УДК 681.3

М. М. ІВАНЮК
Хмельницький національний університет

СТАНОВЛЕННЯ МОДЕЛЮВАННЯ БАЗ ДАНИХ І СУБД

Обґрунтовується концепція, згідно з якою дані мають бути організовані в бази даних з метою адекватного відображення змінливого реального світу і задоволення інформаційних потреб користувачів.

Сприйняття реального світу можна зіставити з послідовністю різних, хоч інколи і взаємозв'язаних, явищ. З давніх давен люди намагались описати ці явища (навіть і тоді, коли їх не розуміли). Такий опис називають "даними". Традиційно фіксування даних здійснювалось з допомогою конкретного способу спілкування (наприклад, з домогою природньої мови або зображень) на конкретному носії (наприклад, камені або папері). Зазвичай дані (факти, явища, події, ідеї або предмети) і їх семантика (інтерпретація) фіксуються сумісно, оскільки природна мова є досить гнучкою для їх представлення. Для прикладу звернемось до твердження "Вартість авіоквитка 128". Тут "128" – дане, а " Вартість авіоквитка " – його семантика. Нерідко дані і семантика виокремленні. Наприклад, "Розклад руху літаків" може бути представлений у вигляді таблиці, у верхній частині якої (окремо від даних) наводиться їх семантика. Таке виокремлення ускладнює работу з даними (спробуйте швидко одержати інформаціцю з нижньої частини таблиці).Застостосування ЕОМ для ввдення і обробки даних зазвичай призводить ще до більшого виокремлення даних від семантики. ЕОМ має справу головним чином з даними. Велика частина інтерпретованої інформації взагалі не фіксується в явній формі (ЕОМ не "знає" «21.50» є вартістю авіобілету чи часом відльоту). У зв'язку з чим це трапилось?

Існує принаймні дві історичні причини, за якими застостосування ЕОМ призвело до виокремлення даних від семантики. По-перше, ЕОМ не мали достатніх можливостей для обробки текстів на природній мові – основній мові інтерпретації даних. По-друге, вартість пам'яті на початку створення ЕОМ була досить значною. Пам'ять використовувалась для збереження самих даних, а семантика традиційно покладалась на користувача. Користувач вводив семантику даних в свою програму, яка "знала", наприклад, що шосте введене значення зв'язане з часом прибуття літака, а четверте – з часом його відльоту. Це суттєво підвищувало роль програми, оскільки поза семантикою дані являли собою лише певну сукупність бітів на запам'ятовуючому пристрої.

Розклад руху літаків - жорстка залежність між даними і використовуючими їх программами, що призводить до значних проблем в роботі з ними. Зустрічаються випадки, коли користувачі однієї і тієї ж самої ЕОМ створюють і використовують в своїх програмах разні набори даних, що містять подібну інформацію. Інколи це викликалось тим, що користувач не знає (або не забажав дізнатись), що в сусідній комнаті або за сусіднім столом сидить співробітник, який вже давно ввів в ЕОМ необхідні дані. Масу проблем породжує також сумісне використання одних і тих же даних.

Розробники прикладних програм (написаних, наприклад, на Бейсіку, Паскалі або Сі) розміщують необхідні їм дані у файлах, організовуючи їх найбільш сприятливим для себе способом. При цьому одні і ті ж дані можуть мати в різних застосуваннях зовсім іншу організацію (різну послідовність розміщення в записі, різні формати одних и тих же полів і т.п.). Усуспільнити такі дані надто важко. Наприклад, будь-яка зміна структури запису файла, що проводиться одним із розробників, призводить до необхідності зміни іншими розробниками тих програм, які використовують записи цього файла.

Активний пошук зручних способів усуспільненя неперервно зростаючого об'єму інформації призвів до створення на початку 60-х років спеціальних програмних комплексів, названих "Системами управлення базами даних" (СУБД). Основна особливість СУБД – наявність процедур для введення і зберігання не лише самих даних, але і опису їх структури. Пойменовану, структуровану сукупність взаємопов'язаних даних, які характеризують окрему предметну область і перебувають під управлінням СУБД, стали називати "Банки даних", а потім "Бази даних" (БД). Бази даних – інтегроване сховище даних (файл), яке використовуються багатьма споживачами і забезпечує незалежність даних від прикладних програм .

