**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ**

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Тарасов О. В.
Федько В. В.
Лосєв М. Ю.

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ. ПРОЕКТУВАННЯ БАЗ ДАНИХ

**Навчально-практичний посібник
для самостійної підготовки студентів**

Частина 1

Харків. Вид. ХНЕУ, 2011

УДК [004.65+004.82](076)
ББК 32.973-018.2я73
Т19

Рецензенти: докт. техн. наук, професор кафедри системотехніки Харківського національного університету радіоелектроніки *Безкоровайний В. В.;* канд. техн. наук, доцент кафедри інформаційних управляючих систем Харківського національного університету радіоелектроніки *Кобзєв В. Г.*

Рекомендовано до видання рішенням вченої ради Харківського національного економічного університету.
Протокол № 7 від 18.04.2011 р.

Тарасов О. В.

Т19 Організація баз даних та знань. Проектування баз даних : навчально-практичний посібник для самостійної підготовки студентів. Ч. 1 / О. В. Тарасов, В. В. Федько, М. Ю. Лосєв. – Х. : Вид. ХНЕУ, 2011. – 200 с. (Укр. мов.)

Розглянуто широке коло питань, пов'язаних з автоматизацією проектування логічної та фізичної структур баз даних на основі CASE-технологій, а також теоретичні засади реляційної алгебри та нормалізації відношень, на яких базується сучасна теорія реляційних баз даних. Наведено приклади вирішення практичних завдань, що пов'язані з проектуванням схеми бази даних CASE-засобами, виконанням нормалізації відношень та операцій реляційної алгебри.

Рекомендовано для студентів напряму підготовки "Комп'ютерні науки".

ISBN

УДК [004.65+004.82](076)
ББК 32.973-018.2я73

Вступ

Сучасні економічні умови господарювання вимагають від фахівців, незалежно від їхньої спеціалізації, всебічного використання новітніх інформаційних технологій, комп'ютеризованих засобів збору, обробки та надання необхідної інформації. Метою цих технологій є значне підвищення якості та оперативності економічних розрахунків, зробити значно ефективнішим процес обґрунтування економічних рішень тощо. У цьому контексті навчальна дисципліна "Організація баз даних та знань" є однією з найважливіших. Вона належить до системних дисциплін і є тим фундаментом, на якому базується проектування та безпосереднє створення інформаційних систем у бізнесі.

Вивчення дисципліни "Організація баз даних та знань" ґрунтується на знаннях та вміннях, які студенти отримали під час вивчення таких дисциплін: "Основи програмування та алгоритмічні мови", "Вища математика", "Об'єктно-орієнтоване програмування", "CASE-технології".

Вона забезпечує такі дисципліни: "Інтелектуальна обробка інформації", "Технології програмування і створення програмних продуктів", "Системи штучного інтелекту", "Системи підтримки прийняття рішень","ІС в сучасному бізнесі" тощо [16].

Розробники реляційних БД дуже часто стикаються з проблемою групування даних у базі даних [2; 30]. До тих самих даних можна запропонувати безліч різних способів групування елементів даних. Одні з них можуть бути кращі, інші – гірші, а треті взагалі можуть призвести до складних і навіть нерозв'язних проблем під час реалізації.

В процесі функціонування база даних може безперервно змінюватися. До неї можуть додаватися нові елементи даних, між ними встановлюватися нові зв'язки і визначатися нові способи їхнього використання. У разі зміни БД необхідно зберігати старе уявлення користувача про дані для того, щоб уникнути необхідності переписувати вже існуючі програми наново. Проте можливі і такі зміни зв'язків між даними, які вимагають модифікації програм, наприклад у разі розщеплювання запису на дві частини, або ж при зміні ключа деяких елементів. Такого роду модифікації і зміни є абсолютно небажаними, оскільки, можуть призвести до руйнування структури БД.

В той же час такі руйнування можна зробити маловірогідними, якщо групування елементів даних і їхні ключі спочатку були добре продумані. Методику такого "гарного" проектування структури БД забезпечує спеціальний формальний апарат, що називається нормалізацією.

Нормалізація дозволяє усунути дублювання, забезпечує несуперечність збережених даних і зменшує трудовитрати на ведення БД.

Процес нормалізації полягає в розбитті (декомпозиції) початкових відношень БД на простіші. При цьому на кожному ступені цього процесу схеми стосунків наводяться до нормальних форм. Для кожного ступеня нормалізації є набори обмежень, яким повинні задовольняти відношення в реляційній базі даних.

Теоретичною основою реляційної моделі даних стала теорія відношень, основу якої заклали два логіки – американець Чарльз Содерс Пірс (1839 – 1914) та німець Ернст Шредер (1841 – 1902). У керівництві по теорії відношень було показано, що множина відношень замкнута відносно деяких спеціальних операцій, тобто утворює разом з цими операціями абстрактну алгебру. Ця найважливіша властивість відношень була використана в реляційній моделі для розробки мови маніпулювання даними, пов'язаного з початковою алгеброю.

Американський математик Е. Ф. Кодд, який народився у Великобританії, в 1970 році уперше сформулював основні поняття і обмеження реляційної моделі, обмеживши набір операцій сімома основними і однією додатковою. Пропозиції Кодда були настільки ефективні для систем баз даних, що за цю модель він був удостоєний престижної премії Т'юрінга в галузі теоретичних основ обчислювальної техніки.

Основною структурою даних в моделі є відношення, саме тому модель дістала назву реляційної (від англійського relation – відношення). Ця модель є домінуючою в даний час і тому на ці питання треба звернути особливу увагу при вивченні дисципліни.

Навчальний посібник написано відповідно до програми навчальної дисципліни "Організація баз даних та знань" [16] і охоплює три базові теми пов'язані з проектуванням та моделюванням предметної області з використання CASE–засобів, нормалізацією відношень та реляційною алгеброю. Освоєння цих важливих питань дозволяє сформувати у

студента професійні компетенції в галузі проектування систем керування базами даних (СКБД), а саме:

1. В рамках аналітико-проектної компетенції – здатність до проведення аналізу та проектування структури і елементів баз даних:

1.1. Обирати СКБД у процесі технічного проектування на основі аналізу технічних, функціональних, сервісних характеристик СКБД, використовуючи науково-технічну, довідкову інформацію.

1.2. Розробляти логічну структуру бази даних за допомогою методу нормалізації відношень.

1.3. Розробляти фізичну структуру бази даних у процесі робочого проектування за допомогою вибраної СКБД, використовуючи сучасні технічні та програмні засоби.

1.4. Розробляти таблиці баз даних і зв’язок між ними в умовах технічного проектування за допомогою відповідного технічного і програмного забезпечення.

2. В рамках програмно-користувальницької компетенції – здатність до ефективного користування базами даних:

2.1. Створювати запити до баз даних з мінімальним часом їх виконання за допомогою програмних засобів СКБД, методів аналізу і нормалізації запитів, використовуючи методи реляційної алгебри та вимоги вибраної СКБД, вибору стратегії виконання запитів, побудови плану запитів.

2.2. Створювати таблиці баз даних, тригери, збережені процедури, індекси за допомогою сучасних програмних і технічних засобів проектування баз даних.

Навчальний посібник складається з шести розділів та глосарія термінів, які широко використовуються у галузі проектування баз даних.

Перший розділ присвячено ознайомленню з основними етапами процесу проектування бази даних. Головна увага приділяється створенню концептуальної моделі системи на етапі концептуального проектування. Базовою в цьому випадку є модель "сутність-зв'язок", яка ґрунтується на важливій семантичній інформації про реальний світ і призначена для логічного подання даних. На прикладах розглянуто процес автоматизації проектування логічної та фізичної схеми бази даних на основі CASE-засобу ErWin Data Modeler.

У другому розділі розглянуто основні поняття, пов’язані з реляційною моделлю даних, яка є основною моделлю проектування баз даних на сучасному етапі, та їхній взаємозв’язок. Крім того, певна увага приділяється питанням підтримки цілісності у реляційних моделях даних.

Третій розділ присвячено викладенню основних математичних операцій, на яких базується реляційна модель даних – реляційної алгебри, яка є фундаментом для подальшого вивчення мови запитів SQL.

Четвертий розділ присвячено дуже важливому питанню у теорії баз даних, а саме, нормалізації відношень. Нормалізація полягає в розбитті (декомпозиції) початкових відношень БД на більш прості з метою усунути дублювання та забезпечити несуперечність збережених даних. Розглянуті питання функціональної та багатозначної залежностей, поняття "декомпозиції без втрат", та безпосередньо процесу нормалізації з визначенням понять першої, другої, третьої нормальних форм та форм більш високого порядку. Розкрито поняття "денормалізації" відношень.

У п'ятому та шостому розділах наведено детальні приклади виконання нормалізації відношень та виконання операцій реляційної алгебри.

Процес вивчення матеріалу навчального посібника слід організувати у такій послідовності:

матеріал першого розділу рекомендується вивчати одночасно з послідовним виконанням наведених у роботі прикладів, використовуючи застосування ErWin Data Modeler. При цьому слід паралельно приділяти увагу теоретичним питанням. Отримані знання та навички, у подальшому, використовуватимуться під час виконання курсового проекту на етапі створення моделі предметної області.

Під час вивчення матеріалу розділів 2, 3 та 4 рекомендується спочатку приділити особливу увагу саме теоретичним питанням, а потім, базуючись на отриманих знаннях, ретельно опрацювати приклади, що наведені у розділах 5 та 6.

Отримані у процесі вивчення викладеного матеріалу професійні компетенції в галузі проектування баз даних дозволять у подальшому на більш високому теоретичному та практичному рівні:

аналізувати довільну предметну галузь та проводити постановку задачі на створення інформаційної системи в цій галузі;

будувати концептуальну модель предметної галузі, логічну та фізичну модель бази даних;

вибирати, обґрунтовувати та реалізувати найкращі рішення стосовно розробки інформаційних систем на основі баз даних;

наповнювати та супроводжувати бази даних і, як наслідок, виконати на високому рівні курсовий проект із дисципліни "Організація баз даних та знань".

Розділ 1. Організація баз даних та знань

1. Етапи проектування баз даних та інструментальні засоби їхньої підтримки

Цілі – ознайомитися з основними етапами проектування баз даних; розібрати основні складові частини моделі "сутність-зв'язок"; вивчити методологію IDEF1X; освоїти інструментарій ERWin; ознайомитися з технологією побудови логічної моделі в ERWin; вивчити методи визначення ключових атрибутів сутностей; вивчити типи зв'язків між сутностями; ознайомитися з методами побудови фізичної моделі; провести перевірку моделі на наявність помилок (валідація); вивчити способи створення та генерації звітів; вивчити засоби експортування, збереження і друкування звітів.

Компетенції, що формуються:

Інструментальні – здатність аналізувати предметну область та синтезувати її модель з використанням інструментальних CASE-засобів.

Загально-професійні – знання стандартів, методів та засобів управління процесами життєвого циклу інформаційних систем. Здатність до проектної діяльності в професійній сфері, уміння будувати і використовувати моделі для опису об’єктів і процесів.

Спеціалізовано-професійні – знання сучасних технологій та інструментальних засобів розробки програмних систем, уміння їх застосовувати на всіх етапах життєвого циклу; знання сучасних теорій організації баз даних, методів і технологій їх розробки, уміння проектувати логічні та фізичні моделі баз даних.

Основні питання

Моделювання предметної області за допомогою ER-діаграм. CASE–системи – розглядаються основні поняття моделі "Сутність-зв'язок" та способи її відображення на діаграмах.

Інтерфейс Erwin Data Modeler – описується користувальницький інтерфейс CASE-засобу ErWin, знання якого дозволяє значною мірою полегшити створення моделей даних.

Створення моделі предметної області "Продаж хлібобулочних виробів" – на прикладі, що використовується у лабораторних роботах, ілюструється процес створення простої моделі бази даних та етап прямого проектування фізичної схеми бази даних.

Створення моделі предметної області "ІС компанії з продажу товарів" – на більш складному прикладі ілюструється процес створення моделі бази даних з урахуванням специфічних особливостей її відображення, а саме: рівні демонстрації моделі, області і збережені відображення в ERwin, особливостей вибору параметрів фізичної моделі тощо.

Синхронізація системного каталогу БД і поточної моделі – розглядається процес синхронізації поточної моделі бази даних з її реальним станом. Це дозволяє виявити розбіжності та привести у відповідність їхній стан.

Створення звітів за моделями даних – описується процес створення різноманітних звітів за спроектованою моделлю з використанням спеціалізованого інструменту ErWin – Data Browser. Крім отримання детальної інформації про модель, ця процедура дозволяє виявити певні помилки у моделі і оперативно їх виправити.

1.1. Моделювання предметної області за допомогою ER-діаграм. CASE–системи

Процес проектування бази даних становить складний і багатоетапний процес, в рамках якого можна виділити кілька основних етапів – концептуальний, логічний та фізичний [3].

1. Концептуальне проектування бази даних – процедура конструювання інформаційної моделі підприємства, незалежно від яких-небудь фізичних умов реалізації.

2. Логічне проектування бази даних – процес конструювання інформаційної моделі підприємства на основі існуючих конкретних моделей даних, незалежною від використовуваної СКБД і інших фізичних умов реалізації. Етап логічного проектування БД полягає в перетворенні концептуальної моделі даних в логічну модель даних підприємства з урахуванням вибраного типу СКБД. Логічна модель даних є джерелом інформації для етапу фізичного проектування. Вона надає розробникові фізичної моделі даних засобу проведення усебічного аналізу різних аспектів роботи з даними, що має виключно важливе значення для вибору дійсно ефективного проектного рішення.

3. Фізичне проектування бази даних – процес створення опису конкретної реалізації бази даних, що розміщується у вторинній пам'яті. Передбачає опис структури зберігання даних і методів доступу, призначених для здійснення найбільш ефективного доступу до інформації. Етап фізичного проектування бази даних передбачає прийняття розробником остаточного рішення про способи реалізації створюваної бази.

Таким чином етап концептуального проектування бази даних починається зі створення концептуальної моделі даних підприємства, повністю незалежної від будь-яких деталей реалізації. До останніх належать: вибраний тип СКБД, перелік програмних застосувань, мова програмування, конкретна обчислювальна платформа та будь-які інші фізичні особливості реалізації. Важливість даного етапу важко переоцінити, оскільки від якості її реалізації багато в чому залежить хід подальшого проектування бази даних.

У рамках етапу концептуального проектування бази даних створюється модель даних, виходячи з уявлень про предметну область кожного з типів користувачів, а для цього розробник системи повинен мати повне уявлення про поняття, предмети, факти та події, якими буде оперувати система, що розробляється. Для того, щоб привести ці поняття до тієї чи іншої моделі даних, необхідно замінити їх інформаційними уявленнями.

Одним з найбільш зручних інструментів уніфікованого подання даних, незалежного від програмного забезпечення що його реалізує, є модель "сутність-зв'язок" (entity-relationship model, ER-model). Автором даного підходу є американський вчений Пітер Чен [21; 22].

Модель "сутність-зв'язок" ґрунтується на важливій семантичній інформації про реальний світ і призначена для логічного подання даних. Вона визначає значення даних у контексті їхнього взаємозв'язку з іншими даними. З цієї моделі можуть бути породжені всі існуючі моделі даних (ієрархічна, мережна, реляційна, об'єктна), тому, в цьому сенсі, вона є найбільш загальною.

Процес створення моделі предметної області є досить нетривіальним завданням і становить багато в чому творчий процес, внаслідок чого його комплексна автоматизація практично неможлива. Однак застосування сучасних програмних засобів, що базуються на CASE-технологіях [12], на етапі створення моделі предметної області, дозволяє здійснити цей процес більш якісно та з меншими витратами трудових ресурсів.

CASE-технологія є методологією автоматизації проектування інформаційних систем (ІС) і її складових елементів, зокрема бази даних (БД), а також набір інструментальних засобів, які дозволяють в наочній формі моделювати предметну область, аналізувати цю модель на усіх етапах розробки і супроводу ІС, а також розробляти застосування, відповідно до інформаційних потреб користувача.

Одним з найбільш популярних засобів інформаційного моделювання і автоматизації є CASE-засіб ErWin Data Modeler фірми Computer Associates [17; 34].

Цей програмний продукт поєднує графічний інтерфейс Windows, інструменти для побудови ER-діаграм, редактори для створення логічного і фізичного опису моделі даних і прозору підтримку провідних реляційних СКБД і настільних баз даних. За допомогою ERwin можна реалізовувати пряме та зворотне проектування баз даних.

Реалізація моделювання в ERwin базується на теорії реляційних баз даних і на методології IDEF1X (Integration DEFinition for Information Modeling) [35].

Методологію IDEF1X було розроблено для ВПС США і тепер використовується, зокрема, в урядових, аерокосмічних і фінансових установах, а також у великій кількості приватних компаній. Методологія IDEF1X визначає стандарти термінології, використовуваної при інформаційному моделюванні, і графічного зображення типових елементів на діаграмах. У 1981 році цей стандарт був формалізований і опублікований організацією ICAM (Integrated Computed Aided

Manufacturing), і відтоді є найбільш поширеним стандартом для створення моделей баз даних по усьому світу.

1.1.1. Поняття про логічні та фізичні рівні

Існує два різні способи мислення і моделювання – логічний рівень (інфологічний) і фізичний рівень (даталогічний). Поняття логічний рівень припускає, що ми мислимо в поняттях реального світу і безпосередньо з нього беремо об'єкти для моделювання. Наприклад, люди, столи, підрозділи, собаки і комп'ютери – це реальні речі. Об'єкти, на які посилаються на логічному рівні, повинні отримувати імена з природної мови, з використанням таких роздільників (пропусків, рисок і тому подібне), які мають сенс. На логічному рівні не має значення, наприклад, якою СКБД користуватися, чи є деяке число цілим чи дійсним, як краще індексувати таблицю.

Після побудови моделі на логічному рівні, настає час подумати про те, як ці логічні моделі виражатимуться в термінах фізичної бази даних, включаючи вибір СКБД, типів даних за замовчуванням і ефективних схем індексування. Це прийнято називати фізичним рівнем моделі ERwin. ERwin підтримує побудову і організацію роботи на цих двох рівнях для однієї діаграми. Використовуючи ту саму модель, можна говорити з кінцевими користувачами в зрозумілій їм термінології і в той самий час генерувати схему бази даних мовою, яка може бути оброблена конкретною СКБД.

1.1.2. Основні поняття моделі "сутність-зв'язок". ER-моделі

Інфологічна модель БД, повинна включати такий формалізований опис предметної області, який легко буде "читатися" як фахівцями з баз даних, так і усіма користувачами. І цей опис має бути настільки містким, щоб можна було оцінити глибину і коректність опрацювання проекту БД, і звичайно, воно не має бути прив'язане до конкретної СКБД. Вибір СКБД – це окреме завдання, і для коректного його вирішення необхідно вже мати проект, який не прив'язаний ні до якої конкретної СКБД.

Інфологічне проектування, передусім пов'язано із спробою представлення семантики предметної області в моделі БД, яка слабо відображається в мережних та ієрархічних моделях даних.

Було запропоновано декілька моделей даних, названих семантичними моделями. В усіх цих моделях були свої позитивні і

негативні сторони, але випробування часом витримала тільки модель "сутність-зв'язок", або Entity Relationships, яка стала фактичним стандартом при інфологічному моделюванні баз даних. Загальноприйнятою стала скорочена назва ER-модель, а більшість сучасних CASE-засобів містять можливість опису даних на основі цієї моделі. Крім того, розроблені методи автоматичного перетворення проекту БД з ER-моделі в реляційну БД, при цьому одночасно виконується перетворення в модель конкретної СКБД.

Усі CASE-системи мають розвинені засоби документування процесу розробки, автоматичні генератори звітів дозволяють підготувати звіт про поточний стан проекту з детальним описом об'єктів БД і їх зв'язків, що істотно полегшує ведення проекту.

Зараз не існує єдиної загальноприйнятої системи позначень для ER-моделі, і різні CASE-системи використовують різні графічні нотації (Чена, Мартина, Баркера та ін.), але розібравшись з однією, можна легко зрозуміти й інші.

Як і будь-яка інша модель – модель "сутність-зв'язок" має декілька базових понять, які утворюють початкові цеглинки, з яких будуються вже складніші об'єкти за заздалегідь визначеними правилами.

Ця модель найбільшою мірою узгоджується з концепцією об'єктно-орієнтованого проектування, і поза сумнівом, є базовою для розробки складних програмних систем.

1.1.3. Базові поняття ER-моделі

Сутність (Entity) – множина екземплярів реальних або абстрактних об'єктів (людей, подій, станів, ідей, предметів та ін.), що мають загальні атрибути або характеристики. Будь-який об'єкт системи може бути представлений тільки однією сутністю, яка має бути унікально ідентифікована. При цьому ім'я сутності повинне відбивати тип або клас об'єкту, а не його конкретний екземпляр (наприклад, АЕРОПОРТ, а не БОРИСПІЛЬ або ХІТРОУ).

Кожна сутність повинна мати унікальний ідентифікатор. Кожен екземпляр сутності повинен однозначно ідентифікуватися і відрізнятися від усіх інших екземплярів цього типу сутності. Кожна сутність повинна мати деякі властивості, а саме:

1) унікальне ім'я; до одного і того ж імені повинна завжди застосовуватися одна і та ж інтерпретація; одна і та ж інтерпретація не

може застосовуватися до різних імен, якщо тільки вони не є псевдонімами;

2) один або декілька атрибутів, які або належать сутності, або наслідуються через зв'язок;

3) один або декілька атрибутів, які однозначно ідентифікують кожен екземпляр сутності.

Кожна сутність може мати будь-яку кількість зв'язків з іншими сутностями моделі.

Зв'язок (Relationship) – пойменована асоціація між двома сутностями, що має значення для даної предметної області. Зв'язок – це асоціація між сутностями, при якій кожен екземпляр однієї сутності асоційований з довільною (у тому числі нульовою) кількістю екземплярів іншої сутності, і навпаки.

Атрибут (Attribute) – будь-яка характеристика сутності, значима для даної предметної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або вираження стану сутності. Атрибут представляє тип характеристики або властивості, що асоціюються з множиною реальних або абстрактних об'єктів (людей, місць, подій, станів, ідей, предметів і так далі). ***Екземпляр*** атрибуту – це певна характеристика окремого елементу множини. Екземпляр атрибуту визначається типом характеристики і її значенням, що називається значенням атрибуту.

Ключ (Key) – атрибут або множина атрибутів, що однозначно ідентифікує конкретний екземпляр сутності (***ключові*** *атрибути)*.

Приклади. Є сутність Співробітник, у якої може бути такий набір атрибутів: Табельний номер, Прізвище, Ім'я, По батькові, Дата народження, Кількість дітей, Наявність родичів за кордоном. Для сутності Співробітник ключовим буде атрибут Табельний номер, оскільки для усіх співробітників цього підприємства табельні номери різні.

Екземпляром сутності Співробітник буде опис конкретного співробітника підприємства. Одне із загальноприйнятих графічних позначень сутності – прямокутник, у верхній частині якого записано ім'я сутності, а нижче перераховуються атрибути, причому ключові атрибути

позначаються, наприклад, підкресленням або спеціальним шрифтом, як показано на рис. 1.1.

Рис. 1.1. **Приклад позначення сутності**

Між сутностями можуть бути встановлені зв'язки – бінарні асоціації, що показують, яким чином сутності співвідносяться або взаємодіють між собою. Зв'язок може існувати між двома різними сутностями або між сутністю та самою ж нею (рекурсивний зв'язок). Вона показує, як пов'язані екземпляри сутностей між собою. Якщо зв'язок встановлюється між двома сутностями, то він визначає взаємозв'язок між екземплярами однієї та іншої сутності. Наприклад, якщо є зв'язок між сутністю "Студент" і сутністю "Викладач" і цей зв'язок – керівництво дипломними проектами, то кожен студент має тільки одного керівника, але один і той же викладач може керувати декількома студентами-дипломниками. Тому це буде зв'язок "один-до-багатьох" (1:M), один з боку "Викладач" і багато з боку "Студент" (рис. 1.2).

Рис. 1.2. **Зв'язок сутностей Викладач та Студент**

У різних нотаціях потужність зв'язку відображається по-різному, але сенс має один. Графічна інтерпретація зв'язку дозволяє відразу прочитати сенс взаємозв'язку між сутностями, вона наочна і легко

інтерпретується. Зв'язки діляться на три типи за потужністю: один-до-одного (1:1), один-до-багатьох (1:М), багато-до-багатьох (М:М).

Зв'язок один-до-одного означає, що екземпляр однієї сутності пов'язаний тільки з одним екземпляром іншої сутності. Зв'язок 1:М означає, що один екземпляр сутності, розташований ліворуч лінії зв'язка, може бути пов'язаний з декількома екземплярами сутності, розташованими праворуч. Зв'язок "багато-до-багатьох" (М:М) означає, що один екземпляр першої сутності може бути пов'язаний з декількома екземплярами другої сутності, і навпаки, один екземпляр другої сутності може бути пов'язаний з декількома екземплярами першої сутності.

Наприклад, зв'язок типу "Вивчає" між сутностями "Студент" і "Дисципліна" є зв'язок типу "багато-до-багатьох" (М:М), оскільки кожен студент може вивчати декілька дисциплін, а кожна дисципліна вивчається множиною студентів.

Між двома сутностями може бути задано скільки завгодно зв'язків з різними смисловими навантаженнями. Наприклад, між двома сутностями "Студент" і "Викладач" можна встановити два зв'язка, один – розглянутий вже раніше "Дипломне проектування", а другий може бути умовно названий "Лекції", і він визначає – лекції яких викладачів слухає цей студент і яким студентам цей викладач читає лекції. Зрозуміло, що це зв'язок типу багато-до-багатьох (рис.1.3).

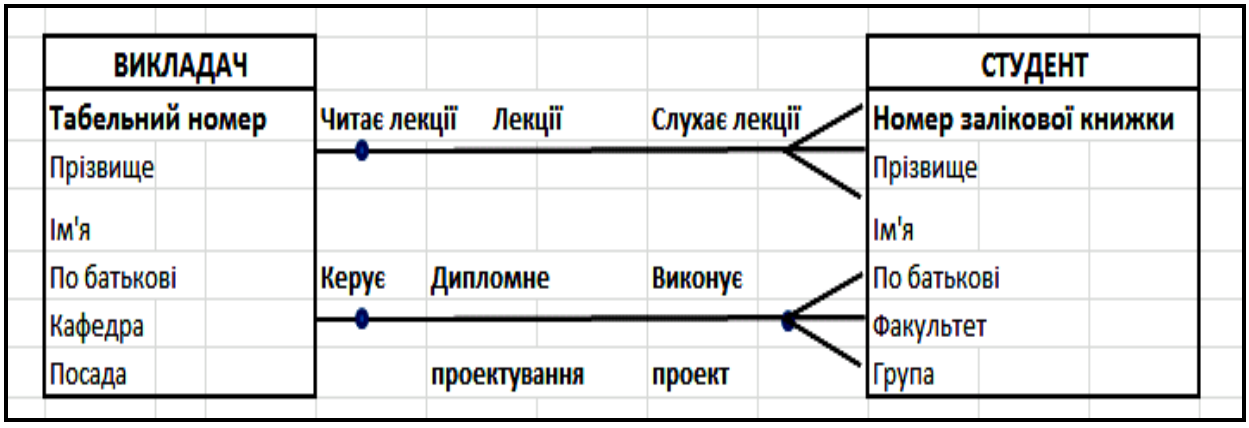


Рис. 1.3. **Множинні зв'язки між сутностями**

Зв'язок будь-якого з цих типів може бути обов'язковим, якщо в цьому зв'язку повинен брати участь кожен екземпляр сутності, і необов'язковою – якщо не кожен екземпляр сутності повинен брати участь в цьому зв'язку. При цьому зв'язок може бути обов'язковим з одного боку і необов'язковим з іншого боку. Обов'язковість зв'язку теж по-різному позначається в різних нотаціях.

1.1.4. Представлення ER-діаграм у нотації IDEF1X

Існує декілька способів відображення ER-діаграм. Найбільш поширеними є метод Баркера і метод IDEFI.

Метод Баркера заснований на нотації, запропонованій автором, і використовується в CASE-засобі Oracle Designer.

Метод IDEFI заснований на підході Чена і дозволяє побудувати модель даних, еквівалентну реляційній моделі в третій нормальній формі. На основі вдосконалення методу IDEFI створена його нова версія – метод **IDEF1X**, розроблений з урахуванням таких вимог, як простота для вивчення і можливість автоматизації. IDEF1X-диаграми використовуються у ряді поширених CASE-засобів (зокрема, ERwin, Design/IDEF
[12; 31]).

У методі IDEF1X сутність є незалежною від ідентифікаторів або просто незалежною, якщо кожен екземпляр сутності може бути однозначно ідентифікований без визначення його зв'язку з іншими сутностями. Наприклад: сутності Клієнт і Замовлення. Для ідентифікації замовлення немає необхідності в посиланні на клієнта, який зробив це замовлення.

Сутність називається залежною від ідентифікаторів або просто залежною, якщо однозначна ідентифікація екземпляра сутності залежить від його відношення до іншої сутності. Приклад: сутності – Замовлення і Позиція замовлення. Для ідентифікації позиції замовлення треба послатися на замовлення, в яке входить ця позиція.

Зв'язок може додатково визначатися за допомогою вказівки ступені або потужності (кількості екземплярів сутності-нащадка, яке може породжувати кожен екземпляр сутності-батька). У IDEF1X можуть бути створені такі потужності зв'язків :

- кожен екземпляр сутності-батька може мати нуль, один або більше за один пов'язаних з ним екземплярів сутності-нащадка;
- кожен екземпляр сутності-батька повинен мати не менше одного пов'язаного з ним екземпляра сутності-нащадка;
- кожен екземпляр сутності-батька повинен мати не більше одного пов'язаного з ним екземпляра сутності-нащадка;

• кожен екземпляр сутності-батька пов'язаний з деяким фіксованим числом екземплярів сутності-нащадка.

Якщо екземпляр сутності-нащадка однозначно визначається своїм зв'язком з сутністю-батьком, то зв'язок називається ідентифікуючим, інакше – неідентифікуючим.

Зв'язок зображується лінією, що проводиться між сутністю-батьком і сутністю-нащадком, з точкою на кінці лінії у сутності-нащадка (рис. 1.4). Потужність зв'язків може набувати таких значень: N – нуль, один або більше, Z – нуль або один, P – один або більше. За замовчуванням потужність зв'язків приймається рівною N.

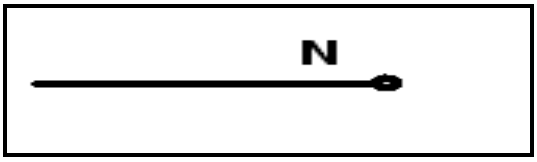


Рис.1.4. **Графічне зображення потужності зв'язку**

Ідентифікуючий зв'язок між сутністю-батьком і сутністю-нащадком зображується суцільною лінією. Сутність-нащадок в ідентифікуючому зв'язку є залежною від ідентифікатора сутності. Сутність-батько в ідентифікуючому зв'язку може бути як незалежним, так і залежним від ідентифікатора сутності (це визначається її зв'язками з іншими сутностями).

Пунктирна лінія зображує неідентифікуючий зв'язок (рис. 1.5). Сутність-нащадок у неідентифікуючому зв'язку буде незалежною від ідентифікатора, якщо вона не є також сутністю-нащадком, в якому-небудь ідентифікуючому зв'язку.

Атрибути зображуються у вигляді списку імен усередині блоку сутності. Атрибути, що визначають первинний ключ, розміщуються спочатку і відділяються від інших атрибутів горизонтальною рисою.

Сутності можуть мати також **зовнішні ключі** (Foreign Key), які використовуються як частина або цілий *первинний ключ* або неключовий *атрибут*. Для позначення зовнішнього ключа всередині блоку сутності поміщають імена атрибутів, після яких записують букви FK в дужках (рис. 1.5).

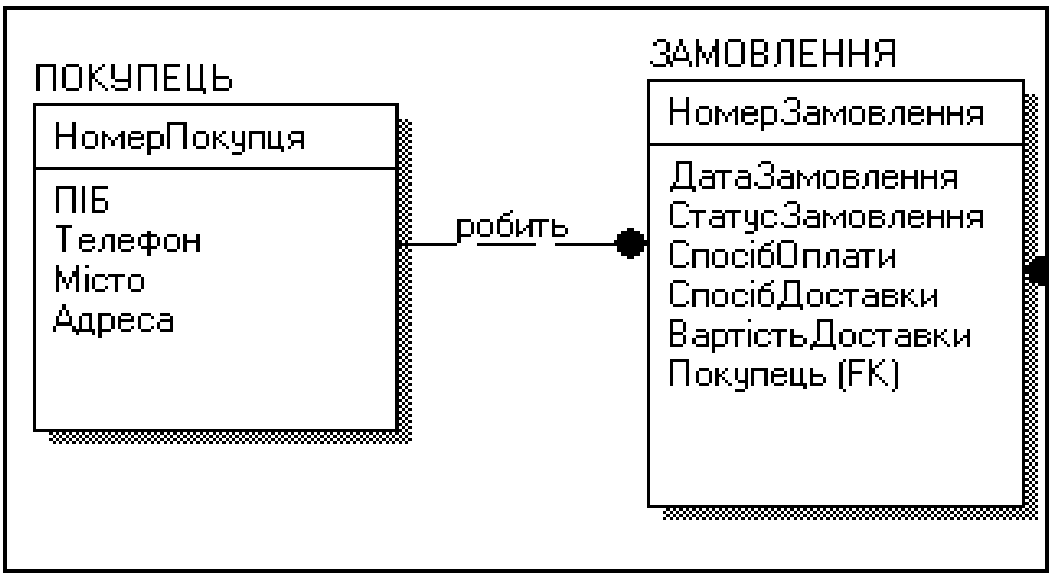


Рис. 1.5. **Неідентифікуючий зв'язок**

1.2. Інтерфейс Erwin Data Modeler

При запуску програми Erwin Data Modeler з'являється її вікно, в якому затінені усі палітри інструментів, окрім меню і кнопок створення нової моделі і відкриття моделі (рис. 1.6).

