**Лекція №10. Постреляційна, багатовимірна і об'єктні моделі даних**

**Постреляційна модель даних.**

***Постреляційна структура даних*.**

Важливим аспектом традиційної реляційної моделі даних є той факт, що елементи даних, які зберігаються на перетині рядків і стовпців таблиці, повинні бути неподільні і єдині. Це значить, що дані не можуть бути розгорнуті в процесі подальшої обробки. Таке правило було закладено в основу реляційної алгебри при її розробці як математичної моделі даних. Подальші дослідження показали, що існує ряд випадків, коли обмеження класичної реляційної моделі суттєво заважають ефективній реалізації додатків.

В основі проблеми лежать три питання:

* робота з полями змінної довжини й групами записів;
* керування відносинами між таблицями й полями;
* відбиття семантичного змісту реальних структур, які будуть змодельовані базою даних.

Основний принцип реляційної моделі – усувати повторювані поля й групи за допомогою процесу, який називається *нормалізацією*.

Досвід розробки прикладних інформаційних систем показав, що відмова від цієї установки веде до якісно корисного розширення моделі даних. Якщо допустити, що значення даних може саме складатися з підзначень, то в результаті виникає поняття багатозначного поля. Найпростіше розглядати набір багатозначних полів у таблиці як самостійну *вкладену* (nested) *таблицю*.

За умови, що вкладена таблиця задовольняє загальним критеріям (наприклад має унікальний ключ), природно відбувається розширення операторів реляційної алгебри. Така модель даних була названа *постреляційною*.

Багато методологій проектування даних дозволяють визначати багатозначні поля й потім видаляти в процесі нормалізації. При цьому таблиці перетворяться в першу нормальну форму або 1NF. Однак видалення багатозначних полів не завжди сприяє поліпшенню прикладних програм. У випадках, коли звичайна форма доступу до поля має на увазі звернення до всіх його значень, базі даних з таблицями в 1NF прийдеться проробити операцію *з’єднання* (Join) щоразу, коли потрібно одержати відповідні значення, що зберігаються в іншій таблиці. Зовсім очевидно, що в подібній ситуації зберігання значень фізично в багатозначних полях може забезпечити більш ефективний доступ до інформації.

На рис.1. наочно продемонстровані переваги зберігання даних у базах uniVerse, що відносяться до непершої нормальної форми, перед більш громіздким зберіганням у базах даних першої нормальної форми. Приклад представляє зберігання частини даних такого типового документа, як накладна.

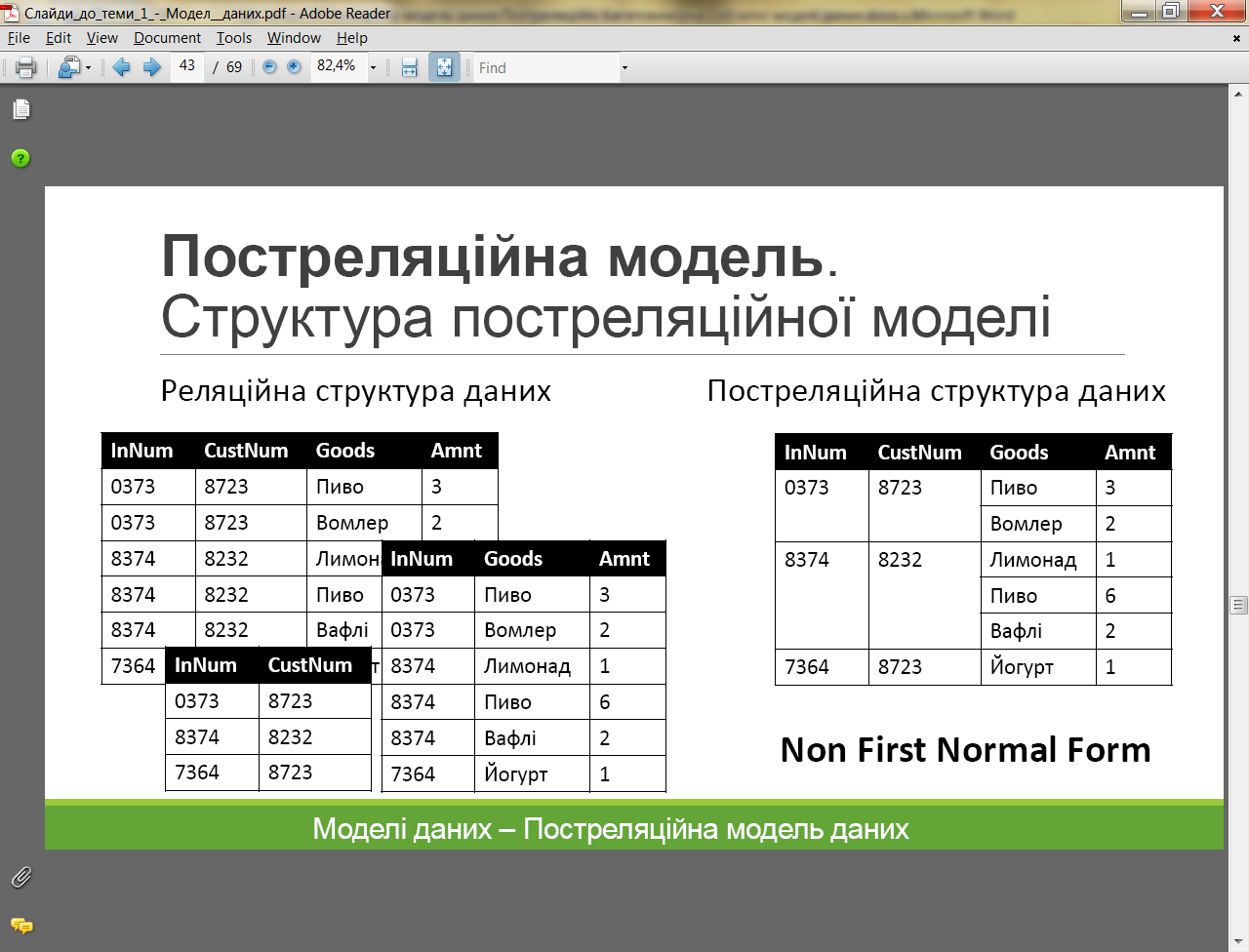


Рисунок 1. – Структури збереження даних з багатозначними полями в uniVerseі в традиційних системах.

Тоді як в uniVerse можна створювати багатозначні поля змінної довжини, всі замовлення можуть зберігатись в одній таблиці. При цьому для зберігання необхідно менше місця та при обробці не вимагається об’єднувати дані, що підвищує ефективність збереження даних.

