Self-Healing regulate particular network devices in order to maximise the signal and network coverage. *Assisted* examines the network in order to reduce the costs and the time necessary for the optimal frequency set-up. Automatic monitoring keeps the network on a highly efficient level and maintain its optimal set-up. Errors and alarms monitoring in the access points and security bridges improves the security level and resistance to attacks. Moreover, it checks the integrity of the network

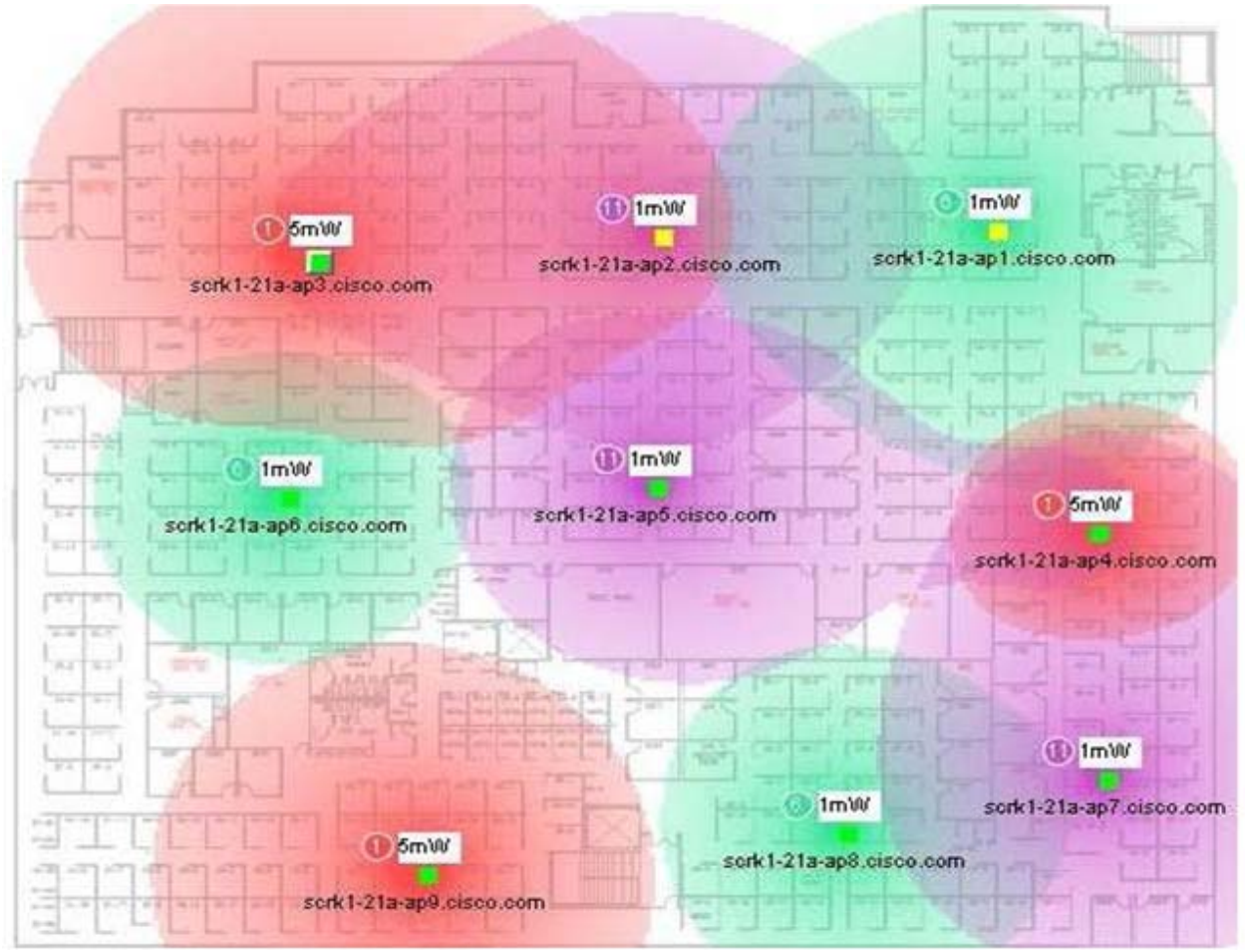

Fig. 4. Network coverage on map in the controller

References

1. Kurytnik I.P. Bezprzewodowa transmisja informacji / I.P. Kurytnik, M. Karpinski. – Warszawa: Wydawnictwo Pomiary Automatyka Kontrola. – 2008. – 228 s. – ISBN: 978-83-926319-4-1.