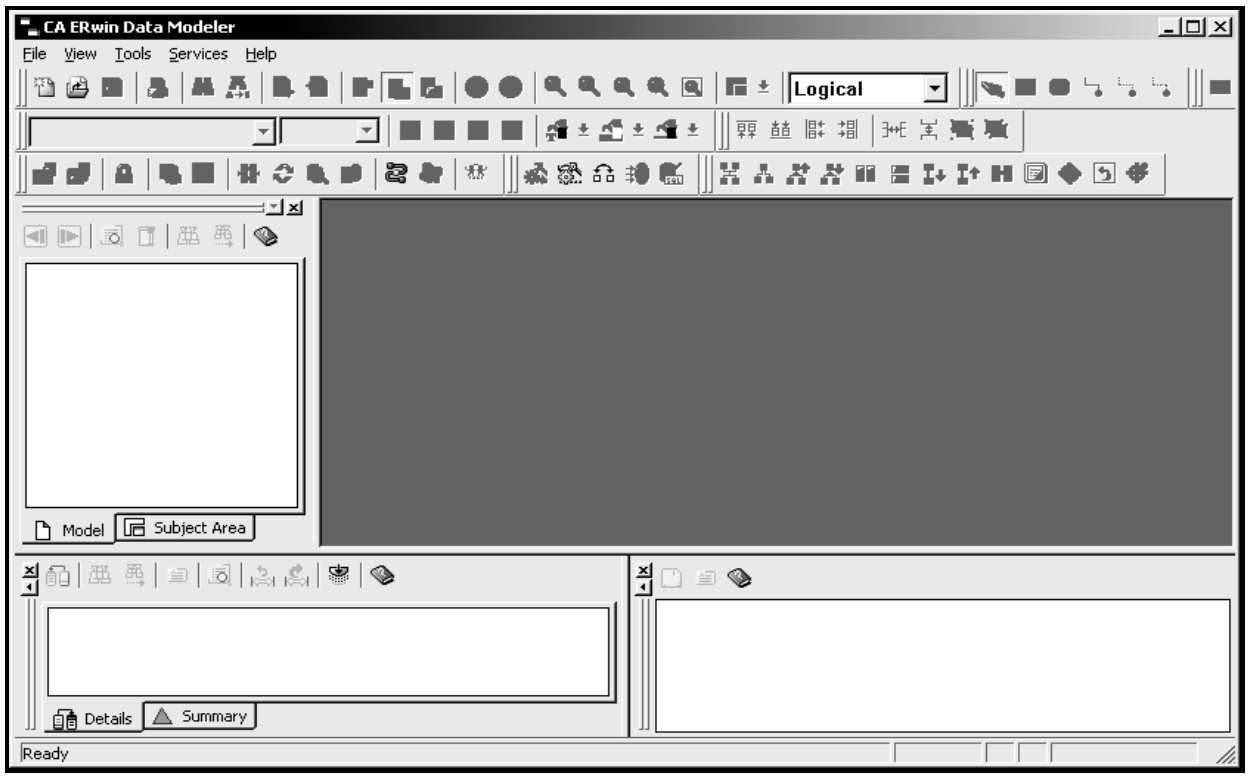


Рис. 1.6 **Початковий вид вікна Erwin Data Modeler**

При створенні нової моделі вимагається вказати її тип: логічний, фізичний або логічний/фізичний (рис.1.7).

Після вибору режиму створення моделі або відкриття вже існуючої моделі, палітра інструментів стає доступною (рис. 1.8).

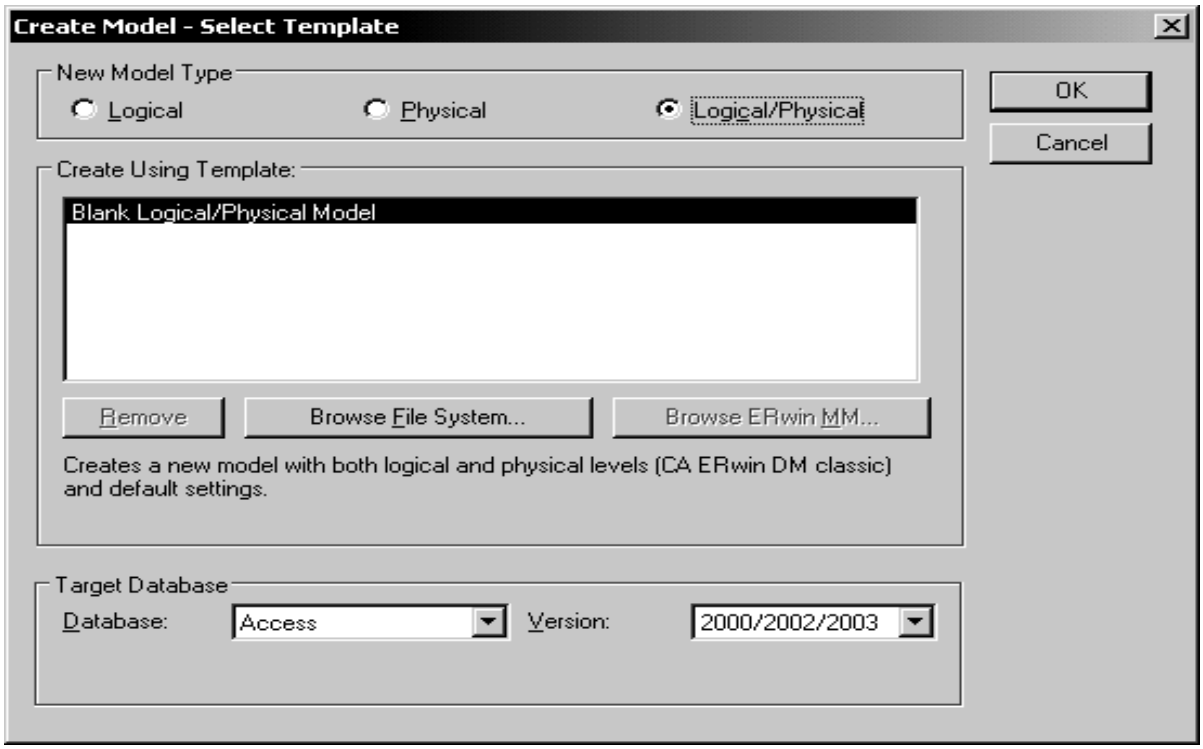


Рис. 1.7 **Вибір типу нової моделі**

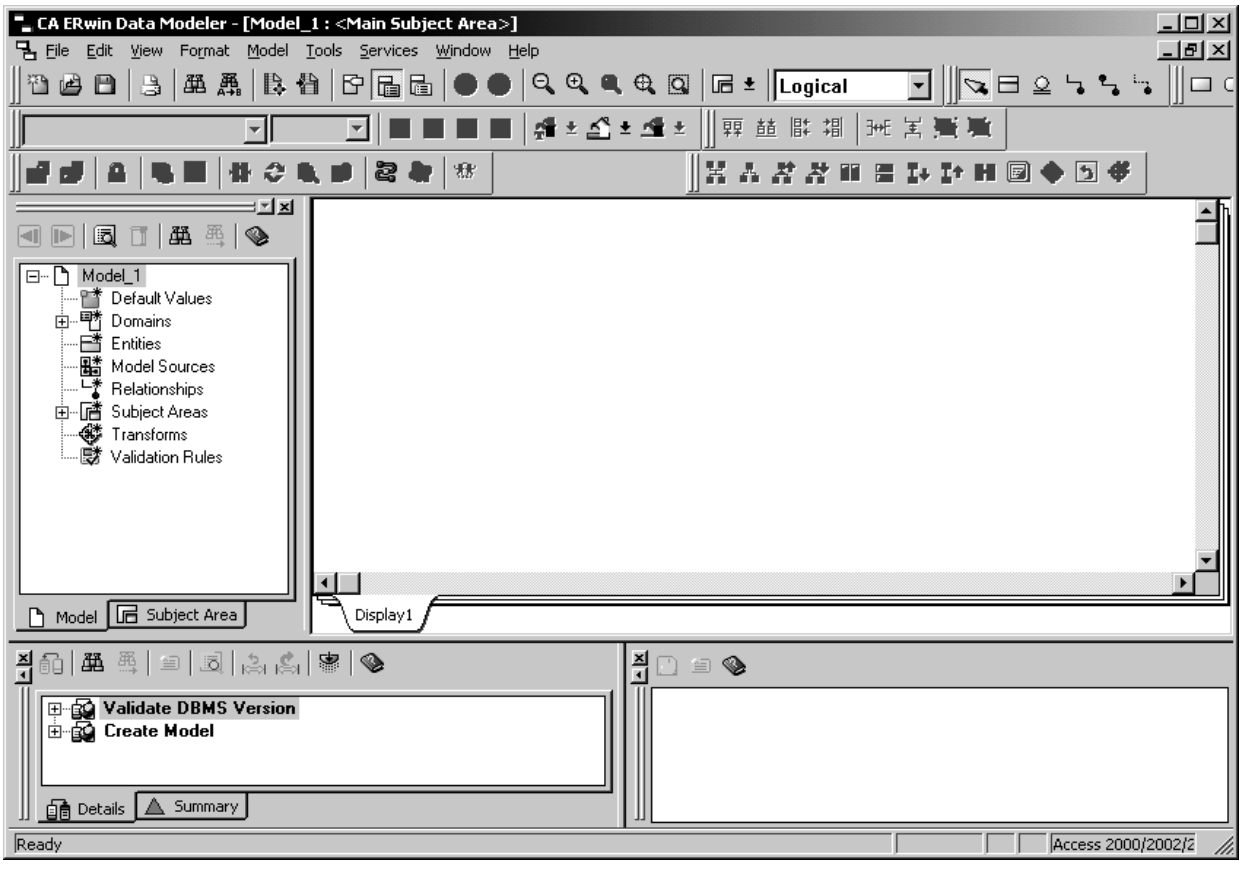
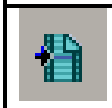
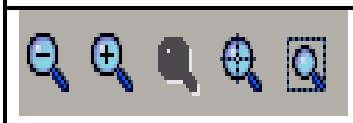
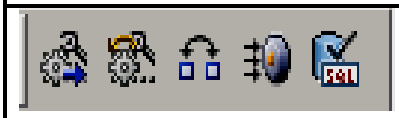
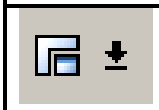


Рис. 1.8. **Вид вікна Erwin Data Modeler для нового проекту**

Основне призначення кнопок на панелі інструментів таке (табл.1.1).

Таблиця 1.1.

Основна панель інструментів ERWin

Кнопки	***Призначення кнопок***
	Створення, відкриття, збереження та друк моделі
	Виклик діалогу Report Template Builder для генерації звітів
	Зміна рівня перегляданню моделі: сутності, атрибути, визначення
	Зміна масштабу перегляданню моделі
	Генерація схеми бази даних, вибір сервера, валідація моделі (тільки на фізичному рівні)
	Перемикання між рівнями моделі – Subject Area

Палітра інструментів виглядає по-різному на різних рівнях відображення моделі.

На логічному рівні панель інструментів виглядає таким чином (рис. 1.9).

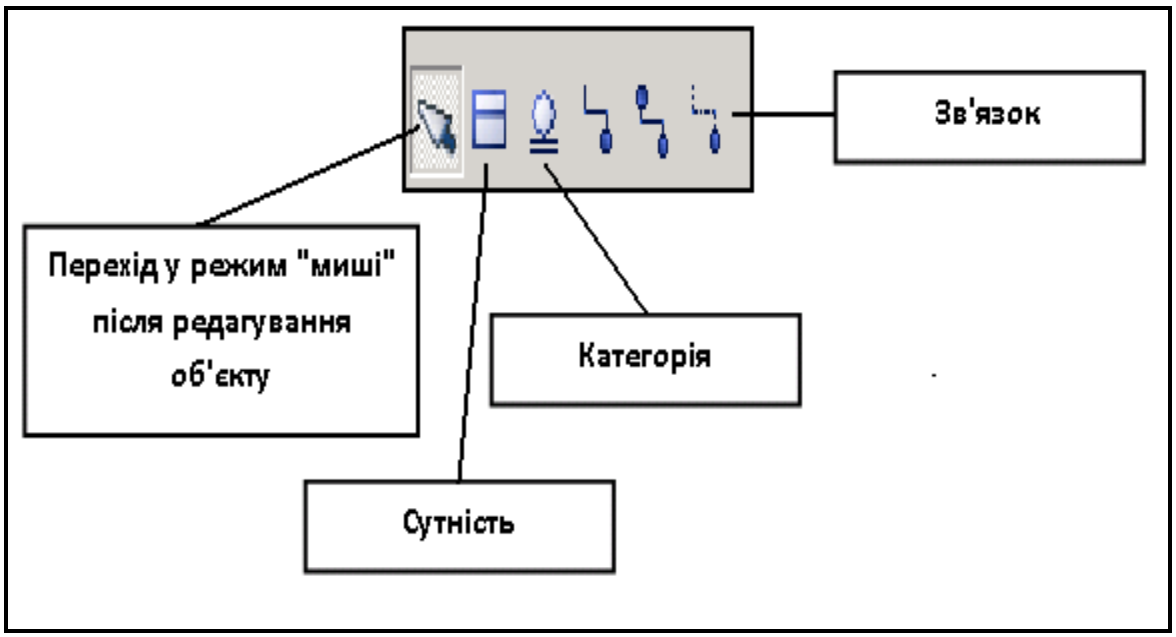


Рис. 1.9. **Палітра інструментів на логічному рівні**

Кнопка покажчика (режим миші) – в цьому режимі можна встановити фокус на якому-небудь об'єкті моделі.

Кнопка внесення сутності – для внесення сутності треба клацнути лівою кнопкою миші по кнопці внесення сутності і один раз по вільному простору на моделі. Повторне клацання приведе до внесення в модель ще однієї нової сутності. Для редагування сутностей або інших об'єктів моделі необхідно перейти в режим покажчика.

Кнопка категорії. Категорія, або категоріальний зв'язок – це спеціальний тип зв'язка між сутностями. Для встановлення категоріального зв'язка треба клацнути лівою кнопкою миші по кнопці категорії, потім один раз клацнути по сутності – родовому предку і, потім по сутності – нащадку.

Кнопка створення зв'язків: ідентифікуючих, "багато-до-багатьох" та неідентифікуючих.

На фізичному рівні палітра інструментів має такий вигляд (рис. 1.10). З'являється можливість створити таблиці та подання.

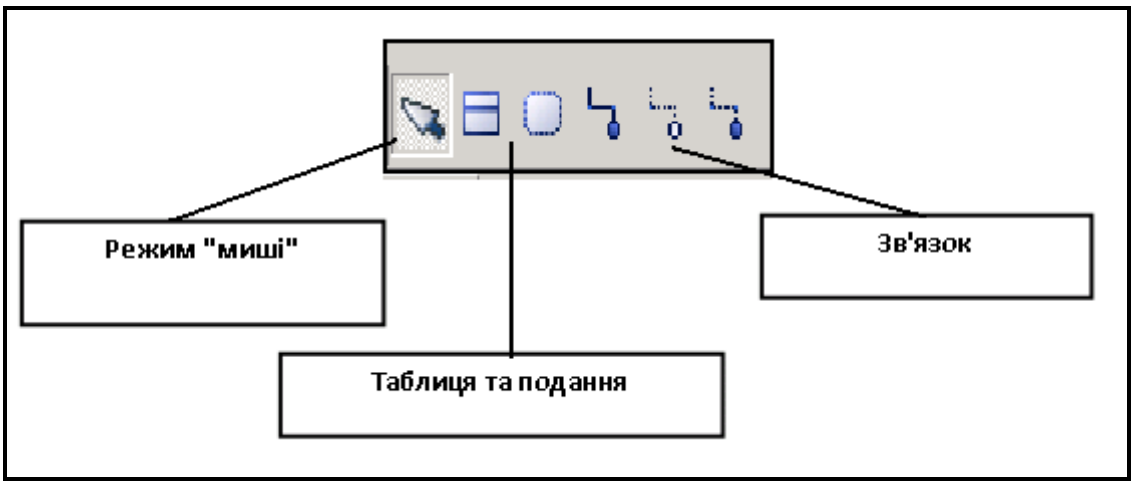


Рис. 1.10. **Палітра інструментів на фізичному рівні**

Для перемикання між логічною і фізичною моделлю даних служить список вибору в центральній частині панелі інструментів ERwin (рис. 1.11). Якщо при перемиканні фізичної моделі ще не існує, вона буде створена автоматично.

Рис. 1.11. **Перемикання між логічною та фізичною моделлю**

1.3. Створення моделі предметної області "Продаж хлібобулочних виробів"

Для ілюстрації роботи з Erwin Data Modeler спочатку спроектуємо просту ІС, яка базується на, розглянутій у попередніх лабораторних роботах [18], системі обліку продажу хлібобулочних виробів.

1.3.1. Створення моделі

Розробку бази даних розпочинаємо зі створення нової моделі в ERwin, для цього треба виконати такі дії:

- запустити ERwin;
- вибрати Create a new model (Створити нову модель);
- вибрати тип моделі. Вибравши тип Physical або Logical/Physical, можна відразу визначити базу даних, в якій буде реалізована модель.

Виберемо тип моделі Logical/Physical. Це дозволить надалі легко перемикатися між логічним і фізичним рівнями. У списку пропонованих баз даних виберемо Access. Пізніше, у разі потреби, середовище реалізації можна буде змінити.

1.3.2. Поняття про логічні та фізичні рівні

Як вже було зазначено раніше, існує два різні способи мислення і моделювання – логічний рівень і фізичний рівень. Поняття логічний рівень передбачає, що ми мислимо в поняттях реального світу і безпосередньо з нього беремо об'єкти для моделювання. На логічному рівні не має значення, наприклад, якою СКБД ми користуватимемося, чи є деяке число цілим або дійсним, як краще індексувати таблицю тощо.

Після того, як модель побудована на логічному рівні, вона виражається в термінах фізичної бази даних, включаючи вибір СКБД, типів даних за замовчуванням і ефективних схем індексування. Це

прийнято називати фізичним рівнем моделі ERwin. ERwin підтримує побудову і організацію роботи на цих двох рівнях для однієї діаграми.

Модель в ERwin, що має тільки логічний рівень може бути синхронізована з декількома моделями, що мають тільки фізичний рівень. Таким чином, на основі однієї логічної моделі можна створювати декілька фізичних, таких, що відповідають різним СКБД.

1.3.3. Робота з діаграмою ERwin

Робоча область ERwin складається з двох частин. Ліворуч розташований навігатор моделі, що надає додаткові можливості пошуку і редагування об'єктів моделі. Основну зону вікна займає безпосередньо сама діаграма. У верхній частині вікна розташовані панелі інструментів. Інтерфейс виконаний в стилі Windows-застосувань, досить простий і інтуїтивно зрозумілий.

Сутності та атрибути. Сутність служить для представлення набору реальних або абстрактних предметів (людей, місць, подій і тому подібне), які мають загальні атрибути або характеристики. Кожна сутність являє собою множину подібних індивідуальних об'єктів, що називаються екземплярами. Атрибут виражає певну властивість об'єкту. На фізичному рівні сутність представляється таблицею, атрибут – колонкою таблиці, екземпляр – рядком таблиці.

Для наведеного прикладу виділимо такі основні логічні об'єкти: Товари і Виробники.

Створення сутності

1. Клацніть значок  на панелі інструментів. Курсор набере вигляду  :

2. Клацніть на будь-якому місці діаграми. На екрані з'явиться нова сутність.

3. За замовчуванням, сутності привласнюється ім'я E/x, де x – унікальний номер сутності. Змінити ім'я можна, клацнувши по сутності лівою кнопкою миші, або вибравши пункт Entity Properties в контекстному меню сутності (рис. 1.12).

4. У властивостях сутності також можна дати опис сутності, ввести замітки і вказати іншу інформацію.

На логічному рівні бажано називати сутності українськими (російськими) словами для полегшення читання схеми усіма учасниками процесу розробки. Пізніше на фізичному рівні можна змінити назви

сутностей і атрибутів на латинські, оскільки більшість СКБД підтримують тільки латинські букви.

Обов'язково складайте описи створюваних сутностей. Це допоможе надалі зрозуміти, що це за об'єкт і зробить схему читабельною для інших учасників процесу розробки. Крім того, побудувавши після створення діаграми звіт, можна отримати готову документацію за схемою.

Для внесення опису і зауважень виберіть в контекстному меню сутності пункт меню Entity Properties. У полі Definition вкажіть визначення сутності (рис. 1.12).

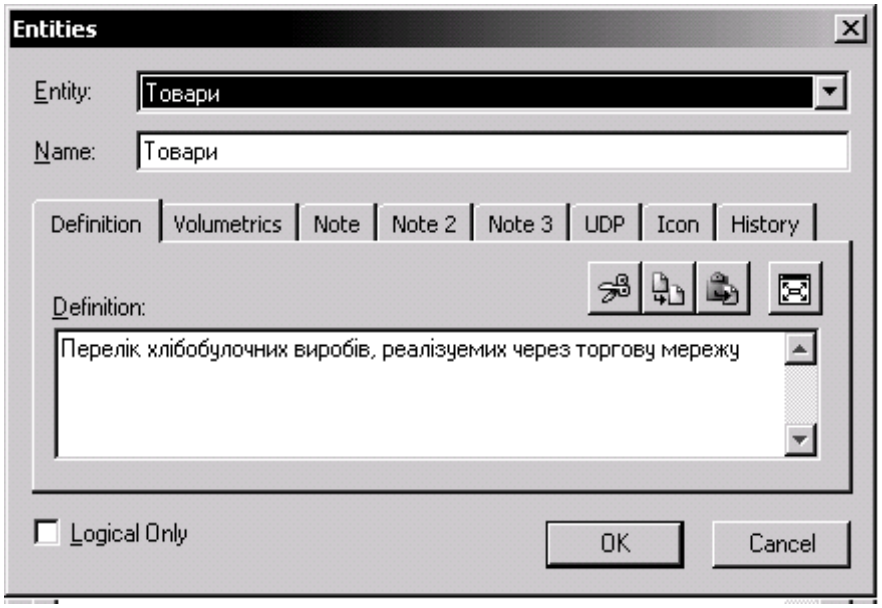


Рис. 1.12 **Вікно створення нової сутності**

На фізичному рівні визначення сутності можна експортувати як частину схеми і використати в реальній БД.

У вкладці Note можна ввести корисне зауваження, що описує будь-яке бізнес-правило або угоду по організації діаграми.

У вкладці Note 2 зазвичай документуються деякі можливі запити, які, як очікується, використовуватимуться по відношенню до сутності у БД. Ця інформація буде корисною при проектуванні БД на фізичному рівні.

Вкладка Note 3 дозволяє вводити приклади даних для сутності в довільній формі.

Створення атрибутів

1. Виділіть сутність.
2. Клацніть правою кнопкою і виберіть Attributes в контекстному меню сутності. З'являється діалог Attributes (рис. 1.13).

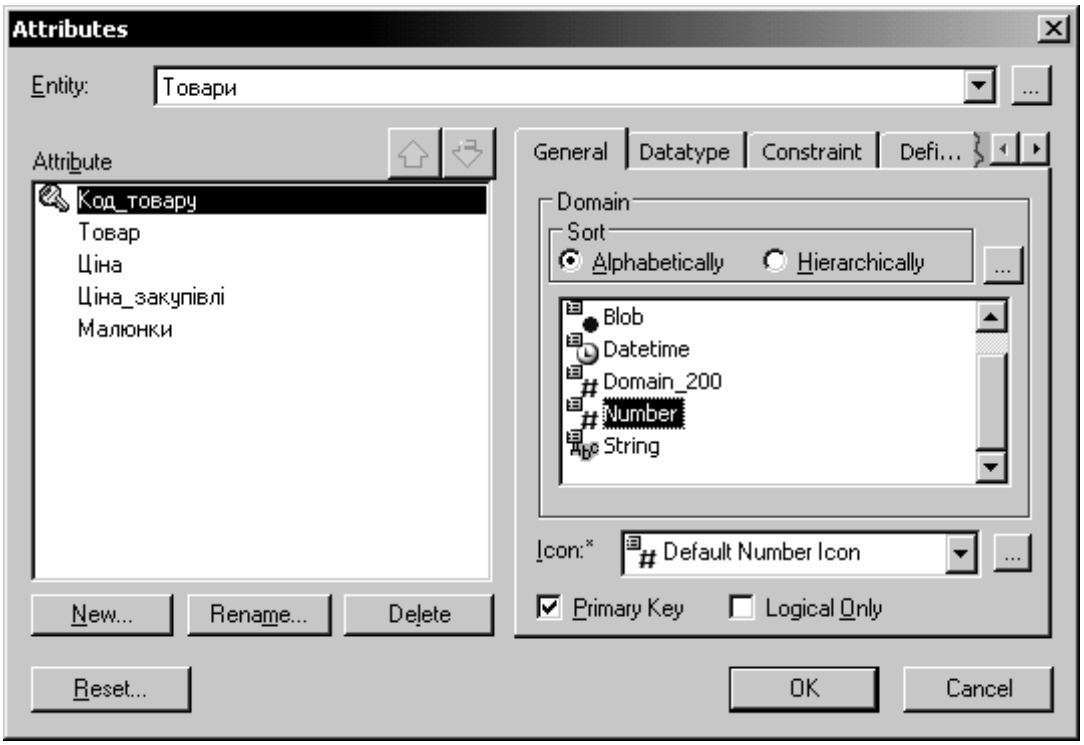


Рис. 1.13. **Вікно створення атрибутів сутності**

Щоб додати новий атрибут клацніть кнопку New, введіть ім'я атрибуту, ім'я колонки до БД і його домена. Домен атрибуту використовуватиметься при визначенні типу колонки на фізичному рівні. Для атрибутів первинного ключа у вкладці General необхідно зробити позначку у вікні вибору Primary Key.

Аналогічно сутностям, необхідно задавати опис атрибутам (вкладка Definition).

Далі створимо сутність Виробники і створимо атрибути для неї (рис. 1.14).

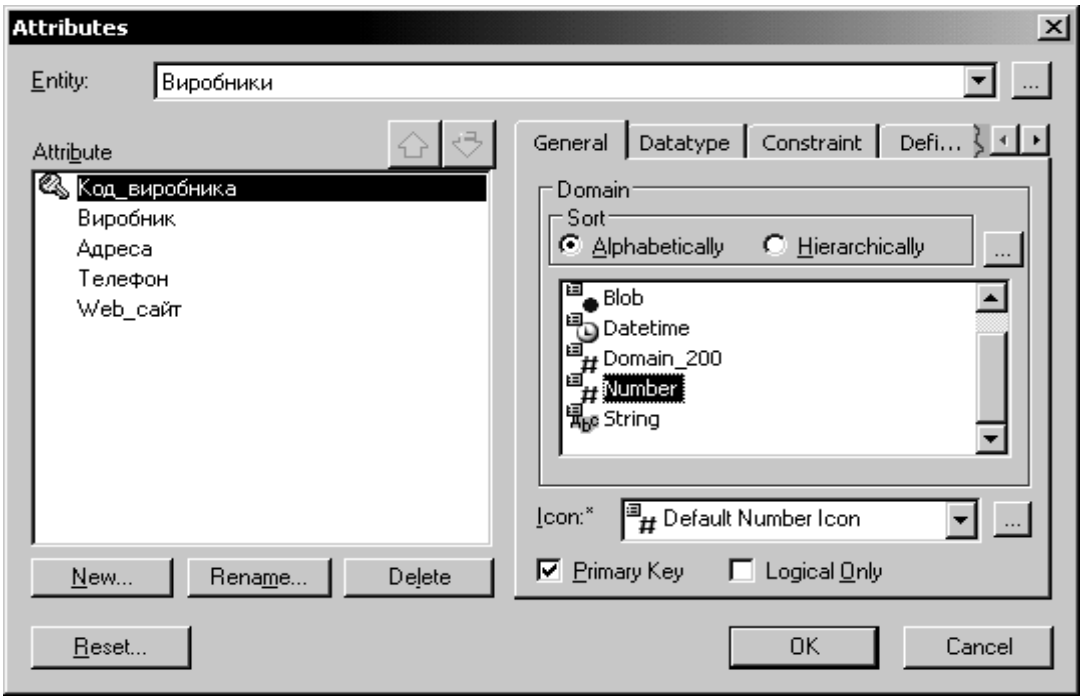


Рис. 1.14. **Створення сутності "Виробники"**

Загальний вид екрану в цьому випадку виглядає так (рис 1.15) якщо встановити рівень перегляду атрибутів (Attribute level)

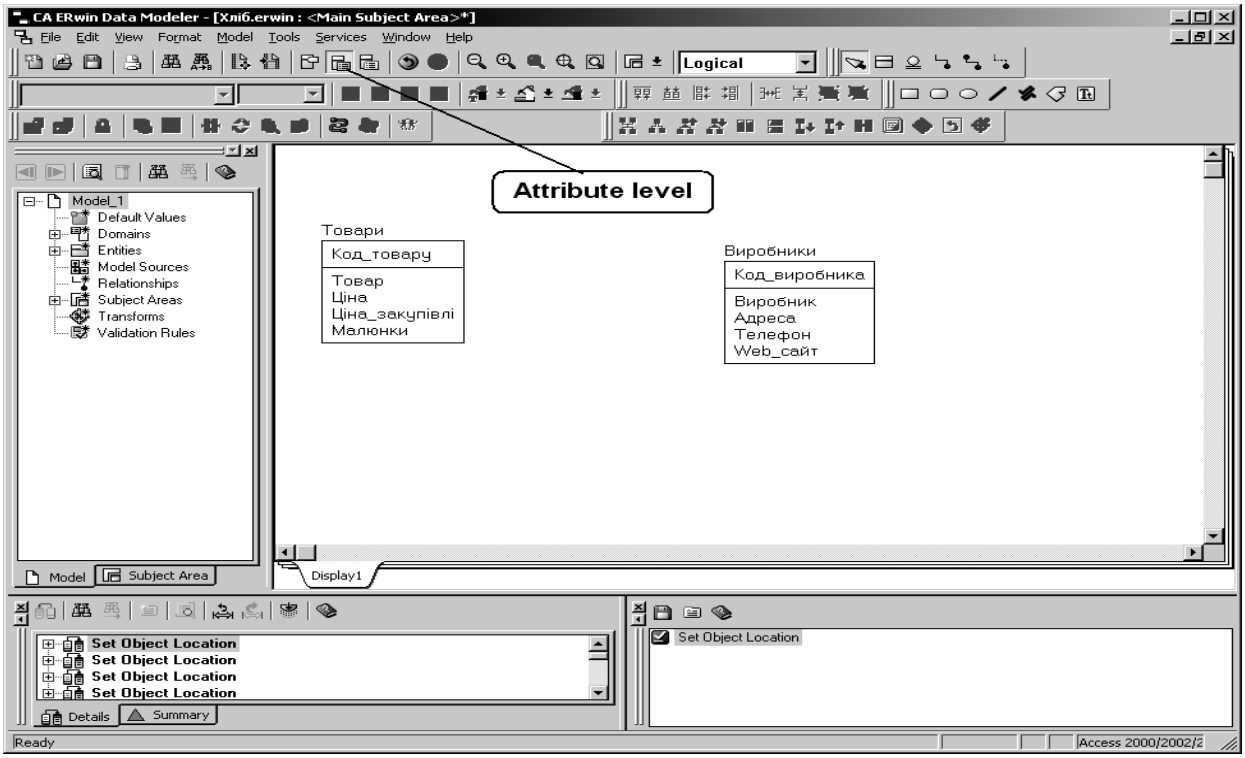


Рис. 1.15. **Вид екрану для двох створених сутностей**

Поняття зв'язку

Створення сутностей і інформації про них – це тільки частина картини. Необхідно показати, як взаємодіють між собою об'єкти моделі. Логічні з'єднання або асоціації між двома сутностями називаються

зв'язками. Дані, що відносяться до зв'язків, дуже важливі. Зв'язок – це співвідношення або між двома сутностями, або між сутністю і цією же сутністю. Кожний зв'язок повинен іменуватися дієсловом або дієслівною фразою, яка полегшує читання діаграми. Якщо взаємозв'язки між сутністю були правильно встановлені, то, для перевірки адекватності логічної моделі, можна скласти речення, що їх описують, наприклад, "Покупець *робить* Замовлення", "Багато замовлень робляться одним покупцем", тощо (рис. 1.16).

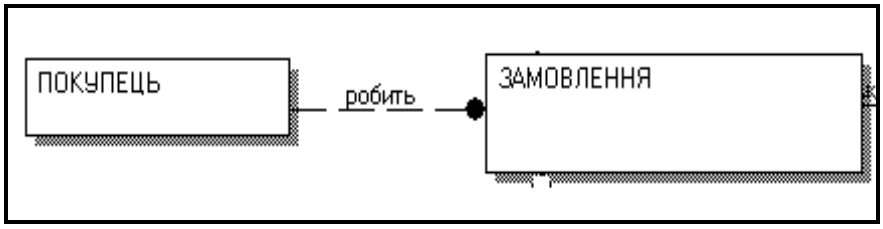


Рис. 1.16. **Відображення зв'язку між сутностями**

Складання таких речень дозволяє перевірити відповідність отриманої моделі вимогам і обмеженням створюваної системи.

Створення зв'язку

На логічному рівні можна встановити три типи зв'язків: ідентифікуючий зв'язок "один до багатьох" –  , неідентифікуючий зв'язок "один до багатьох" –  і невизначений зв'язок "багато-до-багатьох" –  .

Щоб створити зв'язок необхідно:

1. Клацнути по одній з вище приведених кнопок на панелі інструментів.
2. Клацнути по батьківській сутності.
3. Клацнути по дочірній сутності.

Типи зв’язків

Ідентифікуючий зв'язок – зв'язок, при якому екземпляр дочірньої сутності визначається через свій зв'язок з батьківською сутністю. Атрибути первинного ключа батьківської сутності стають атрибутами первинного ключа дочірньої. Операція доповнення атрибутів дочірньої

сутності при створенні зв'язка називається міграцією атрибутів. У дочірній сутності нові атрибути позначаються як зовнішній ключ (FK).

При встановленні ідентифікуючого зв'язка між сутностями дочірня сутність стає залежною. *Залежна сутність* – це сутність, екземпляри якої не можуть бути унікальним чином ідентифіковані без визначення її зв'язка з іншою сутністю або сутностями. Залежні сутності в Erwin зображується у вигляді прямокутника із закругленими кутами.

У даному прикладі ми не вводимо ідентифікуючих зв'язків. Кожен екземпляр всіх сутностей однозначно визначається своїм номером.