Про таблиці, що містять багатозначні поля, говорять, що вони *перебувають у непершій* нормальній формі, або NF2 (Non First Normal Form). Як було помічене раніше, за умови, що використовувані поля підкоряються певним правилам, що дозволяють звертатися до них, як до таблиці, вбудовані в інші таблиці, форма NF2 не порушує принципи реляційної алгебри. Більше того, така інформація повністю доступна, тому що розширені оператори, які працюють із таблицями NF2, дозволяють витягати вбудовані таблиці й розглядати дані як інформацію, що зробила з таблиць 1NF. І все-таки в багатьох випадках форма 1NF буде скоріше виключенням, а не правилом. У більшості випадків набагато більш ефективно здійснювати доступ до багатозначних полів одночасно з іншими даними, знаючи, що їх завжди можна витягати й розглядати як окрему таблицю в тих випадках, коли це може знадобитися.

Модель даних СУБД uniVerse фірми Ardent підтримує асоційовані багатозначні поля, які часто називають множинними групами. Тобто ви можете зв’язати кілька стовпців із множинними значеннями в єдине ціле, яке називається асоціацією. При цьому в рядку перше значення одного стовпця асоціації відповідає першим значенням усіх інших стовпців асоціації, у такому ж зв’язку перебувають усі другі значення стовпців і т.д.

### *Операції над даними*.

Багатозначні поля й асоціації не можуть вкладатися друг у друга. Розширення синтаксису SQL дозволяє здійснювати доступ до множинних полів як розширення реляційної моделі, але, звичайно, можна застосовувати також стандартні й інша мова запитів uniVerse – Retrieve. У цілому багатозначність полів є дуже корисною властивістю при створенні комерційних додатків, де інформація нерідко представлена у вигляді списків предметів.

Приклади використання оператора Select для вибірки даних з реляційних і постреляційних таблиць.

а) SELECT

INVOICES.INVNO, CUSTNO, GOODS, QTY FROM

INVOICES, INVOICE. ITEMS WHERE

INVOICES.INVNO= INVOICES.ITEMS.INVNO;

б) SELECT INVNO, CUSTNO, GOODS, QTY

FROM

INVOICES;

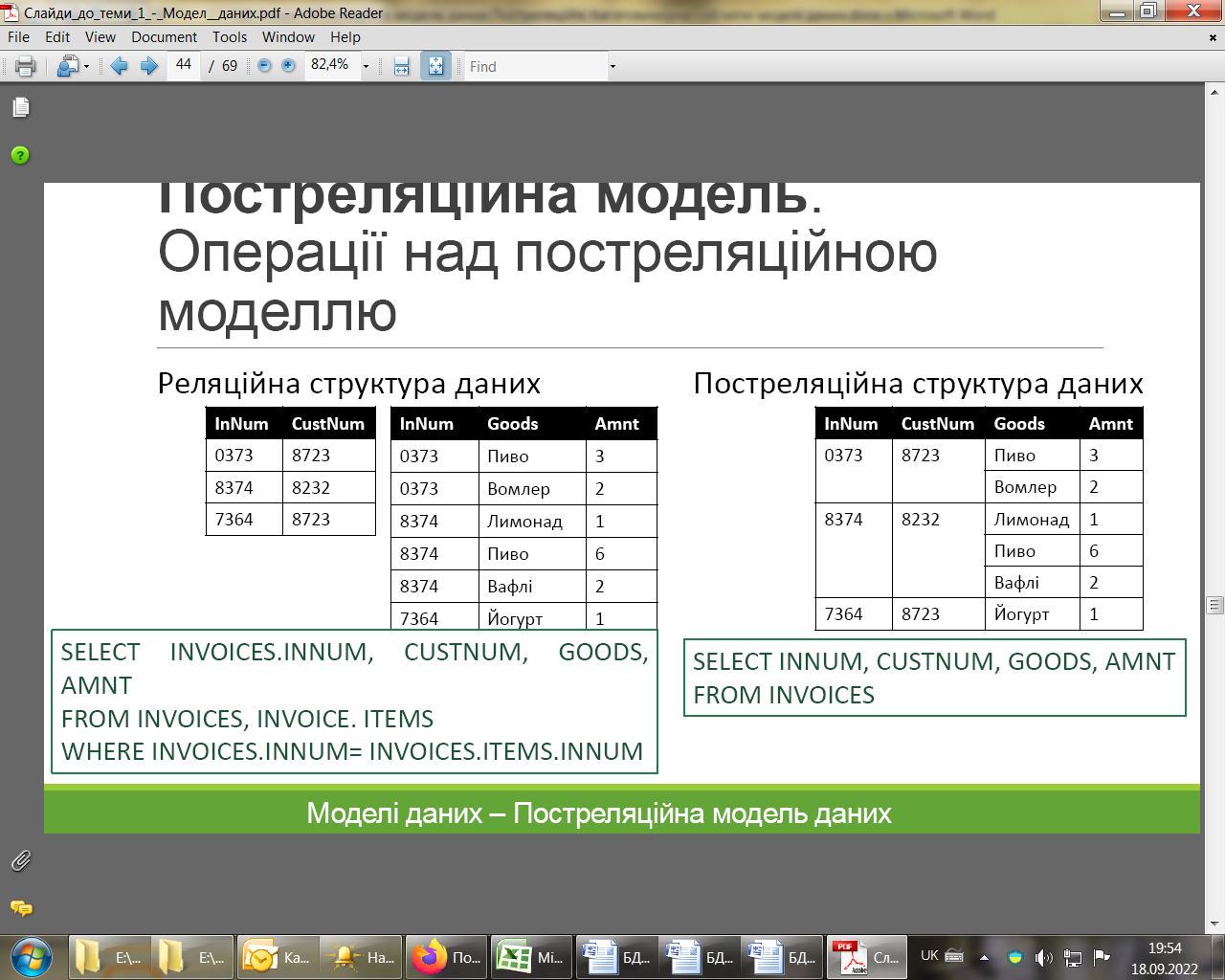


Рисунок 2 – Приклади використання оператора Select

Модель даних uniVerse являє собою розширену форму реляційної моделі даних, яка допускає використання даних у формі NF2. У цій моделі всі дані зберігаються у формі таблиць, як і у звичайній реляційній базі даних. Єдина відмінність полягає в тому, що таблиці NF2 можуть містити вкладені таблиці.

Як і в більшості реляційних баз даних, таблиці описані в словниках, що містять логічні описи, які представляють стовпці таблиць. Можна сказати, що реляційні й постреляційні СУБД різняться тільки способами зберігання й індексування даних, у всім іншому – це те саме.