Нехай, наприклад, вимагається зберігати розклад руху літаків і ряд інших даних, пов'язаних з організацією работи аеропорту (БД "Аеропорт"). Використовуючи для цього одну із сучасних СУБД, можна підготувати опис розкладу

СТВОРИТИ ТАБЛИЦЮ	Розклад
Номер_рейсу	Ціле
Дні тижня	Текст (8)
Пункт відправлення	Текст (24)
Час відльоту	Час
Пункт призначення	Текст (24)
Час прибуття	Час
Тип літака	Текст (8)
Вартість квитка	Валюта

і ввести його разом з даними в БД "Аеропорт".

Мова запитів СУБД дозволяє звертатись до даних як з програм, так і з терміналів. Сформувавши запит

```
ВИБРАТИ Номер_рейсу, Дні тижня, Час відльоту
ІЗ ТАБЛИЦІ Розклад
ДЕ Пункт відправлення = 'Львів'
І Пункт призначення = 'Київ'
І Час відльоту > 17,
```

одержимо розклад "Львів-Київ" на вечірній час, а за запитом

```
ВИБРАТИ КІЛЬКІСТЬ (Номер_рейсу)
З ТАБЛИЦІ Розклад
ДЕ Пункт відправлення = “Київ”
І Пункт призначення = “Житомир”
```

одержимо кількість рейсів "Київ- Житомир".

СУБД надає можливість доступу до даних будь-якому користувачу, включаючи і тих, які практично не мають і (або) не бажають мати уявлення про: фізичне розміщення в пам’яті даних та їх описів; механізми пошуку запитуваних даних; проблеми, які виникають при одночасному запиті одних і тих же даних багатьма користувачами (прикладними програмами); способи забезпечення захисту даних від некоректних оновлень і (або) несанкціонованого доступу; підтримка баз даних в актуальному стані та в багатьох інших функціях СУБД. Більшість доступних на сьогоднішній день СУБД, такі як Oracle, Informix, InterBase, DB2, MS SQL Server, MySQL та інші можуть управляти не лише декількома колонками, рядками і таблицями в базі даних, але і декількома базами.

Характерною ознакою першого типу СУБД, який в 1960-х роках отримав стандартне використання при обробці даних, була обробка файлів. Реальні дані зберігались в масивах, які за своєю основою були текстовими. При збільшенні розмірів таких файлів швидкість і ефективність доступу до них падала. На початку 1970-х років обробка файлів була замінена СУБД ієрархічного і мережевого типу. Ієрархічні СУБД для зберігання даних використовували структуровані дерева, а мережеві СУБД - записи. Обидва типи СУБД забезпечували більш швидкий і ефективний доступ до стабільної бази даних . Однак вони не забезпечували незалежності даних. У зв’язку з цим їх з часом витіснили так звані реляційні СУБД.

Існує чотири основних типи користувачів СУБД: адміністратор бази даних, системний адміністратор, розробник додатків і користувач додатків. Адміністратор БД – це людина, яка несе загальну відповідальність за функціонування СУБД. Системний адміністратор несе відповідальність за операційну систему і машину, на якій виконується СУБД. Розробник додатків створює додатки, з допомогою яких здійснюється доступ до СУБД. Користувач додатків – це людина, яка використовує додаток для доступу до даних в СУБД і виконує певні маніпуляції над даними. Всі користувацькі додатки, з допомогою яких здійснюється доступ до СУБД, розглядаються як клієнти, а сама СУБД вважається сервером. Клієнт/серверна обробка даних звична у світі СУБД, оскільки сервер створює свій власний процес, який виконується весь день і очікує запити від клієнтів. На основі запиту клієнта СУБД виконує одну із чотирьох основних задач: вибірка, вставка, модифікація і видалення даних.

Створення та впровадження в практику сучасних інформаційних систем висуває нові задачі проектування, які неможливо розв’язати традиційними прийомами та методами.

Моделювання баз даних – це ітераційний, багатоетапний процес прийняття обгрунтованих рішень в процесі аналізу інформаційної моделі предметної області, вимог до даних з боку прикладних програмістів і користувачів, синтезу логічних і фізичних структур даних, аналізу та обгрунтування вибору програмних і апаратних засобів. Етапність проектування даних пов’язана з багаторівневою організацією даних (рис. 1).

Очевидно, що побудову бази даних необхідно розпочинати з аналізу предметної області та з’ясуванню вимог до неї окремих користувачів (співробітників організації, для яких створюється база даних). При цьому, в першу чергу, необхідно вивчити усю первинну та вихідну документацію з точки зору визначення того, які саме дані потрібно зберігати в БД.