Неідентифікуючий зв'язок – це такий зв'язок, при якому екземпляр дочірньої сутності не ідентифікується через свій зв'язок з батьківською сутністю. Атрибути первинного ключа батьківської сутності стають неключовими атрибутами дочірньої. При встановленні неідентифікуючого зв'язка дочірня сутність залишається незалежною.

Незалежна сутність – це сутність, екземпляри якої можуть бути унікальним чином ідентифіковані без визначення її зв'язка з іншою сутністю.

У ілюстрованому прикладі зв'язок "Виробники виробляють Товари" матиме зв'язок "багато-до-багатьох", який може бути створений тільки на рівні логічної моделі (рис. 1.17, рис. 1.18).

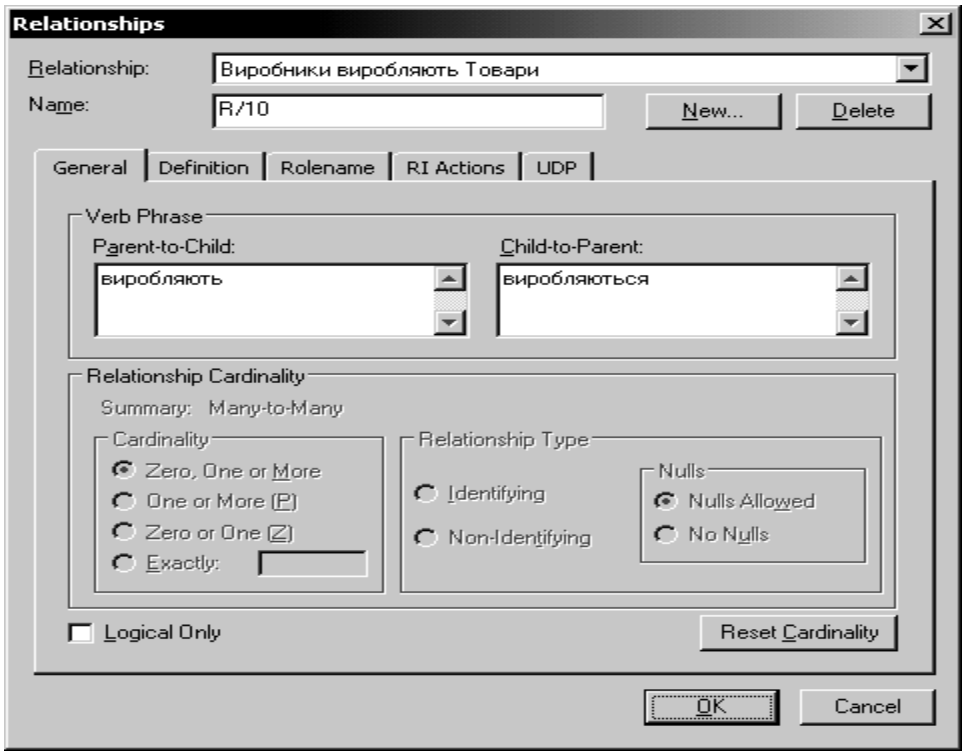


Рис. 1.17. **Вікно створення зв'язку**

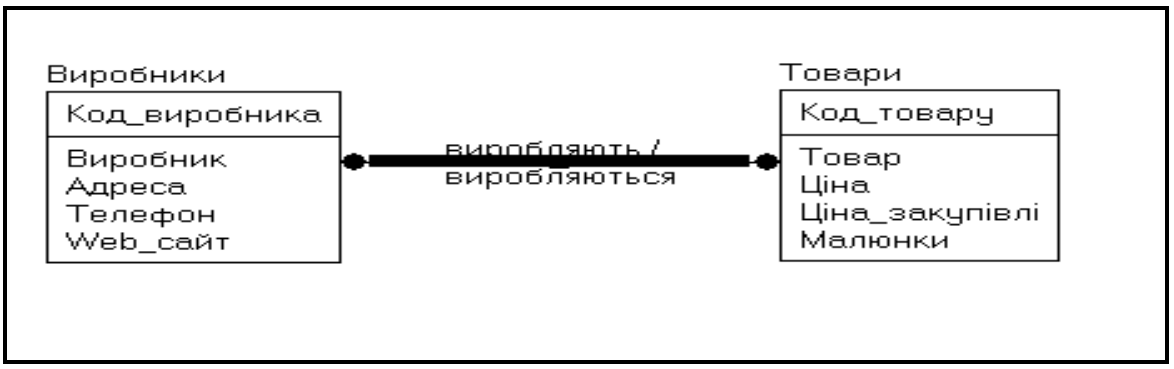


Рис. 1.18. **Відображення зв'язку на діаграмі**

Відображення на екрані дієслівних фраз

За замовчуванням дієслівна фраза не відображається на екрані. Для того, щоб показати на екрані дієслівну фразу, що відноситься до зв'язка, клацніть правою кнопкою миші по будь-якому місці у вікні діаграми і виберіть пункт Relationship Display -> Verb Phrase.

Типи сутностей та ієрархія наслідування

Як було вказано вище, зв'язки визначають, чи є сутність незалежною або залежною. Розрізняють декілька типів залежних сутностей.

Характеристична – залежна дочірня сутність, яка пов'язана тільки з однією батьківською і у цьому сенсі зберігає інформацію про характеристики батьківською сутності.

Прикладом характеристичної сутності може служити сутність Замовлення.

Асоціативна – сутність, пов'язана з декількома батьківськими сутностями. Така сутність містить інформацію про зв'язки сутностей.

Іменуюча – окремий випадок асоціативної сутності, що не має власних атрибутів (тільки атрибути батьківських сутностей, що мігрували як зовнішній ключ).

Категоріальна – дочірня сутність в ієрархії наслідування.

Ієрархія наслідування (або ієрархія категорій) є особливим типом об'єднання сутностей, які розділяють загальні характеристики.

Створення фізичної моделі даних

Для створення фізичної моделі слід позбавитися від зв'язків "багато-до-багатьох". Для цього виділимо зв'язок і запустимо майстер перетворення зв'язка "багато-до-багатьох" (рис. 1.19).

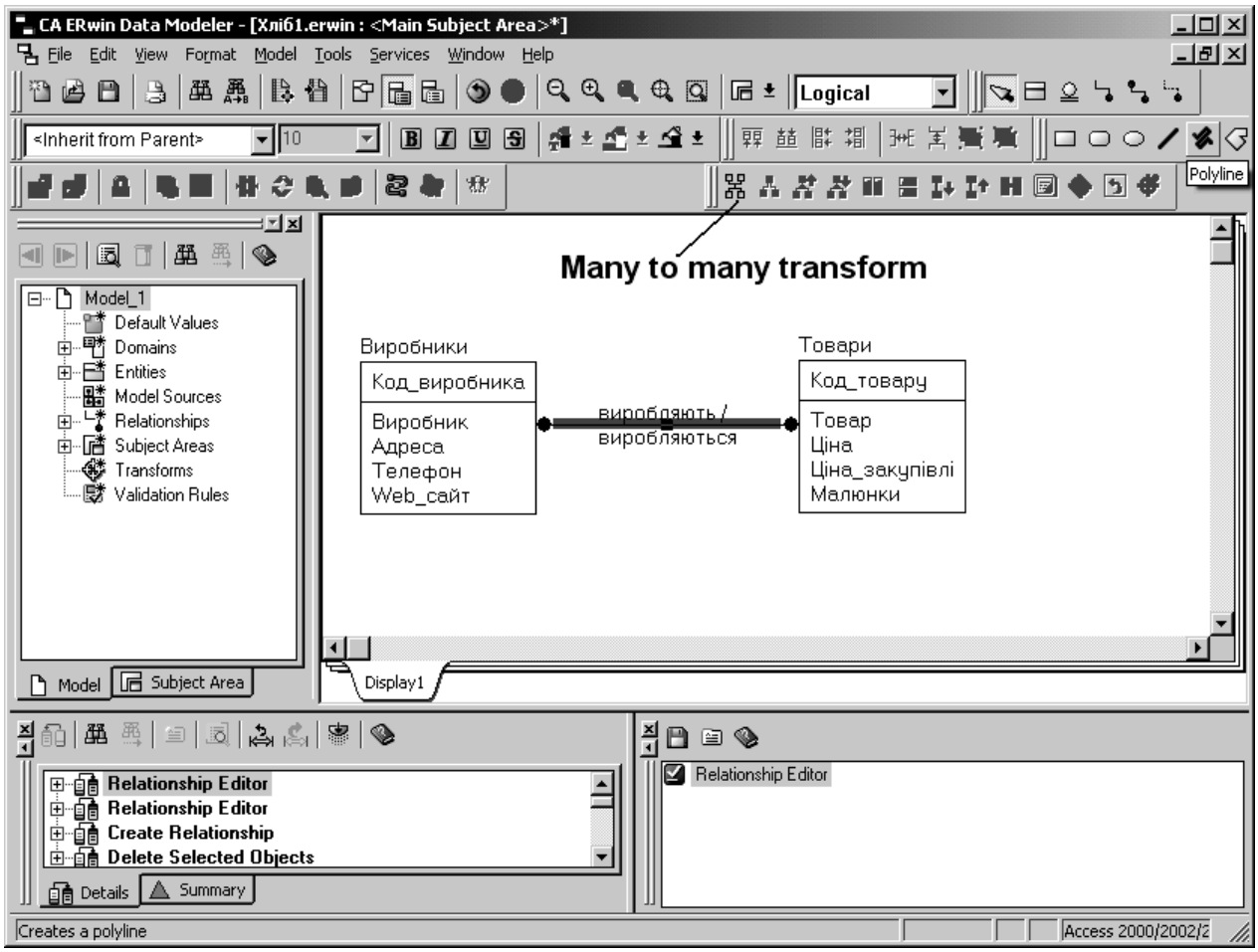


Рис. 1.19. **Перетворення зв'язка "багато-до-багатьох"**

Після запуску майстра в його початковому вікні натискаємо кнопку "Далі" і переходимо у вікно "Transform information", де вказуємо ім'я перетворення та його опис (рис. 1.20).

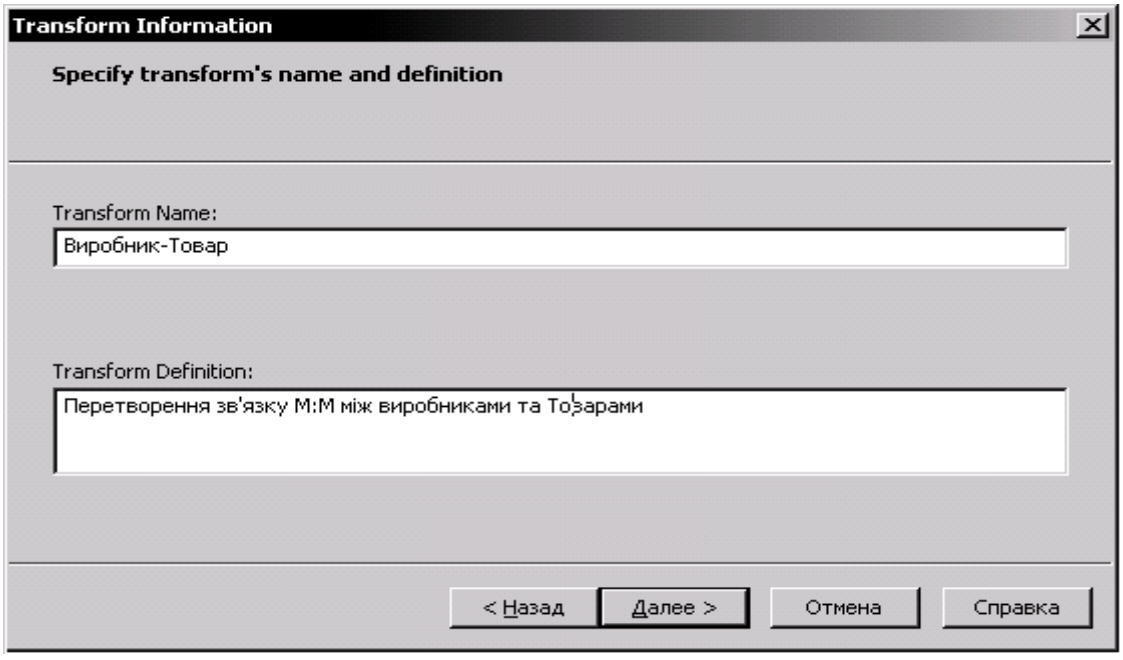


Рис. 1.20. **Вікно для внесення інформації про перетворення зв'язку**

Натискаємо кнопку "Далі" і переходимо до наступного вікна, в якому даємо ім'я новостворюваній асоціативній сутності і вводимо її опис (рис. 1.21).

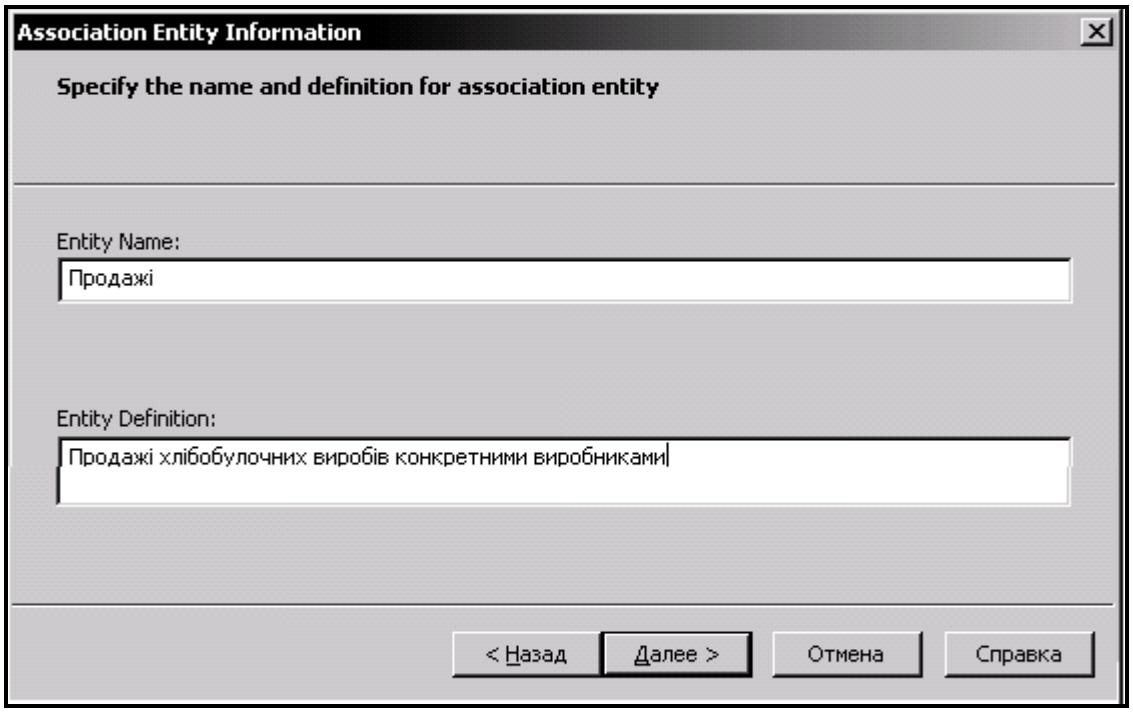


Рис. 1.21. **Опис новостворюваної асоціативної сутності**

Натискаємо кнопку "Далее" а потім "Готово". У результаті отримуємо таку діаграму (рис. 1.22)

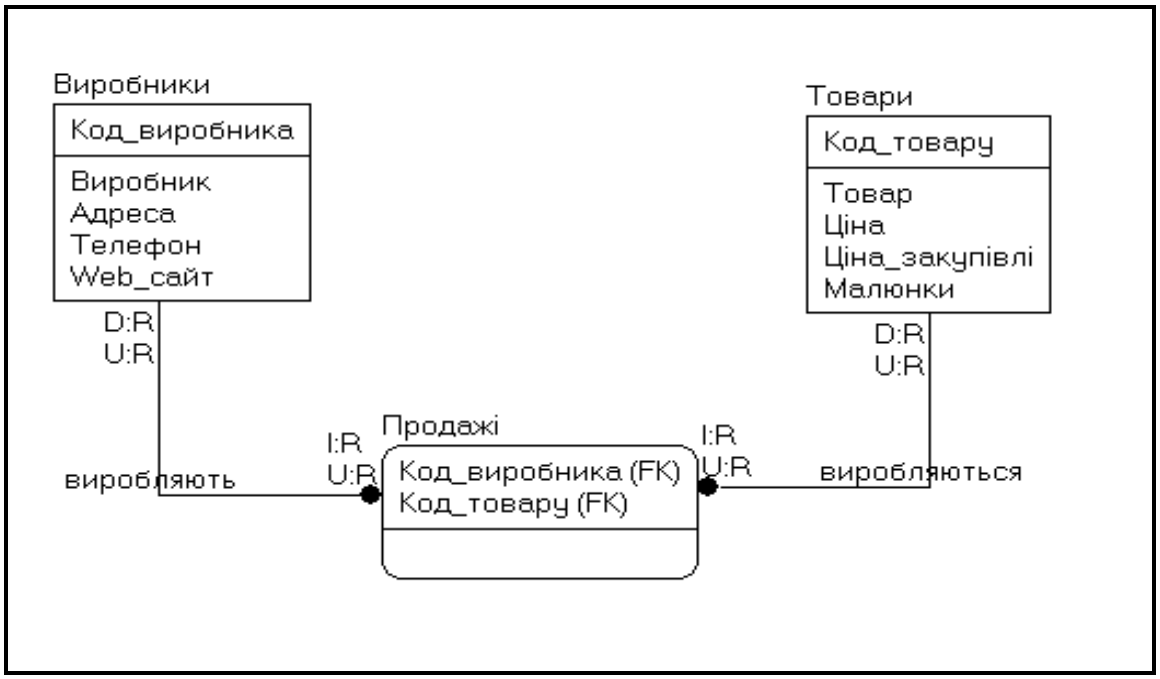


Рис. 1.22. **Вид створеної діаграми після перетворення зв'язку**

З'являється нова сутність "Продажі", така, що забезпечує асоціативний зв'язок між Виробниками і Товарами. Поява такої сутності позбавляє нас від зв'язка "багато до багатьох". Первинні ключі початкових сутностей стають складеним первинним ключем нової асоціативною сутності.

Після цього додамо до сутності Продажі нові атрибути (рис. 1.23).

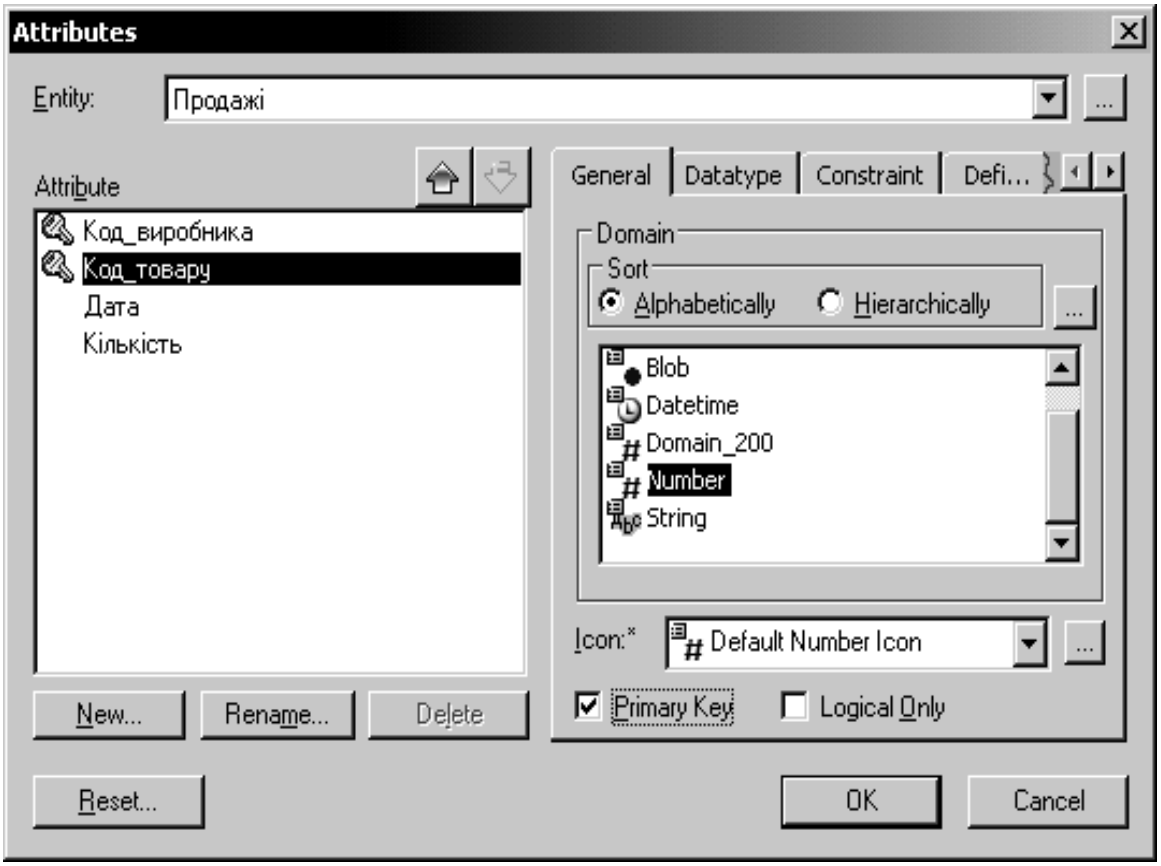


Рис. 1.23. **Додавання нових атрибутів до асоціативної сутності**

Генерація фізичної схеми БД

Процес генерації фізичної схеми БД з логічної моделі даних називається прямим проектуванням (Forward Engineering). При генерації фізичної схеми ERwin включає тригери посилальної цілісності, збережені процедури, індекси, обмеження та інші можливості, доступні при визначенні таблиць у вибраній СКБД.

Для генерації системного каталогу БД перейдіть з логічної моделі на фізичну, виберіть пункт меню Tools ->Forward Engineer/Schema

Generation або клацніть кнопку  на панелі інструментів Database (рис. 1.24).

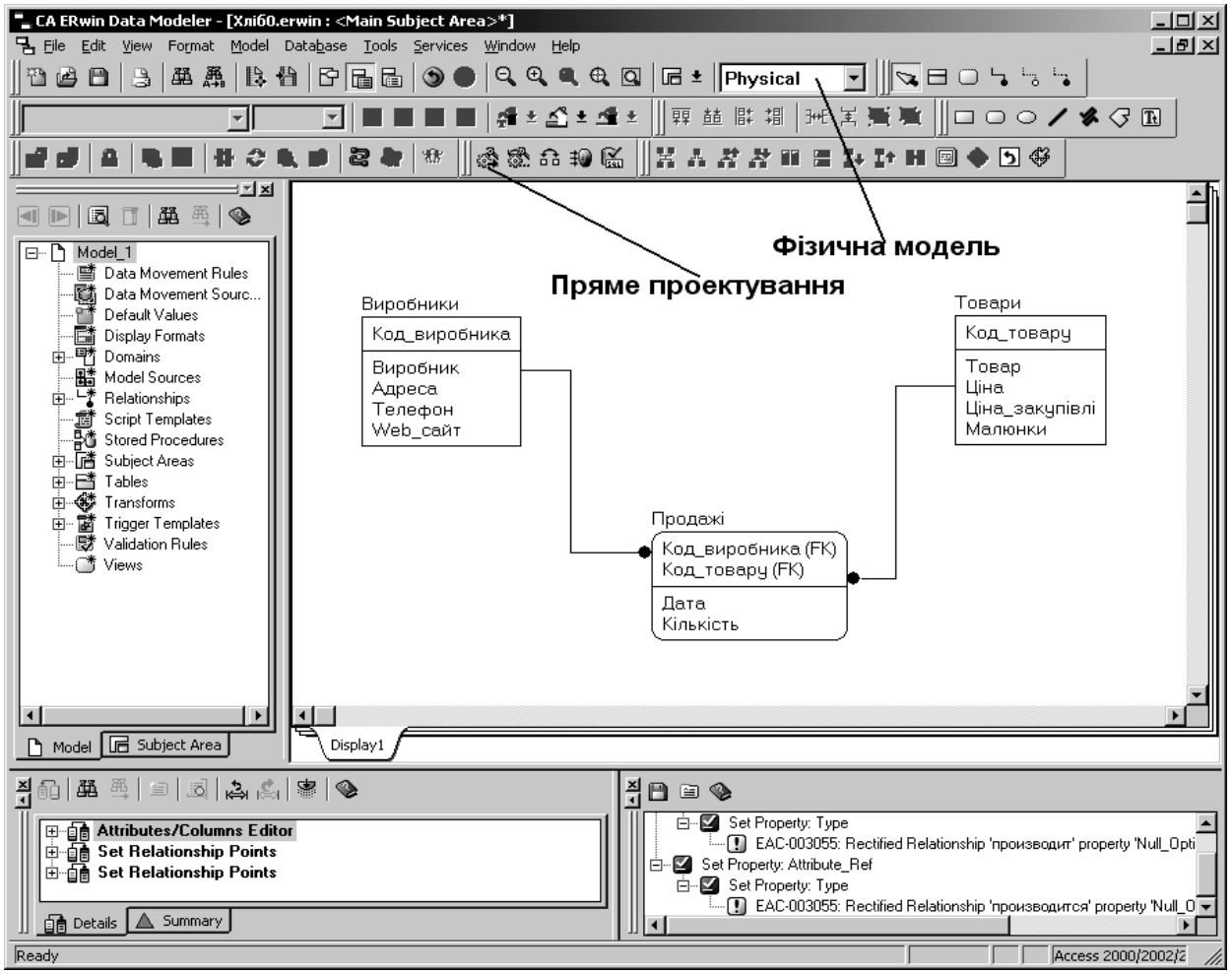


Рис. 1.24. **Вибір генерації фізичної схеми бази даних**

Ви увійдете до діалогу Schema Generation (рис. 1.25).

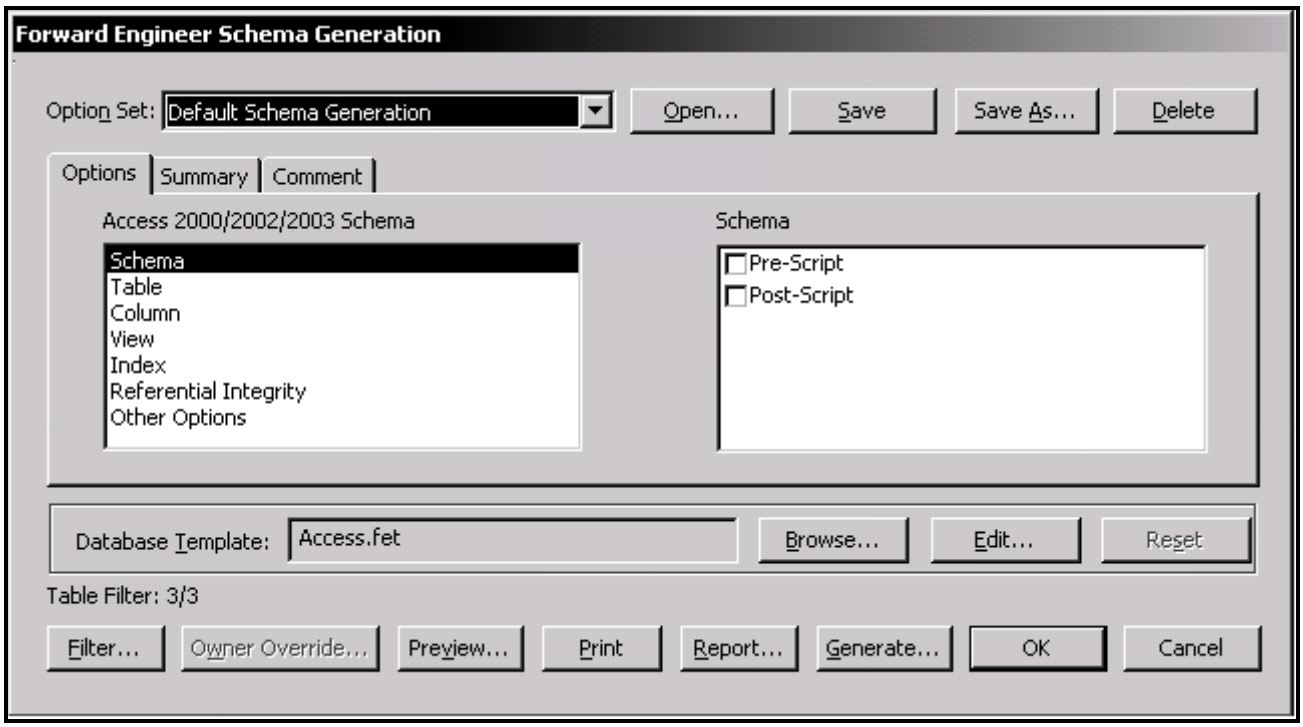


Рис. 1.25. **Вікно діалогу при прямому проектуванні бази даних**

Діалог Schema Generation має три закладки:

1. Закладка Options служить для завдання опцій генерації об'єктів БД- тригерів, таблиць, подання, колонок, індексів тощо. Для завдання опцій генерації якого-небудь об'єкту слід вибрати об'єкт у лівому списку закладки, після чого увімкнути відповідну опцію в правому списку.

2. В закладці Summary відображаються усі опції, задані в закладці Options. Список опцій в Summary можна редагувати так само, як і в Options.

3. Закладка Comment дозволяє внести коментар для кожного набору опцій.

Для генерації схеми БД треба клацнути кнопку Generate та увійти до діалогу Connection для встановлення сеансу зв'язку з сервером і запуску SQL-скрипта (рис. 1.26).

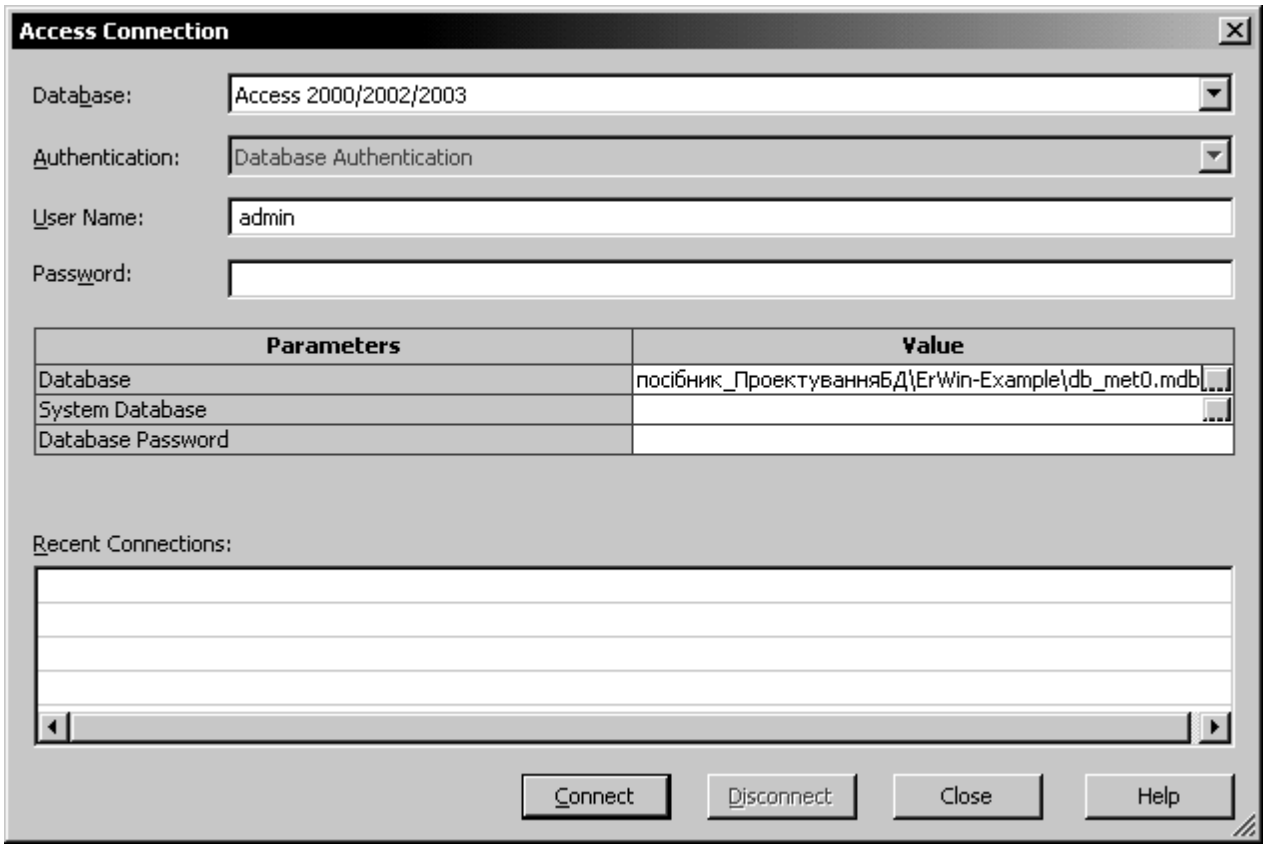


Рис. 1.26. **Вікно вибору бази даних при прямому проектуванні**

Для того, щоб згенерувати базу даних в Access треба:

1. Створити порожню базу даних в MS Access.
2. Вибрати створену базу даних на локальному диску комп'ютера в полі Database.
3. Клацнути кнопку Connect.
4. Відбувається запуск процесу генерації таблиць у вибраній БД (рис. 1.27).