Звичайно компанії автоматично вибирають реляційну модель, не даючи собі праці пошукати більш продуктивний спосіб обробки даних. Як тільки вони починають аналізувати свої проблеми, з’ясовується, що ефективність постреляційної технології в керуванні більшими масивами швидко мінливої інформації, наприклад у бізнес-додатках, набагато вище.

Крім того, uniVerse не вимагає, щоб дані в поле були певної довжини або щоб кількість полів у записі було фіксовано. Це означає, що всі дані й таблиці відрізняє більша гнучкість щодо розміру і їх можна легко видозмінювати.

### *Структура вкладених таблиць*.

Використання вкладених таблиць повністю відбиває основне завдання реляційної моделі – забезпечення користувача простою логічною структурою даних. Фактично, вкладені таблиці ще більш спрощують логічне подання даних і є природнім розширенням реляційної моделі. Розширення реляційної бази даних пропоноване фірмою Ardent дозволяє використовувати вкладені таблиці як атрибути із множинними значеннями й зв’язані групи таких атрибутів.

Для створення таблиці  може бути використане наступне SQL речення:

**CREATE TABLE**ORDER

(INNUM INT NOT NULL PRIMARY KEY,

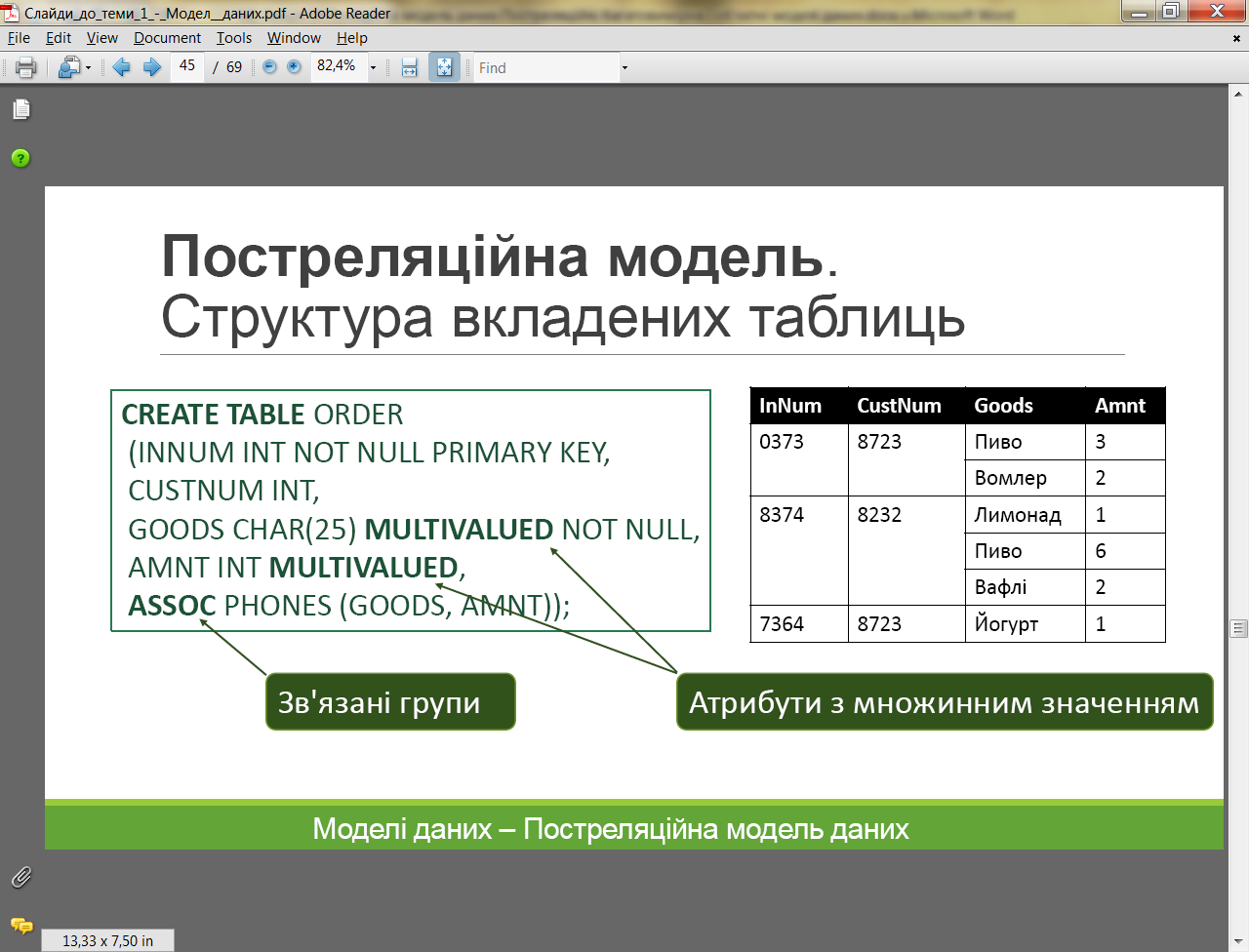
CUSTNUM INT,

GOODS CHAR(25) MULTIVALUED NOT NULL,

AMNT INT MULTIVALUED,

ASSOC PHONES (GOODS, AMNT));

У цьому випадку атрибути із множинними значеннями визначаються словом MULTIVALUED, а зв’язані групи – словом ASSOC. Звичайно, якщо необхідне подання даних у першій нормальній формі, Ardent надає механізм для *динамічної нормалізації* даних (у процесі запитів). Для даної мети в SQL синтаксис розширений ключовим словом UNNEST.