Рис. 1. Рівні моделей даних предметної області

Об'єднуючи частинні уявлення про зміст бази даних, що одержуються в результаті опитування користувачів, а також свої уявлення про дані, які знадобляться в майбутніх застосуваннях, АБД створює спочатку узагальнений неформальний опис створюваної бази даних. Цей опис, що виконується з допомогою природної мови, математичних формул, таблиць, графіків і інших засобів, зрозумілих усім людям, які працюють над проектуванням бази даних, називають інфологічною моделлю даних (рис. 1). Інфологічна модель – засіб структуризації предметної області й розуміння концепції семантики даних. Така людино-орієнтована модель цілком незалежна від фізичних параметрів середовища зберігання даних. Врешті-решт цим середовищем може бути пам'ять людини, а не ЕОМ.

Інфологічна модель не повинна змінюватись до тих пір, доки якісь зміни в реальному світі не викличуть в ній змін деякого поняття. Решта моделей, зображених на рис. 1 – комп'ютерно-орієнтовані. За допомогою них СУБД надає можливість програмам і користувачам здійснювати доступ до даних лише за їх іменами, не зважаючи на місце їх розташування. Необхідні дані відшукуються СУБД на зовнішніх запам'ятовуючих пристроях за фізичною моделлю даних.

Оскільки вказаний доступ здійснюється з допомогою конкретної СУБД, то моделі мають бути описані на мові опису даних цієї СУБД. Такий опис, що створюється АБД за інфологічною моделлю даних, називають даталогічною моделлю даних.

Наприкінці 60-х років з'явились роботи, в яких обговорювались можливості застосування різних табличних даталогічних моделей даних, тобто можливості використання звичних і природних способів представлення даних. Найбільш вагомою з них була стаття співробітника фірми IBM д-ра Э.Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), в якій вперше був вжитий термін "реляційна модель даних". Будучи математиком за освітою, Э.Кодд запропонував використовувати для обробки

даних апарат теорії множин (об'єднання, переріз, різницю, декартовий добуток). Він показав, що будь-яке представлення даних зводиться до сукупності двовимірних таблиць особливого виду, відомого в математиці як відношення – relation (англ.). Найменьша одиниця даних реляційної моделі – це окреме атомарне (неподільне) для даної моделі значення даних. Справді, в одній предметній області прізвище, ім'я та по батькові можуть разглядатись як єдине значення, а в іншій – як три різних значення. Множину атомарних значень одного і того ж типу називають доменом. Суть доменів полягає в наступному. Якщо значення двох атрибутів береться з одного й того ж домена, то, ймовірно, мають місце порівняння, які використовують ці два атрибута (наприклад, для організації транзитного рейсу можна дати запит "Видати рейси, в яких час вильоту з Москви в Сочі більше часу прибуття із Архангельска в Москву". Якщо ж значення двох атрибутів береться з різних доменів, то їх порівняння, ймовірно, не має змісту: чи варто порівнювати номер рейсу з вартістю білета?

Відношення на доменах D1, D2, ... (не обов'язково, щоб всі вони були різні) складається з рядка заголовка й тіла. На рис. 2 наведений приклад відношення для розкладу руху літаків. *Заголовок* складається з такої фіксованої множини атрибутів A1, A2, ..., в якій існує взаємо однозначна відповідність між цими атрибутами та визначаючими їх доменами Di (i=1,2,...).