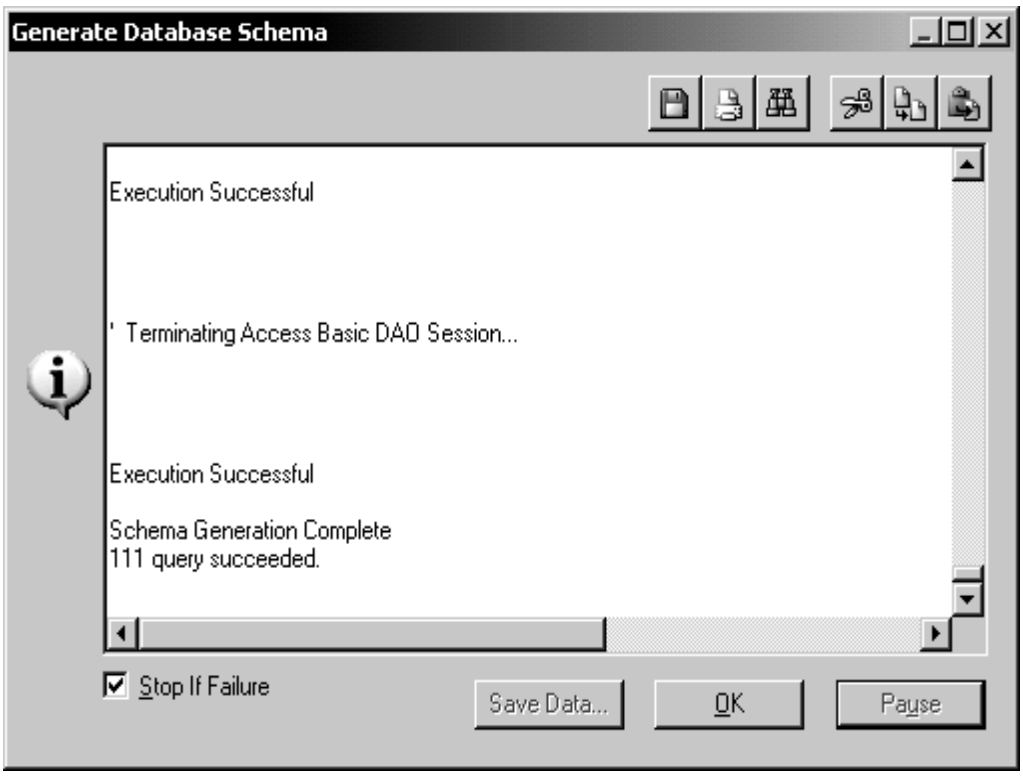


Рис. 1.27. **Ілюстрація процесу генерації схеми бази даних**

Відкривши створену базу даних в ACCESS, побачимо таку схему (рис. 1.28).

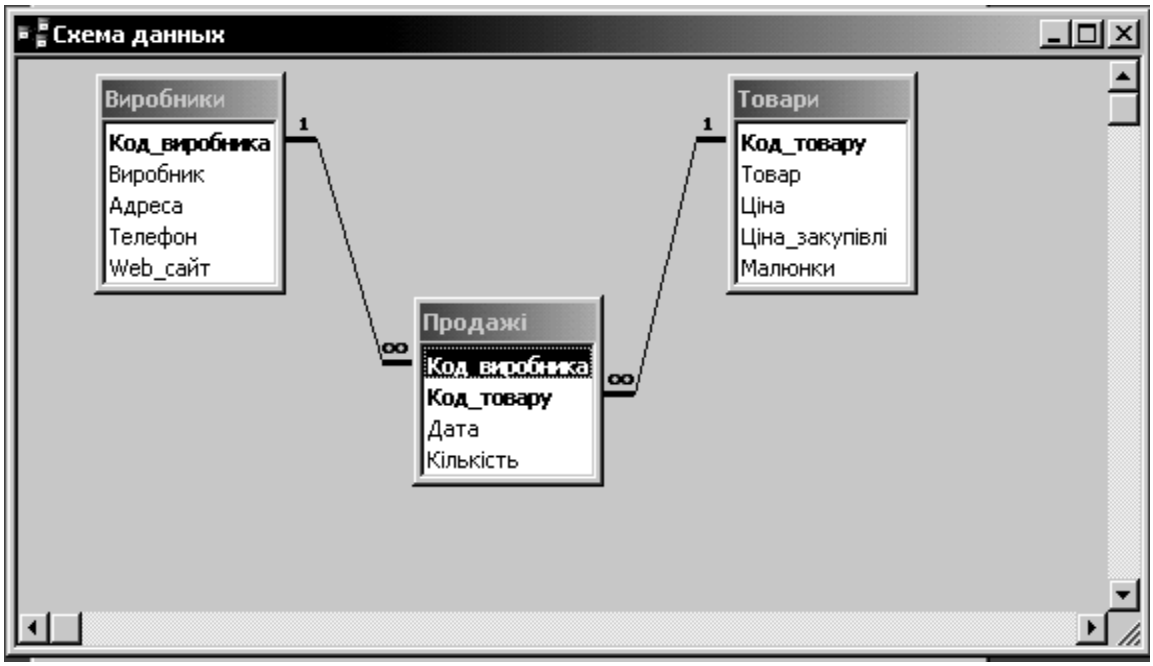


Рис. 1.28. **Схема створеної бази даних у ACCESS**

На етапі проектування ER-моделі можна було атрибути складеного первинного ключа перенести вниз (не робити первинним), а додати штучний ключ "Код продажу" (рис 1.29). В цьому випадку, згенерована

схема, повністю б відповідала тій базі, на якій виконувалися перші лабораторні роботи.

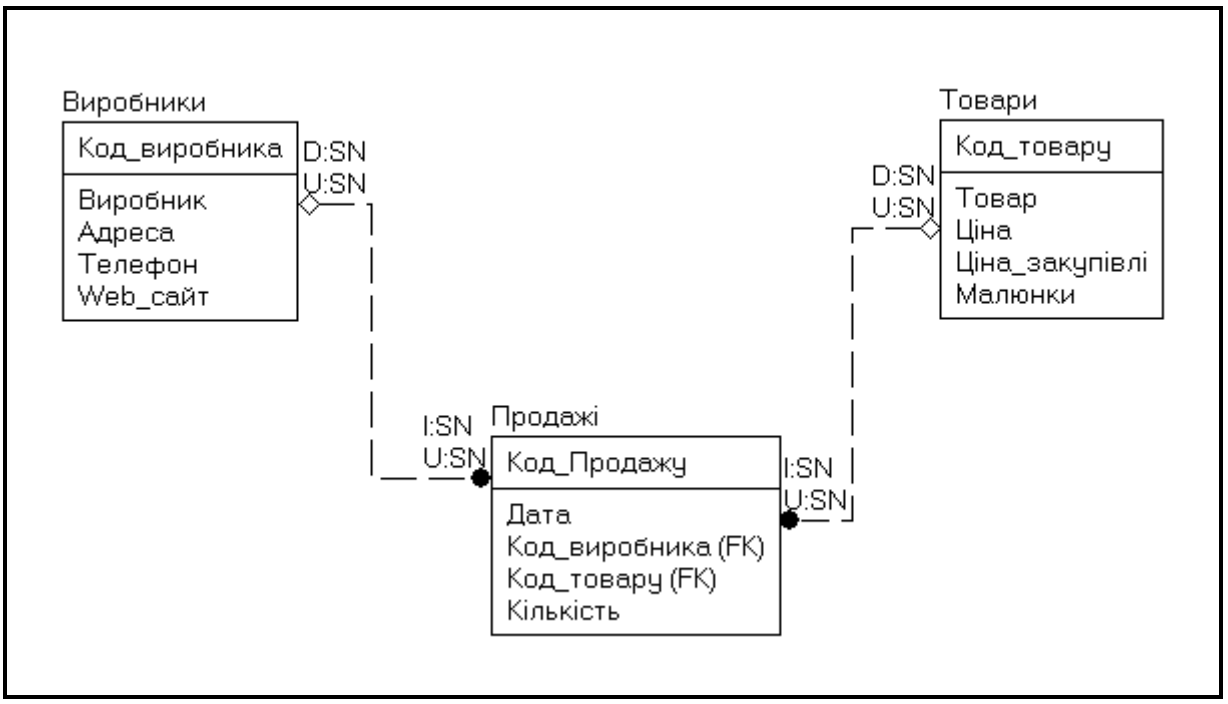


Рис. 1.29. **Фрагмент ER-діаграми зі штучним ключем**

1.4. Створення моделі предметної області "ІС компанії з продажу товарів"

Розглянемо тепер побудову ER-моделі для більш складного приклада.

Для ілюстрації роботи з Erwin Data Modeler спроектуємо ІС для компанії, що займається продажем товарів. Перед початком розробки було проведено дослідження предметної області і визначені такі основні положення відносно бізнес-процесів які будуть автоматизовуватись.

1. Номенклатура продукції, що продається, складає близько 10 000 найменувань. Товари об'єднуються в ієрархічно організований каталог з необмеженою кількістю рівнів ієрархії.

2. Замовлення на купівлю товару приймаються по телефону і обробляються менеджерами організації. Замовлення включає необмежену кількість товарів. Покупцем може виступати фізична або юридична особа.

3. Замовлення доставляються кур'єрами у разі доставки по місту і транспортними компаніями у разі доставки в інші регіони країни. Відвантаження товарів оформляється витратною накладною.

4. Увесь товар враховується на складі. Постачання товару здійснюється постачальниками і оформлюється прибутковими накладними. Окремого довідника товарів в компанії не ведеться.

5. В системі має бути передбачено ведення бази даних, яка повинна зберігати інформацію про покупців, постачальників, товари, замовлення, служби доставки.

1.4.1. Створення моделі

Розробку бази даних розпочинаємо із створення нової моделі в ERwin, для цього необхідно:

1. Запустити ERwin;
2. Вибрати Create a new model (Створити нову модель);
3. Вибрати тип моделі. Вибравши тип Physical або Logical/Physical, можна відразу визначити базу даних, в якій буде реалізована модель. Виберемо тип моделі Logical/Physical.

Сутності та атрибути

Як вже відзначалося раніше сутність служить для представлення набору реальних або абстрактних предметів (людей, місць, подій і тому подібне), які мають загальні атрибути або характеристики. Кожна сутність становить множину подібних індивідуальних об'єктів, що називаються екземплярами. Атрибут виражає певну властивість об'єкту. На фізичному рівні сутність представляється таблицею, атрибут – колонкою таблиці, екземпляр – рядком таблиці.

Для ілюстрованого прикладу виділимо такі логічні об'єкти: Замовлення, Товар, Каталог, Покупець, Служба доставки, Постачальник, Накладна. Створені об'єкти можна назвати базовими.

Для цих об'єктів створимо і опишемо відповідні сутності, аналогічно тому, як вони створювалися в попередньому прикладі. Наприклад (рис. 1.30):

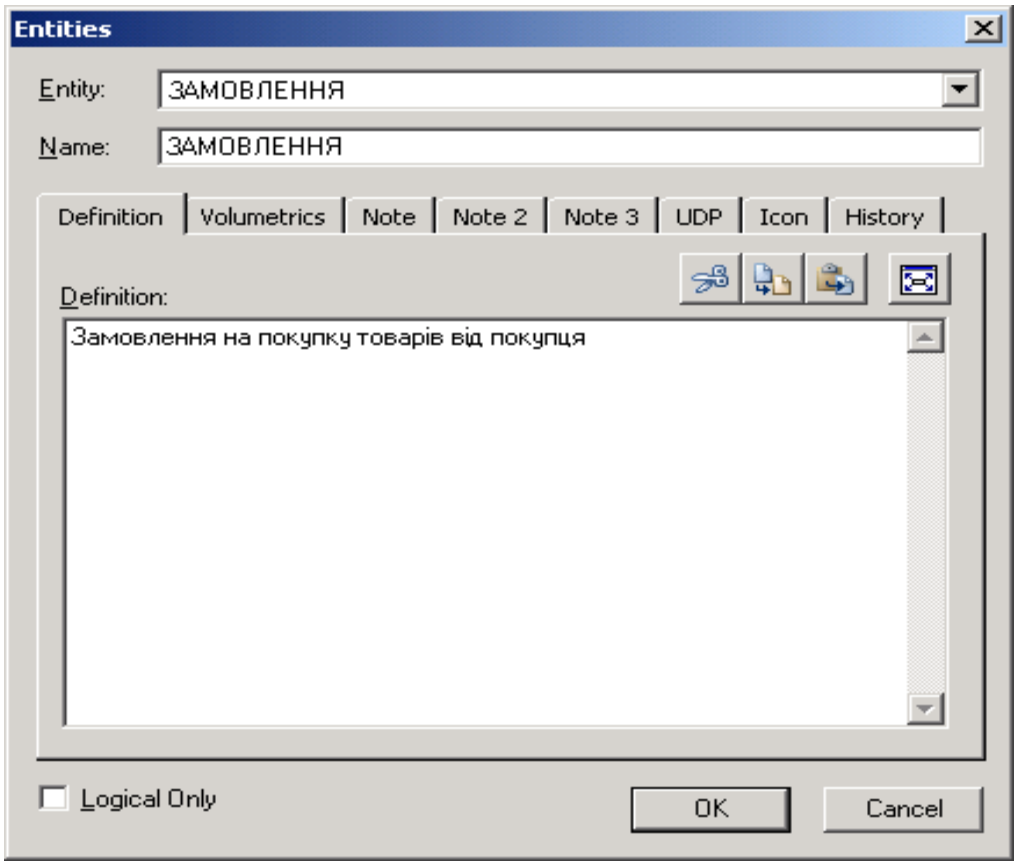


Рис. 1.30. **Створення сутності Замовлення**

Для кожної сутності визначимо перелік атрибутів і їх властивості. Наприклад, для сутності Замовлення, це виглядатиме таким чином (рис. 1.31):

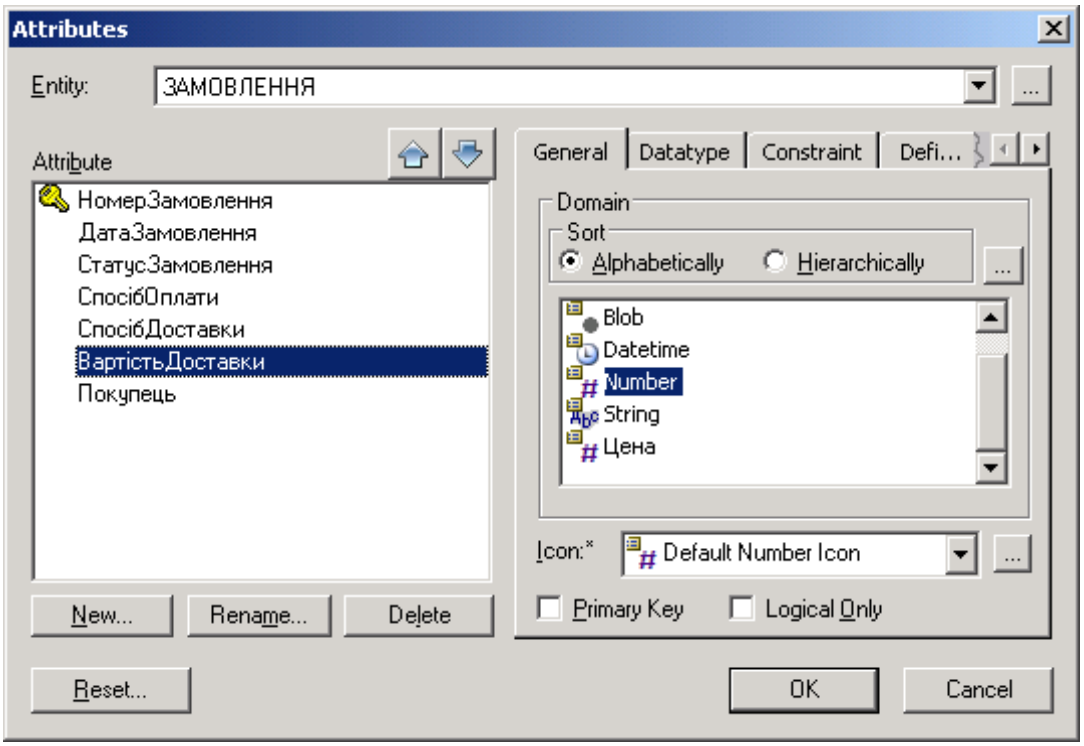


Рис. 1.31. **Атрибути сутності Замовлення**

Створимо атрибути для визначених вище сутностей і отримаємо таку схему (рис. 1.32):

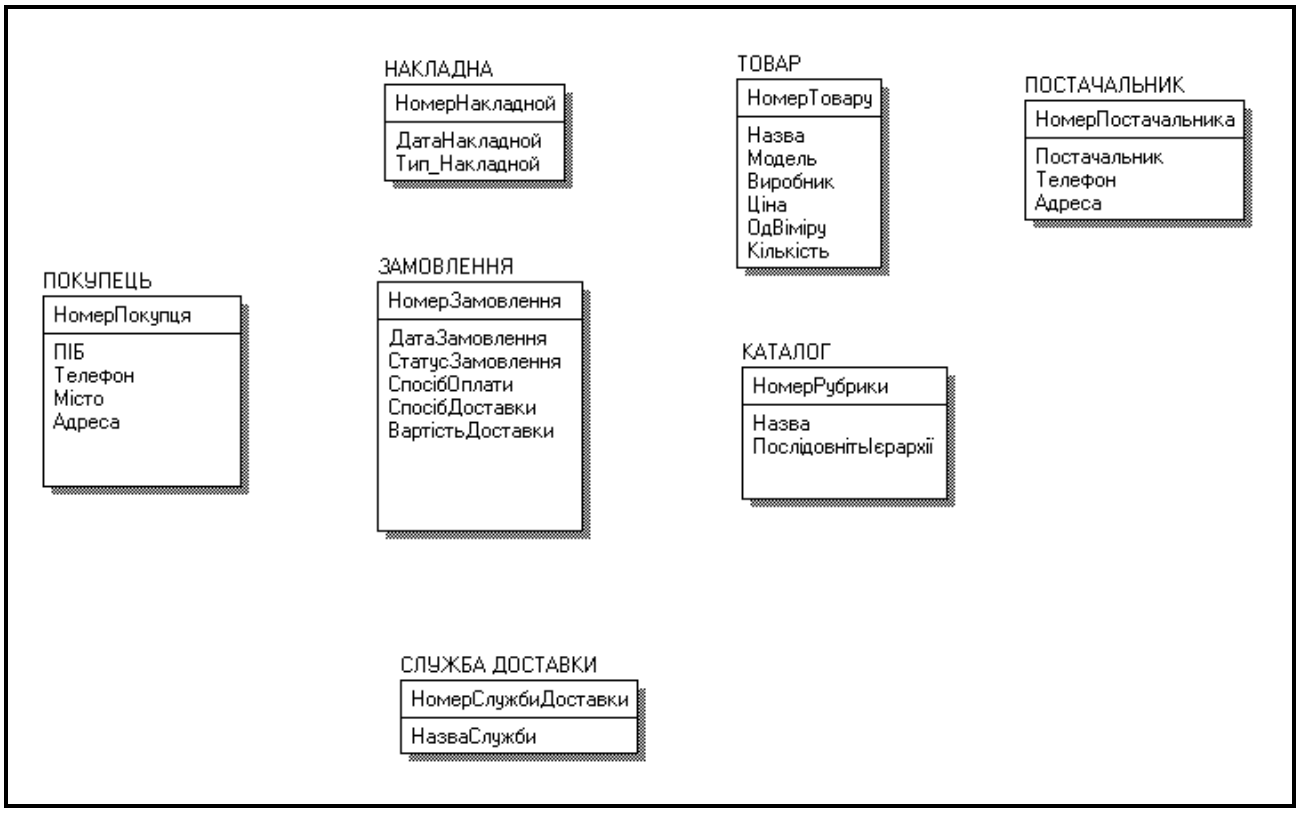


Рис. 1.32. **Діаграма сутностей без зв'язків**

Створення зв'язка

Як вже відзначалося раніше на логічному рівні можна встановити три типи зв'язків: ідентифікуючий зв'язок "один до багатьох", неідентифікуючий зв'язок "один до багатьох" та невизначений зв'язок "багато-до-багатьох".

В наведеному прикладі зв'язок "Покупець робить Замовлення" буде не ідентифікуючим (рис. 1.33).

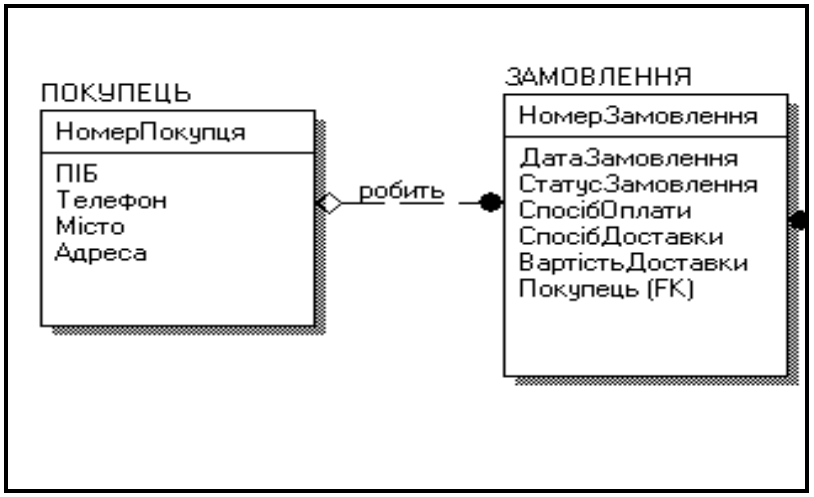


Рис. 1.33. **Вид неідентифікуючого зв'язку**

Проте, інформація про замовлення безглузда, якщо немає інформації про покупця. Тобто атрибут Номер покупця в Замовленні має бути наявним обов'язково. Така ситуація відбивається обов'язковістю зв'язку (Властивість Nulls).

Для того, щоб встановити обов'язковість зв'язку, необхідно клацнути правою кнопкою миші по зв'язку і вибрати пункт Relationship Properties в контекстному меню (рис. 1.34).

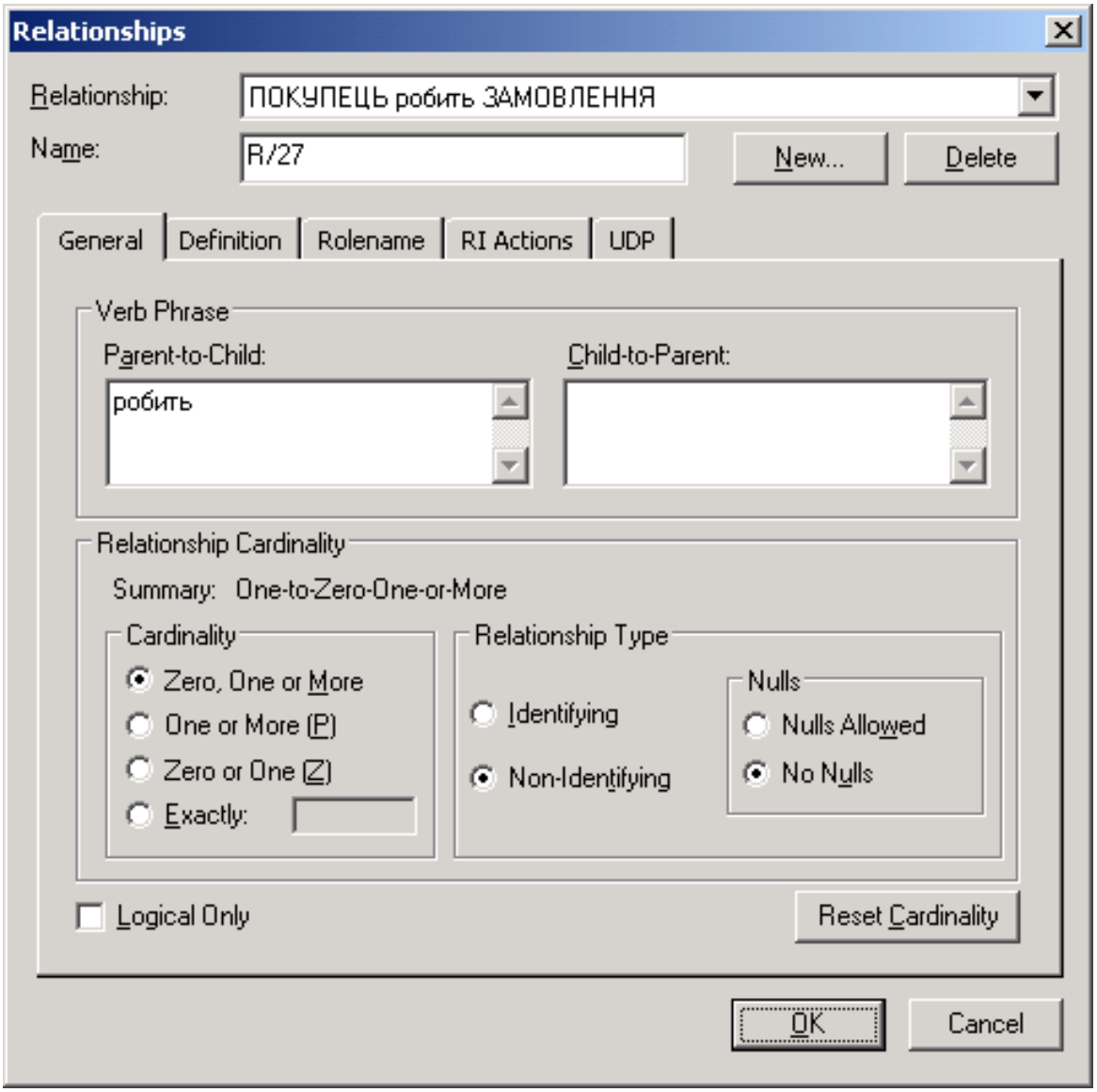


Рис. 1.34. **Встановлення обов'язковості зв'язку**

У блоці Nulls вибрати позицію No Nulls. В цьому випадку значок на початку стрілки зникне (рис. 1.35).

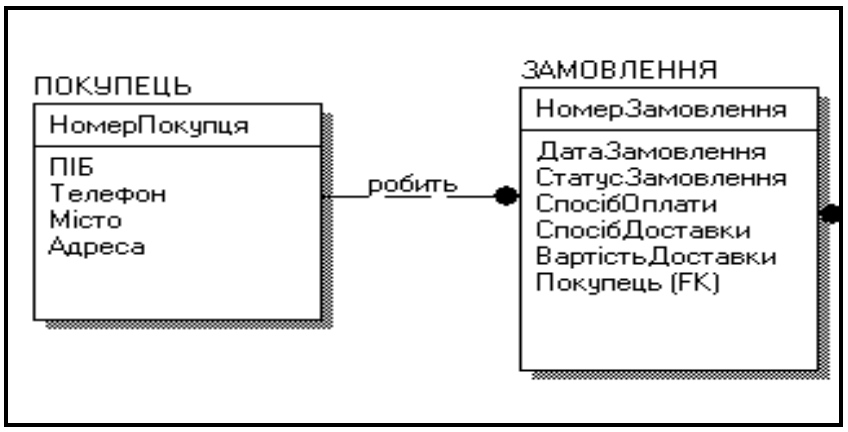


Рис. 1.35. **Відображення обов'язкового зв'язку**

Властивість обов'язковості вказується тільки для неідентифікуючих зв'язків. При ідентифікуючому зв'язку атрибути первинного ключа батьківської сутності мігрують в первинні ключі дочірньої сутності і автоматично набувають ознаки Not Null.

Зв'язок "багато-до-багатьох" може бути створений тільки на рівні логічної моделі. Прикладом зв'язку "багато-до-багатьох" може служити зв'язок між сутностями Замовлення і Товар. Одне замовлення може включати багато товарів, а один товар може входити у багато замовлень (рис. 1.36).

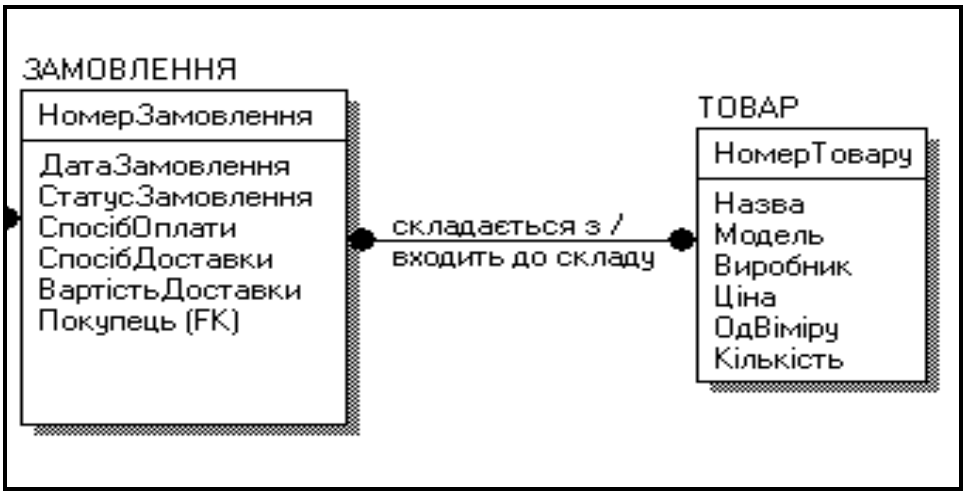


Рис. 1.36. **Позначення зв'язку на діаграмі**

Властивості зв'язку

Для редагування властивостей зв'язку слід клацнути правою кнопкою миші по зв'язку і вибрати пункт Relationship Properties в контекстному меню.

Потужність зв'язку (Cardinality) служить для позначення відношення числа екземплярів батьківської сутності до числа екземплярів дочірньої. Визначається тільки для зв'язку "один-до-багатьох". За замовчуванням на діаграмі не відображається потужність зв'язку. Для того, щоб показати потужність зв'язку, необхідно в контекстному меню діаграми вибрати пункт Relationship Display -> Cardinality.

Ім'я зв'язку – фраза, що характеризує відношення між батьківською ті дочірньою сутностями. Для зв'язку "один-до-багатьох" досить задати ім'я, що характаризує відношення від батьківської до дочірньої сутності (Parent - to - Child). Для зв'язку "багато-до-багатьох" слід вказати імена як Parent - to - Child, так і Child - to - Parent.

Повне визначення зв'язка можна задати на вкладці Definition.

Ім'я ролі (функціональне ім'я) – синонім атрибуту зовнішнього ключа, який показує, яку роль відіграє атрибут у дочірній сутності. Наприклад, ми можемо присвоїти зовнішньому ключу Номер покупця в сутності Замовлення функціональне ім'я Покупець (рис. 1.37).

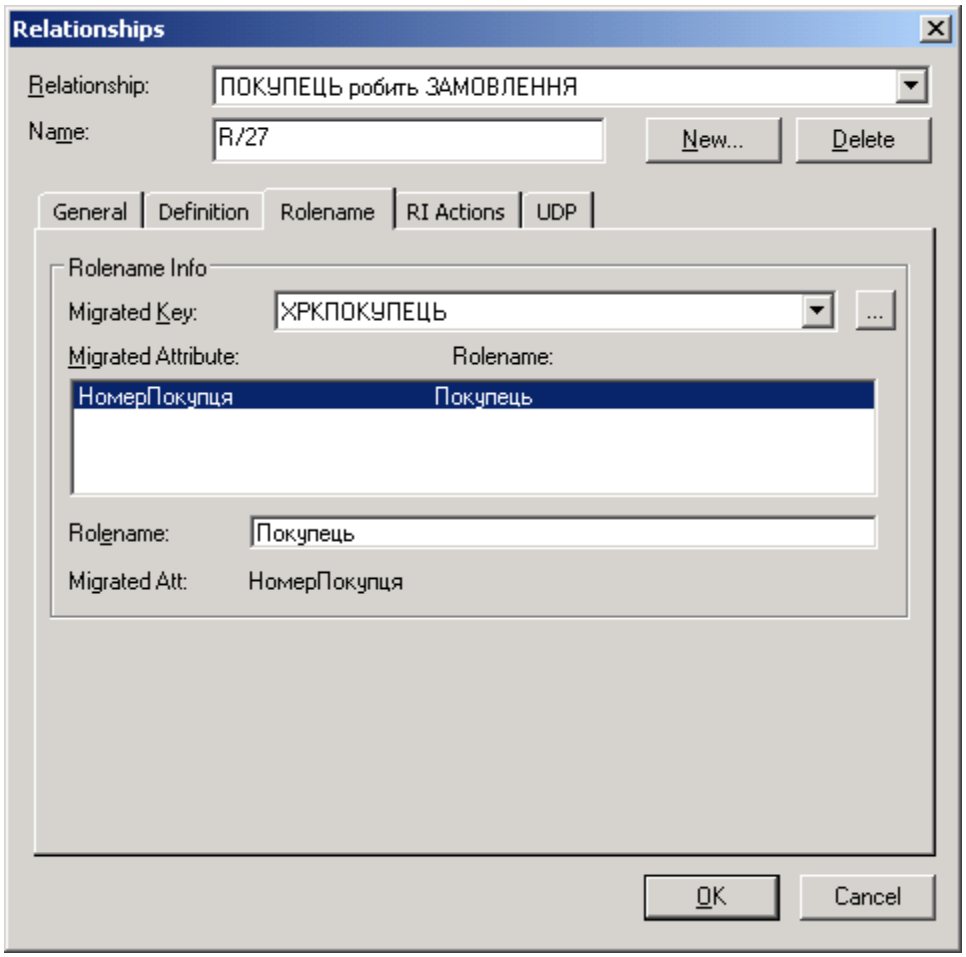


Рис. 1.37. **Вікно опису зв'язку**

За замовчуванням на діаграмі відображується тільки ім'я ролі. Щоб відобразити повне ім'я, що включає ім'я ролі і базове ім'я атрибуту, розділені точкою, необхідно в контекстному меню діаграми вибрати пункт Entity Display -> Rolename/Attribute.

Ім'я ролі необхідно вказувати обов'язково в рекурсивних зв'язках, коли одна і та ж сутність являється і батьківською і дочірньої одночасно.

Прикладом такого зв'язку може слугувати сутність Каталог. Тут рекурсивним зв'язком відображується входження рубрики каталогу в іншу рубрику цього ж каталогу. Зовнішній ключ, номером рубрики, що являється, грає тут роль підрубрики (рис. 1.38, рис. 1.39).