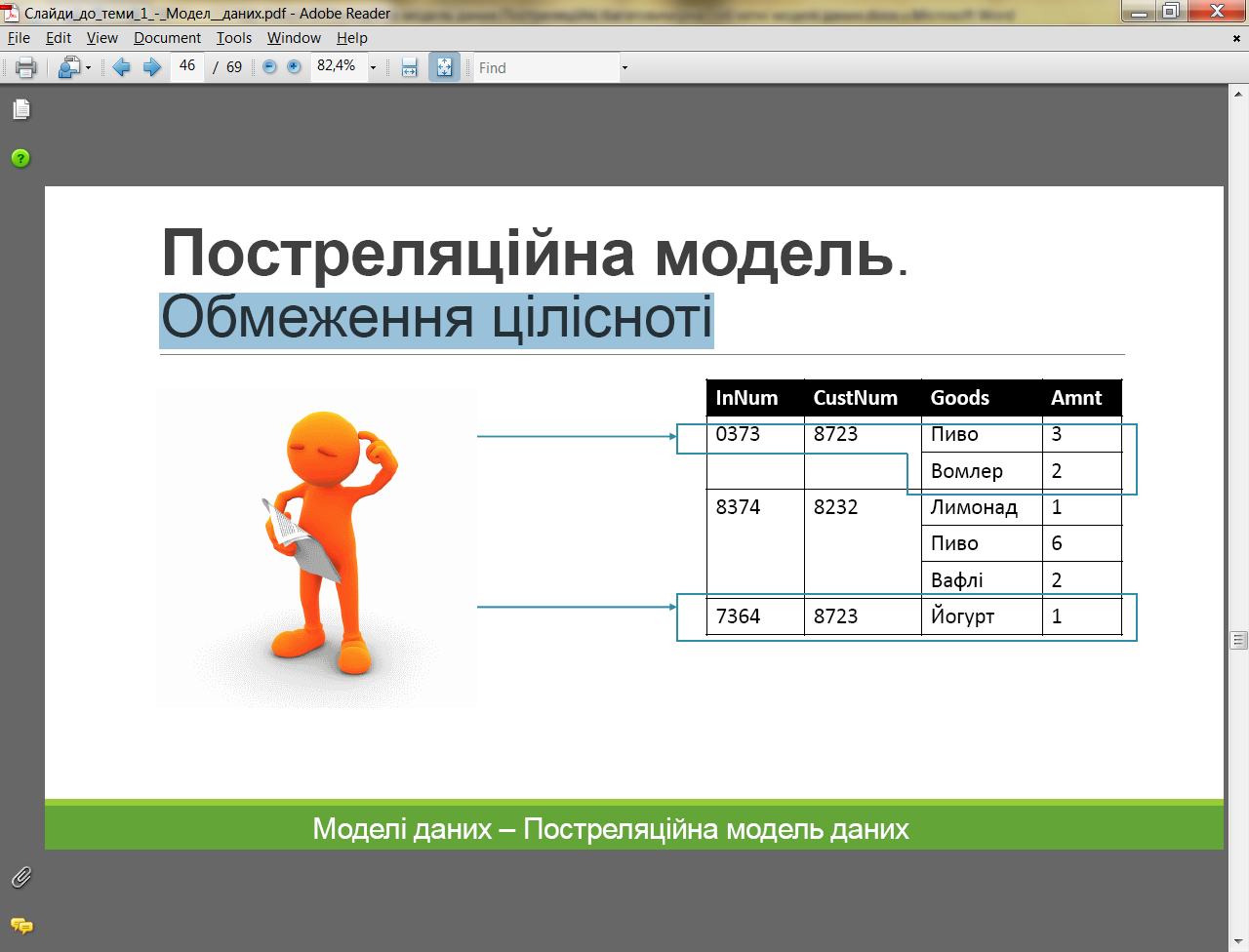


### *Структура вкладених реляційних відношень*

Визначення відношень пов’язане з необхідністю підтримки цілісності даних. У випадку використання вкладених таблиць, визначення відношень стають більш неявним ніж явним і обмеження цілісності в них здійснюються автоматично, без додаткових визначень. Дійсно, у даній моделі немає необхідності зберігати зовнішні ключі у вкладеній і в основній таблиці оскільки фактично одна таблиця є частиною іншої й при вибірці записів з основної таблиці, автоматично вибираються дані із вкладеної.

Так само відбувається й з посилальними обмеженнями цілісності – при видаленні запису основної таблиці, природно, віддаляються всі вкладені в цей запис таблиці. Крім того з’являється цілий ряд додаткових переваг. Так можливість зберігати у вкладеній таблиці безліч зовнішніх ключів рятує від необхідності використовувати й підтримувати таблиці перехресних посилань у відносинах безліч до безлічі, що дає незаперечні переваги на вибірках з більших масивів даних.

Обмеження цілісності:



### *Переваги та недоліки постреляційної моделі даних*

Перевагою постреляційної моделі є можливість представлення сукупності зв’язаних реляційних таблиць однією постреляційною таблицею. Це забезпечує високу наочність представлення інформації й підвищення ефективності її обробки.

Недоліком постреляційної моделі є складність вирішення проблеми забезпечення цілісності й несуперечності збережених даних.

## Об’єктно-орієнтована модель даних

### *Структура моделі даних*

Об’єктно-орієнтована модель даних враховує семантику об’єктів, яка застосовується в об’єктно-орієнтованому програмуванні. Основними модельними поняттями є об’єкти і літерали.

Об’єкт має унікальний ідентифікатор, який не змінюється і не використовується після видалення об’єкта. Об’єкти можуть бути розбиті на типи: атомарні, колекції або структуровані типи. Тип також є об’єктом. Об’єкт інкапсулює стан і поведінку. Поведінку об’єкта – це операції, які можуть бути виконані або самим об’єктом, або над ним. У сукупності ці операції називаються методами. Стан об’єкта визначається значеннями, які маються у набору властивостей об’єкта. Є два типи властивостей – атрибути та зв’язки. Атрибут визначається для об’єктів одного типу. Він не є об’єктом, але може приймати в якості значень літерал або ідентифікатор об’єкта. Об’єкт може зберігати всі зв’язки, якими він пов’язаний з іншими об’єктами, включаючи зв’язок «багато – до – багатьох». Зв’язки представлені за допомогою посилальних атрибутів. Запит одного об’єкта до іншого називають повідомленням.

Об’єкти, що мають однакові атрибути і відповідають на одні й ті ж повідомлення, утворюють клас. Спадкування дозволяє визначити один клас як окремий випадок більш загального класу. Поліморфізм означає допустимість в об’єктах різних типів мати методи з однаковими іменами.

Типи літералів можна розбити на атомарні, колекції, структуровані типи та об’єкти без типу. Літерали не можуть існувати окремо. Вони завжди вбудовані в об’єкт. За допомогою механізму наслідування допускається створення нових абстрактних типів даних на основі вже існуючих.

Логічно структура об’єктно-орієнтованої бази даних схожа на структуру ієрархічної бази даних (рис. 3). Основна відмінність полягає в методах маніпулювання даними.

