів. *OID* виконує наступні функції: забезпечує надання електронного ключа для застосування цифрового підпису, ідентифікацію абонента мережі та визначає тип документа. Вимоги щодо обов'язковості певної частини елементів, які описують ідентифікатор і тип документа, забезпечують його повноту та валідність. Розглянемо приклад заголовку документа:

```
<Одержувач-Identifier>
        <id root='1.2.804.5.1.2301' extension='1801'
  displayable='true'/>
            <rootOrg
        name='МоніторингФінансовоїЗвітності'
        code='FSR1'/>
            <telecom value='(tel):+38044xxxxxxx;
        (url):pfs@finstat.ua'/>
  </Одержувач-Identifier>

  <Абонент-Identifier>
              <id root='1.2.840.5.3.1234567890' extension='1801.1.2010'/>
```

<typeID **root**='1.2.804.5.1.xx.1801' **extension**='*FS*1801230'/>
<confidentiality code='*R*' codeSystem='
=1.2.16.804.5.3.1801'/>
<title verNumber='1'>Фінансова звітність за 2010</title>
<effectiveTime value='20101012103000+0200'/>
<telecom value=' (*tel*):38044xxxxxx;
(*mailto*):*info@org-ABC.com*'/>
</Абонент-*Identifier*>

У заголовку маємо префікс 1.2.804, який призначено для зразків підпису електронних документів інфраструктури ідентифікаторів об'єктів (*PKI – Private Key Infrasructure*) українського сегменту світового простору (*ISO.member-body.UA*). Слід відзначити, що для визначення *PKI* замість *ISO.member-body.UA* можна використати префікс 2.16.804 стандарту *ISO-ITU-T* для визначення простру об'єктів прикладної системи або ідентифікатори *GUID*. Згідно з стандартом ідентифікатор документа *id* складається з кореня (*root*), привласненого організації, і розширення (*extension*). Наприклад, документ з номером 1801.1.2010 має наступний ідентифікатор:

<*id* ***root***='1.2.840.5.3.1234567890'
extension='1801.1.2010'/>,

тут *root* є *OID* установи з реєстраційним номером ЄДПРО 1234567890.

Привласнення ідентифікаторів *OID* потрібно для кодування інформації і розробки автоматизованих систем обміну документами з цифровим підписом. Для цього можна використовувати шаблон <*root OID*><*system*><*location*><*typeID*>, наприклад з коренем 2.16.840.5.1. Тоді кодування у системі (2301) з організацією (1234567890) для документа (1801001) буде 2.16.840.5.1.2301.1234567890.1801001.

Заголовок також має містити назву, дату створення документа і посилання на організацію, до якої він належить. При розширенні номенклатури документів системи потрібно вводити поняття типів документів. Значення типу документа вибирається з класифікатора та має обов'язковий атрибут *code*, а ідентифікатор типу документа *typeID* містить атрибути *root*, *extension* для опису його структури.

Елементи документа містять набори атрибутів. Разом з ідентифікацією об'єктів необхідно створювати класифікатори атрибутів ресурсу. Слід зазначити, що хаотичний розвиток інформаційних ресурсів призводить до ситуації, коли одні й ті ж атрибути кодуються за різними класифікаторами. Це виключає можливість інтеграції різних ІР та ускладнює використання даних. Для вирішення проблеми власник ІР має чітко визначити правила побудови дерева об'єктних ідентифікаторів при класифікації атрибутів. Тоді засоби конвертації щодо навігаційної складової *XML*-файлу можуть застосовувати програмні засоби автоматичного визначення атрибутів абонента для прийму інформації.

Програмний модуль обробки *XML*-даних

Іноді власники інформаційних систем моніторингу не мають можливості або не бажають використовувати *on-line* доступ до БД із застосуванням *WEB*-сервісів. Тоді формується файл даних визначеної структури, який передається до власника ІР наявними засобами зв'язку. Одним із методів для обробки *XML*-даних при формуванні ІР є програмні модулі *CLR* [3].