2. Курітник І.П. Безпровідна трансляція інформації / І.П. Курітник, М. Карпінський. – Тернопіль: Крок. – 2010. – 376 с. – ISBN 978-966-2362-16-9.

Надійшла 26.4.2011 р.

УДК 004.652.5

О.О. ЛИСЕНКО
Тернопільський національний економічний університет

ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ ОБ'ЄКТНО-ОРІЄНТОВАНИХ БАЗ ДАНИХ

У статті зроблена спроба огляду досягнень технологій об'єктно-орієнтованих баз даних, а також обговорюються недоліки, які повинні бути вирішеними, щоб об'єктно-орієнтовані бази даних стали настільки ж поширеними, як реляційні бази даних.

This paper serves as an overview on the achievements of object-oriented database technology, and also discusses the weaknesses that have to be yet resolved by the object-oriented database community before object-oriented database technology can become as widespread as relational databases.

Ключові слова: моделі даних, об'єктне програмування.

Вступ

Об'єктно-орієнтовані системи керування базами даних (ООСКБД) повстали як результат симбіозу

можливостей баз даних та об'єктно орієнтованих мов програмування.

На сьогодні найбільш поширеними у програмуванні є мови, які використовують об'єкти і тим самим роблять ООСКБД ідеально придатними для використання об'єктно-орієнтованими програмістами. Останні ж можуть розвивати продукт своєї праці, зберігати його у вигляді об'єктів, копіювати чи змінювати існуючі об'єкти для створення нових в ООСКБД. Інформація ж на сьогодні включає в себе не тільки числові чи текстові дані, але й відео, аудіо, графіки і фотографії, все те, що вважається складними типами даних. Реляційні СКБД з самого початку були не здатні у повній мірі підтримувати ці типи даних. В той же час, ООСКБД, будучи інтегрованими з мовою програмування, можуть забезпечити збереження програмного коду та інформації в одному середовищі, тому що вони використовують ту ж модель представлення.

Використання ООСКБД на сьогодні надає найбільшу перевагу тим організаціям, що зберігають складні дані як об'єкти. Зокрема, це компанії, що роблять свій бізнес на мультимедійних презентаціях, використовують системи автоматизованого проектування тощо.

Деякі з об'єктно-орієнтованих баз даних (ООБД) були призначені для спільної роботи з такими поширеними об'єктно-орієнтованими мовами програмування як Ruby, Python, Perl, Java, C #, C + +, Objective-C і Smalltalk; . інші ж використовують свої власні мови програмування. Проте усі ООСКБД дотримуються точно такої ж моделі даних, як і їх об'єктно-орієнтовані мови програмування.

ООСКБД зростала у першій половині 1970-х років в результаті дослідження можливостей збереження графічних структурованих об'єктів у власностворе ній СКБД. Термін "система керування об'єктно-орієнтованою базою даних" вперше з'явився десь біля 1985 року [1]. Реалізація великої кількості відомих науково-дослідних проектів, таких як Encore-Ob/Server (Brown University), EXODUS (University of Wisconsin– Madison), IRIS (Hewlett-Packard), ODE (Bell Labs), ORION (MCC), Vodak (GMD-IPSI), та Zeitgeist (Texas Instruments) призвела до появи численних публікацій в області ООСКБД. Особливо плідним у науковому плані став проект ORION. Вон Кім з MCC скомпілював найкращі з цих публікацій у книжку, яку було видано в The MIT Press [2].

Найбільш вагомі результати досліджень у рамках реалізації вище перелічених проектів були отримані на межі 80-х та 90-х років минулого століття такими науковцями як Т.Атвуд, А.Аткінсон, Д.Баррі, Д.Волк, Вон Кім, М.Гарвей, Р.Кеттел, Д.Макнаб, Р.Лорі, М.Рапапорт, Е.Руденштейнер, М.Стоунбракер, Д.Фішман, А.Чаудрі та іншими. На жаль, автору не вдалося відшукати роботи вітчизняних дослідників об'єктно-орієнтованих СКДБ, а от серед російських авторів, які опублікували результати своїх досліджень вже у 21-му столітті можна назвати А.Андреєва, Д.Берьозкіна, Ю.Кантоністова, С.Кузнєцова та інших.