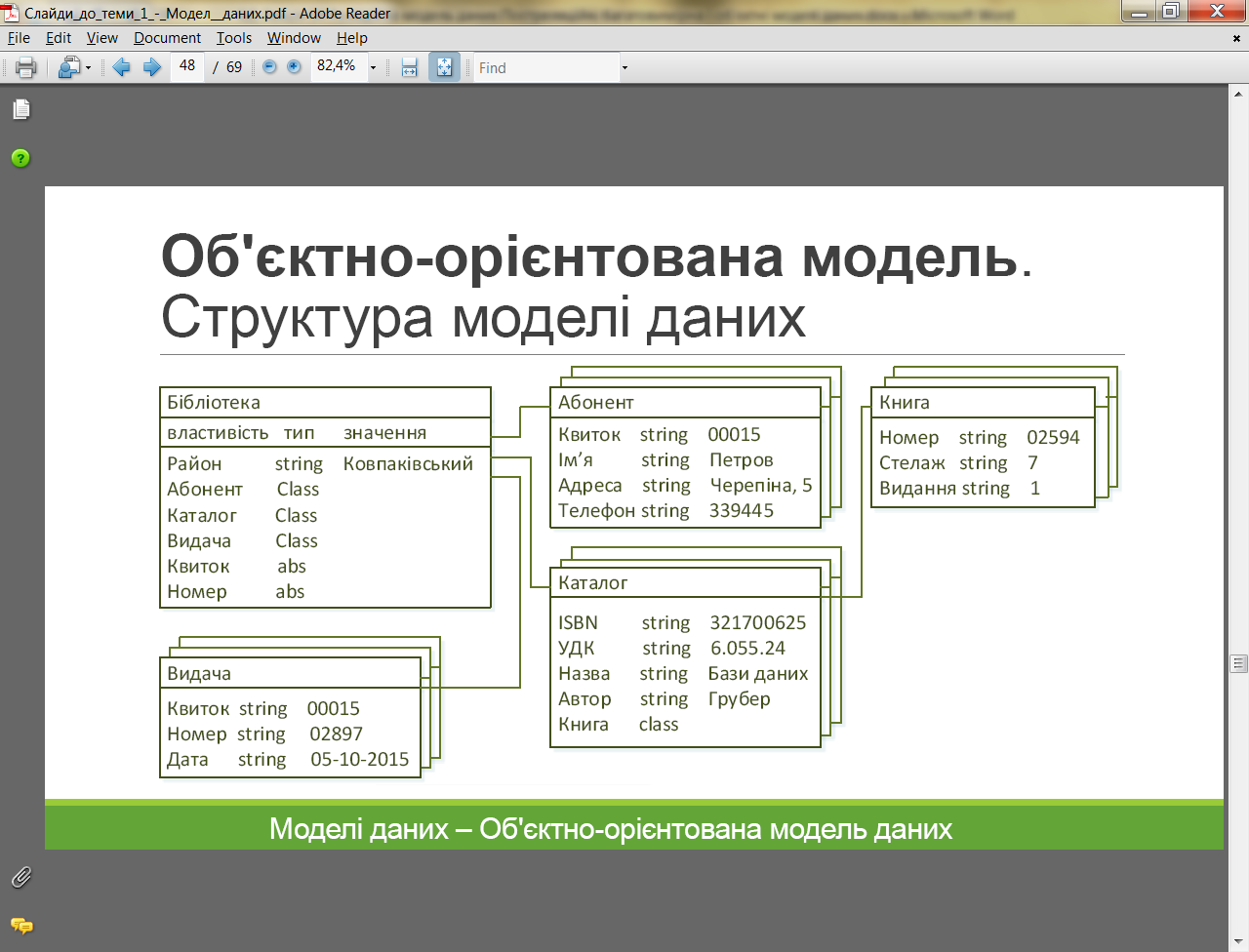


Рисунок 3 - Об'єктно-орієнтована схема даних

***Операції маніпулювання даними***

В стандарті ODMG в якості базового засобу маніпулювання об’єктними базами даних пропонується мова OQL (Object Query Language). Це невелика, але досить складна мова запитів. Розробники в цілому характеризують її таким чином:

* OQL спирається на об’єктну модель ODMG (мається на увазі, що в ній підтримуються засоби доступу до всіх можливих структур даних, що допускаються в структурній частині моделі).
* OQL дуже близька до SQL/92. Розширення відносяться до об’єктно-орієнтованих понять, таких як складні об’єкти, об’єктні ідентифікатори, шляхові вирази, поліморфізм, виклик операцій і відкладене зв’язування.
* У OQL забезпечуються високорівневі примітиви для роботи з множинами об’єктів, але, крім того, є настільки ж ефективні примітиви для роботи зі структурами, списками і масивами.
* OQL є функціональним мовою, що допускає необмежену композицію операцій, якщо операнди не виходять за межі системи типів. Це є наслідком того факту, що результат будь-якого запиту володіє типом, що належить до моделі типів ODMG, і тому до результату запиту може бути застосований новий запит.
* OQL не є обчислювально повною мовою. Вона являє собою просту мову запитів.
* Оператори мови OQL можуть викликатися з будь-якої мови програмування, для якого в стандарті ODMG визначені правила зв’язування. І, навпаки, у запитах OQL можуть бути присутніми виклики операцій, запрограмованих на цих мовах.
* У OQL не визначаються явні операції оновлення, а використовуються виклики операцій, визначених в об’єктах для цілей оновлення.
* У OQL забезпечується декларативний доступ до об’єктів. З цієї причини OQL-запити можуть добре оптимізуватися.
* Можна легко визначити формальну семантику OQL.

Приклад виконання запиту на отримання номерів керівників відділів і тих службовців їх відділів, зарплата яких перевищує 2000 грн.

SELECT DISTINCT STRUCT (ВДЛ\_РУК: D.ВДЛ\_РУК, СЛУ:

(SELECT E

FROM D.CONSISTS\_OF AS E

WHERE E.СЛУ\_ЗАРП> 20000.00)

)

FROM ВІДДІЛИ D

Тут передбачається, що для атомарного об’єктного типу ВІДДІЛ визначений екстент типу безлічі з ім’ям ВІДДІЛИ. У запиті перебираються всі існуючі об’єкти типу ВІДДІЛ, і для кожного такого об’єкта відбувається перехід по зв’язку до літеральної безлічі об’єктів типу СЛУЖБОВЕЦЬ, відповідних службовцям, які працюють в даному відділі. На основі цієї безлічі формується «усічена» безліч об’єктів типу СЛУЖБОВЕЦЬ, в якій залишаються тільки об’єкти-службовці із зарплатнею, більшою від 2000.00 грн. Результатом запиту є літеральне значення-безліч, елементами якого є значення-структури з двома літеральними значеннями, перше з яких є атомарне літеральне значення типу INTEGER, а друге – літеральне значення-множина з елементами-об’єктами типу СЛУЖБОВЕЦЬ.

Більш точно, результат запиту має тип set < struct {integer ВДЛ\_РУК; bag <СЛУЖБОВЕЦЬ> СЛУ}>.

У сукупності результатом допустимих в OQL виразів запитів можуть бути:

* колекція об’єктів;
* індивідуальний об’єкт;
* колекція літеральних значень;
* індивідуальне літеральне значення.