Використання *CLR*-модулів обумовлено тим, що засоби *T-SQL* не мають доступу до файлового простору локальної мережі. В принципі, достатньо було б використати функцією читання даних з *XML*-файлу програмним модулем *CLR*, якби не існуюче обмеження на довжину *XML*-поля, що передається до *T-SQL*-процедури. Виходом є створення всієї бізнес-логіки на боці *CLR*-модуля та завантаження даних із підтримкою цілісності.

Схема прийому-передачі *XML*-даних використовує *CLR*-модуль, який активізує сервер БД і передає параметри *XML*-файлу. Алгоритм функціонування програмного модуля наступний:

- пошук елементів, які визначають атрибути абонента повідомлення та контент *XML*-файлу;
- активізація *T-SQL* щодо формування набору даних для завантаження у БД та одночасна перевірка цілісності даних;
- пошук тегу початку інформаційного тіла документа;
- читання наступних параметрів: код показника у класифікаторі; позначку початку/закінчення звітного періоду; значення одиниці виміру показника; значення і формат показника звітності;

• завантаження даних документа в таблиці БД;

• закінчення роботи при знаходженні кінцевого тегу документа.

Процедура спирається на структуру *XML*-файлу і при невідповідному форматі або порушенні цілісності даних аварійно закінчує роботу. Оскільки схема документа містить логічну модель обміну інформацією, програмний модуль *CLR* аналізує синтаксис документа та зіставляє його зі схемою БД. Програмний модуль має властивості завантаження інформації з *XML*-документа до БД у декодованій формі, а також формує *XML*-документ із контенту БД використовуючи відповідний клас *XML*-документів.

Висновки. Запропоновану у статті процедура імпорту даних супроводжується кроками щодо збереження семантики *XML*-схеми для фрагментації *XML*-документів на прикладі використання *XBRL* і формування інформаційного ресурсу. Семантичні обмеження схеми *XML*-документа зберігаються у реляційній схемі у формі функціональних зв'язків і багатозначних залежностей. Зворотна процедура формування *XML*-документа з контенту БД передбачає конвертацію реляційних даних у *XML*-формат. Крім того, визначено необхідність узгодженості обміну даними за допомогою об'єктних ідентифікаторів для навігації інформаційного потоку щодо автоматизації процесу консолідації даних.

1. *Extensible* Business Reporting Language (XBRL) 2.1. – http://www.xbrl.org/Specification/XBRLRECOMMENDATION-2003-1231+Corrected-Errata-2008-07-02.htm
2. *Хорозов О.А.* Розробка формату передачі даних ділової звітності // УСиМ. – 2010. – № 5. – С. 65–74.
3. *Introduction* to SQL Server CLR Integration (ADO.NET). – http://msdn.microsoft.com/en-us/library/ms254498.aspx

Поступила 26.04.2011

Тел. для справок: (+38095) 277-8194 (Киев)
E-mail: *oleh753@hotmail.com*

●

О.А. Хорозов

Использование реляционной СУБД для структурированных *XML*-документов

Модель данных *XML*

Язык *XML* имеет безусловные преимущества при обмене данными наряду с возможностью объединения данных документа в *XML*-файле и отделения информационного наполнения документа от средств форматирования, т.е. данные *XML*-документа могут быть отделены от способа представления (разметки), а информация может быть использована базой данных (БД) для последующей обработки и построения аналитических отчетов.

Объектное моделирование информационной системы предусматривает использование диаграммы отношений сущность–связь и отображения ее на реляционную структуру БД. Проблема с использованием файлов или потока данных *XML*-формата заключается в преобразовании между реляционными и *XML*-моделями данных с применением БД относительно древовидных *XML*-файлов. Схема *XML*-файла должна отражать семантику диаграммы сущность–связь объектной модели, тогда физические данные реляционной БД будут адекватны *XML* и наоборот *XML*-данные загружаются в БД без потери семантических связей.