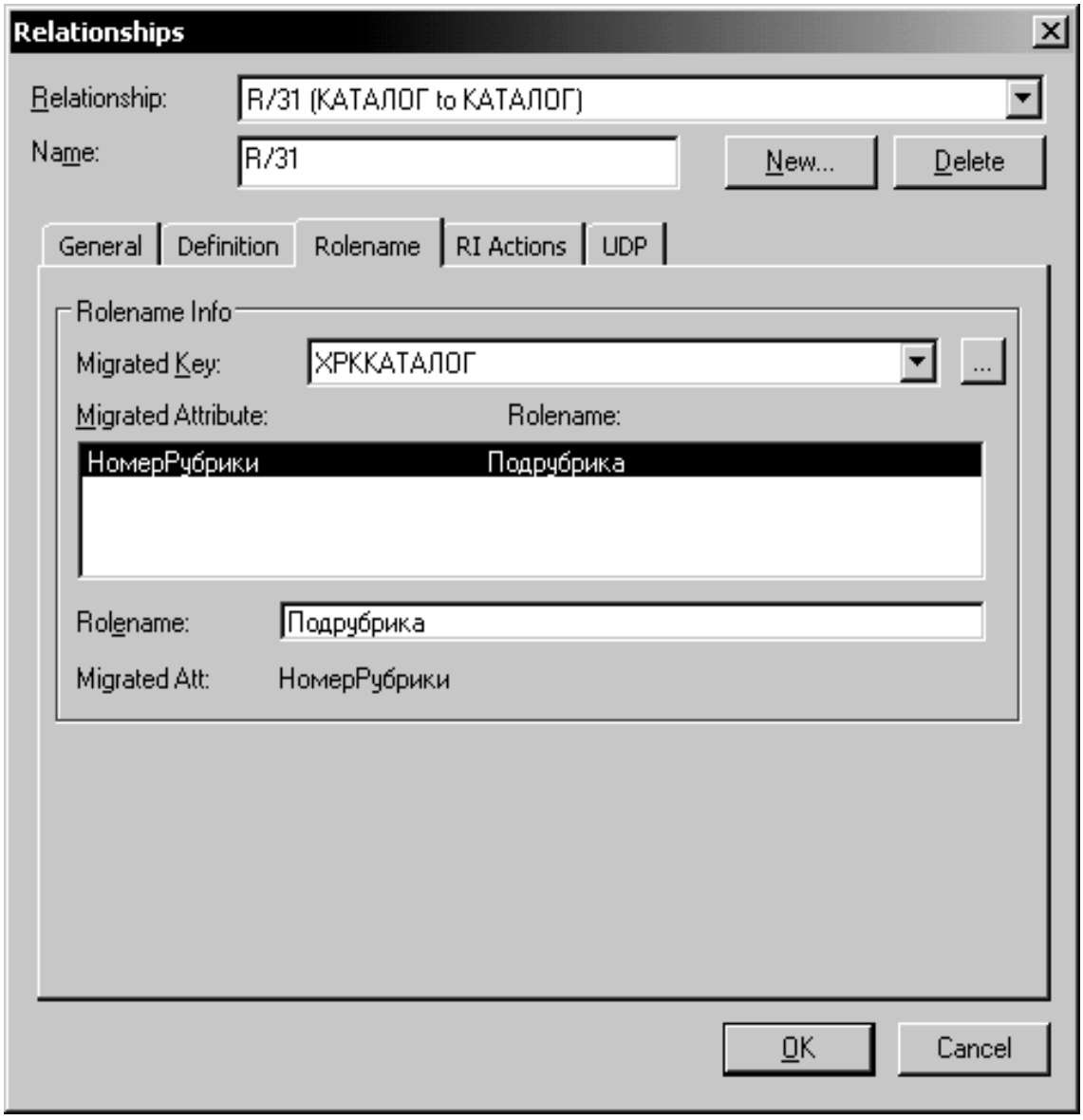


Рис. 1.38. **Створення рекурсивного зв'язку**

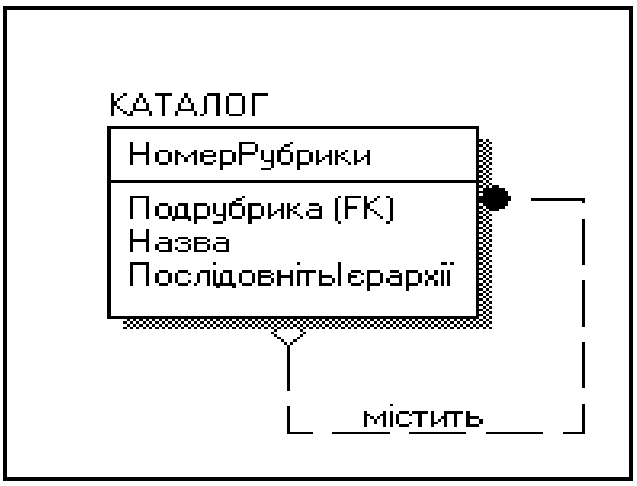


Рис. 1.39. **Вид рекурсивного зв'язка на діаграмі**

Ім'я ролі необхідно обов'язково вказувати також у тому випадку, коли встановлюються зв'язки від одного атрибута батьківської сутності до двох різних атрибутів дочірньої. Наприклад, існують дві сутності Підприємства (довідник) та Договори. У сутності Договори є два атрибути, які зв'язані з довідником Підприємтва і вказують – між якими підприємствами був укладений договір на виконання робіт.

1.4.2. Ключі

Первинний ключ

Первинний ключ (Primary key) – це атрибут або група атрибутів, що однозначно ідентифікують екземпляр сутності. Атрибути первинного ключа на діаграмі знаходяться в списку атрибутів сутності вище за горизонтальну лінію.

Створення первинного ключа

Для того, щоб зробити атрибут первинним ключем, треба в редакторові атрибутів сутності відмітити галочку Primary Key в нижній частині закладки General. На діаграмі атрибут можна внести до складу первинного ключа, скориставшись режимом перенесення атрибутів:

1. Виберіть необхідну сутність.

2. Наведіть курсор на атрибут, який треба внести до складу ключових. Курсор набере вигляду  .

3. Клацніть лівою кнопкою миші і, утримуючи її, перенесіть в списку атрибутів сутності вище за горизонтальну лінію.

Так само можна вимикати атрибути із складу ключових, а також переносити атрибути з однієї сутності в іншу.

У одній сутності може опинитися декілька атрибутів або наборів атрибутів, що претендують на роль первинного ключа. Такі претенденти називаються потенційними ключами (Candidate key).

При виборі первинного ключа перевага повинна віддаватися простішим ключам, тобто ключам, що містять меншу кількість атрибутів.

Багато сутностей мають тільки один потенційний ключ. Такий ключ стає первинним. Деякі сутності можуть мати більше за один можливий ключ. Тоді один з них стає первинним, а інші – альтернативними ключами.

Наприклад, в сутності Товар первинним ключем може стати Номер товару або Модель товару, оскільки обидва атрибути є унікальними для цього об'єкту і однозначно ідентифікують об'єкт сутності. Але для первинного ключа ми вибрали атрибут Номер товару, оскільки він більш простий: це тільки число, тоді як модель – це складна комбінація букв і цифр. Крім того, теоретично модель товару може змінитися, а номер товару не зміниться ніколи.

Альтернативний ключ

Альтернативний ключ (Alternative Key) – це потенційний ключ, що не став первинним.

Кожному ключу відповідає індекс, ім'я якого також привласнюється автоматично. Імена ключа і індексу за бажанням можна змінити вручну.

На діаграмі атрибути альтернативних ключів позначаються як (Akn.m.), де n – порядковий номер ключа, m – порядковий номер атрибуту в ключі.

Створення альтернативного ключа

У нашому прикладі альтернативним ключем стане атрибут Модель у сутності Товар.

Для того, щоб створити альтернативний ключ:

1. Клацніть правою кнопкою миші по сутності і виберіть в контекстному меню пункт Key Group. Ви увійдете в редактор Key Groups.
2. Клацніть кнопку New і при необхідності змініть ім'я створюваного ключа. Клацніть кнопку OK.

3. Виберіть в списку Available Attributes необхідний атрибут (у нашому випадку Модель) і клацніть кнопку  . Таким чином атрибут переміститься в список Key Group Members (рис. 1.40).

4. Клацніть кнопку OK.

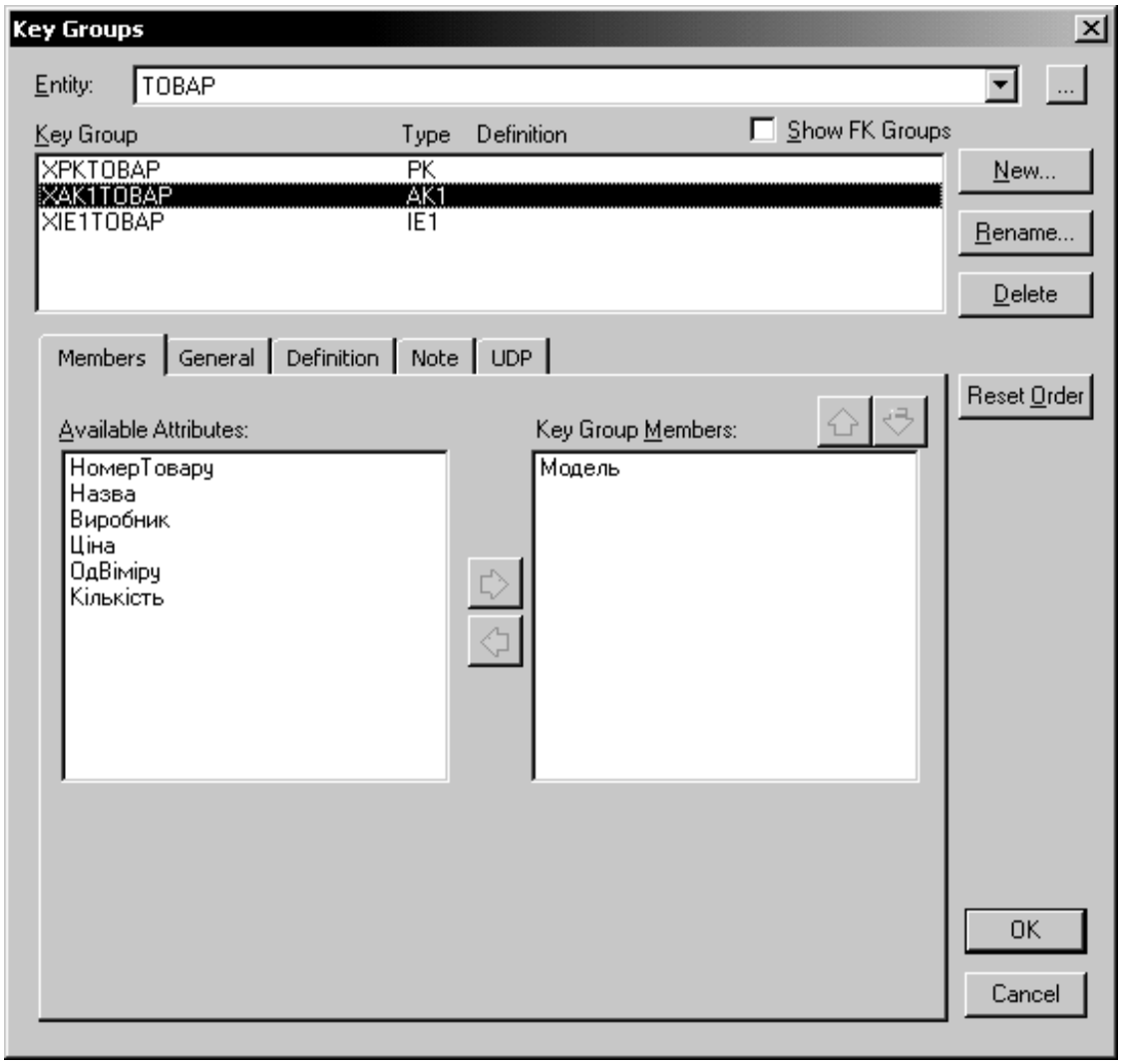


Рис. 1.40. **Приклад створення альтернативного ключа**

Інверсійний вхід

При роботі ІС часто буває необхідно забезпечити доступ до декількох екземплярів сутності, об'єднаним якою-небудь однією ознакою. Для підвищення продуктивності в цьому випадку використовуються не унікальні індекси. ERwin дозволяє на рівні логічної моделі призначити атрибути, які братимуть участь в не унікальних індексах. Атрибути, що беруть участь в не унікальних індексах, називаються інверсійними входами (Inversion Entries).

Інверсійний вхід – це атрибут або група атрибутів, які не визначають екземпляр сутності унікальним чином, але часто використовуються для звернення до екземплярів сутності. ERwin генерує неунікальний індекс для кожного інверсійного входу. Часто інверсійний вхід називають вторинним ключем.

Створення інверсійного входу

Включимо атрибути, які часто використовуватимуться для пошуку записів в таблицях БД, до складу інверсійних входів.

У нашому прикладі інверсійними входами будуть:

a) атрибут ПІБ в сутності Покупець;

b) атрибути Дата замовлення і Статус замовлення в сутності Замовлення;

c) атрибут Виробник в сутності Товар.

Для того, щоб створити інверсійний вхід:

1. Клацніть правою кнопкою миші по сутності і виберіть в контекстному меню пункт Key Group. Ви увійдете в редактор Key Groups (рис. 1.41).

2. Клацніть кнопку New і виберіть прапорець Inversion Entry (non - unique) у блоці Key Group Type в нижній частині вікна New Key Group.

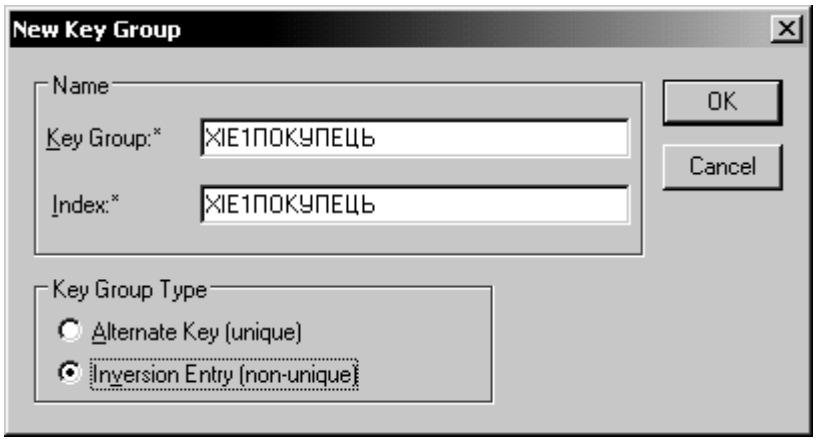


Рис. 1.41. **Вікно редактора Key Groups**

3. При необхідності змініть ім'я створюваного ключа і клацніть кнопку OK.

4. Виберіть в списку Available Attributes необхідний атрибут, клацніть кнопку  . Таким чином атрибут переміститься в список Key Group Members (рис.1.42).

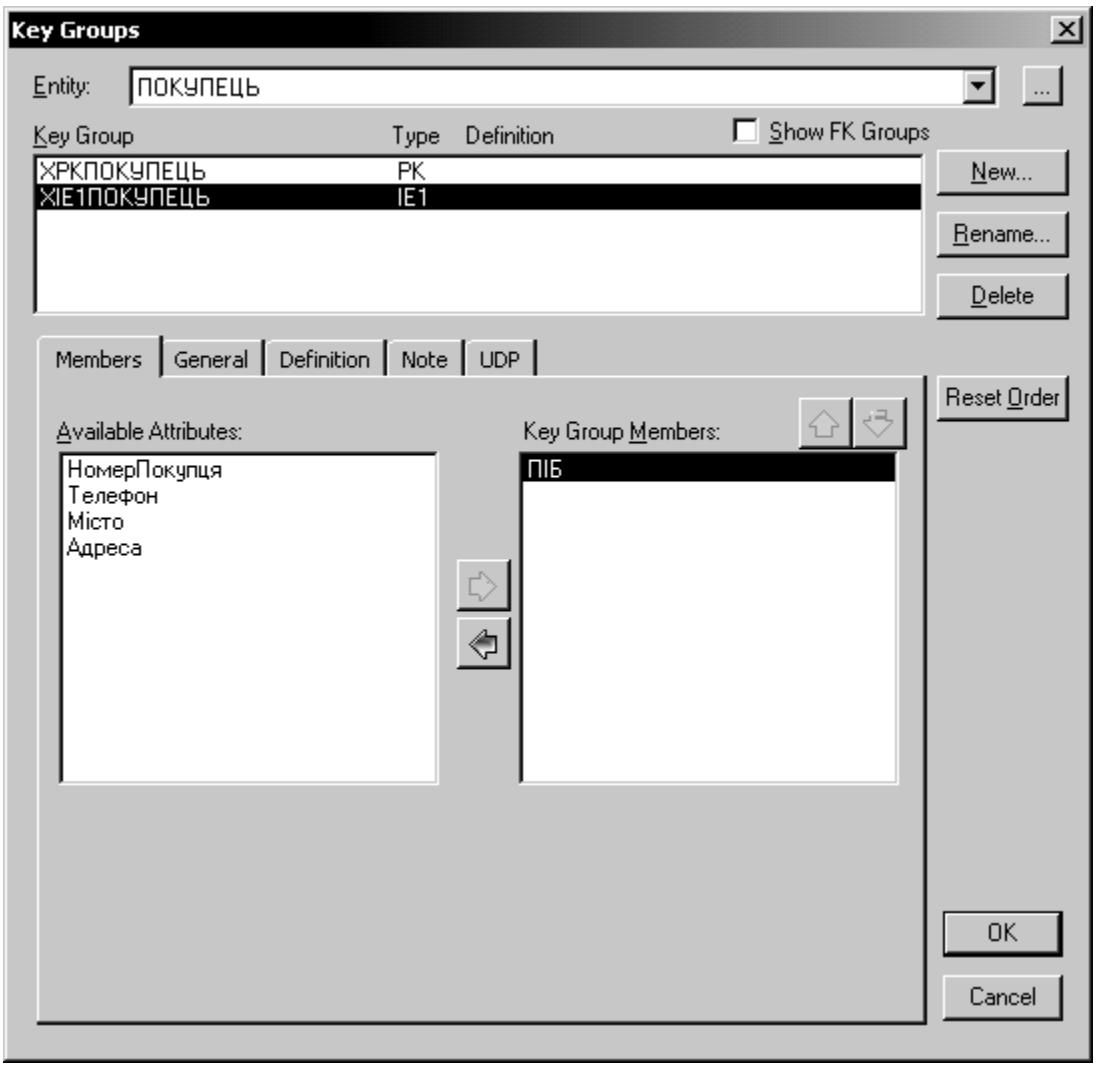


Рис. 1.42. **Вибір атрибута для створення ключа**

5. Клацніть кнопку OK.

1.4.3. Типи сутностей та ієрархія наслідування

Як було вказано вище, зв'язок визначає, чи є сутність незалежною або залежною.

Розрізняють декілька типів залежних сутностей – характеристична, асоціативна, іменуюча, категоріальна і ієрархія наслідування.

У даному прикладі в організації ведеться облік прибуткових та витратних накладних. Очевидно, що накладні мають ряд схожих характеристик, але пов'язані з різними сутностями. Витратні накладні

пов'язані із замовленнями, а прибуткові – з товаром, що приймається від постачальника.

Із загальних властивостей накладної можна сформувати узагальнену сутність (родовий предок) Накладна, щоб представити інформацію, загальну для усіх типів службовців. Специфічна для кожного типу інформація може бути розташована в категоріальних сутностях (нащадках) Витратні накладні і Прибуткові накладні.

Для кожної категорії можна вказати дискримінатор – атрибут родового предка, який показує, як відрізнити одну категоріальну сутність від іншої (атрибут Тип) (рис. 1.43).

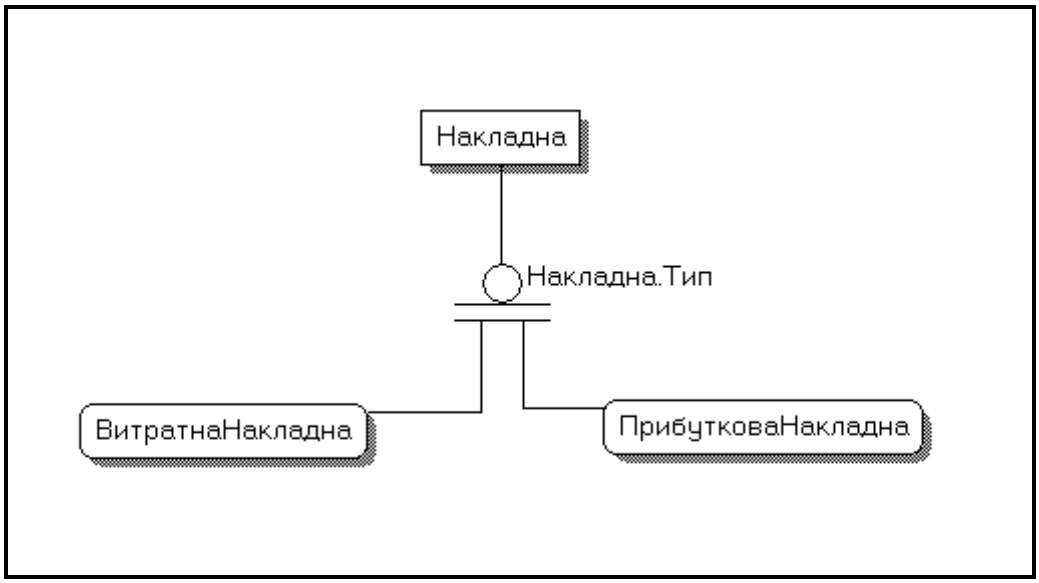


Рис. 1.43. **Відображення категоріального зв'язку**

Ієрархії категорій діляться на два типи – повні і неповні. У повній категорії одному екземпляру родового предка обов'язково відповідає екземпляр в якому-небудь нащадку, тобто накладна завжди є або прибутковою, або витратною.

Якщо категорія ще не побудована повністю і в родовому предку можуть існувати екземпляри, які не мають відповідних екземплярів в нащадках, то така категорія буде неповною.

Для створення категоріального зв'язку необхідно:

1. Створити сутність – родовий предок (Накладна).

2. Створити сутності – нащадки (Витратна накладна і Прибуткова накладна).

3. Клацнути кнопку на панелі інструментів.

4. Клацнути спочатку по родовому предкові, а потім по нащадкові.

5. Для встановлення другого зв'язку в ієрархії категорії слід спочатку клацнути по символу категорії, потім по другому нащадкові (рис. 1.44).

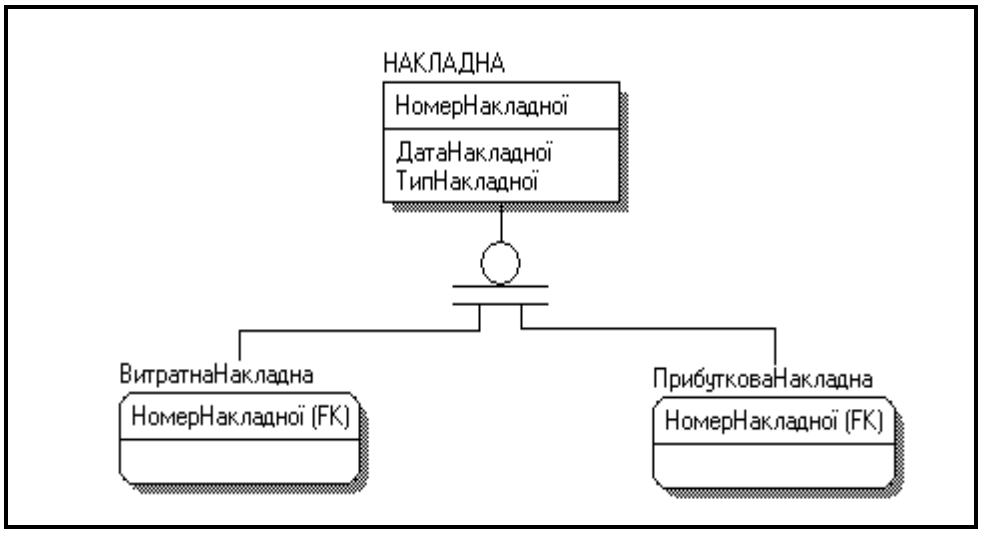


Рис. 1.44. **Відображення категоріального зв'язку з атрибутами**

Для редагування категорій треба клацнути правою кнопкою миші по символу категорії і вибрати в контекстному меню пункт Subtype Properties (рис. 1.45).

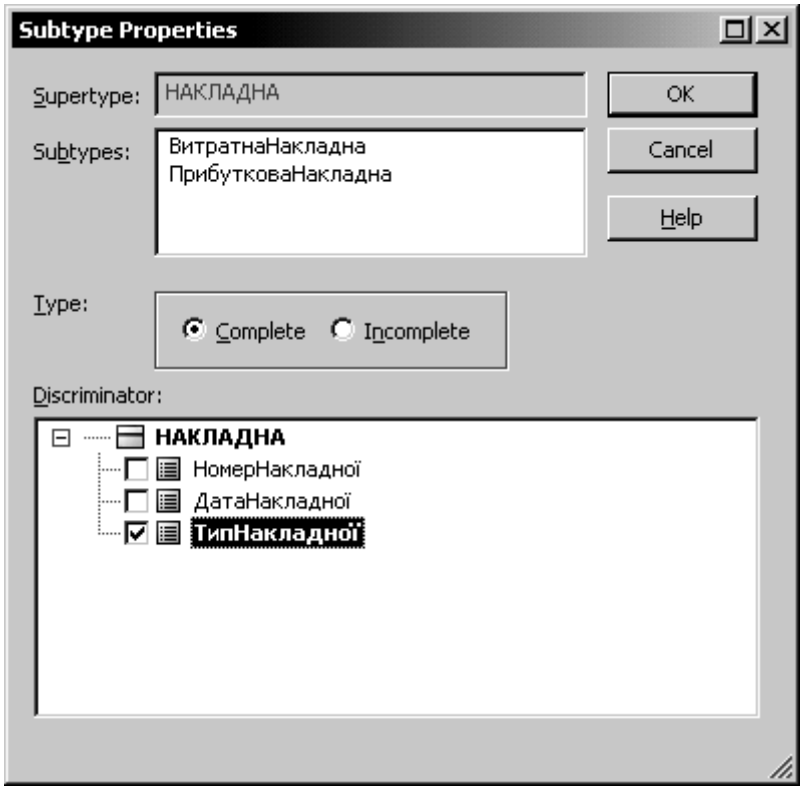


Рис. 1.45. **Вікно редагування категорій**

У діалозі Subtype Properties можна вказати атрибут – дискримінатор категорії (дерево Discriminator) і тип категорії – повна/неповна (прапорці вибору Complete/Incomplete).

Для кожного предка можна вказати атрибути, властиві цьому типу сутності. Наприклад, для прибуткової накладної необхідно вказати, від якого постачальника був прийнятий товар.

Таким чином, діаграма набере такого вигляду (рис. 1.46):

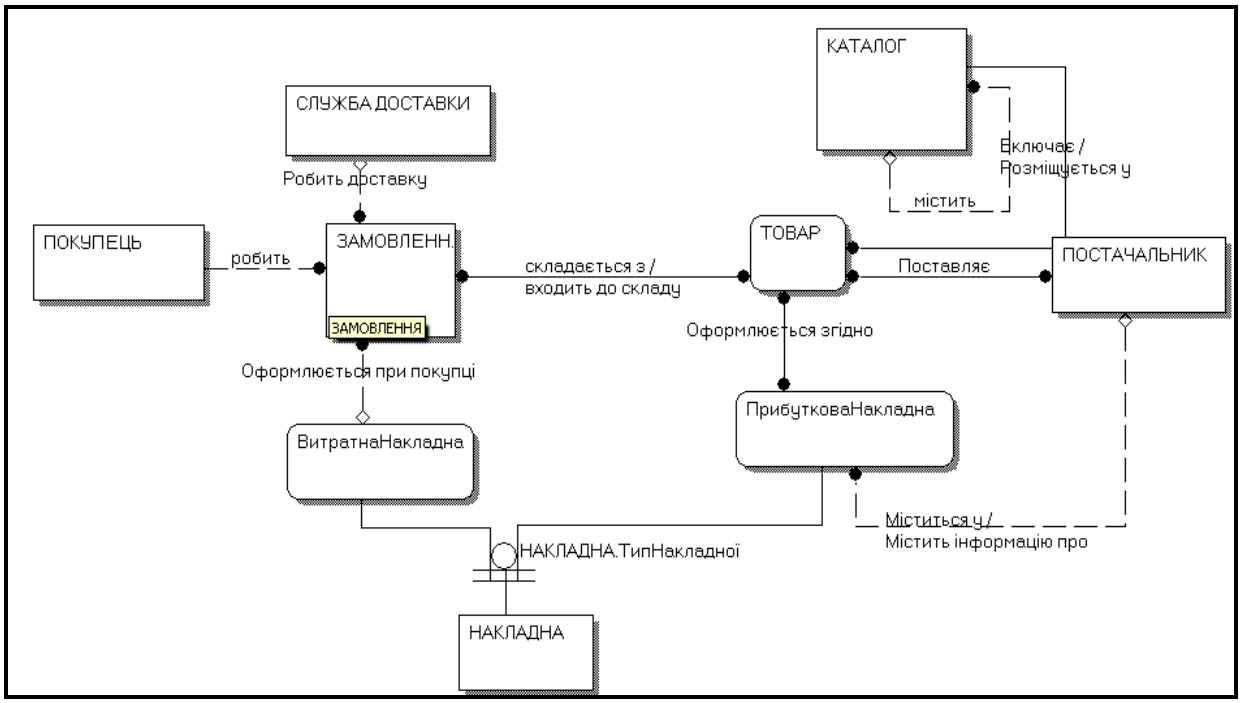


Рис. 1.46. **Вид створеної діаграми**

1.4.4. Рівні демонстрації зображення в ERwin

ERwin дозволяє переглядати інформацію про модель на різних рівнях і пропонує декілька режимів перегляду зображення. Ці рівні і режими бувають корисні при демонстрації різних типів інформації на різних стадіях побудови моделі і при передачі інформації про модель різним аудиторіям. Ці рівні задаються в меню Display Level. Ви можете відкрити меню Display Level, клацнувши правою кнопкою миші на будь-якому місці на фоні вікна діаграми, або вибравши меню Format -> Display Level.

У логічній моделі існують такі рівні демонстрації :

1. Рівень демонстрації сутності (Entity). На цьому рівні ім'я кожної сутності виводиться на екран, поміщене у вікно сутності. Інша інформація про сутності на екрані не з'являється. В даний момент, коли ми тільки починаємо планувати модель, ми використовуватимемо саме цей рівень. Також це представлення корисне, коли вам потрібна картина, що дає поняття про увесь бізнес-процес у цілому (рис. 1.46)

2. Рівень демонстрації визначень сутності (Definition). Корисний рівень при презентації моделі, оскільки на цьому рівні розмір вікна сутності збільшується і в нього включається визначення сутності (рис. 1.47).

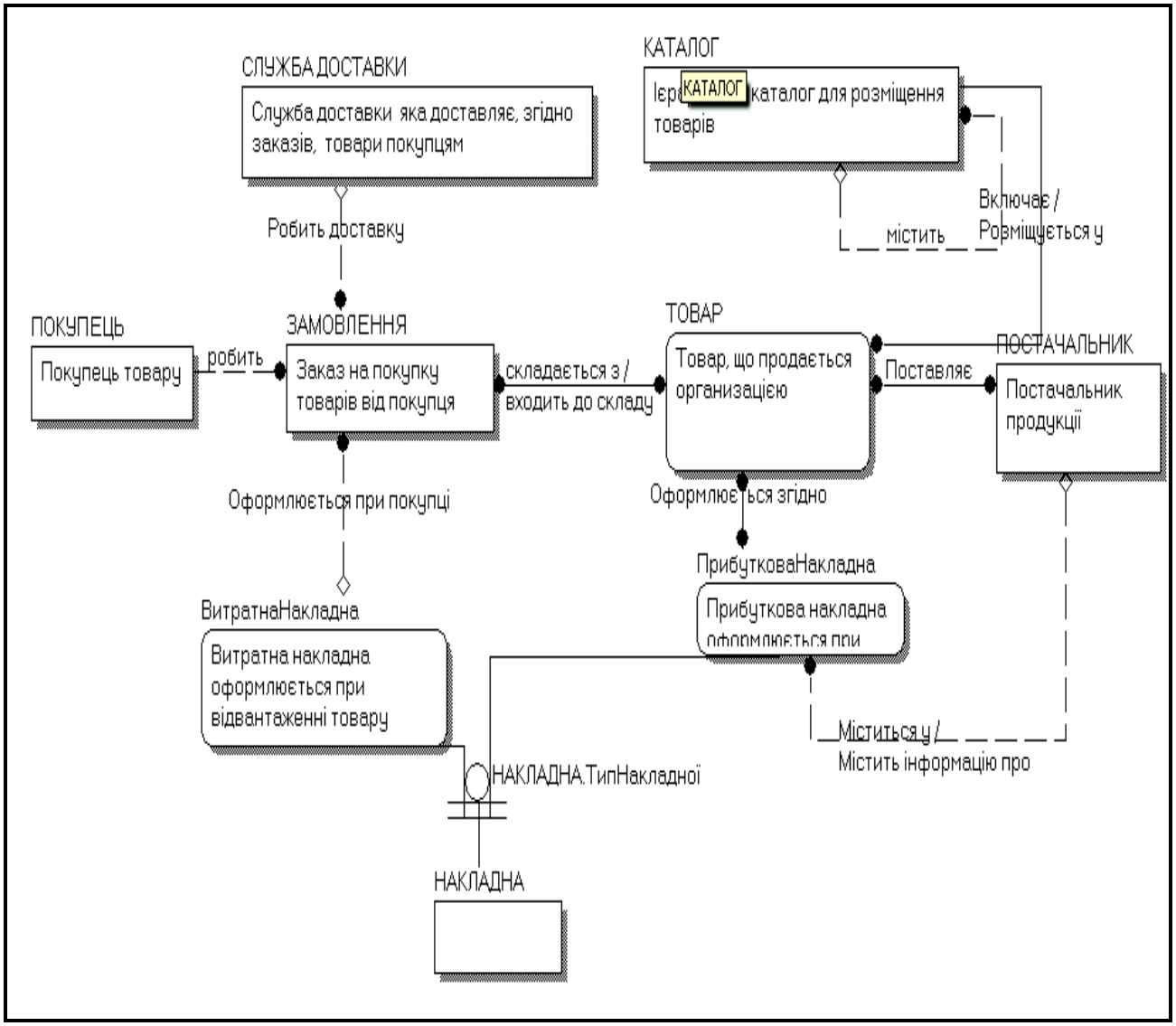


Рис. 1.47. **Рівень демонстрації визначень сутності**

3. Рівень демонстрації атрибутів сутностей. На рівні демонстрації атрибутів кожна сутність в діаграмі відкривається, показуються її

атрибути, причому атрибути, які є первинними ключами, розташовані над рисою, а неключові атрибути – під рисою у вікні сутності (рис. 1.48).