До числа ранніх комерційних продуктів входили Gemstone, Gbase та Vbase. Вони були інтегровані з різними мовами програмування: GemStone – Smalltalk, Gbase – LISP, Vbase – COP. Протягом більшої частини 1990-х років C++ домінував на ринку як об'єкт управління об'єктно-орієнтованими базами даних. Наприкінці 1990-х була додана можливість використовувати Java і вже зовсім недавно – C#.

Починаючи з 2004 року, об'єктні бази даних вступили до свого другого періоду росту, який був спричинений широкою доступністю та простотою у використанні, тому що їх програмне забезпечення було повністю написане на таких широковідомих об'єктно-орієнтованих мовах програмування як Smalltalk, Java або C # [3].

Виклад основного матеріалу

Виникнення напрямку ООБД зумовлене перш за все потребами практики: необхідністю розробки складних інформаційних прикладних систем, для яких технології попередніх систем БД не були цілком придатними.

Звичайно, ООБД виникли не на порожньому місці. Відповідний базис забезпечили як попередні роботи в області БД, так і напрямки мов програмування з абстрактними типами даних чи об'єктно-орієнтованих мов програмування, що вже давно розвиваються.

Щодо зв'язку з попередніми роботами в області БД, то складається враження, що найбільш суттєвий вплив на дослідження в області ООБД надають опрацювання реляційних СКБД і хронологічно наступних за ними сімейств БД, в яких підтримується управління складними об'єктами [4]. Крім того, винятковий вплив на ідеї та концепції ООБД і, як здається, всього об'єктно-орієнтованого підходу надав підхід до семантичного моделювання даних.

Слід зауважити, що будь-яка СКБД грунтується на певній моделі даних. З кінця 70-х років XX-го ст. найбільшу популярність отримала реляційна модель даних. Реляційні СКБД і понині грають чільну роль на світовому ринку СКБД. Проте деяка частина розробників додатків, які використовують СКБД, висловлюють незадоволення невідповідністю реляційної моделі сьогоднішнім вимогам, що пред'являються до термінів розробки проектів, швидкості обробки запитів до баз даних [5]. Особливо це проявляється при проектуванні систем, в яких зберігаються складні неструктуровані дані. У свій час найбільші розробники СКБД фактично визнали це, і почали спішно вбудовувати у свої продукти підтримку об'єктно-орієнтованого програмування. З міркувань сумісності з колишніми напрацюваннями, лідери індустрії СКБД пропонували змішаний підхід – об'єктно-реляційний. Сталося так, що багато з поважаючих себе фірм повернулися обличчям до об'єктних технологій і продуктивно співпрацювали з розробниками об'єктно-орієнтованих СКБД. Так IBM і Oracle радикально підійшли до проблеми і переробили ядра своїх СКБД з метою додати в нього об'єктні властивості [6].

Інший шлях вибрав Informix, який придбав серйозну об'єктно-реляційну СКБД Illustra і вмонтував її в свої продукти. У результаті вийшов продукт, що дістав назву універсального серверу [7].

Проте є очевидним той факт, що мови, які підтримують складні структури даних, вимагають адекватної моделі бази даних. Незважаючи на існування об'єктних доповнень і можливостей розширення, ядро бази залишається орієнтованим на роботу з реляційними даними, що негативно позначається на продуктивності, змушуючи СКБД щоразу проводити збірку та розбирання об'єктів при обміні зі сховищем [8].

У чому ж принципова відмінність реляційних і об'єктних баз? Мері Луміс, одна з ідеологів СКБД Versant, свого часу дуже коротко і точно сформулювала актуальність об'єктного підходу до баз даних: "Модель даних ближча сутностей реального світу. Об'єкти можна зберегти і використовувати безпосередньо, не розкладаючи їх по таблицях. Типи даних визначаються розробником і не обмежені набором визначених типів ". В об'єктних СКБД дані об'єкту, а також його методи поміщаються в сховищі як єдине ціле. Об'єктна СКБД саме той засіб, який забезпечує запис об'єктів в базу даних "як є". What You have coded is what You put in database – "Все, що Ви запрограмували, Ви ставите в базу даних" – ось девіз такої СКИД [7].