***Обмеження цілісності в об’єктній моделі***

Відповідно до загальної ідеології об’єктно-орієнтованого підходу в моделі ODMG два об’єкти вважаються співпадаючими в тому і тільки в тому випадку, коли є одним і тим же об’єктом, тобто мають один і той же OID. Об’єкти одного об’єктного типу з різними OID вважаються різними, навіть якщо володіють повністю співпадаючими станами. Тому в об’єктній моделі відсутній аналог обмеження цілісності суті реляційної моделі даних. Цікаво, що при визначенні атомарного об’єктного типу можна оголосити ключ – набір властивостей об’єктного класу, однозначно ідентифікує стан кожного об’єкта, що входить в екстент цього класу. Для класу може бути оголошено кілька ключів, а може не бути оголошено жодного ключа навіть за наявності визначення екстента. Але при цьому визначення ключа не трактується в моделі як обмеження цілісності; стверджується, що оголошення ключа сприяє підвищенню ефективності виконання запитів.

Цілісність підтримується, якщо між двома атомарними об’єктними типами визначається зв’язок виду «один-до-багатьох». У цьому випадку об’єкти на стороні зв’язку «один» розглядаються як предки, а об’єкти на стороні зв’язку «багато» – як нащадки, і ООСУБД зобов’язана стежити за тим, щоб не утворювалися нащадки без предків.

***Переваги та недоліки об’єктно-орієнтованої моделі даних*.**

Перевагами об’єктно-орієнтованої моделі є покращення можливості моделювання об’єктів реального світу. Об’єктні типи даних, а також об’єктні таблиці представляють потужний єдиний рівень інтерпретації об’єктів ділової сфери і дозволяють відмовитися від поділу на частини бізнес-даних для зберігання їх в БД при використанні реляційної моделі.

Недоліками моделі є висока понятійна складність, відсутність стандарту об’єктно-орієнтованої моделі через недостатню її теоретичну розробку.

**Об’єктно-реляційна модель даних.**

***Структура моделі даних*.**

У зв’язку з недостатньою розробленістю об’єктно-орієнтованої моделі на практиці застосовується об’єктно-реляційна модель, що є ніби сумішшю реляційної та об’єктно-орієнтованої методологій для представлення даних. Ця модель являє собою розширену реляційну модель, в якій зняті обмеження неподільності даних, які зберігаються в записах таблиць.

Допускаються багатозначні поля – поля, значеннями яких є самостійні таблиці, вбудовані в основну таблицю. Крім цього підтримуються такі концепції об’єктно-орієнтованого програмування, як «абстракція», «клас», «екземпляр», «інкапсуляція», «метод», «перевантаження» і «повідомлення».

Хоча спадкування і є однією з найбільш важливих характеристик об’єктів, але в об’єктно-реляційній моделі воно не підтримується.

В модель вводиться спеціальний об’єктний тип, за допомогою якого можна створити абстрактний тип даних будь-якого ступеня складності.

Використовуючи вкладені об’єктні типи, можна створювати структури, в яких використовуються всі види зв’язків: «один до одного», «один до багатьох» і навіть «багато до багатьох». Хоча це і може призвести до певної надмірності, такий підхід дає переваги в порівнянні з використанням безлічі нормалізованих таблиць в чисто реляційній моделі.

***Переваги та недоліки об’єктно-реляційної моделі даних*.**

Перевагою об’єктно-реляційної моделі є можливість використання існуючих реляційних баз даних з знову розроблюваними об’єктними додатками.

До недоліків моделі можна віднести складність вирішення проблеми забезпечення цілісності і несуперечності збережених даних.

**Багатовимірна модель даних**

**1.8.1. Багатовимірна структура даних.**

Багатовимірна модель даних є вузькоспеціалізованою моделлю, яка призначена для оперативної аналітичної обробки інформації. В основі моделі лежить не двомірна, як в реляційній моделі, а багатовимірна таблиця і багатовимірне логічне представлення структури інформації при описі даних і в операціях маніпулювання даними. У порівнянні з реляційною моделлю багатовимірна організація даних має більш високу наочність та інформативність.

За визначенням, запропонованим Б. Коддом, *багатовимірне концептуальне представлення (multi-dimensional conceptual view)* – це множинна перспектива, що складається з кількох незалежних вимірювань, уздовж яких можуть бути проаналізовані визначені сукупності даних. Одночасний аналіз за кількома вимірюваннями визначається як *багатовимірний аналіз*.

*On-Line Analytical Processing (OLAP)* – технологія оперативної аналітичної обробки даних, що використовує методи і засоби для збору, зберігання та аналізу багатовимірних даних з метою підтримки процесів прийняття рішень.

  12 правил, що визначають OLAP, згідно із концепцією Б. Кодда.

1. Багатовимірність – OLAP-система на концептуальному рівні має представляти дані у вигляді багатовимірної моделі, що спрощує процеси аналізу і сприйняття інформації.
2. Прозорість – це спосіб організації даних, джерела, засоби обробки і зберігання.
3. Доступність – OLAP-система має надавати користувачу єдину, узгоджену і цілісну модель даних, забезпечуючи доступ до даних незалежно від того, як і де вони зберігаються.
4. Постійна продуктивність при розробленні звітів – продуктивність OLAP-систем не має дуже зменшуватися при збільшені кількості вимірювань, за якими виконується аналіз.
5. Клієнт-серверна архітектура – OLAP-система має бути здатна функціонувати у клієнт-серверному середовищі, оскільки більшість даних, які потрібно обробляти, зберігається децентралізовано. Серверний компонент інструменту OLAP має бути достатньо інтелектуальним і дозволяти будувати загальну концептуальну схему на основі узагальнення й консолідації різних логічних і фізичних схем корпоративних БД для забезпечення ефекту прозорості.
6. Рівноправність вимірювань – OLAP-система має підтримувати багатовимірну модель, в якій усі вимірювання рівноправні. За необхідності додаткові характеристики можуть бути надані окремим вимірюванням, але така можливість має бути надана будь-якому вимірюванню.
7. Динамічне управління розрідженими матрицями – OLAP-система має забезпечувати оптимальну обробку розріджених матриць. Швидкість доступу повинна зберігатися незалежно від розташування осередків даних і бути постійною величиною для моделей, що мають різну кількість вимірювань і різний ступінь розрідженості даних.
8. Підтримка розподіленого режиму доступу – OLAP-система має надавати можливість працювати кільком користувачам спільно з однією аналітичною моделлю або створювати для них різні моделі з єдиних даних. При цьому можливі усі операції.
9. Підтримка перехресних операцій – OLAP-система мас забезпечувати збереження функціональних відношень, описаних за допомогою певної формальної мови між осередками гіперкуба при виконанні будь-яких операцій зрізу, обертання, консолідації або деталізації. Система має автоматично виконувати перетворення встановлених відношень, не вимагаючи від користувача їх перевизначення.
10. Інтуїтивна маніпуляція даними – OLAP-система має надавати спосіб виконання операцій зрізу, обертання, консолідації і деталізації над гіперкубом без необхідності для користувачів здійснювати дії з інтерфейсом. Вимірювання, визначені в аналітичній моделі, мають містити всю необхідну інформацію для виконання операцій.
11. Гнучкі можливості отримання звітів – OLAP-система має підтримувати різні способи візуалізації даних, тобто звіти мають подаватися у будь-якому можливому вигляді.
12. Необмежена розмірність і кількість рівнів агрегації – одночасно може використовуватися близько 20 вимірювань.