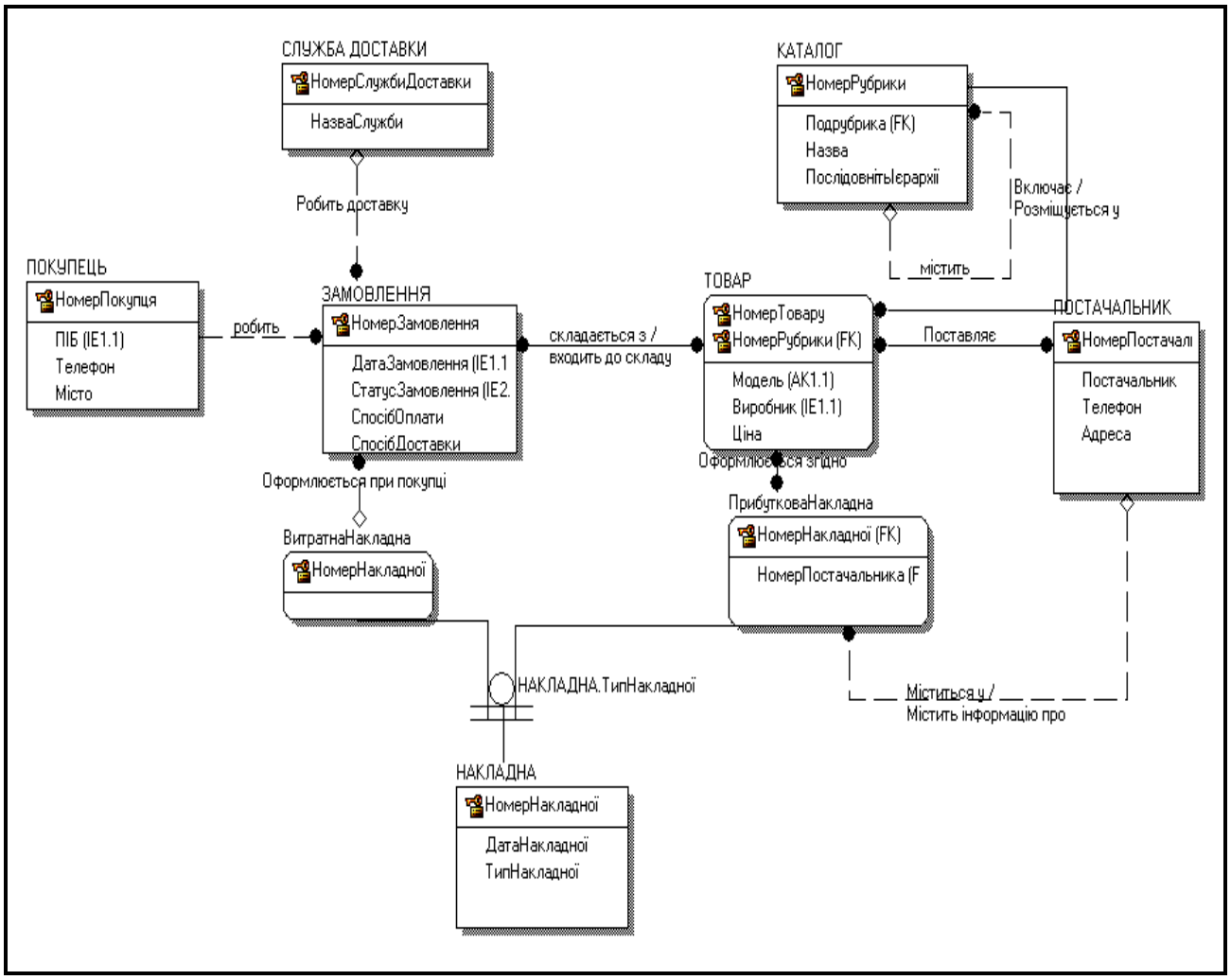


Рис. 1.48. **Рівень демонстрації атрибутів сутностей**

4. Рівень демонстрації первинних ключів (Primary Key). Різновидом рівня демонстрації атрибутів є рівень первинного ключа. На цьому рівні демонструються тільки ті атрибути, які є первинними ключами.

5. Рівень піктограми (Icon). Для презентації можна використати можливість, наявну в ERwin: розташування зображення типу bitmap у вікні сутності. Цей рівень зображення можна використати для того, щоб пояснити наочно, який тип бізнес-організації представляє модель даних. Для піктограми сутності повинен використовуватися файл Windows bitmap (розширення .BMP). За бажання можна для кожної сутності задати свій значок bitmap (вкладка Icon у властивостях сутності).

Для того, щоб помістити піктограми у вікно сутності слід клацнути правою кнопкою мишки за вільним місцем у вікні моделі і вибрати пункт Display Level -> Icon (рис. 1.49)

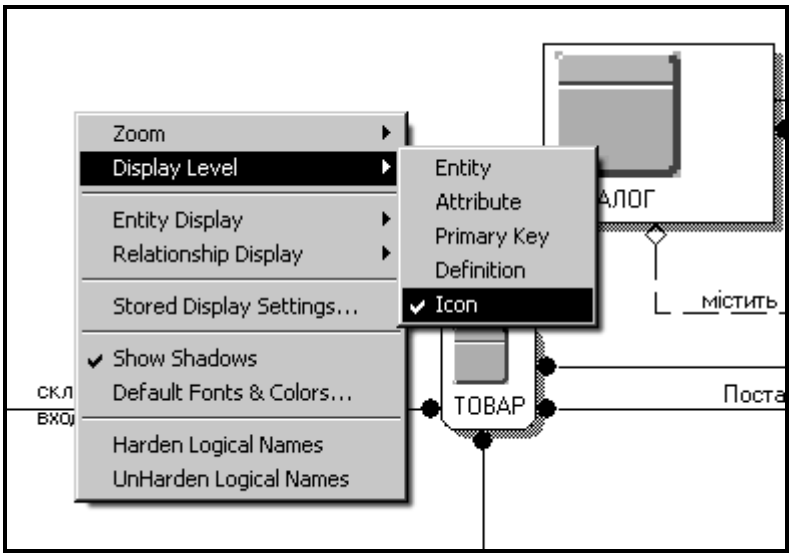


Рис. 1.49. **Порядок вибору піктограм для сутностей**

Далі клацаємо правою кнопкою миші по бажаній сутності, вибираємо пункт Entity Properties і у вікні, що відкрилося, переходимо на вкладку Icon (рис. 1.50).

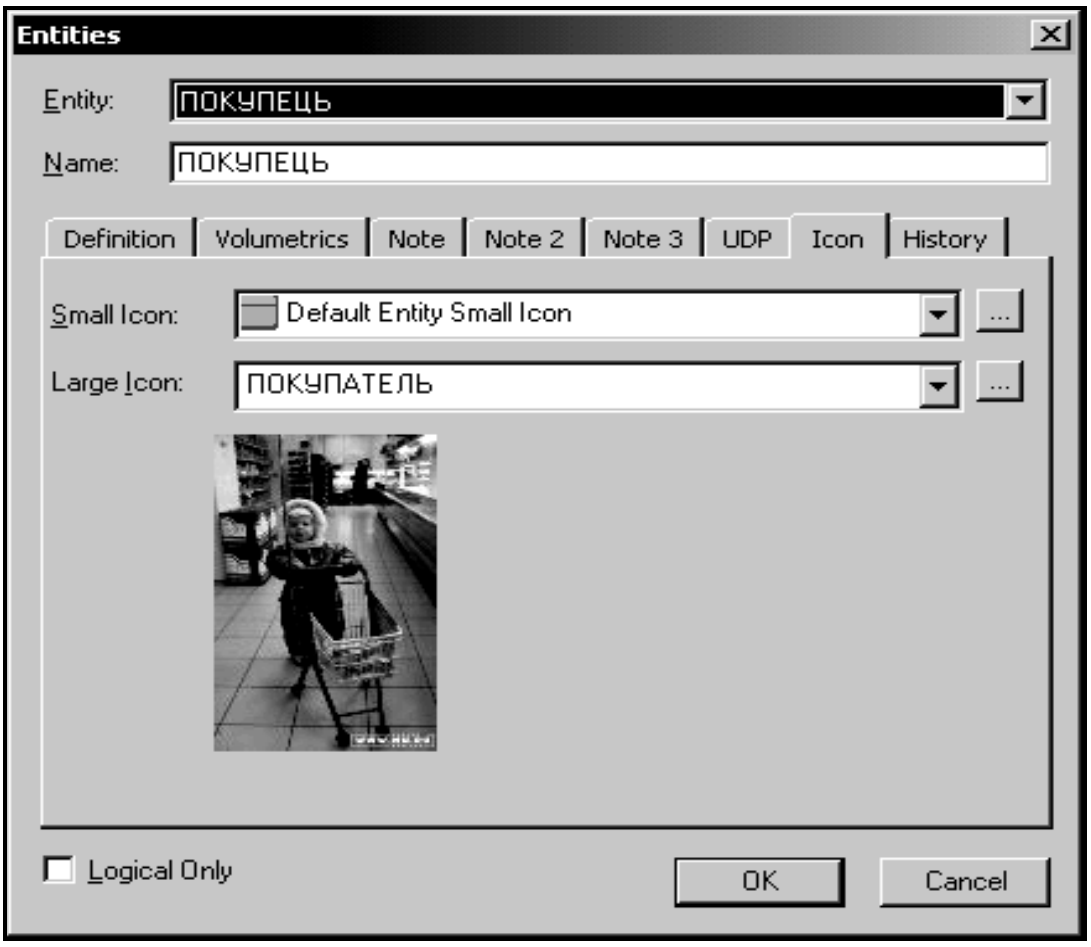


Рис. 1.50. **Вікно з вибраною піктограмою (малюнком)**

і далі вибираємо бажане зображення у вигляді bmp-файла як піктограму до сутності.

В результаті виконаних дій – отримаємо таку діаграму (рис. 1.51):

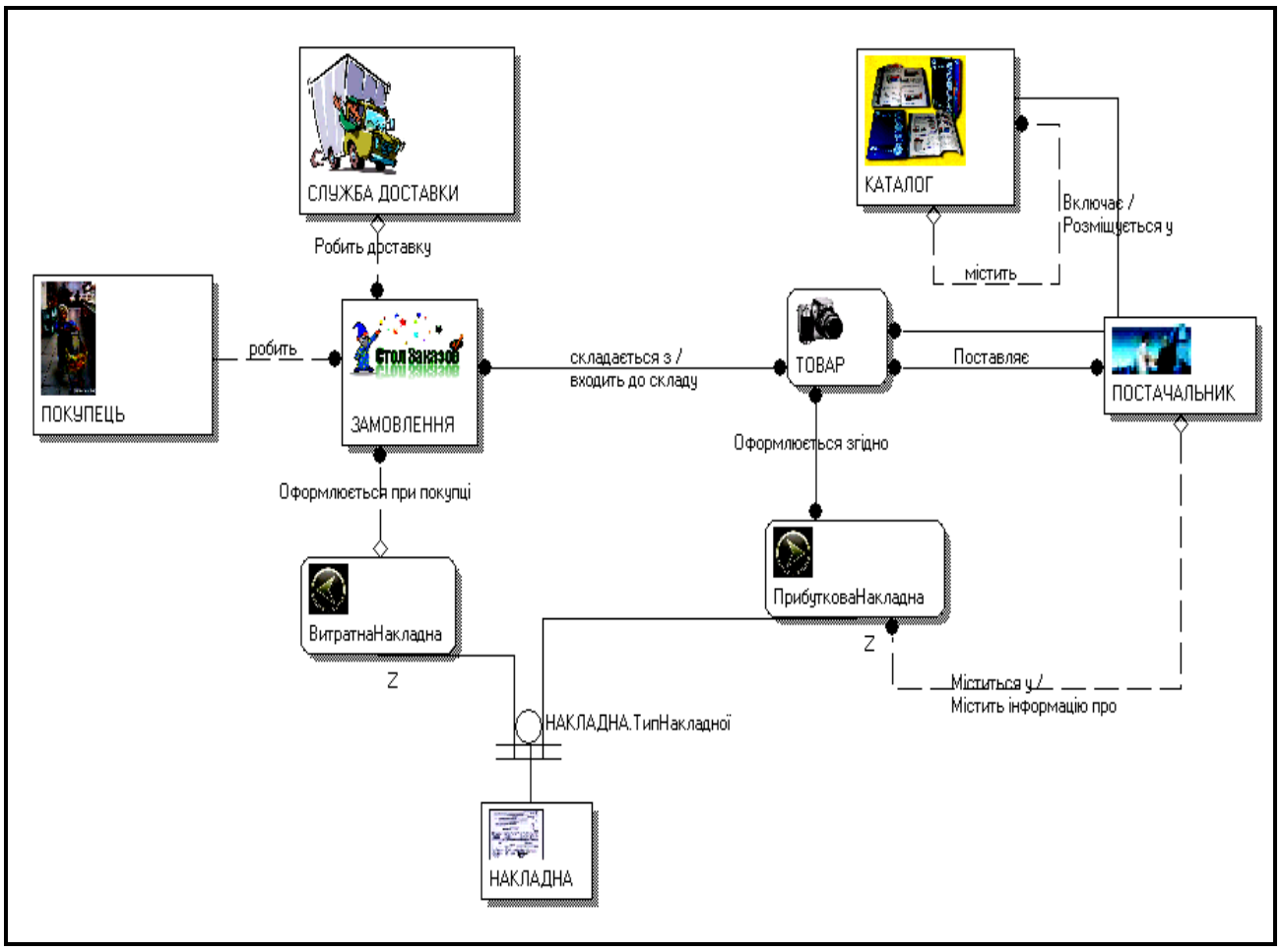


Рис. 1.51. **Діаграма з піктограмами**

1.4.5. Області і збережені відображення в ERwin

Поняття області

Область – це іменована версія моделі даних, яка може включати усі сутності, зв'язки, підтипи і текстові блоки або будь-яку підмножину об'єктів у повній моделі даних. Наприклад, можна створити область, в якій будуть тільки ті сутності і зв'язки між ними, які використовуються конкретною одиницею бізнесу або процесом.

За замовчуванням початкова модель даних отримує назву Головної області (Main Subject Area). Коли ви створюєте іншу область, то ви вибираєте об'єкти, які хочете в неї включити, і привласнюєте їй ім'я, що відповідає призначенню області.

Створення нової області

Виділимо у нашій схемі область, яку можна використати для ілюстрації бізнес-функцій відділу продажів. Нова область включатиме

сутності, що відносяться до оформлення замовлення: Покупець, Замовлення, Товар, Служба доставки.

1. Виберіть пункт меню Model -> Subject Areas або виберіть пункт Subject Areas в навігаторові моделі. Ви увійдете в редактор Subject Areas (рис. 1.52).

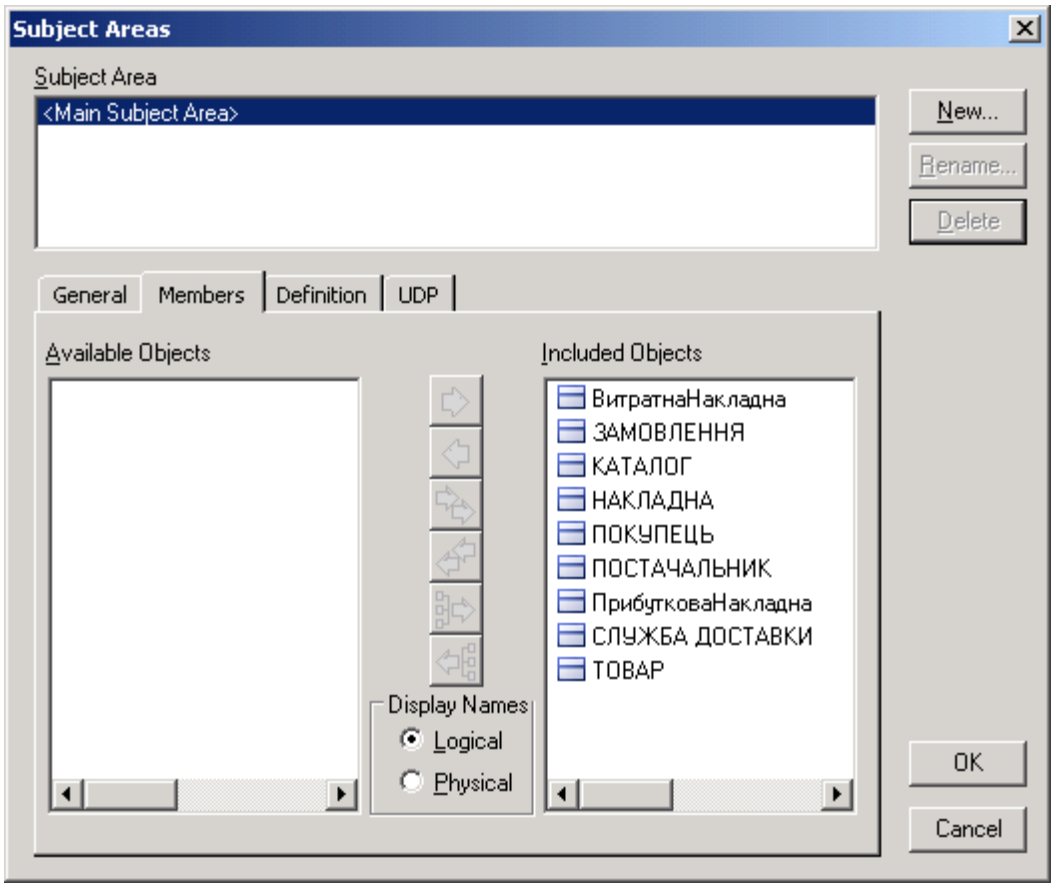


Рис. 1.52. **Вид вікна Subject Areas**

2. Клацніть кнопку New і введіть ім'я нової області. Якщо ви відмітите галочку Copy members from <Main Subject Area>, те в нову область будуть включені усі об'єкти Головної області.

3. Відредагуйте список об'єктів, що включаються в область. У вікні Available Objects приведений список об'єктів, які можна включити в область. У вікні Included Objects перераховані об'єкти, включені в область. Використовуючи кнопки, розташовані між цими списками, можна переміщати сутності з одного списку в інший (рис. 1.53).

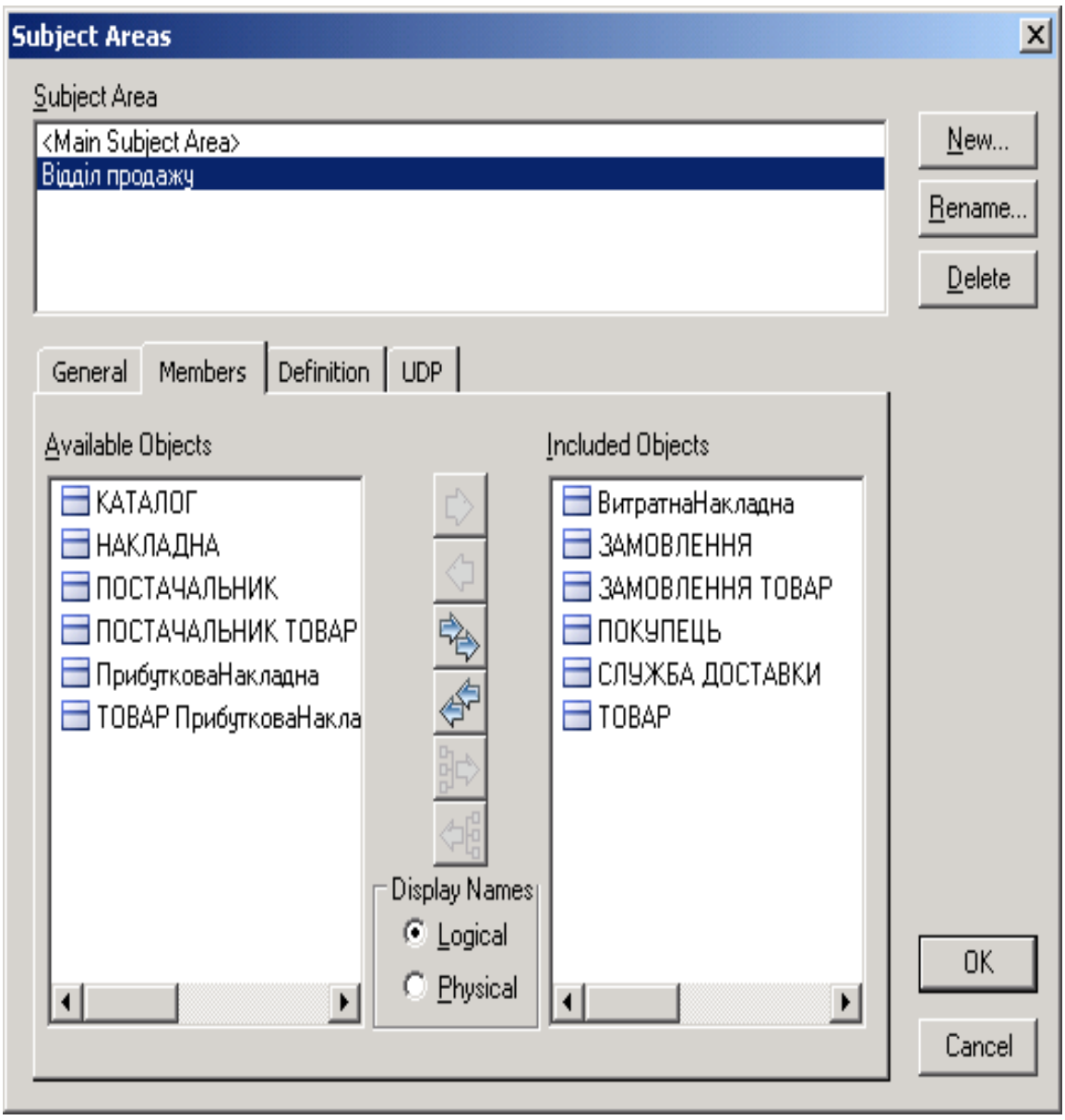


Рис. 1.53. **Вибір об'єктів у вікні Subject Areas**

4. Клацніть кнопку OK.

Редагування області

Зміна властивостей області – автора, складу, опису, властивостей, визначених користувачем, а також видалення областей здійснюється за допомогою редактора Subject Area.

Також можна скористатися контекстним меню області в навігаторові моделі, вибравши необхідну дію (рис. 1.54).

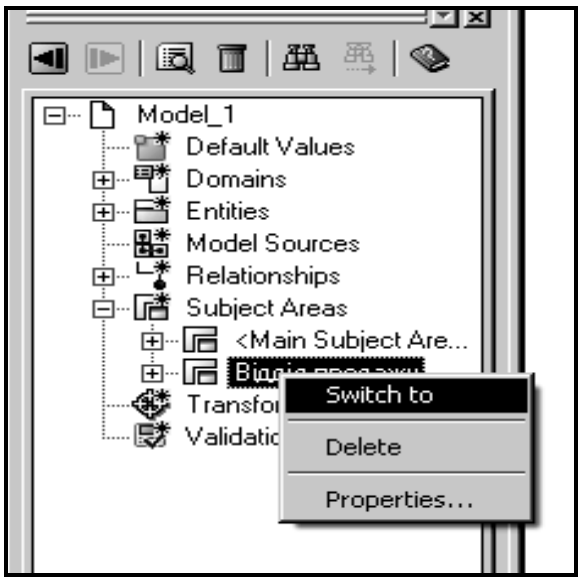


Рис. 1.54. **Контекстне меню у навігаторі Subject Areas**

Переміщення між областями

1. Клацніть стрілку поряд зі значком  на панелі інструментів.

2. У списку областей, що відкрився, клацніть потрібну область (рис. 1.55).

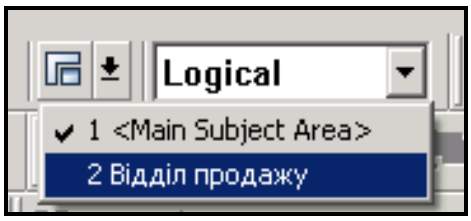


Рис. 1.55. **Вибір списку областей у Subject Areas**

Використання областей для управління великими діаграмами

Головна область включає усі сутності структури даних компанії. ERwin дозволяє Вам розбити головну область на одну або більше спеціалізованих областей, які відбивають окремі бізнес-функції і завдання. Зазвичай об'єкти, що включаються в спеціалізовану область, відносяться до певної одиниці бізнесу, наприклад, до служби фінансів, маркетингу або виробництву.

Області можуть виявитися особливо корисними при проектуванні та управлінні великою і складною моделлю даних. Розбиття головної області на декілька менших областей дозволяє різним групам, що

входять в організацію, сконцентруватися на процесах і завданнях, що відносяться до їх області бізнесу.

Поняття збереженого зображення

Збережене зображення (чи збережене відображення) – альтернативне представлення області, що висвітлює якийсь аспект усієї структури даних. Воно використовується для того, щоб працювати з різними представленнями діаграми, які допомагають фокусуватися на окремих бізнес-процесах. За замовчуванням початкове відображення області отримує назву Display1.

Збережені зображення містять ті ж об'єкти, що і початкова область, але тут можна розміщувати і оформлювати об'єкти довільним чином без зміни початкової діаграми.

Використання збереженого зображення допомагає краще визначити завдання або функції, пов'язані з конкретним бізнес-процесом. Наприклад, ви можете переносити об'єкти, пов'язані з процесом, на більш помітне місце на діаграмі. Змінивши розташування об'єктів, можна краще сфокусуватися на конкретній ділянці батьківської області.

Якщо ви хочете показати діаграму на декількох різних рівнях демонстрації зображення, то ви можете створити збережене зображення, яке містить цю діаграму на іншому рівні демонстрації. Наприклад, якщо Головна область показана на рівні атрибутів, то ви можете створити збережене зображення для показу цієї ж моделі на рівні демонстрації сутностей, піктограм тощо.

Примітка: Ви можете переносити будь-який графічний об'єкт в збереженому зображенні на нове місце, і це не відіб'ється на розташуванні об'єктів в Головній області або в інших зображеннях, заснованих на цій же моделі даних. Але якщо ви змінили шрифти або кольори об'єктів на збереженому зображенні, то ця зміна поширюється на Головну область і інші збережені зображення та області.

Створення нового збереженого зображення

Створимо збережене відображення для кожного рівня демонстрації.

1. Перейдіть в область, для якої створюється збережене відображення.

2. Клацніть правою кнопкою миші на будь-якому вільному місці діаграми і виберіть пункт меню Stored Display Settings.

3. Клацніть кнопку New і введіть ім'я збереженого відображення.

4. Клацніть кнопку OK (рис. 1.56).

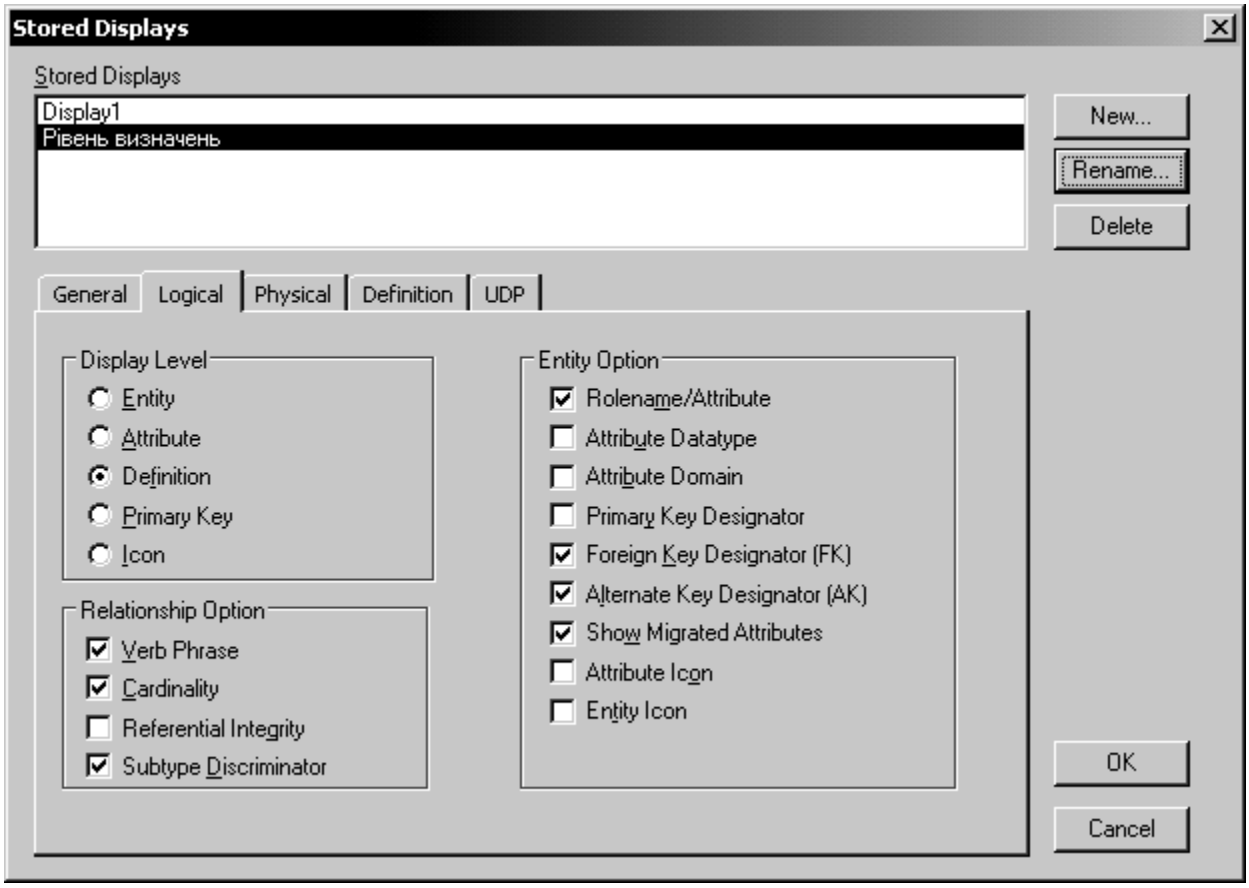


Рис. 1.56. **Вікно створення збереженого зображення**

Список збережених зображень для поточної області відображається у вигляді закладок в нижній частині області діаграми (рис. 1.57).

Рис. 1.57. **Вид закладок із списком збережених зображень**

Після створення нового збереженого відображення вкладка з його ім'ям з'являється праворуч від раніше створених відображень.

Щоб перейти з одного відображення в інше, клацніть відповідну вкладку.

Редагування збережених зображень

Якщо виникає необхідність в зміні властивостей збереженого відображення, наприклад при додаванні або видаленні його окремих елементів, то це можна здійснити за допомогою редактора Stored Display Settings.

Також можна скористатися контекстним меню збереженого відображення, в навігаторові моделі, вибравши необхідну дію (рис. 1.58).

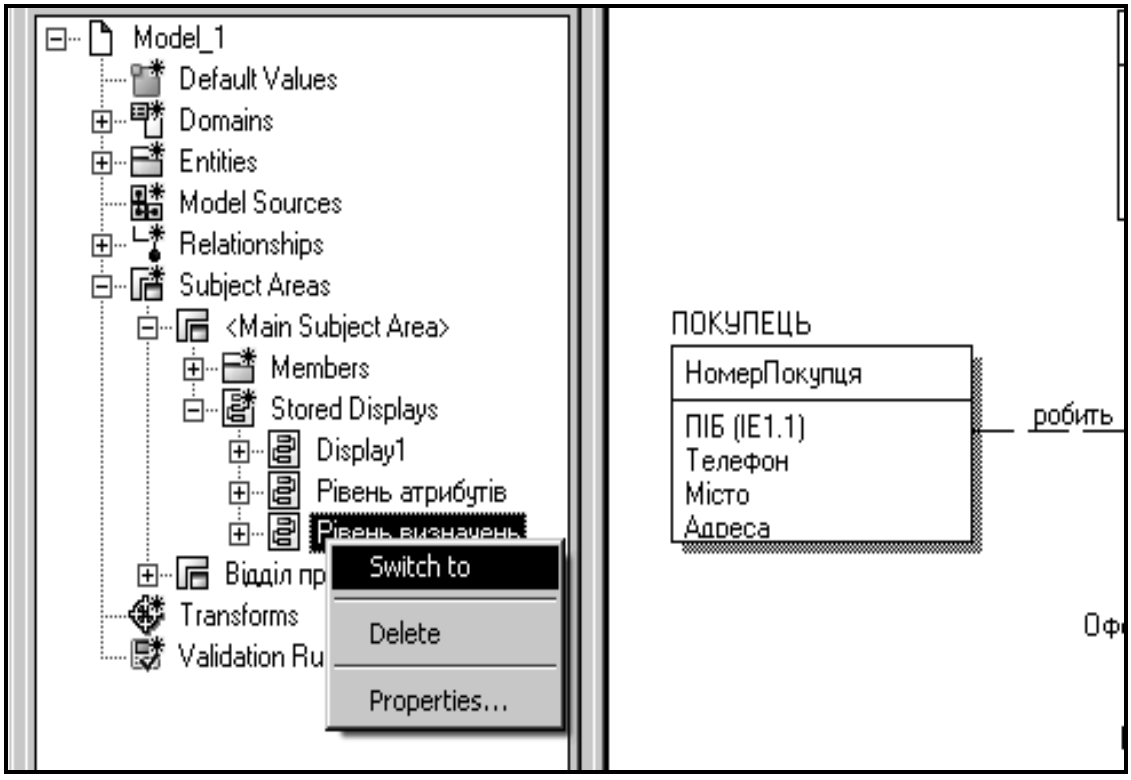


Рис. 1.58. **Перехід до редагування збереженого зображення**

Автоматична зміна розмірів сутностей

Після створення нового зображення усі об'єкти на діаграмі розташовуються в тому ж порядку і з такими ж розмірами, як на початковому відображенні області.

При зміні рівня демонстрації діаграми часто необхідно змінити розміри сутностей. Для цього можна скористатися функцією автоматичної зміни розмірів сутностей:

1. Виберіть пункт меню Format -> Preferences (рис. 1.59).