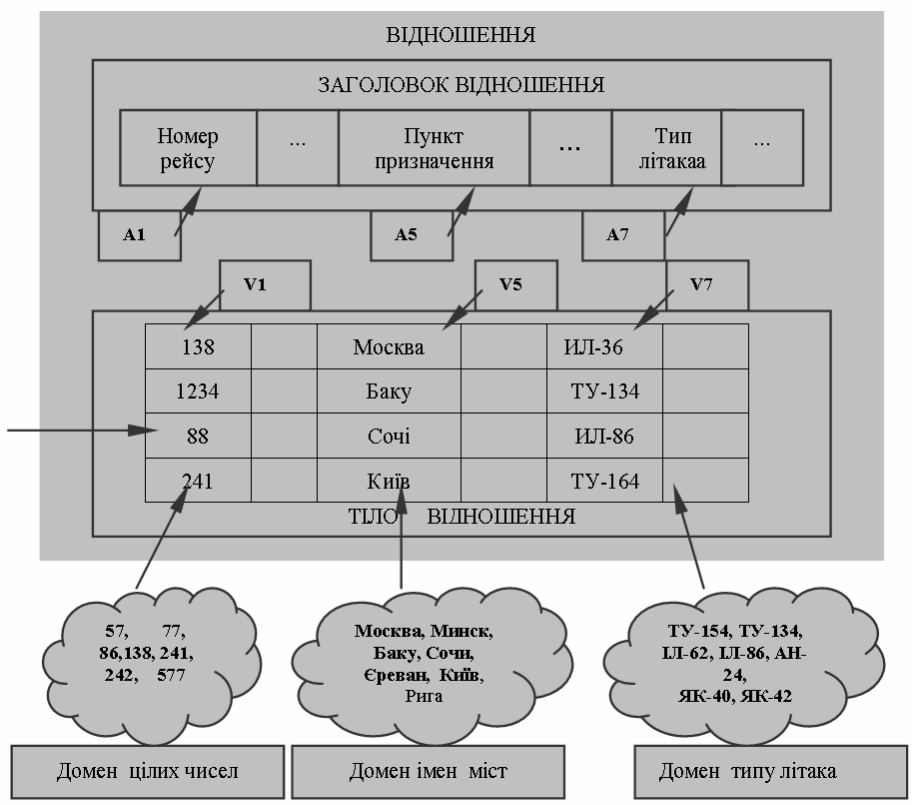

Рис. 2. Відношення з математичної точки зору

Тіло складається зі змінної в часі множини *кортежей*, де кожний кортеж складається в свою чергу з множини пар атрибут-значення (Ai:Vi), (i=1,2,...), по одній такій парі для кожного атрибута Ai у заголовку. Для будь-якої заданої пари атрибут-значення (Ai:Vi) Vi є значенням з єдиного домену Di, який пов'язаний з атрибутом Ai. *Степінь відношення* – це число його атрибутів. Відношення степеня один називають унарним, степеня два – бінарним, степеня три – тернарним, ..., а степеня n – n-арним. Степінь відношення "Рейс" . *Кардинальне число* або *потужність відношення* – це число його кортежей. Кардинальне число відношення змінюється у часі на відміну від його степеня. Оскільки відношення – це множина, а множина за означенням не має співпадаючих елементів, то ніякі два кортежи відношення не можуть бути дублікатами один одному у будь-який навмання вибраний момент часу. Нехай R – відношення з атрибутами A1, A2, ..., An. Кажуть, що множина атрибутів K=(Ai, Aj, ..., Ak) відношення R є можливим ключом R тоді і тільки тоді, коли виконуються дві незалежних від часу умови: *унікальність* (в довільний момент часу ніякі два різних кортежи R не мають одного й того ж значення для Ai, Aj, ..., Ak); *мінімальність* (ні один із атрибутів Ai, Aj, ..., Ak не може бути виключений з K без порушення унікальності).

Кожне відношення має, принаймні, хоча б один можливий ключ, оскільки, як правило, комбінація всіх його атрибутів задовільняє умову унікальності. Один із можливих ключів (вибраний довільним шляхом) приймається за його первинний ключ. Інші можливі ключі, якщо вони мають місце, називаються альтернативними ключами.

Вищезгадані та деякі інші математичні поняття служать теоретичною базою для створення основ теорії проектування баз даних та реляційних СУБД, розробки відповідних мовних засобів і програмних систем з високою продуктивністю. Проте для масового користувача реляційних СУБД найбільш успішними виявились

використання неформальних еквівалентів цих понять: Відношення – Таблиця (інколи Файл), Кортеж – Рядок (інколи Запис), Атрибут – Стовпець, Поле. При цьому вважається, що "запис" означає "екземпляр запису", а "поле" - "ім'я й тип поля".

Список використаних джерел

1. Автоматизированные информационные технологии в экономике: Учебник / Под ред. Г.А. Титоренко. – М.: Комоютер, ЮНИТИ, 1998. – 400 с.
2. Бойко В. В., Савинков В. М. Проектирование информационной автоматизированной системы на основе СУБД. – М.: Финансы и статистика, 1982.
3. Гетц К., Джильберт М. Программирование в Microsoft Office: Полное руководство по VBA. – К.: Издательская группа BHV, 2007.
4. Дейт К. Введение в системы баз данных. – М.: Наука, 1998. – 520 с.
5. Єрьоміна Н.В. Проектування баз даних. – К.: КНЕУ, 1998. – 208 с.