В общем виде *XML* имеет структуру дерева, которая описывается набором вложенных тегов (узлов) с атрибутами. Имена узлов указывают на данные *XML*-документа. Используя язык документооборота, класс – это описание шаблона документа, а экземпляр – заполненный шаблон. Классы делятся на выходные (первичные) и аналитические. Поддержка целостности данных при удалении узла дерева сводится к удалению его дочерних элементов в рамках документа. Естественно, что совокупность иерархических и горизонтально направленных связей не должна образовывать циклических графов.

Проблема заключается в том, что *XML* поддерживает только текстовый формат данных, даже если он представляет другие типы данных, такие как дата или число. Если программное приложение требует измененного формата данных (что вероятно), он должен провести анализ *XML*-документа, прежде чем использовать данные. Телекоммуникационное программное обеспечение, используя *XML* для передачи данных, должно нести нагрузку, связанную с риском неопределения связи *ODBC*, независимо от типа БД. Как правило, программное обеспечение конвертирует данные из текста (*XML*) в другие типы (БД), и наоборот. Решение этой проблемы – использование «осведомленной» об *XML* БД, в которой фактически разметка документа рассматривается отдельно от сущностей. Поскольку *XML*-документ, будучи информационным объектом, не имеет собственных методов, то работа с экземплярами документов происходит с помощью анализатора, в который их загружают. В качестве анализатора можно использовать систему управления базами данных (СУБД) с

представлением модели данных в соответствии со структурой документа.

Общая архитектура документа предусматривает три уровня, иерархически связанные между собой: транзакционный, навигационный и содержательный с типами данных. Использование информации на прикладном уровне в формате *XML* подразумевает хранение иерархически структурированных данных с помощью СУБД. Хранение данных означает обработку отдельных данных без выборки всего документа и применения операций *SQL*. При обработке данных используют внутренние средства СУБД, а для представления – прикладные процедуры формирования документов в различных форматах, например *HTML*, *PDF*, *XML*.

Отметим, что язык *XML* описывает данные произвольного документа, а реляционный язык представляет отдельные фрагменты документа. Структура *XML*-документа представляет собой связанные между собой элементы, а *XML*-схемы содержат много необязательных элементов, не существующих в документе, но которые должны храниться в БД для сохранения целостности данных. Благодаря этому различию целесообразно использовать механизм трансляции *XSD*-графа на реляционную схему и определить, как операции над *XML* отображаются в *T-SQL*. Формальное решение проблемы заключается в интеграции *XML*-схемы с БД и использовании технологии *XQuery* как средства обработки *XML*-данных. Если данные «размечать» именами тегов, а атрибуты использовать только для ссылок, то получается довольно однородная структура. Кроме того, часть элементов документа можно отнести к метаданным в качестве словаря таксономии, что приведет к упрощению его структуры. К этим элементам относятся кортежи, отождествляющие связанные между собой элементы и ответственные за иерархическую структуру.

В настоящее время для хранения *XML*-данных и селективного поиска информации применяют объектно-ориентированные языки программирования и реляционные БД. В этом смысле перспективы использования объектно-ориентированных БД (ООБД) вызывают серьезные сомнения, поскольку технология *XML* развивается гораздо быстрее, чем технология ООБД. Преобразование информации в БД, поддерживающей язык *XML*, представлено на рис. 1.

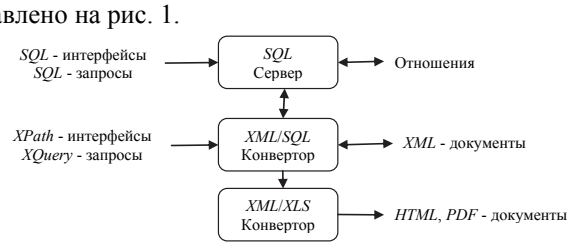

Рис. 1. Реляционная СУБД с *XML*-данными

Для построения модели данных используем концепцию реляционной БД. Внешним представлением реляционного языка есть набор двухмерных таблиц с неопределенным набором кортежей и ограниченным набором атрибутов. В контрасте документы *XML* имеют иерархическую структуру.