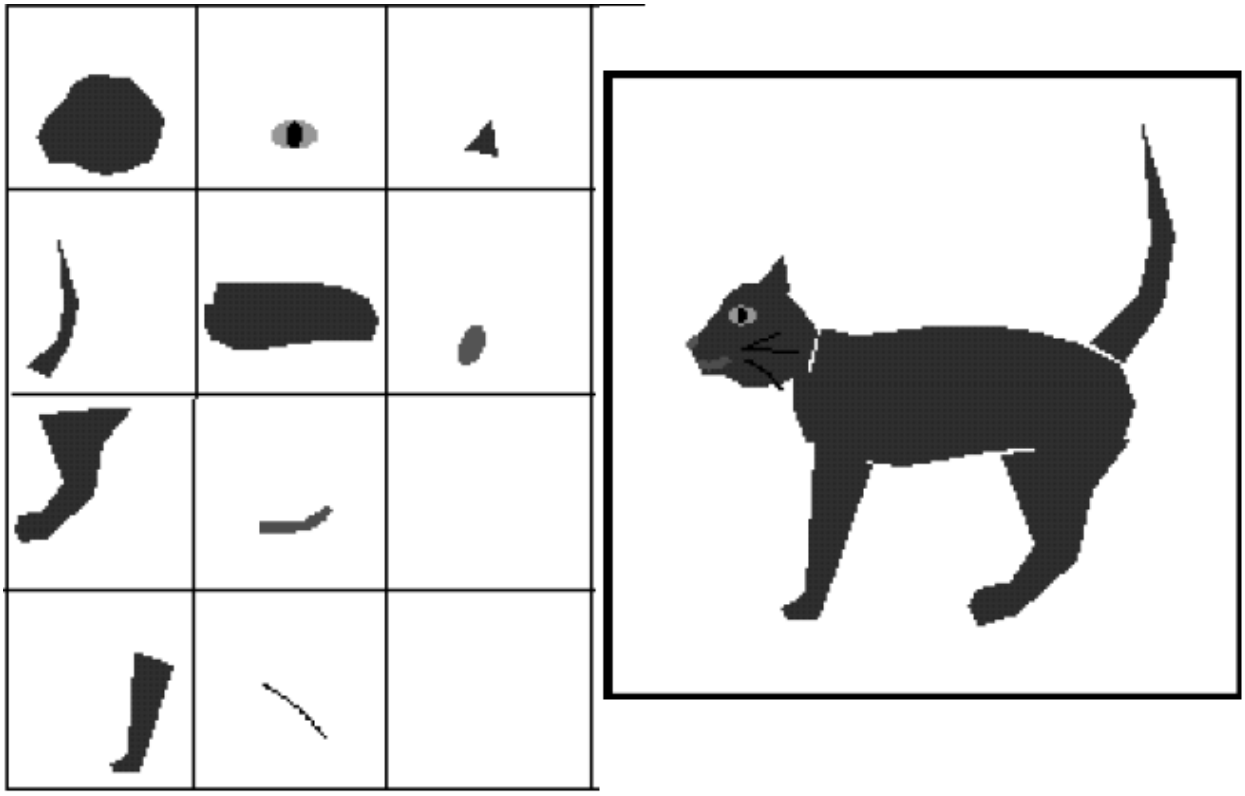

Рис 1. Схематичне зображення способу збереження об'єкту "кішка» в реляційній (зліва) та об'єктно-орієнтованій БД (справа)

Для ілюстрації вище сказаного на рис. 1 відображено, як зберігається об'єкт «кішка» в реляційній БД та об'єктно-орієнтованій. В останньому випадку вона збережена разом з методом «піймати мишу».

Зупинимося тепер на загальних поняттях об'єктно-орієнтованого підходу і його переломлення в ООБД. Як відмічається у найбільш загальній і класичній постановці [9], об'єктно-орієнтований підхід базується на наступних концепціях:

- об'єкту та ідентифікатора об'єкту;
- атрибутів і методів;
- класів;
- ієрархії і успадкуванні класів.

Будь-яка сутність реального світу в об'єктно-орієнтованих мовах і системах моделюється у вигляді об'єкту. Будь-який об'єкт при своєму створенні отримує генерований системою унікальний ідентифікатор, який пов'язаний з об'єктом протягом усього часу його існування і не змінюється при зміні стану об'єкту.

Кожен об'єкт має стан і поведінку. Стан об'єкту – це набір значень його атрибутів. Поведінка об'єкту – набір методів, що оперують над станом об'єкту. Значення атрибуту об'єкту – це також деякий об'єкт або безліч об'єктів. Стан і поведінка об'єкту інкапсульовані в об'єкті; взаємодія між об'єктами здійснюється на основі передачі повідомлень та виконанні відповідних методів.