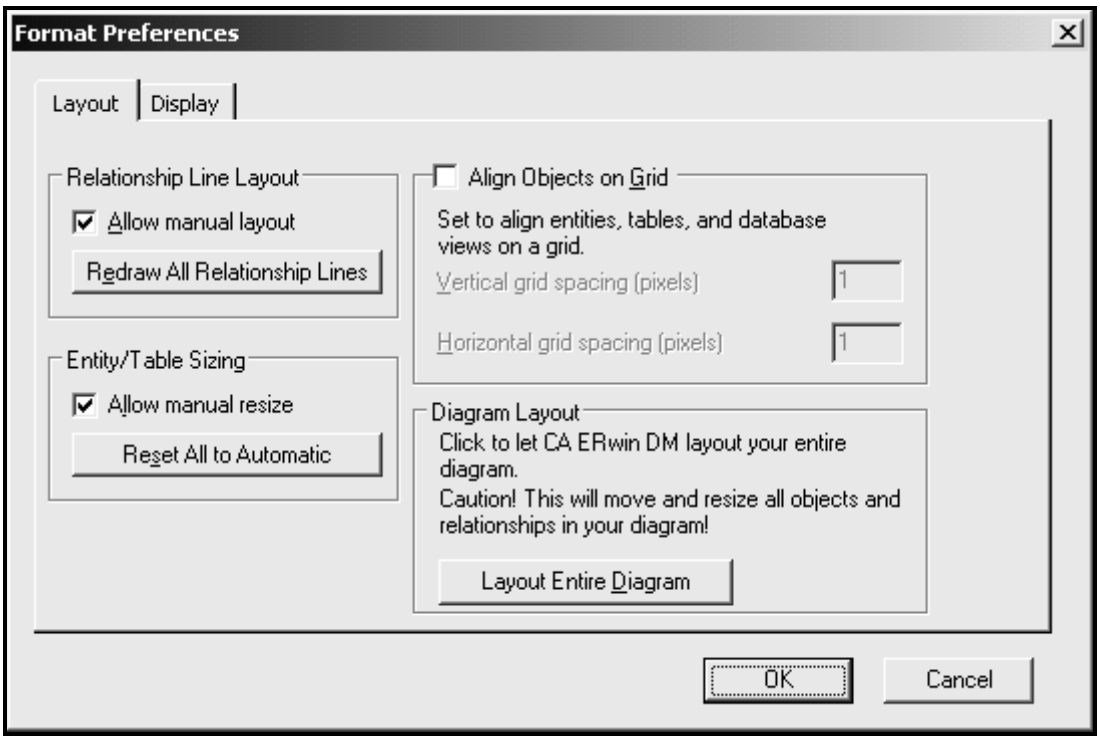


Рис. 1.59. **Вид вікна Format Preferences**

2. Для автоматичного оновлення розмірів об'єктів діаграми клацніть кнопку Reset All to Automatic і підтвердіть оновлення.

3. Щоб дозволити зміну розмірів об'єктів, відмітьте галочку Allow manual resize.

Також діалог Format Preferences дозволяє:

А. Визначити, як повинні розташовуватися зв'язки на діаграмі;

Б. Вирівняти об'єкти діаграмі по лініях сітки;

В. Перемальовувати усю діаграму;

Г. Встановити максимальні розміри для відображення описів, коментарів і виразів на діаграмі;

Д. Настроїти відображення тіней об'єктів;

Е. Настроїти відображення зв'язків і зовнішніх ключів сутностей.

1.4.6. Домени

Домен можна визначити як сукупність значень, з яких беруться значення атрибутів. Кожен атрибут може бути визначений тільки на одному домені, але на кожному домені може бути визначена множина атрибутів.

У поняття домена входить не лише тип даних, але і область значень даних. Наприклад, домен Ціна можна визначити як позитивне число і визначити атрибут Ціна в сутності Товар таким, що належить цьому домену.

В ERwin домен може бути визначений тільки один раз і використовуватися як у логічній, так і у фізичній моделях.

Створення домена

1. Виберіть пункт меню Model -> Domain Dictionary. Ви увійдете у редактор Domain Dictionary (рис. 1.60).

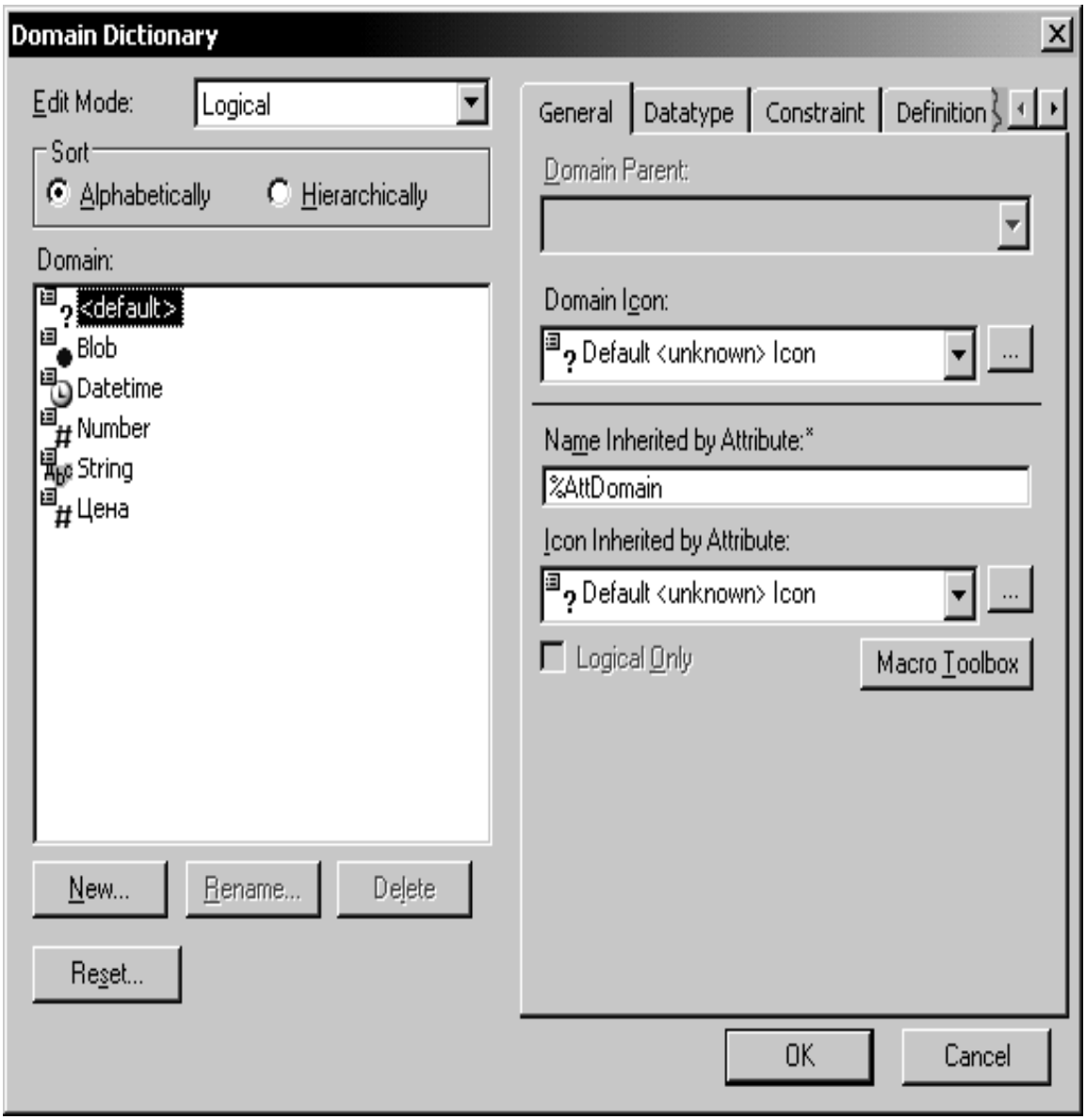


Рис. 1.60. **Вид вікна з переліком доменів**

2. Клацніть кнопку New. З'являється діалог New Domain.

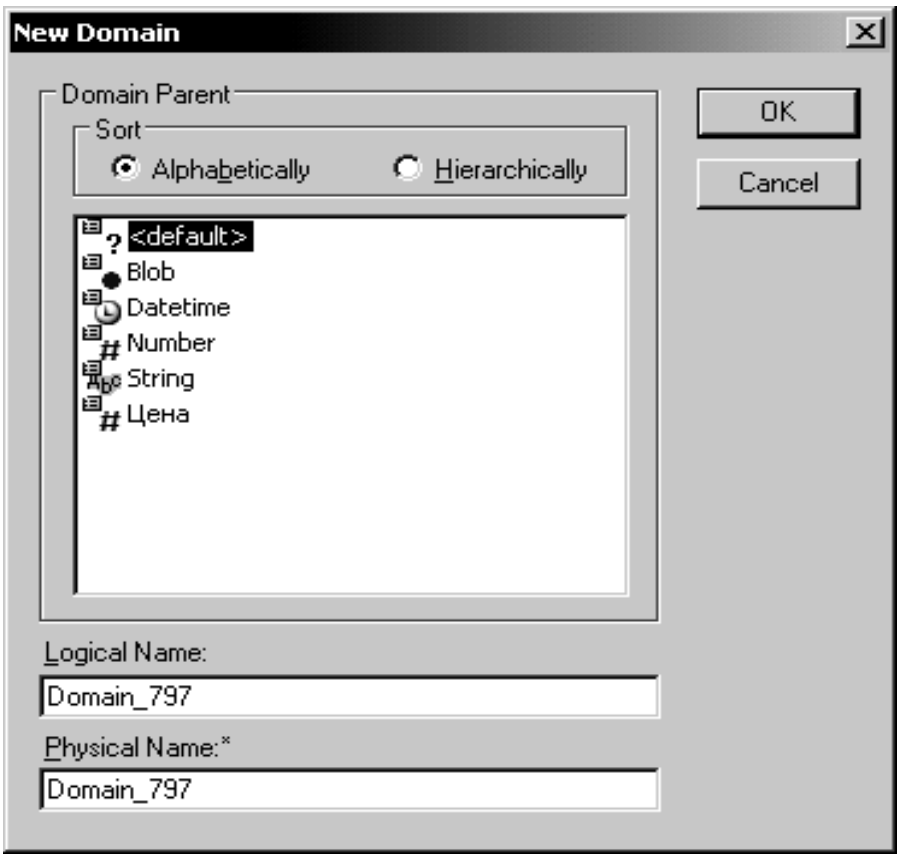


Рис. 1.61. **Вікно створення нового домена**

3. Виберіть батьківський домен зі списку Domain Parent. Новий домен можна створити на основі вже створеного користувачем домена, або на основі спочатку існуючого. За замовчуванням ERwin має чотири зумовлених домена – String, Number, Blob, Datetime. Новий домен наслідує усі властивості батьківського домена. Ці властивості надалі можна перевизначити.

4. Вкажіть ім'я домена в полі Logical Name. Можна також вказати ім'я домена на фізичному рівні в полі Physical Name. Якщо фізичне ім'я не вказане, за замовчуванням воно набуває значення логічного імені.

5. Клацніть кнопку OK.

На логічному рівні домени можна описати без конкретних фізичних властивостей. На фізичному рівні вони набувають специфічних властивостей, які можна змінити вручну. Так, для домена Ціна, який на логічному рівні базується на типі Number, можна на вкладці Datatype уточнити його тип, як MONEY() (рис. 1.62).

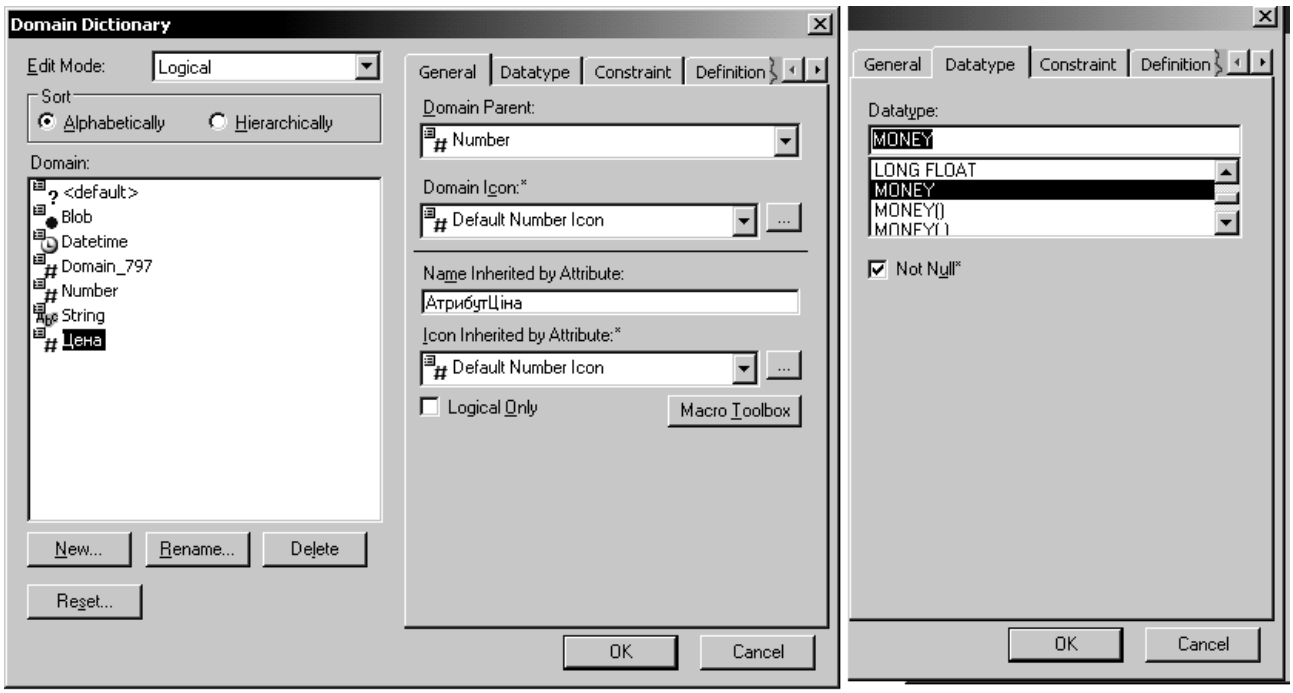


Рис. 1.62. **Вікно уточнення типу даних для домена**

Тоді на фізичному рівні, наприклад для СКБД ACCESS, домену буде присвоєний тип CURRENCY.

У редакторові Domain Dictionary можна зв'язати домен з іконкою, з якою він відображатиметься в списку доменів, а також на діаграмі в режимі відображення значків атрибутів.

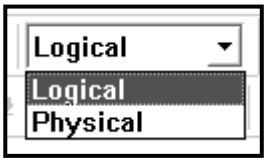
Кожен домен може бути описаний в закладці Definition, забезпечений коментарем в закладці Note або властивістю, визначеною користувачем в закладці UDP.

ERwin має спеціальний інструмент, який значно полегшує створення нових атрибутів у моделі, використовуючи опис доменів.

1. Виберіть у списку доменів у навігаторі моделі домен, на основі якого має бути створений атрибут.
2. Перенесіть його в потрібну сутність методом drag&drop.
3. У вказаній сутності буде створений новий атрибут з ім'ям, яке було задано у полі Name Inherited by Attribute на вкладці General у властивостях домена. Якщо значення поля не задане, за замовчуванням приймається ім'я домена.

Физичні властивості домена

При виклику діалога Domain Dictionary на фізичному рівні (для переходу на фізичний рівень слід вибрати значення Physical) можна редагувати фізичні



властивості домена. При цьому в діалозі з'являється нова закладка БД, що відповідає вибраному серверу. У нашому випадку з'являться закладки Access,….Access.

На закладці Access можна задати конкретний тип даних, що відповідають домену, правила привласнення нульових значень, правила валідації (правила перевірки допустимих значень) і завдання значення за замовчуванням.

Правила валідації і значення за замовчуванням мають бути заздалегідь описані та іменовані. Для виклику діалогів редагування правил валідації і значень за замовчуванням служать кнопки праворуч від відповідного списку вибору (Valid і Default) вкладки Constraint (рис. 1.63).

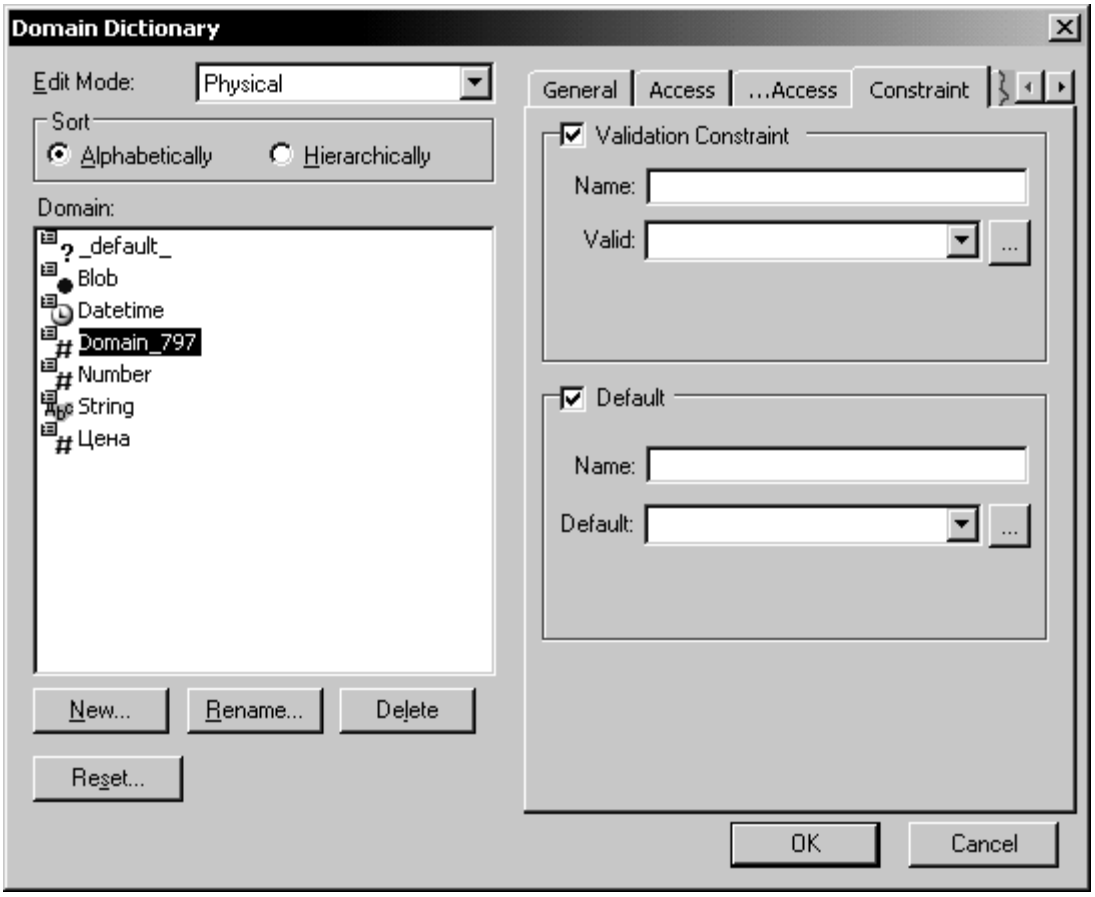


Рис. 1.63. **Вікно для переходу в редагування правил валідації**

1.4.7. Правила посилальної цілісності

Поняття та види посилальної цілісності

Правила посилальної цілісності (Referential Integrity) – логічні конструкції, які виражають бізнес-правила використання даних і є правилами вставки, заміни та видаленнями рядків з таблиць.

При генерації схеми БД на основі опцій логічної моделі будуть згенеровані правила декларативної посилальної цілісності, які мають бути приписані для кожного зв'язка, і тригери, що забезпечують посилальну цілісність.

У нашому прикладі сутності Покупець і Замовлення пов'язані ненульовим зв'язком, тобто поле Номер покупця в таблиці Замовлення повинно бути заповнено обов'язково. Правила цілісності визначають, що буде із записом про замовлення, якщо ми видалимо запис про покупця, який його зробив.

Існують такі правила для видалення рядків з таблиці:

1) RESTRICT – забороняє видалення екземпляра батьківської сутності, доки у дочірній сутності є екземпляри, залежні від нього;

2) CASCADE – видалення екземпляра батьківської сутності призводить до видалення усіх екземплярів дочірньої сутності, залежних від нього;

3) SET NULL – при видаленні екземпляра батьківської сутності значення пов'язаного поля в усіх екземплярах дочірньої сутності, залежних від нього, замінюються значенням NULL;

4) SET DEFAULT – при видаленні екземпляра батьківської сутності значення пов'язаного поля в усіх екземплярах дочірньої сутності, залежних від нього, замінюються значенням за замовчуванням;

5) NONE – при видаленні екземпляра батьківської сутності значення пов'язаного поля в усіх екземплярах дочірньої сутності залишається незмінним.

Аналогічні правила застосовуються для заміни і вставки рядків в таблицях.

Встановлення правил посилальної цілісності:

1. Зайдіть у властивості зв'язка, клацнувши правою кнопкою миші по зв'язку і вибравши пункт Relationship Properties в контекстному меню.

2. Зайдіть на вкладку RI Actions.

3. Задайте правила посилальної цілісності (рис. 1.64).

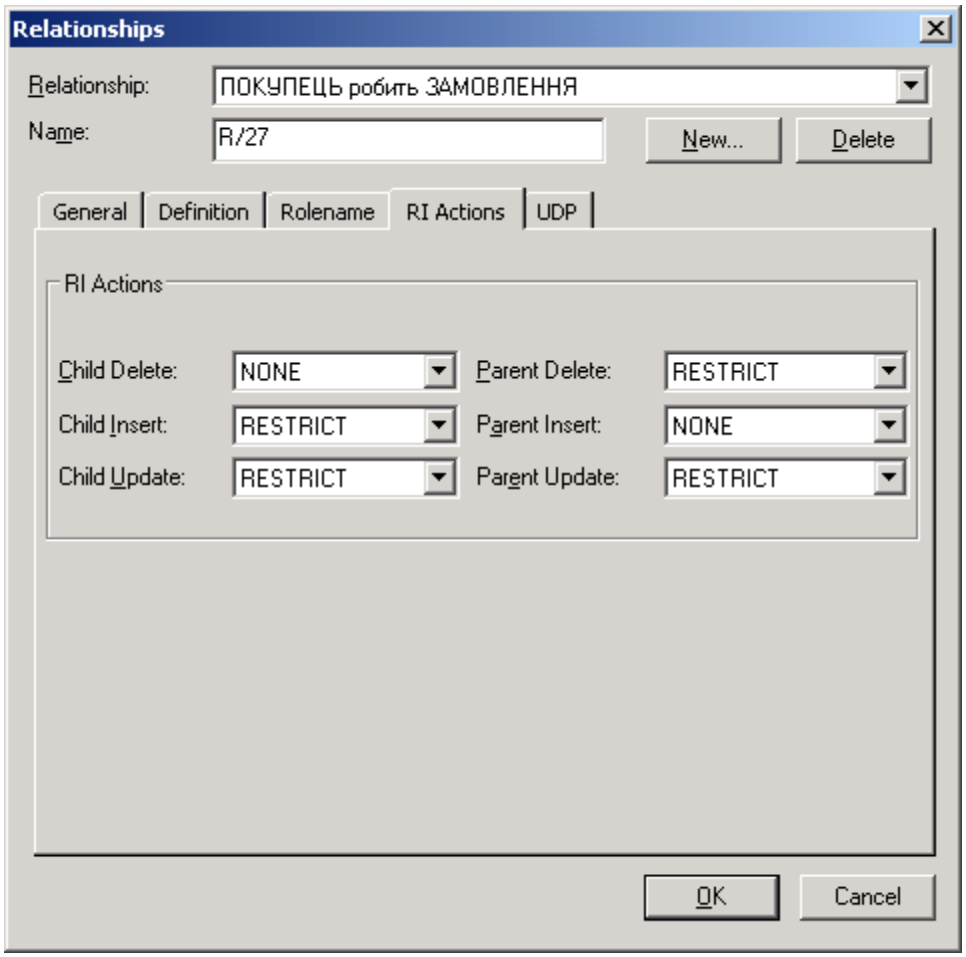


Рис. 1.64. **Вікно встановлення правил посилальної цілісності**

Створення фізичної моделі даних

Так само, як використовувалися редактори ERwin для завдання імен і визначень сутностей при проектуванні логічної моделі даних, можна задавати відповідні імена для фізичних таблиць, колонок, зв'язків, а також призначати точні типи даних, які підтримує конкретна СКБД.

1.4.8. Вибір СКБД

Фізичний рівень представлення моделі залежить від вибраного сервера.

Для вибору СКБД виберіть пункт меню Database -> Choose Database (цей пункт меню доступний тільки на фізичному рівні). Ви увійдете до діалогу Target Server (рис. 1.65).

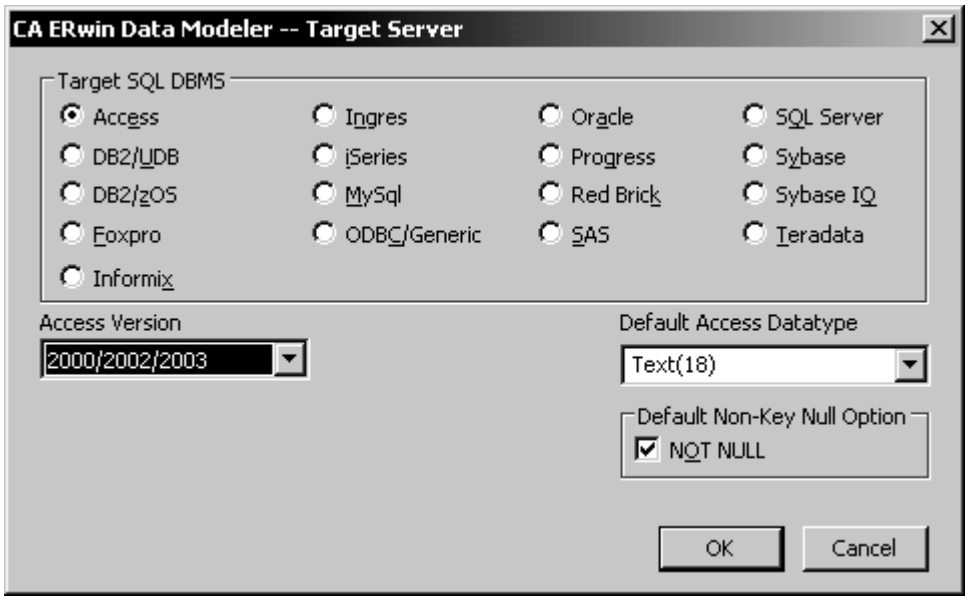


Рис. 1.65. **Вікно вибору сервера СКБД**

ERwin підтримує практично усі поширені СКБД, усього більше 20 реляційних і не реляційних БД. Для вибору СКБД треба відмітити прапорець поряд з ім'ям СКБД.

Діалог Target Server дозволяє задати тип даних, використовуваний за замовчуванням і опцію NULL для нових колонок. Тип даних можна вибрати в списку Default Datatype, який автоматично заповнюється типами даних, підтримуваних вибраним сервером.

Опція NOT NULL у блоці Default Non – Key Null Option дозволяє заборонити значення NULL для неключових колонок.

Перетворення типів цих СКБД

На початку роботи з новою діаграмою ERwin використовує інформацію, яка задана за замовчуванням для конкретної СКБД.

При зміні СКБД, ERwin пропонує автоматично перетворити тип даних, пов'язаний з кожним атрибутом, на еквівалентні типи даних нової СКБД. Для цього після вибору нової СКБД треба підтвердити автоматичне перетворення типів даних.

Якщо в новій СКБД не задано еквівалентного типу даних для одного або декількох атрибутів моделі (наприклад, початкова СКБД підтримує графічний тип даних, а в новій СКБД немає еквіваленту цьому типу), ERwin надає можливість згенерувати звіт зі списком неперетворюваних типів даних і провести їх ручне перетворення.

Визначення інформації фізичної схеми

За замовчуванням ERwin генерує імена таблиць і індексів за шаблоном на основі імен відповідних сутностей і ключів логічної моделі, враховуючи синтаксичні обмеження, що накладаються на СКБД.

Додавання нових таблиць, колонок і зв'язків відбувається аналогічно створенню сутностей, атрибутів і зв'язків на логічному рівні.

Таблиці

Редактор Tables дозволяє задати властивості таблиць моделі – її ім'я, синоніми, правила валідації, процедури і так далі.

Для виклику редактора Tables клацніть правою кнопкою миші по таблиці і виберіть пункт Table Properties -> і конкретну властивість, наприклад General або Validation (рис. 1.66).

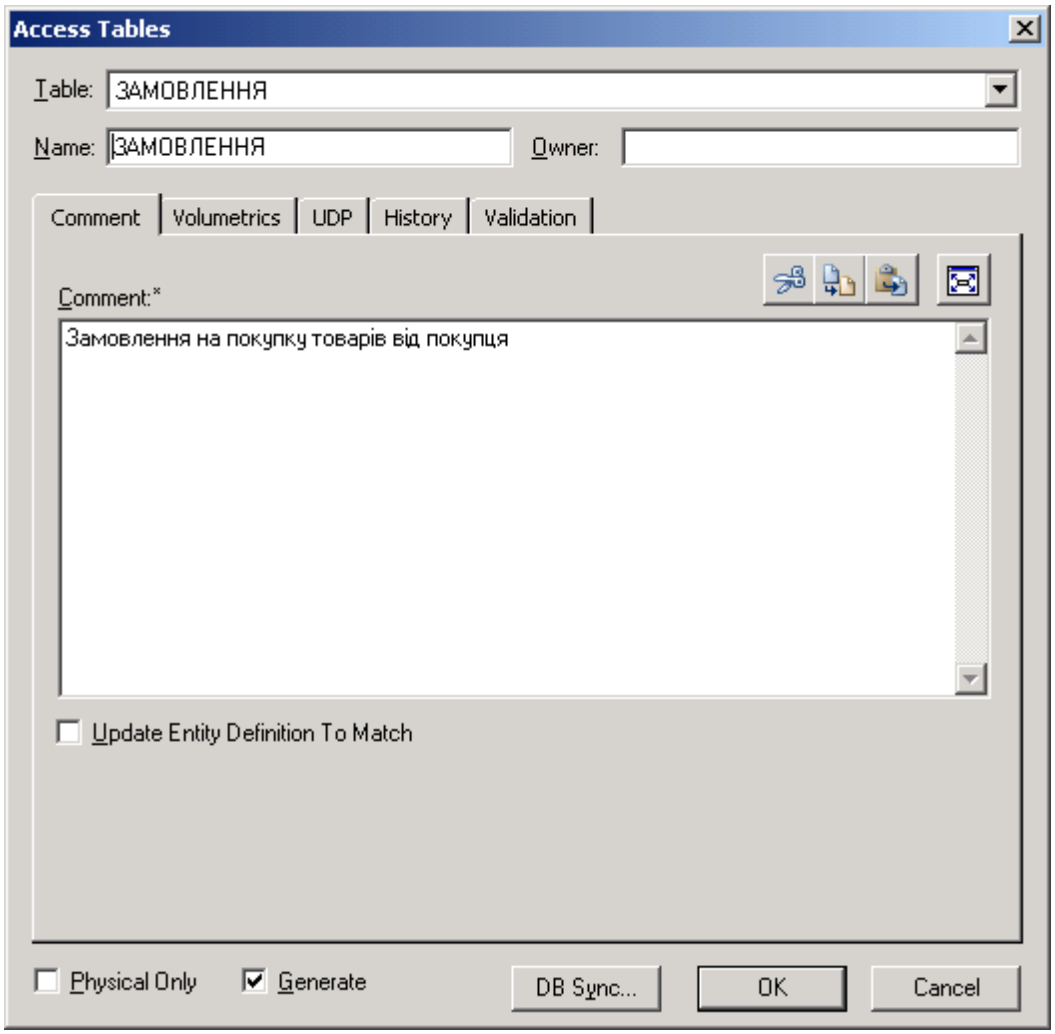


Рис. 1.66. **Вікно редактора таблиць**

Якщо вибрана опція Generate, то при генерації схеми БД виконуватиметься команда CREATE TABLE.

Кнопка DB Sync служить для негайної синхронізації моделі з системним каталогом БД.

Склад і зміст закладок редактора Tables залежить від вибраної СКБД.

Колонки

Для завдання властивостей колонок, відмінних від значення за замовчуванням, служить редактор Columns.

Колонці фізичної таблиці можуть бути присвоєні такі характеристики: ім'я колонки, тип даних, режим нульових значень, правило валідації, значення за замовчуванням і так далі.

Щоб викликати редактор Columns, клацніть правою кнопкою миші по таблиці і виберіть пункт Columns у контекстному меню (рис. 1.67).

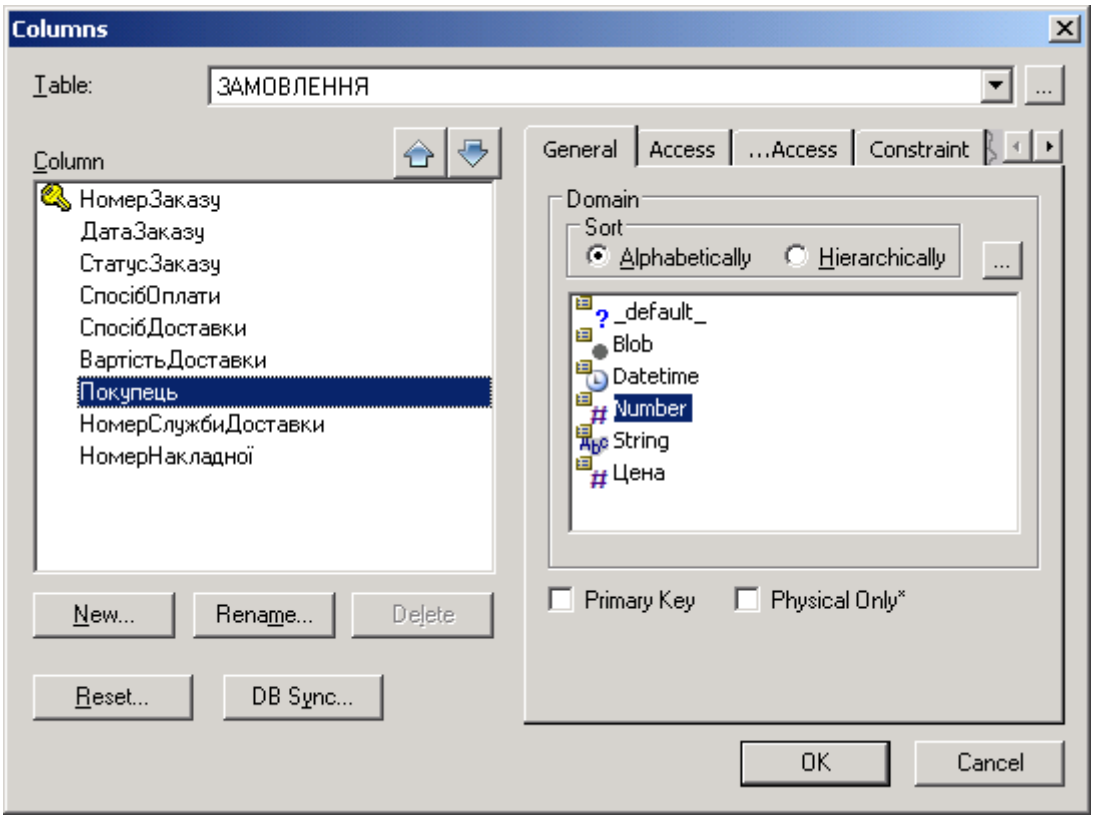


Рис. 1.67. **Вікно редактора Columns (атрибутів)**

Закладка, що відповідає вибраній СКБД, дозволяє задати тип даних, опцію NULL, правила валідації і значення за замовчуванням. Для

СКБД Access, AS/400 і Teradata створюються додаткові закладки для завдання властивостей (рис. 1.68).