Для применения формата *XML* ведущими производителями СУБД добавлен тип данных XML, что позволяет разбирать *XML*-документы на сервере БД. Схемы БД поддерживают структуру *XML*-документов и обеспечивают доступ к конкретному узлу документа посредством программного интерфейса. При работе с *XML*-документами один из промежуточных интерфейсов к СУБД – транслирование путей *XPath* в *SQL*-запросы и обратно. В этом случае узлы *XML*-документа сохраняются в таблицах БД, а при запросах *XQuery* документ *XML* восстанавливается из этих таблиц. Для хранения иерархических структур *XML*-документов можно использовать: тип узла (элемент, атрибут, пространство имен и корневой узел), имя узла (имя элемента, префикс пространства имен); значение узла (*NULL* для корневого узла и элементов).

Для определения проекции *XML*-документа на реляционные таблицы используют запросы *XQuery* и аннотации в *XML*-схеме. Схема определяет типы данных как для элементов, так и атрибутов, и включает в себя сведения о первичных ключах объектов. Кортежи таблицы БД отражают элементы сложного типа схемы, а значения столбцов – атрибуты или элементы простого типа. Конвертация *XML*-файла в реляционную структуру происходит с помощью специально созданной прикладной процедуры.

Грамматический анализ (*parsing*) *XML* для хранения в реляционной СУБД известен как расщепление (*shredding*). Для приведения *XML*-данных к реляционным необходима фрагментация набора данных в соответствии со схемой БД. Механизм отображения (*mapping*) *XML*-файла на реляционную структуру существенно упрощается, если БД соответствует классу *XML*-файлов с одинаковым набором атрибутов элементов тела документа. Тогда загрузка *XML*-данных из внешнего источника в БД происходит посредством компонент *.NET Framework* и программного кода прикладного сервера и использует язык *SQL* для работы с информацией.

Программные средства реляционных СУБД по обработке *XML*-данных

На сервере БД (например, *MS SQL* 2005/2008) можно писать логику, используя интеграцию с *CLR* (*Common Language Runtime* – компонент *.NET Framework*) для поддержки бизнес-правил. Бизнес-логика задается с помощью управляющего кода *C#*. *NET* (*VB.NET*) или процедур (*storing procedures*) и функций *T-SQL*, выполняющих обработку данных.

Рассмотрим преимущества и недостатки использования *T-SQL* или *CLR*:

- расщепление *XML*-файлов на *SQL*-сервере с использованием *T-SQL*:

  плюсы: *T-SQL* – родной для сервера, поэтому выполнение процедуры будет быстрее, чем *CLR*-решения;

  минусы: функциональность языка *XML* беднее в сравнении с языками программирования, структура *XML*-файла должна быть передана в процедуру; *T-SQL* не

может работать с дисковым пространством операционной системы;

• расщепление *XML*-файлов на *SQL*-сервере с использованием *CLR*:

плюсы: универсальность языка *CLR* (*C#* или *VB.NET*), расширение возможностей для обработки сложных *XML*-файлов; минусы: необходимо развернуть и поддерживать сборку на сервере; отвод большего ресурса оперативной памяти для выполнения процедур.

Сервер *SQL*, используя *CLR*, запускает интерпретатор кода. Код «*Just-in-time*» компилируется в машинный код и выполняется в собственном пространстве, а не интерпретируется. Идея использования *CLR* состоит в том, что процедура пишется на языке *C#* или *VB .NET* для выборки необходимого набора данных, который затем записывается в таблицы БД. Методология *CLR*, как правило, используется только для чтения данных. Причина заключается в том, что *T-SQL* лучше, чем *CLR* работает с процедурой записи данных в таблицы и поэтому память не переполняется за пределами буферной зоны сервера.