Множина об'єктів з одним і тим же набором атрибутів і методів утворює клас об'єктів. Об'єкт повинен належати тільки одному класу, якщо не враховувати можливості успадкування. Допускається наявність примітивних наперед визначених класів, об'єкти-екземляри яких не мають атрибутів: цілі числа, рядки тощо. Клас, об'єкти якого можуть слугувати значеннями атрибуту об'єктів іншого класу, називається доменом цього атрибуту.

Допускається породження нового класу на основі вже існуючого класу – успадкування. У цьому

випадку новий клас, званий підкласом існуючого класу (суперкласу) успадковує всі атрибути і методи суперкласу. У підкласі, крім того, можуть бути визначені додаткові атрибути і методи. Розрізняються випадки простого та множинного успадкування. У першому випадку підклас може визначатися лише на основі одного суперкласу, у другому випадку суперкласів може бути декілька. Якщо в мові або системі підтримується одиничне успадкування класів, то набір класів утворює деревоподібну ієрархію. При підтримці множинного успадкування класи пов'язуються в орієнтований граф з коренем, який називається решіткою класів. Об'єкт підкласу вважається приналежним будь-якому суперкласу цього класу.

Однією з пізніших ідей об'єктно-орієнтованого підходу є ідея можливого перевизначення атрибутів і методів суперкласу в підкласі (перевантаження методів). Ця можливість збільшує гнучкість, але породжує наступну додаткову проблему: при компіляції об'єктно-орієнтованої програми можуть бути невідомі структура і програмний код методів об'єкту, хоча його клас (у загальному випадку – суперклас) відомий. Для вирішення цієї проблеми застосовується так званий метод пізнього зв'язування, що означає, по суті справи, інтерпретаційний режим виконання програми з розпізнаванням деталей реалізації об'єкту під час виконання посилки повідомлення до нього. Введення деяких обмежень на спосіб визначення підкласів дозволяє домогтися ефективної реалізації без потреб в інтерпретації.

Очевидно, що при такому наборі базових понять, якщо не враховувати можливості успадкування класів, об'єктно-орієнтований підхід дуже близький до підходу мов програмування з абстрактними (або довільними) типами даних [5].

З іншого боку, якщо абстрагуватися від поведінкового аспекту об'єктів, об'єктно-орієнтований підхід вельми близький до підходу семантичного моделювання даних (навіть і по термінології). Фундаментальні абстракції, що лежать в основі семантичних моделей, неявно використовуються і в об'єктно-орієнтованому підході. На абстракції агрегації грунтується побудова складних об'єктів, значеннями атрибутів яких можуть бути інші об'єкти. Абстракція групування – основа формування класів об'єктів. На абстракціях спеціалізації/узагальнення заснована побудова ієрархії або решітки класів.

Згадаємо, що першою формалізованої і загальновизнаною моделлю даних була реляційна модель Кодда [10]. У цій моделі, як і у всіх наступних, виділялися три аспекти – структурний, цілісний і маніпуляційний. Структури даних в реляційній моделі грунтуються на плоских нормалізованих відносинах, обмеження цілісності виражаються за допомогою засобів логіки першого порядку і, нарешті, маніпулювання даними здійснюється на основі реляційної алгебри чи рівнозначного їй реляційного числення. Останнє є основою мови запитів SQL і розширення ODBC, які широко застосовуються для під'єднання користувальницького інтерфейсу в 2-х і 3-х шарових додатках клієнт-серверної архітектури.

Як відзначають багато дослідників, своїм успіхом реляційна модель даних багато в чому зобов'язана тому, що спиралася на строгий математичний апарат теорії множин, відносин і логіки першого порядку. Розробники будь-якої конкретної реляційної системи вважали своїм обов'язком показати відповідність своєї конкретної моделі даних загальної реляційної моделі, яка виступала як міра "реляційності" системи.

Відома й деревоподібна модель, яка базується на системах, що представляють дані безпосередньо у вигляді дерева і можуть оперувати з мережевими даними (які можуть мати зв'язки "багато-до-багатьох"). З математичної точки зору ці бази даних описуються математичної теорією графів. У багатьох реалізаціях цих систем можна створити (можливо обмежене) відображення в реляційну модель, яка дозволяє використовувати SQL/ODBC як інтерфейс на рівні запитів.