Основні поняття багатовимірної СУБД: агрегація, історичність і прогнозованість даних.

* ***Агрегація даних*** означає розгляд інформації на різних рівнях її узагальнення. В інформаційних системах ступінь детальності подання інформації для користувача залежить від його рівня: аналітик, користувач-оператор, керуючий, керівник.
* ***Історичність даних*** передбачає забезпечення високого рівня статичності (незмінності) власне даних і їх взаємозв’язків, а також обов’язковість прив’язки даних до часу.
* ***Прогнозованість даних*** передбачає завдання функцій прогнозування і застосування їх до різних часових інтервалів.

Багатовимірна модель даних – це перш за все багатовимірне логічне подання даних при їх описі і операціях маніпулювання даними, а не багатовимірність їх візуалізації. В порівнянні з реляційною моделлю багатовимірна організація даних володіє більш високою інформативністю (рис. 4).



|  |  |  |
| --- | --- | --- |
| а) реляційне подання даних | б) багатовимірне подання даних | в) тривимірна модель даних |

Рисунок 4 – Реляційне та багатовимірне подання даних

До числа основних понять багатовимірної моделі відносяться вимір і комірка. *Вимір* – це безліч однотипних даних, що утворюють одну з граней багатовимірної таблиці (аналог домену в реляційній моделі). Наприклад, дні, місяці, квартали, роки – це часові виміри, які найчастіше використовуються в аналізі. Прикладами географічних вимірів є міста, райони, регіони і т. ін.

У багатовимірній моделі даних виміри відіграють роль індексів, які використовуються для ідентифікації конкретних значень (показників), що знаходяться в комірках гіперкуба.

Комірка – це поле, значення якого однозначно визначається фіксованим набором вимірів. У багатовимірній СУБД Oracle Express Server показник може бути визначений, як:

* змінна (Variable) – значення таких показників один раз вводяться з будь-якого зовнішнього джерела або формуються програмно, а потім у явному вигляді зберігаються в багатовимірній базі даних;
* формула (Formula) – значення таких показників обчислюються за деякою заздалегідь специфікованою формулою. Тобто для показника, який має тип «формула», у БД зберігається не його значення, а формула, за якою це значення може бути обчислено.

Використовуються два основних варіанти організації даних: гіперкубічна і полікубічна.

У гіперкубічній схемі передбачається, що всі багатовимірні таблиці мають однакову розмірність і співпадаючі виміри.

У полікубічній схемі може бути визначено декілька таблиць з різною розмірністю і різними вимірами. Системи, які підтримують полікубічну модель (прикладом є Oracle Express Server), припускають наявність у багатовимірній БД декількох гіперкубів з різною розмірністю та різними вимірами.

Наприклад, значення показника «Робочий час менеджера» не залежить від виміру «Модель автомобіля» та однозначно визначається двома вимірами: «Час» та «Менеджер».

У полікубічній моделі в цьому випадку можуть бути присутні два різні гіперкуби:

* двовимірний (рис. 5) – для показника «Робочий час менеджера» з вимірами «Час», «Менеджер»;
* тривимірний (рис. 6) – для показника «Обсяг продажів» з вимірами «Час», «Менеджер», «Модель автомобіля».

У разі гіперкубічної моделі передбачається, що всі показники повинні визначатися одним і тим же набором вимірів. Тобто тільки через те, що «Обсяг продажів» визначається трьома вимірами, при описі показника «Робочий час менеджера» доведеться перебудувати модель і використати ще один вимір –«Модель комп’ютера».

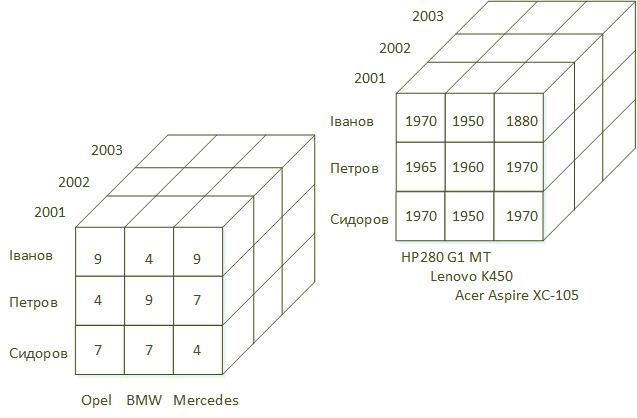


Рисунок 5 - Приклад гіперкубічної моделі

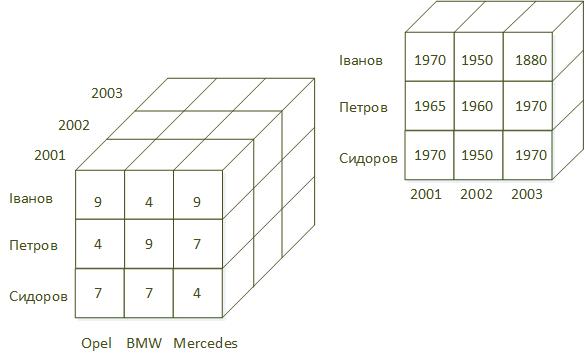
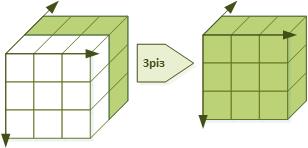


Рисунок 6 - Приклад полікубічної моделі

***Операції над багатовимірною структурою даних*.**

 Багатовимірна модель застосовує ряд спеціальних операцій, до яких відносяться: формування «зрізу», «обертання», агрегація і деталізація.

* Операція *зріз (slice)* являє собою підмножину гіперкуба, отриману в результаті фіксації одного або декількох вимірів. Формування «зрізів» виконується для обмеження використовуваних користувачем значень, так як всі значення гіперкуба практично ніколи одночасно не використовуються (рис. 7).
* Операція *обертання (rotate)* застосовується при двовимірному поданні даних. Суть її полягає в зміні порядку вимірів при візуальному представленні даних (рис. 8).



|  |  |
| --- | --- |
| Рисунок 7 - Ілюстрація до операції зрізу | Рисунок 8 - Ілюстрація до операції обертання |

Ця операція забезпечує можливість візуалізації даних у формі, найбільш комфортної для їх сприйняття. Наприклад, аналітик має можливість вивести звіт, в якому моделі автомобілів перераховані по осі X, а менеджери по осі V, і поміняти місцями координати (виконавши обертання на 90 градусів).