Платформа *.Net Framework* предоставляет большие возможности в работе с *XML*: чтение, запись и сериализация *XML*, поддержка *XPath*, *XQuery*, *XSLT*, а также *DataSet*. Например, с помощью запросов *XQuery* можно получить данные для последующего формирования *XML*-файла с соответствующей схемой *XSD*. Последняя формируется с помощью объектов *DataSet*, позволяющими представлять реляционные данные в иерархической форме.

Когда используется класс *XmlSerializer* для получения *XML*-данных, он фактически проецируется на систему *CLR*. Чтобы перенести данные между объектами и *XML*, требуется отображение конструкций языка программирования в схему *XML* и наоборот. Класс *XmlSerializer* и инструменты типа *Xsd.exe*, обеспечивают связь между двумя технологиями как на этапе разработки, так и при выполнении процедур.

Для разработки прикладных приложений предлагается использовать библиотеку *ADO.NET*, содержащую классы пространства имен *System.XML*. Библиотека *ADO.NET* предоставляет возможность переводить *XML*-документы в таблицы и наоборот, отображать данные из реляционных таблиц в *XML*-документ. В частности, объекты класса *DataSet* предоставляют возможность: читать данные в *XML*-формате, заполнять *DataSet* данными из *XML*-файлов, манипулировать данными. Фактически объект *Dataset* сохраняет данные в локальной памяти и заполняется через провайдера данных. Провайдеры данных *.NET* устанавливают соединения и извлекают данные источника.

Принципы работы классов *ADO.Net* с БД следующие: устанавливают соединение с источником данных, используя *SQLConnection*; создают объекты *SQLCommand*, которые посылают на выполнение; используют *SQLDatareader* для выборки данных; закрывают *SQLDatareader* и *SQLConnection*. Класс *SqlDataAdapter* используется совместно с классами *SQLConnection* и *SQLCommand* при подключении к БД и служит мостиком между *DateSet* и *SQL Server* для получения данных с помощью инструкций *T-SQL*. Функции *SqlDataAdapter* заключаются в заполнении, преобразовании и обновлении данных.

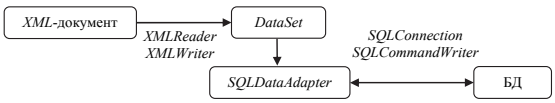

Рис. 2. Функциональная нагрузка *SqlDataAdapter*

Для выполнения сценария обработки *XML*-данных необходимо: сформировать *XML*-файл, основанный на схеме; импортировать *XML*-данные в сервер и сохранять первоначальный *XML*-файл в БД; получить набор данных (после их обработки на сервере) для формирования *XML*-документа. Этот класс используется в случае, когда отсутствует необходимость придерживаться принципа целостности данных.

Формирование информационного ресурса (хранилища) *XML*-документов

Рассмотрим технологию формирования хранилища финансовой статистической отчетности с использованием СУБД. Поскольку язык *XML* используется как средство передачи информации, естественно применить этот формат для формирования и представления отчетности субъектами хозяйствования. Язык деловой отчетности *XBRL* – один из расширений *XML*, позволяет осуществлять обмен информацией между различными программными средствами и операционными системами. При обмене информацией между участниками документооборота *XBRL* обеспечивает трансляцию бухгалтерских концептов в набор данных и сохраняет семантику значений финансовых показателей. Таксономия *XBRL* представляет собой коллекцию метаданных, определяющих модель данных, включая отношения между элементами. Существуют различные расширения таксономий согласно национальным стандартам. Полное описание спецификации *XBRL* версии 2.1 находится на сайте *www.xbrl.org* [1].