Об'єктно-орієнтовані бази даних відносно нові і як і раніше досить примітивні в деяких моментах. У більшості з сучасних систем відчувається брак зручності як для користувача, так і для програміста, і сама по собі теорія баз даних не має такої гарної математичної основи як реляційні або деревоподібні моделі. З іншого боку, існують твердження, що загальна об'єктно-орієнтована модель даних у класичному розумінні і не може бути визначена через непридатність класичного поняття моделі даних до парадигми об'єктної орієнтованості.

В той же час, об'єктно-орієнтовані бази даних дозволяють представляти складні об'єкти більш безпосереднім чином, ніж реляційні системи. Системи ООБД дозволяють користувачам визначати абстракції; полегшувати проектування деяких зв'язків; усувати потреби в визначених користувачами ключах; підтримувати новий набір предикатів порівняння; в деяких випадках усувати потреби у з'єднаннях; в деяких ситуаціях забезпечувати більш високу продуктивність, ніж системи, засновані на реляційній моделі; забезпечувати підтримку версій і тривалих транзакцій. Нарешті, розроблена об'єктна алгебра – хоча, можливо, поки і не настільки детально, як реляційна алгебра.

Запинимось тепер на особливостях програмування об'єктної СКБД. Зауважимо, що створення програм для об'єктних баз істотно відрізняється від написання додатків, взаємодіючих з реляційними СКБД. Об'єктна СКБД, як правило, підтримує один або кілька об'єктно-орієнтованих мов – C + +, Java, Smalltalk, Object Lisp тощо. У своїх програмах розробники використовують об'єкти і структури, які поміщаються в базу даних. Щоб зберегти їх в базі не потрібно особливих зусиль. Творці об'єктних СКБД намагаються максимально полегшити життя розробників програм, тому збереження об'єктів забезпечується прозоро. Програміст використовує єдину мову програмування для створення логіки додатку, розробки інтерфейсу й спілкування з базою даних. У поєднанні з візуальними засобами розробки створення прикладних програм може бути проведено з мінімальними витратами коштів і часу [7].

Продемонструємо відмінність у написанні програм на невеликому прикладі. Нехай треба занести в

базу даних відомості про людину: її ім'я, вік і прізвище безпосереднього начальника. Для визначеності будемо вважати, що відомості зберігаються у реляційній таблиці PERSONAL, яка має поля NAME (ім'я), AGE (вік) і CHIEF (керівник). Щоб додати в реляційну базу людину на прізвище Іванов, 30 років, у якого керівник має прізвище Сидоров, на мові SQL треба написати приблизно такий вираз:

```
INSERT INTO personal
VALUES («Іванов», 30, «Сидоров»)
```

Мається на увазі, що відомості про Сидорова присутній в цій же таблиці, у нього теж є керівник, а у того – теж і так далі. Як бачимо, все робиться доволі просто, але коли потрібно, наприклад, отримати інформацію про усіх керівників Іванова, без створення відповідного програмного коду чи на стороні клієнта, чи на сервері (збережені процедури) тут не обійтись. А за цим стоять певні втрати вільної пам'яті чи швидкодії системи.

Ситуація, коли для програмування використовується мова високого рівня, а запити відправляються з нього в реляційну базу, не надто відрізняється від наведеної вище. Наприклад, фрагмент програми на C++, що виконує додавання нового запису до таблиці БД може бути приблизно таким:

```
TTable * pTable = new TTable ();
pTable -> Assign ("personal");
TVarRecord newrec;
newrec.Add ("Іванов");
newrec.Add (30);
newrec.Add ("Сидоров");
pTable -> AppendRecord (&newrec);
pTable ->Close ();
delete pTable;
```

Нарешті, запишемо те ж саме, але збережемо дані в об'єктній базі. Розглянутий нижче приклад ілюструє підхід, який використовується в типовій об'єктній СКБД. Дані про людину присутні в об'єкті класу TPerson, оголошеного наступним чином:

```
persistent class TPerson {
char                *m_pszName;
int                 m_iAge;
TPerson             *m_pChief;
public:
TPerson (char *pszName, int Age, TPerson * pChief) {
// реалізація конструктору
}
};
```