Використання *ієрархічних відношень (Hierarchical Relationship)*.

Безліч відносин може мати ієрархічну структуру, яка відображує залежність вимірювань один від одного.

Наприклад:

День →Місяць → Квартал → Рік;

Менеджер → Підрозділ → Регіон → Фірма → Країна;

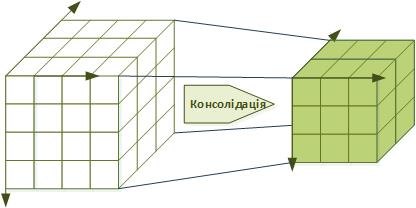
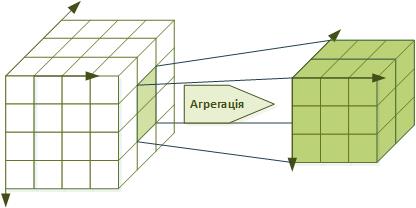
Модель автомобіля →Завод-виробник → Країна.

Часто зручніше не оголошувати нові виміри і потім встановлювати між ними множину відношень, а використовувати механізм ієрархічних відношень. У цьому випадку всі потенційно можливі значення з різних вимірів об’єднуються в одну множину.

* Операція *агрегації (Drill Up)* – це операція підйому за рівнями консолідації даних у процесі аналізу або переходу від деталізованих даних до агрегованих (рис. 9). З точки зору користувача, «Підрозділ», «Регі­он», «Фірма», «Країна» є точно такими ж вимірами, як і «Менеджер». Але кожний з них відповідає новому, більш високому рівню агрегації значень показника «Обсяг продажів».

Наприклад, подивившись, наскільки успішно в 2002 р. Сидоров продавав моделі BMV та Opel, керуючий може захотіти дізнатися, як виглядає співвідношення продажу цих моделей на рівні підрозділу, де Сидоров працює. А потім отримати аналогічну довідку по регіону або фірмі.

* Операція *деталізації (Drill Down)* – операція спуску за рівнями консолідації даних або переходу від агрегованих до деталізованих даних (рис. 10). Наприклад, почавши аналіз на рівні регіону, користувач має можливість отримати більш точну інформацію про роботу кон­кретного підрозділу або менеджера.



|  |  |
| --- | --- |
| Рисунок 9 - Ілюстрація до операції агрегації | Рисунок 10 - Ілюстрація до операції деталізації |

***Переваги та недоліки багатовимірної моделі даних*.**

Основною перевагою багатовимірної моделі даних є зручність і ефективність аналітичної обробки великих обсягів даних, пов’язаних з часом. При організації обробки аналогічних даних на основі реляційної моделі відбувається нелінійний зростання трудомісткості операцій залежно від розмірності БД та суттєве збільшення витрат оперативної пам’яті на індексацію.

Недоліком багатовимірної моделі даних є її громіздкість для найпростіших завдань звичайної оперативної обробки інформації.

**Сфери використання.**

Багатовимірні моделі даних мають три важливі області застосування, пов’язаних з проблематикою аналізу даних:

1. Сховища даних інтегрують для аналізу інформації з декількох джерел.
2. Системи оперативної аналітичної обробки дозволяють оперативно отримати відповіді на запити, що охоплюють великі обсяги даних у пошуках загальних тенденцій.
3. Додатки видобутку даних служать для виявлення знань за рахунок напівавтоматичного пошуку раніше невідомих шаблонів і зв’язків у базах даних.

Багатовимірні СУБД є вузькоспеціалізованими СУБД, призначеними для інтерактивної аналітичної обробки інформації.

З іншого боку, є істотні обмеження на використання БСУБД:

1. БСУБД не дозволяють працювати з великими базами даних. На сьогоднішній день їх реальна межа – 10-20 гігабайт.
2. БСУБД порівняно з реляційними малоефективно використовують зовнішню пам’ять. Комірки гіперкуба зберігаються в них у вигляді логічно впорядкованих масивів (блоків фіксованої довжини), причому саме такий блок є мінімальною одиницею, яка індексується. Хоч у багатовимірних СУБД блоки, які не містять жодного певного значення, не зберігаються, це вирішує проблему тільки частково. Оскільки дані зберігаються у впорядкованому вигляді, невизначені значення не завжди віддаляються повністю, а лише в тому випадку, коли за рахунок вибору порядку сортування дані вдається організувати в максимально великі безперервні групи. Але порядок сортування, який частіше за все використовується в запитах, може не співпадати з порядком, в якому вони повинні бути відсортовані, а бути сформований з метою максимального усунення неіснуючих значень. Таким чином, при проектуванні багатовимірної БД часто доводиться жертвувати або швидкодією (а це одна з перших переваг та головна причина вибору саме багатовимірної СУБД), або зовнішньою пам’яттю.
3. Для багатовимірних СУБД поки відсутні єдині стандарти на інтерфейс, мови опису та маніпулювання даними.
4. Багатовимірні СУБД не підтримують реплікацію даних, яка часто використовується як механізм завантаження даних.

Використання БСУБД виправдане тільки при умовах, коли:

1. Рівень агрегації даних досить високий. При цьому об’єм початкових даних для аналізу не дуже великий (не більше кількох гігабайт).
2. Набір інформаційних вимірів стабільний (оскільки будь-яка зміна в їх структурі майже завжди вимагає повної перебудови гіперкуба).

**Висновки**

Модель даних визначає правила, за якими структуруються дані. Можна визначити три основні компоненти в моделі даних:

* Структуризація даних – сукупність засобів визначення допустимих структур даних. Визначається різноманітністю та кількістю типів об’єктів моделі даних, обмеженнями на структуру даних.
* Множина операцій, яку можна застосовувати до допустимого стану бази даних для пошуку чи модифікації даних.
* Множина обмежень цілісності даних, яка в явному чи неявному вигляді визначає множину допустимих станів бази даних.

Вибір моделі даних залежить від певної множини факторів, до яких слід віднести:

* обсяг інформації,
* складність задач, що мають бути вирішені,
* наявне технічне та програмне забезпечення,
* засоби маніпулювання даними,
* цілісність та захист даних,
* критерії якості (надійність, точність, т.п.),
* можливість розвитку тощо.

Необхідно також пам’ятати, що вибір моделі даних визначає, зазвичай, вибір СУБД.