Каждый документ *XBRL*-формата содержит объекты, отображающие его контекст и данные. Наличие разнообразных иерархических структур документов приводит к проектированию целого набора конверторов по загрузке данных в БД. При использовании данных для дальнейшего анализа адаптер загрузки *XML*-файлов в БД должен предусмотреть проверку согласованности элементов документа с классификатором, поскольку экземпляр документа может иметь расширение таксономии.

Подход, позволяющий избежать представления данных через сложные иерархические структуры, – привлечение системы связей *XLink*. Согласно технической спецификации *XBRL* таксономия состоит из схемы (*Schema*) и системы связей (*Linkbases*). Спецификация *XBRL* определяет элементы и атрибуты *XML*, которые используются для описания информации. Базовый синтаксис документов опирается на схему, содержащую определение элементов, тогда как система связей определяет отношения между элементами. При объявлении корневого элемента с именем *document* указывается ссылка на схему и базы связей, а именно:

```
<?xml version="1.0" encoding="utf-8"?>
<ua-pfs:document xmlns:ua-pfs='http://xbrl.org.ua/ua-pfs/2009-05-15'
        xmlns:iso4217='http://www.xbrl.org/2003/iso4217'
        xmlns:xbrli='http://www.xbrl.org/2003/instance'
        xmlns:xbrll = 'http://www.xbrl.org/2003/linkbase'
        xmlns:xlink = 'http://www.w3.org/1999/xlink'
        xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xmlns:samp='http://www.ABC.com/Taxonomy'>
<xlink:schemaRef xlink:type='simple' xlink:href='ABC-Taxonomy.xsd'/>.
```

Элементы документа имеют атрибуты: имя, тип данных, код, контекст. Последний необходим для понимания финансового факта, содержащегося в элементе.

Спецификация *XBRL* не содержит рекомендаций по использованию методов передачи информации. Вопрос целостности и конфиденциальности лежит вне сферы *XBRL*, основная задача которой – отображение содержимого отчетности в согласованном формате. Поэтому система консолидации отчетности должна учитывать особенности протокола обмена данными *XML*-формата [2].

При обмене информацией между абонентами основная проблема заключается в том, чтобы данные могли быть определены независимо от платформы программного обеспечения. Обычно для создания отчетных документов используют программные средства бухгалтерского учета или формат электронных таблиц типа *XLS*. Формирование отчетности с помощью *XBRL* требует соответствующего словаря таксономии. Если бухгалтерский учет ведется на базе современной СУБД, то не составляет труда получить отчет в *XBRL*-формате или формировании *XML*-файла из *XLS* таблиц с кодировкой макроса для данных формата *utf*-8.

Клиентская часть системы выполняет операцию формирования файла *XML* и криптографического преобразования (наложение/проверка цифровой подписи, шифрования/дешифрования). Абонент взаимодействует с серверной частью системы через *Web*-службу синхронного обмена сообщениями, которая поддерживает *XML*-формат данных с помощью *ASP*-страниц. Субъект инициирует выполнение функции обращения к *Web*-сервису и передачи данных. После завершения процесса инициатору запроса передается сообщение.

Информационная система мониторинга финансовой отчетности предусматривает прием данных при взаимодействии с абонентами и агрегацию показателей деятельности субъектов хозяйствования в единое хранилище для сравнительного анализа и создания аналитических материалов. Программное обеспечение блока сбора данных субъектов учета выполняет следующие функции: поддержку протокола обмена данными и проверку формата ввода данных, прием и загрузку данных из *XML*-файлов в СУБД; введение справочников, классификаторов и журнала изменений данных.

Структура БД, кроме хранения первичных документов, должна обеспечить введение реестра показателей отчетности, статистических кодов предприятий и учитывать необходимость расчета индикативных показателей. Фрагмент структуры БД представлены на рис.3.