Звернемо увагу на слово persistent в оголошенні класу. Воно вже згадувалося вище і означає, що об'єкти типу TPerson будуть зберігатися в базі даних. Український еквівалент терміну persistent – "стабільний ", тому далі будемо називати такі об'єкти стабільними. Стабільність – це властивість об'єкту зберігати стан між сеансами роботи програми. Об'єктна база даних – це, по суті, сховище стабільних об'єктів. Досягається це, залежно від реалізації, або введенням у мову програмування нового ключового слова, або наданням спеціального методу для створення об'єктів, або успадкуванням від спеціального зумовленого типу. Програмісту, таким чином, досить оголосити об'єкт "стабільним ", а далі СКБД бере на себе чорнову роботу з відстеження змін, скасування посилань на віддалені об'єкти, створення версій об'єктів. Додавання нового об'єкту, в якому зібрані відомості про Іванова, в нашому випадку виглядає так:

```
TPerson * pIvanov = new TPerson ("Іванов", 30, pSydorov);
```

Особливу увагу слід звернути на останній параметр. У об'єктному СКБД посилання на безпосереднього начальника зберігається у вигляді вказівника. Для переходу до об'єкту "Сидоров" не виконується ніякого пошуку.

Крім стабільності, об'єктна СКБД надає широкий набір засобів управління базою. Серед них слід відмітити: управління транзакціями, коли програміст на рівні початкового тексту задає початок і кінець процесу трансакції та блокування бази або набору об'єктів, які в ній збережені.

На стороні клієнта можна дізнатися, як протікає ініційований клієнтом запит. Отримавши таку інформацію, користувач або буде чекати завершення процесу, або перерве його. Як мова запитів, використовуються різні реалізації Object Query Language (OQL) – об'єктної мови запитів. Це, як правило, розширення SQL, доповнене об'єктними властивостями, засобами опису типів даних, ітерації по об'єктах в СКБД.

Рис. 2. Об'єднання можливостей СКБД та об'єктного програмування [7].

Цілком природнім є той факт, що переваги об'єктних СКБД накладені на переваги об'єктних мов програмування створюють нову, значно вищу, якість продукту (рис. 2).

Як відомо, успадкування, інкапсуляція даних та поліморфізм є найважливішими механізмами об'єктно-орієнтованого програмування і подальших коментарів не потредують.

Ідентифікація об'єктів – це присвоєння об'єкту унікального ідентифікатора, таким чином, кожен об'єкт в СКБД унікальний. Зазвичай унікальний ідентифікатор невидимий, його "знає" тільки СКБД, але він не змінюється ні за яких обставин, не дивлячись на модифікацію об'єкту аж до його видалення.

Тепер коротенько пояснимо термінологію, що використовується для опису функціональності об'єктної бази даних.

Під цілісністю даних розуміємо як структурну, так і логічну цілісність бази даних. Крім того, для об'єктних СКБД важлива підтримка відповідності між об'єктами в базі і об'єктами, переданими клієнту.

Паралелізм – це механізм, який гарантує успішне розв'язання конфліктів, що виникають при одночасному доступі до одних і тих же даних.

З того, що про стабільність вже згадувалося раніше відомо, що це властивість СКБД зберігати дані програми між сеансами роботи. Стосовно ж до об'єктним СКБД вона означає збереження даних об'єктів, оголошених як стабільні.

Відновлюваність – це обов'язковий атрибут СКБД. Система повинна адекватно реагувати на збої в прикладних програмах, які виконують обмін даними з базою, збої операційної системи і пошкодження носіїв інформації. Ведення журналу змін, транзакції, зберігання надлишкових даних забезпечують достатню надійність роботи бази даних.

Транзакції – це механізм, що забезпечує фіксацію стану бази на момент початку будь-якої критичної операції і повернення до цього стану у разі невдалого виконання операції.

Безпека даних є одним з ключових понять у світі СКБД, оскільки часто в базах зберігається конфіденційна, а іноді й секретна інформація.

У будь-якій базі даних обов'язково повинен бути присутнім механізм виконання запитів. Мова запитів повинна бути простою у користуванні та залежати від структури бази даних.

Версійність забезпечує підтримку на рівні СКБД багатьох версій того самого об'єкту. Наприклад, дані об'єкту можуть багаторазово змінюватися, а в базі зберігатимуться всі версії об'єкту.

Архівування дозволяє задіювати засоби програмного стиснення даних коли в базі нагромаджуються надто великі обсяги інформації [7].

Слід зауважити, що більшість авторів цитованих видань переважно позитивно оцінюють перспективи об'єктно-орієнтованої моделі БД. В той же час деякі дослідники налаштовані більш скептично. Так, Сіха Багуї [11] розчарований, що об'єктно-орієнтовані методи не виправдали надій щодо створення свого роду квантового переходу у технології баз даних.

Він вважає, що в об'єктно-орієнтованих базах даних відсутні базові засоби, до яких користувачі систем баз даних звикли і тому очікують бачити. Серед іншого негативу відзначається: відсутність інтероперабельності між реляційними БД і ООБД; мінімальна оптимізація запитів; відсутність стандартної алгебри запитів; відсутність засобів забезпечення запитів; відсутність підтримки уявлень; проблеми з безпекою; відсутність підтримки динамічних змін визначень класів; незадовільна підтримка обмежень

цілісності; обмежені можливості налаштування продуктивності; недостатня підтримка складних об'єктів; обмежена інтеграція з існуючими об'єктно-орієнтованими системами програмування; обмежений виграш в продуктивності. тощо.

У більшості своїй спростовує негативні висновки [11] відомий російський дослідник ООБД і редактор перекладу статті – С. Кузнєцов. Останній пояснює висновки Сіха Багуї застарілими джерелами цитувань, відмінностями у термінології ООБД та фактом невеликої кількістю публікацій з означеної теми протягом нульових років XX-ст., що дехто може сприйняти як занепад технології, а інші – як перехід технології до зрілого та стійкого стану [11].

Висновки

Об'єктно-орієнтований підхід до систем БД був широко розрекламований у 80-90 роках минулого століття як майбутня парадигма, яка замінить реляційну модель через кілька років. Він дійсно дає велику гнучкість в порівнянні з реляційною моделлю і дозволяє зосередити увагу при розробці та документуванні на описі окремих типів предметів, а не на загальній структурі бази даних. Це обіцяє бути особливо корисним при побудові складних систем БД, які мають різні типи даних, особливо, якщо часто зустрічаються зв'язки "багато-до-багатьох", які за своєю природою не дуже підходять для таблиць.

На жаль, об'єктно-орієнтований підхід все ще не настільки широко випробуваний і не настільки "дозрів", як реляційна чи деревоподібна моделі. Не останню роль у практичній відсутності досліджень з ООБД у наші дні відіграє й часто не виправдана критика об'єктно-орієнтованого програмування вцілому [12].

Література

1. Three example references from 1985 that use the term: T. Atwood, "An Object-Oriented DBMS for Design Support Applications, " Proceedings of the IEEE COMPINT 85, pp. 299-307, September 1985.

2. Kim, Won. Introduction to Object-Oriented Databases. The MIT Press, 1990. ISBN 0-262-11124-1.

3. Object database: (From Wikipedia, the free encyclopedia): [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Object_database.

4. Raymond Lorie, Won Kim, Dan McNabb, Wil Plouffe. Supporting Complex Objects in a Relational System for Engineering Databases // In Query Processing in Database Systems, ed. Won Kim, David S. Reiner, Dan. S. Batory, Springer-Verlag, 1985. – 145-155.

5. Сергей Кузнєцов. Объектно-ориентированные базы данных – основные концепции, организация и управление: краткий обзор: [Електронний ресурс]. – Режим доступу: http://citforum.ru/database/articles /art_24.shtml.

6. Oracle8. Энциклопедия пользователя.: Пер. с англ./Компания Advanced Information System и др. К: Издательство «Диасофт», 1998. – 864 с.

7. Андреев А.М. Объектно-ориентированные базы данных: среда разработки программ плюс хранилище объектов / А.М. Андреев, Д.В. Березкин, Ю.А. Кантонистов: [Електронний ресурс]. – Режим доступу: http://www.inteltec.ru/publish/articles/objtech/oodbms_o.shtml.

8. Chaudhri, A.B. & Osmon, P. (1996) Databases for a New Generation in "Object Expert" March/April '97 pp 33-38.

9. Won Kim. Object-Oriented Databases: Definition and Research Directions // IEEE Trans. Data and Knowledge Eng. – 2, N 3. – 1990. – 327-341.

10. E. F. Codd. A Relational Model of Data for Large Shared Data Banks // Commun. ACM. – 26, N 1. – 1970. – 377-387.

11. Сиха Багуи. Объектно-ориентированные базы данных: достижения и проблемы: Редактор перевода – С.Д. Кузнецов [Електронний ресурс]. – Режим доступу: http://www.osp.ru/text/print/article/184042.html.

12. Игорь Савчук. Почему объектно-ориентированное программирование провалилось? [Електронний ресурс]. – Режим доступу: http://citforum.ru/ gazeta/165/#comments.

Надійшла 27.4.2011 р.