Хранилище метаданных документов предусматривает классификацию по регионам, отраслям, видам деятельности, организациям, периодам, отчетам и их разделам. Предоставленная структура БД отражает сущность документа через справочники элементов отчетности, таблицы контекста и единиц измерения, значения которых определяются атрибутами элементов. Для учета хозяйствующих субъектов использованы следующие статистические коды предприятий (организаций): ЕДПРОУ (регистровый номер); КОАТУУ (территория); КОПФГ (форма хозяйствования); СПОДУ (орган управления); ЗКНГ (отрасль); КВЕД (вид экономической деятельности); ДКУД (управленческая документация). Эти классификаторы позволят группировать предприятия по регионам, отраслям и видам деятельности при сравнительном анализе. Для анализа субъектов учета введены дополнительные измерения как комбинации первичных показателей отчетности и рассчитаны индикаторы деятельности предприятий по соответствующим функциям. Хранение ключевых индикаторов эффективности ведется с учетом рекомендованных значений индикаторов.

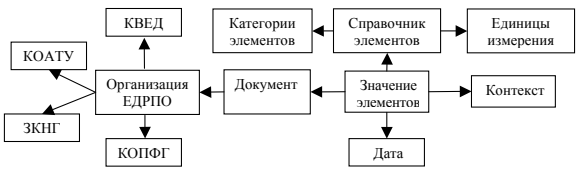

Рис. 3. Фрагмент диаграммы БД

В процессе формирования статистического ресурса программный блок обработки данных выполняет следующие функции: расчет аналитических показателей и выявление расхождений с рекомендованными значениями; группировку аналитических показателей по различным аспектам деятельности предприятий и группировку предприятий для сравнения их показателей по регионам, отраслям и видам деятельности; формирование аналитических отчетов с учетом динамики показателей во времени.

Система идентификаторов и классификаторов информационного ресурса

Информационный ресурс (ИР) финансовой отчетности состоит из совокупности объектов с атрибутами (например, информационный объект *документ*). Каждому объекту необходимо присвоить идентификатор, кодирующий документы и классификаторы. Объектный идентификатор (*OID*) однозначно идентифицирует объект в адресном пространстве объектных идентификаторов. *OID* выполняет следующие функции: обеспечивает предоставление электронного ключа для применения цифровой подписи, идентификацию абонента сети и определяет тип документа. Требования относительно обязательности определенной части элементов, описывающих идентификатор и тип документа, обеспечивающих его полноту и валидность. Рассмотрим пример заголовка документа:

```
<Получатель-Identifier>
    <id root='1.2.804.5.1.2301' extension='1801' displayable='true'/>
      <rootOrg name='МониторингФинансовойОтчетности'
        code='FSR1'/>
      <telecom value='(tel):+38044xxxxxxx; (url):pfs@finstat.ua'/>
</ Получатель-Identifier>
```

```
<Абонент-Identifier>
        <id root='1.2.840.5.3.1234567890' extension='1801.1.2010'/>
        <typeID root='1.2.804.5.1.xx.1801' extension='FS1801230'/>
        <confidentiality code='R' codeSystem='1.2.16.804.5.3.1801'/>
        <title verNumber='1'>Финансовая отчетность за 2010</title>
        <effectiveTime value='20101012103000+0200'/>
        <telecom value=' (tel):38044xxxxxx;
        (mailto):info@org-ABC.com'/>
</Абонент-Identifier>
```

В заголовке есть префикс 1.2.804, предназначенный для образцов подписи электронных документов инфраструктуры идентификаторов объектов (*PKI – Private Key Infrasructure*) украинского сегмента мирового пространства (*ISO.member-body.UA*). Следует отметить, что для определения *PKI* вместо *ISO.member-body.UA* можно использовать префикс 2.16.804 стандарта *ISO-ITU-T* для определения пространства объектов прикладной системы или идентификаторы *GUID*. Согласно стандарту идентификатор документа *id* состоит из корня (*root*), присвоенного организации, и расширения (*extension*). Например, документ с номером 1801.1.2010 имеет следующий идентификатор:

<id root='1.2.840.5.3.1234567890'*extension*='1801.1.2010'/>,

здесь *root – OID* учреждения с регистрационным номером 1234567890.

Присвоение идентификаторов *OID* нужно для кодирования информации и разработки автоматизированных систем обмена документами с цифровой подписью. Для этого можно использовать шаблон *<root OID><system><location><typeID>*, например с корнем 2.16.840.5.1. Тогда кодирование в системе (2301) с организацией (1234567890) для документа (1801001) будет 2.16.840.5.1.18.1234567890. 1801001.

Заголовок также должен содержать название, дату создания документа и ссылку на организацию, к которой он относится. При расширении номенклатуры документов системы необходимо вводить понятие типов документов. Значение типа документа выбирается из классификатора и имеет обязательный атрибут *code*, а идентификатор типа документа *typeID* содержит атрибуты *root*, *extension* для описания его структуры.

Элементы документа содержат набором атрибутов. Наряду с идентификацией объектов необходимо создавать классификаторы атрибутов ресурса. Следует отметить, что хаотическое развитие информационных ресурсов приводит к ситуации, когда одни и те же атрибуты кодируется по разным классификаторам. Это исключает возможность интеграции различных ИР и затрудняет использование данных. Для решения проблемы владелец ИР должен четко определить правила построения дерева объектных идентификаторов при классификации атрибутов. Тогда средства конвертации по навигационной составляющей *XML*-файла могут использовать программные средства автоматического определения атрибутов абонента для приема информации.

Программный модуль обработки *XML*-данных

Иногда владельцы информационных систем мониторинга не имеют возможности или не желают использовать *on-line* доступ к БД с применением *WEB*-сервисов. Тогда формируется файл данных определенной структуры, который передается владельцу ИР имеющимися средствами связи. Один из методов для обработки *XML*-данных при формировании ИР – программные модули *CLR* [3].

Использование *CLR*-модулей обусловлено тем, что средства *T-SQL* не имеют доступа к файловому пространству локальной сети. В принципе, было бы достаточно использовать функцию чтения данных из *XML*-файла программным модулем *CLR*, если бы не существующее ограничение на длину *XML*-поля, которое передается в *T-SQL*-процедуры. Выход – создание всей бизнес-логики на стороне *CLR*-модуля и загрузки данных с поддержкой целостности.

Схема приема-передачи *XML*-данных использует *CLR*-модуль, который активизирует сервер БД и передает параметры *XML*-файла. Алгоритм функционирования программного модуля следующий:

- поиск элементов, определяющих атрибуты абонента сообщения и содержание *XML* файла;
- активизация *T-SQL* по формированию набора данных для загрузки в БД и одновременная проверка целостности данных;
- поиск тег начала информационного тела документа;
- чтение следующих параметров: кода показателя в классификаторе; отметки начала/окончания отчетного периода; значения единицы измерения показателя; значения и формата показателя отчетности;
- загрузка данные документа в таблицы БД;
- завершение работы при нахождении конечного тега документа.

Процедура опирается на структуру *XML*-файла и при неподходящем формате или нарушении целостности данных аварийно завершает работу. Поскольку схема документа включает в себя логическую модель обмена информацией, программный модуль *CLR* анализирует синтаксис документа и сопоставляет его со схемой БД. Программный модуль обладает свойствами загрузки информации в БД из *XML*-документа в декодированной форме, а также формирует *XML*-документ из контента БД, используя соответствующий класс *XML*-документов.

Заключение. Предложенная в статье процедура импорта данных сопровождается шагами по сохранению семантики *XML*-документов на примере использования *XBRL* и формирования информационного ресурса. Семантические ограничения *XML*-схемы документа сохраняются в реляционной схеме в форме функциональных связей и многозначных зависимостей. Обратная процедура формирования *XML*-документа из контента БД предусматривает конвертацию реляционных данных в *XML*-формат. Кроме того, определена необходимость согласованности обмена данными при помощи объектных идентификаторов для навигации информационного потока по автоматизации процесса консолидации данных.