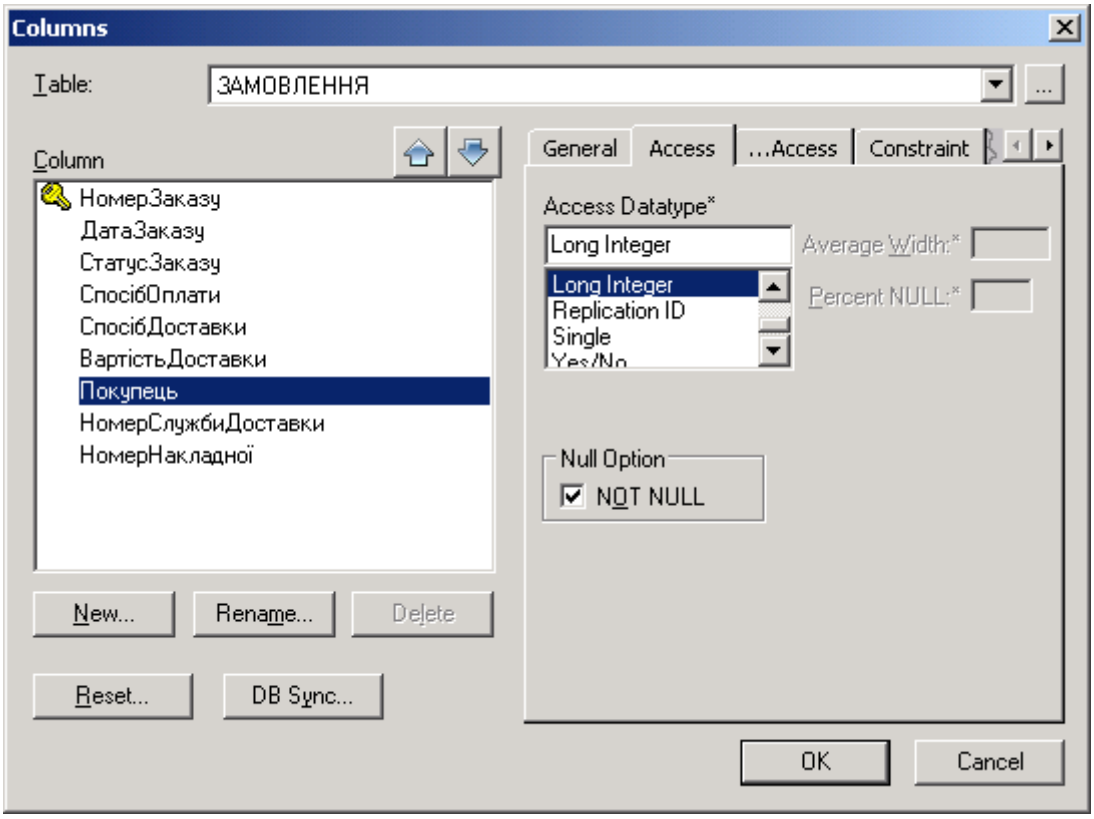


Рис. 1.68. **Вікно встановлення типу даних для атрибута**

Подання

Подання (view), або, як їх іноді називають, тимчасові або похідні таблиці, є об'єктами БД, дані в яких не зберігаються постійно, як у таблиці, а формуються динамічно при зверненні до подання. Подання не може існувати саме по собі, а визначається тільки в термінах однієї або декількох таблиць. Застосування подання дозволяє розробникові БД забезпечити кожному користувачеві або групі користувачів свій погляд на дані, що вирішує проблеми простоти використання і безпеки даних.

ERwin має спеціальні інструменти для створення і редагування подання. Палітра інструментів на фізичному рівні містить кнопки створення подання –  і встановлення зв'язків між таблицями і поданнями –  .

Подання і зв'язок між ними і таблицями створюються на діаграмі аналогічно створенню таблиць і зв'язків між ними.

По суті подання – це ті ж SQL-запити, які потрібні для вибірки інформації, що представляється користувачеві. Наприклад, для відображення картки замовлення недостатньо вибрати усі поля з таблиці замовлень. Необхідно також вказати інформацію про покупця, службу доставки, сформованої витратної накладної. Для цього можна створити подання, пов'язане з таблицями товарів, замовлень та доставки. Властивості подання нагадують майстер створення запитів в Access (рис. 1.69).

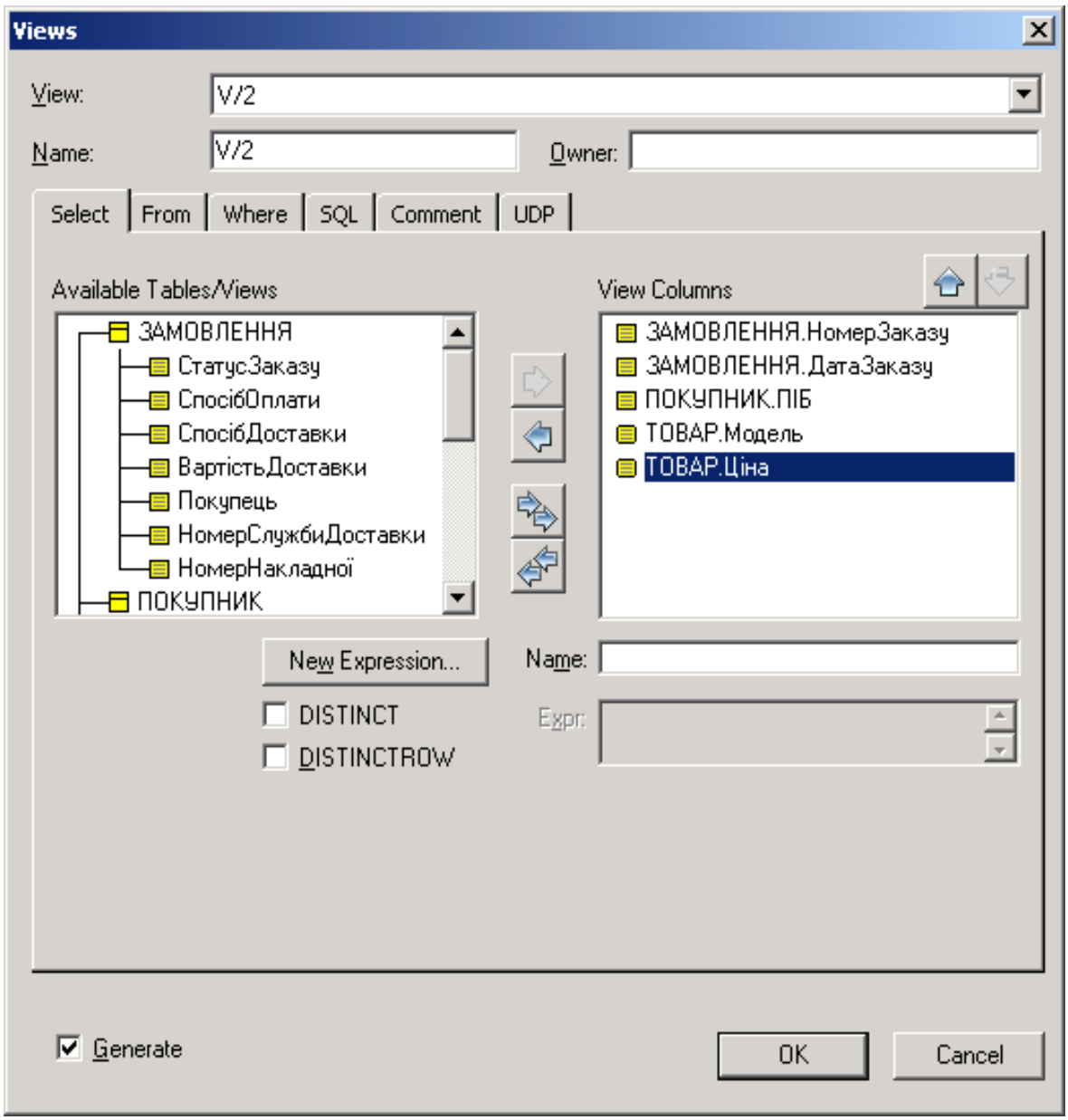


Рис. 1.69. **Вікно створення подання**

Для редагування подання служить діалог Views. Для його виклику слід клацнути правою кнопкою миші по поданню і вибрати в меню пункт

Database View Properties. Вид створеного подання на діаграмі показаний на рис 1.70.

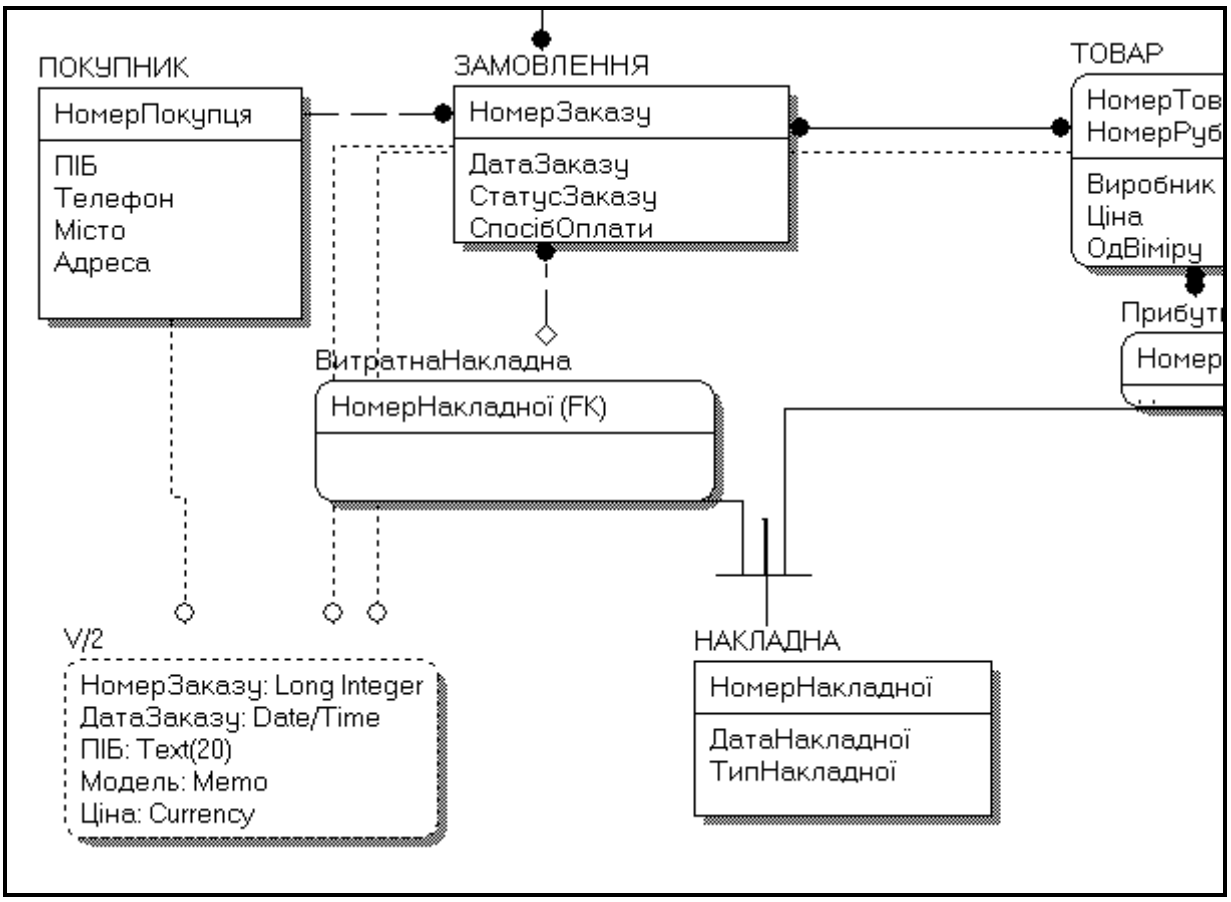


Рис. 1.70. **Вид створеного подання на ER-діаграмі**

Правила валідації та значення за замовчуванням

ERwin підтримує правила валідації для колонок, а також значення, що привласнюються колонкам за замовчуванням. Правило валідації задає список допустимих значень для конкретної колонки і/або правила перевірки допустимих значень. Значення за замовчуванням – це значення, яке прийме колонка, якщо під час введення даних інше значення не було задане явним чином.

З кожною колонкою або доменом можна зв'язати значення за замовчуванням (якщо вибрана СКБД підтримує домени).

Наприклад, створимо правило валідації для колонки Тип таблиці Служба доставки. У описі предметної області задано, що замовлення можуть доставлятися або кур'єрами, або транспортними компаніями.

Щоб визначити правила валідації для колонки:

1. Увійдіть в редактор Columns і на вкладці, що відповідає вибраній СКБД, виберіть потрібне правило у списку Valid на вкладці Constraint.

Якщо правило ще не створене, клацніть кнопку ..., розташовану вище від поля Valid. Ви увійдете в редактор Validation Constraint.

2. Клацніть кнопку New і введіть ім'я нового правила (рис. 1.71).

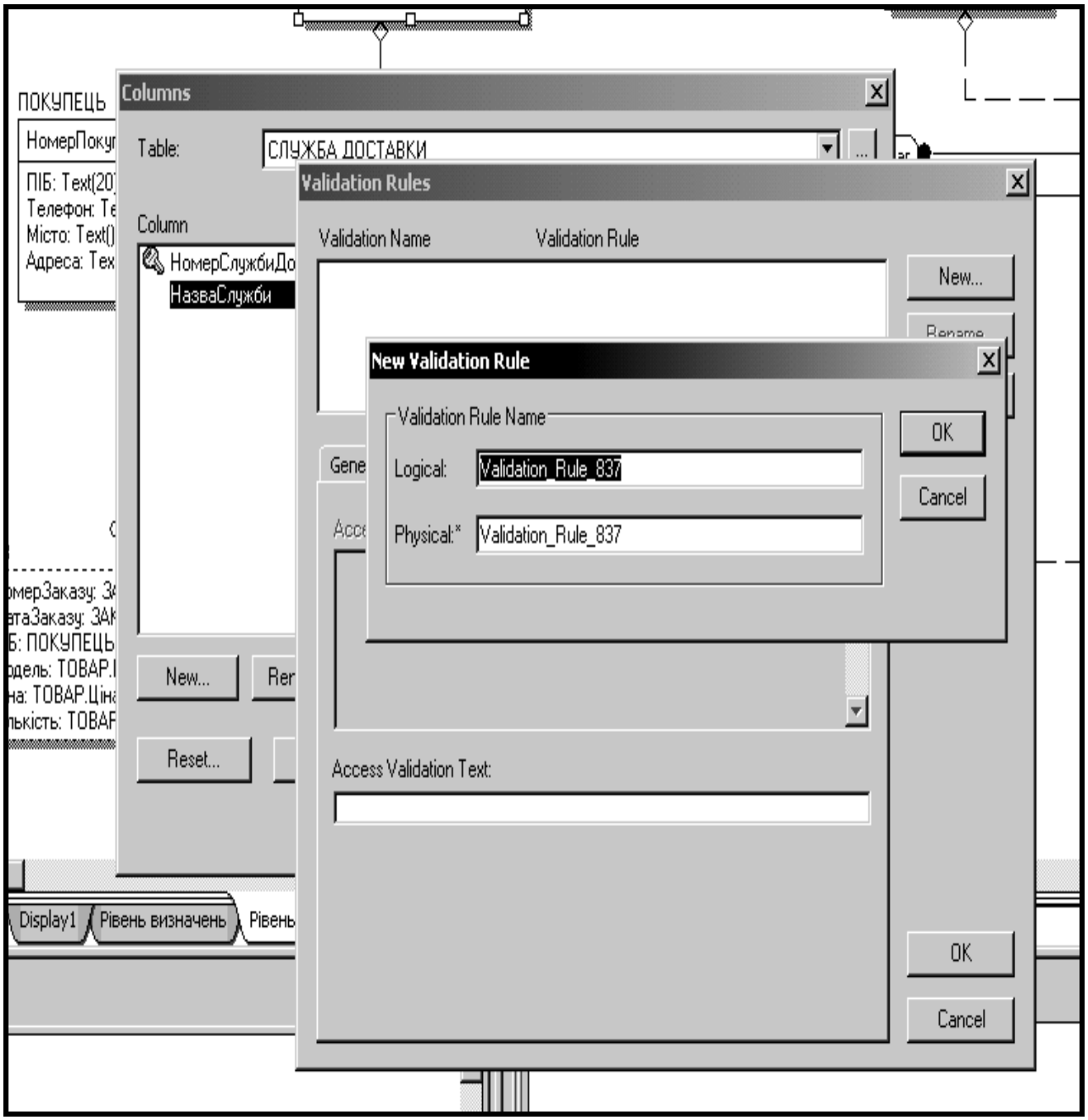


Рис. 1.71. **Створення правила валідації**

3. На вкладці General виберіть опцію Valid Values List і введіть допустимі значення поля (рис. 1.72).

4. Клацніть кнопку OK.

Редактор Validation Rules також дозволяє задати максимальне і мінімальне значення поля або ввести правила валідації вручну.

Аналогічно створюються правила для завдання значення за замовчуванням.

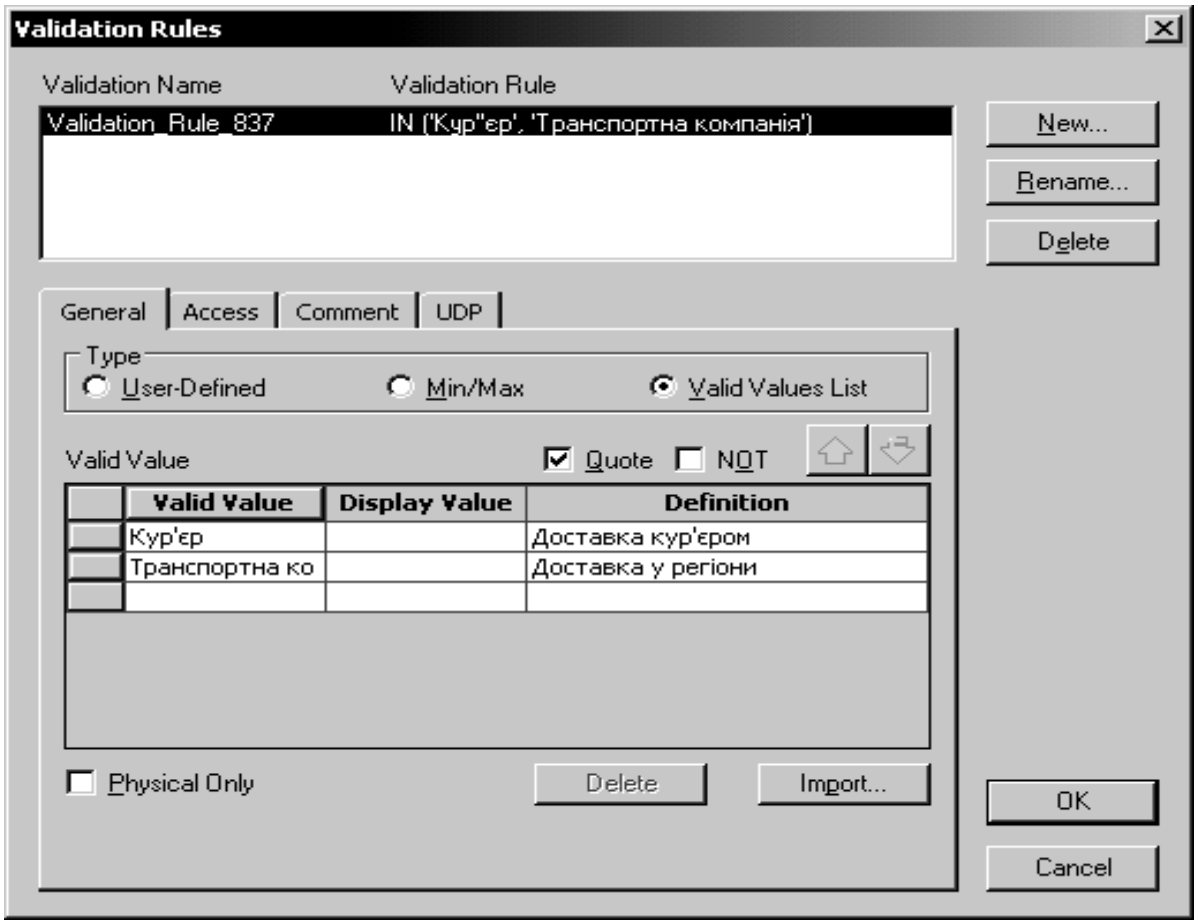


Рис. 1.72. **Редагування правила валідації**

Індекси

Щоб розв'язати проблему пошуку даних, СКБД використовує особливий об'єкт, що називається індексом. Він подібний до змісту книги, який вказує на усі номери сторінок, присвячених конкретній темі. Індекс містить відсортовану по колонці або декільком колонкам інформацію і вказує на рядки, в яких зберігається конкретне значення колонки.

При генерації схеми фізичної БД ERwin автоматично створює окремий індекс на основі первинного ключа кожної таблиці, а також на основі усіх альтернативних ключів, зовнішніх ключів і інверсійних входів, оскільки ці колонки найчастіше використовуються для пошуку даних.

При створенні логічної схеми даних ми створили інверсійний вхід, що містить поле ПІБ в таблиці Покупці. Для цього поля буде створений індекс. У індексі імена покупців будуть відсортовані в алфавітному

порядку. Індекс міститиме посилання, що вказує, в якому місці таблиці зберігається цей рядок.

При виконанні запиту на пошук покупця СКБД переглядає індекс, замість того щоб переглядати по порядку усі рядки таблиці Покупці. Оскільки значення в індексі зберігаються в певному порядку, переглядати треба набагато менший об'єм даних, що значно зменшує час виконання запиту.

Адміністратор СКБД повинен аналізувати найчастіше виконувані запити і створювати індекси з різними колонками і порядком сортування для збільшення ефективності пошуку при роботі конкретних застосувань.

Змінити характеристики існуючого індексу або створити новий індекс можна в редакторові Indexes. Для його виклику слід клацнути правою кнопкою миші по таблиці і вибрати в контекстному меню пункт Indexes.

Створення і редагування індексів відбувається аналогічно створенню і редагуванню альтернативних ключів і інверсійних входів на логічному рівні моделі. При цьому вибрана опція Unique при створенні індексу означає, що поле має бути унікальним, і такий індекс відповідатиме альтернативному ключу на логічному рівні. При відключеній опції Unique створюватиметься неунікальний індекс, що відповідає інверсійному входу. На вкладці, відповідній вибраній СКБД можна задати додаткові опції індексу.

1.4.9. Позбавлення від зв'язків "багато-до-багатьох" і категоріальних зв'язків

Перед процесом генерації схеми бази даних для вибраного типу СКБД необхідно позбавитися від зв'язків "багато-до-багатьох" і категоріальних зв'язків. Позбавлення від зв'язків "багато-до-багатьох" було розглянуто в попередньому розділі. Тепер розглянемо як позбавитися від категоріальних зв'язків.

Виділіть створений категоріальний зв'язок для накладних і на панелі інструментів клацніть на кнопці " Supertype-Subtype Identity" (рис. 1.73).

У вікні Supertype/Subtype Indentity Transform Wizard натискаємо кнопку "Далее" і переходимо у вікно опису перетворення категоріального зв'язку (рис. 1.74).

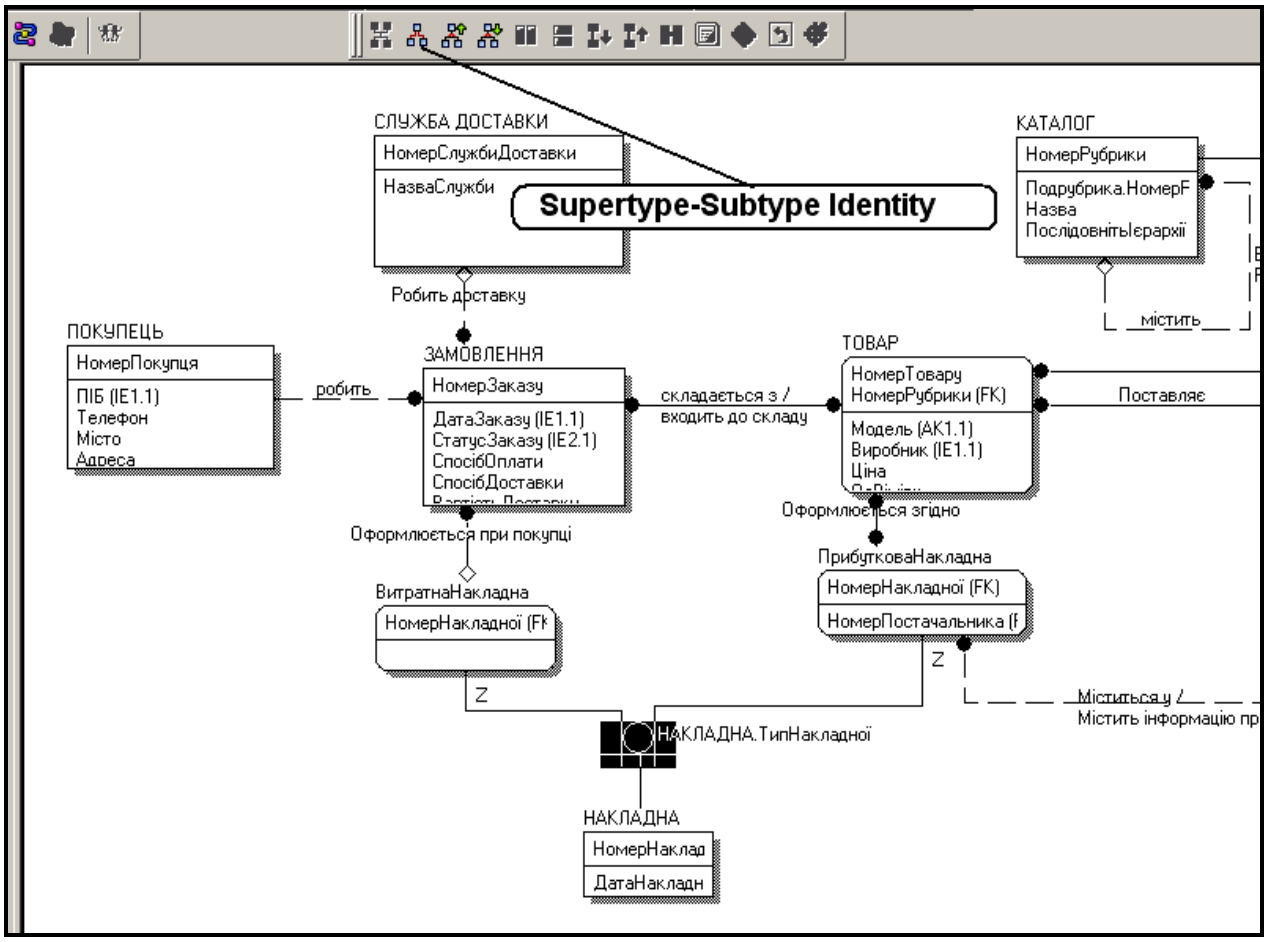


Рис. 1.73. **Позбавлення від категоріальних зв'язків**

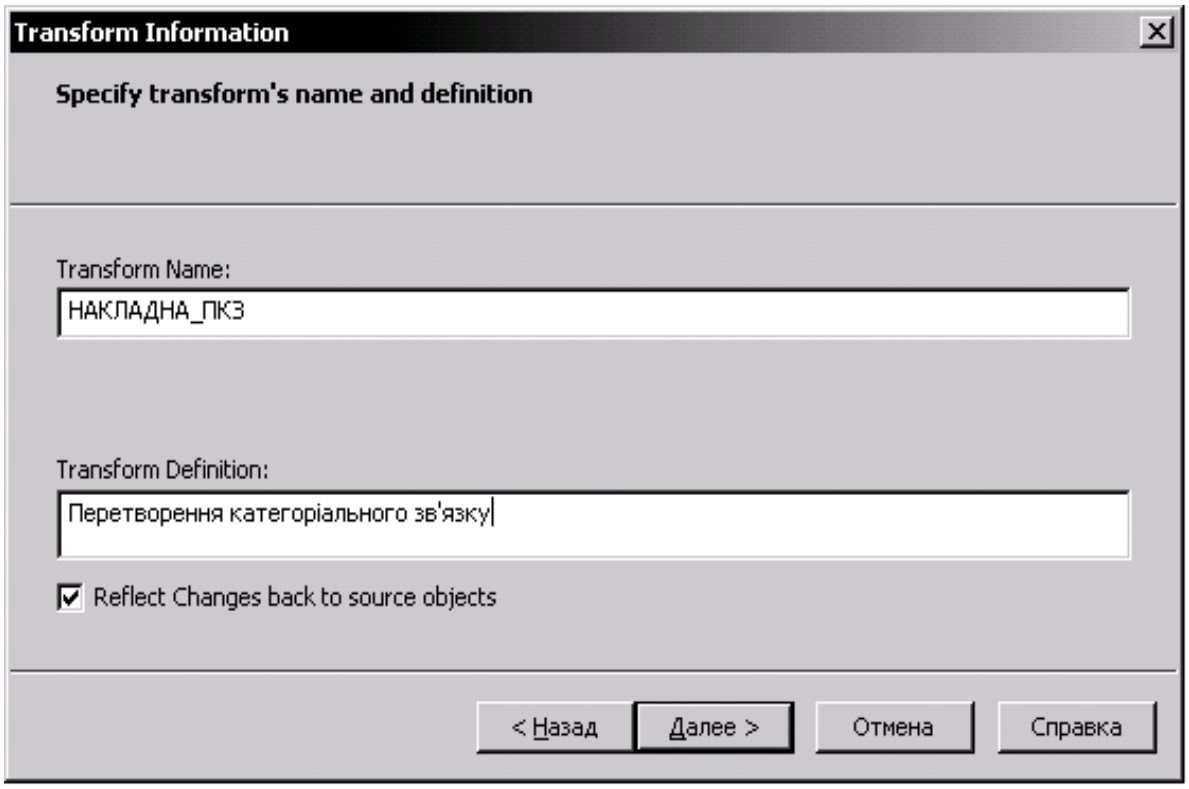


Рис. 1.74. **Вікно опису перетворення категоріального зв'язку**

Вводимо ім'я перетворення і його визначення і тиснемо кнопку "Далее". Потрапляємо у вікно "Summary" і тиснемо кнопку "Готово".

У результаті отримуємо таку модель (рис. 1.75):

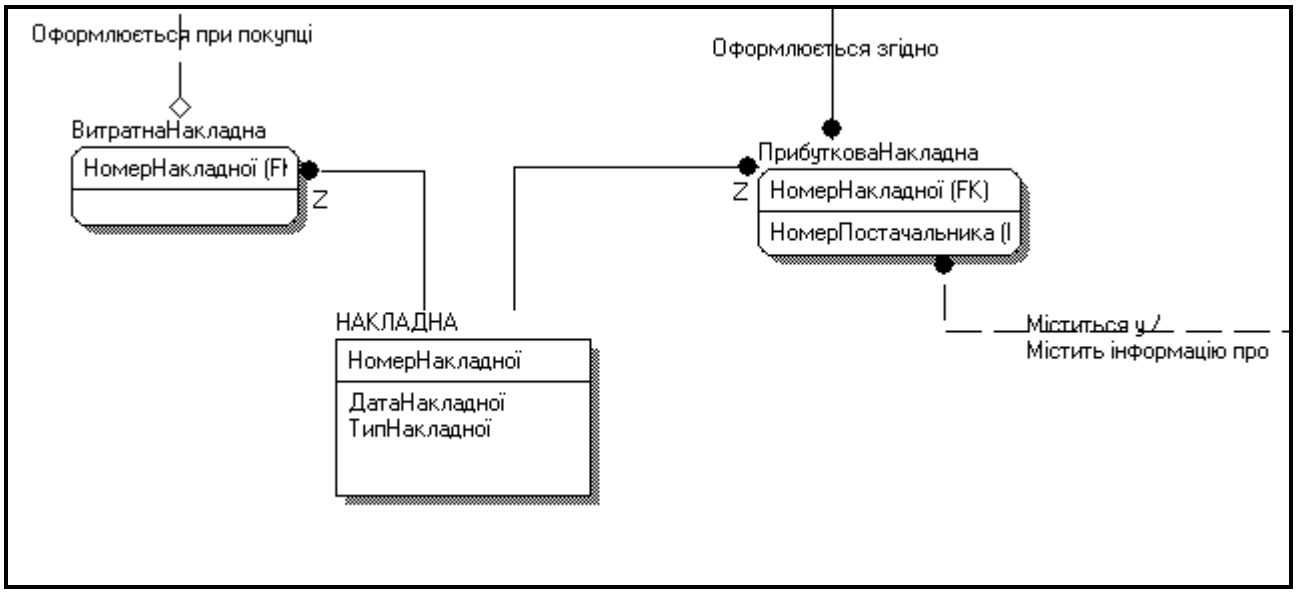


Рис. 1.75. **Фрагмент моделі з перетвореним категоріальним зв'язком**

Надалі, позбавившись від усіх зв'язків "багато-до-багатьох" можна приступати до формування схеми бази даних у рамках конкретної СКБД.

1.4.10. Пряме та зворотне проектування

Процес генерації фізичної схеми БД з логічної моделі даних називається прямим проектуванням (Forward Engineering). При генерації фізичної схеми ERwin включає тригери посилальної цілісності, збережені процедури, індекси, обмеження і інші можливості, доступні при визначенні таблиць у вибраній СКБД.

Процес генерації логічної моделі з фізичної БД називається зворотним проектуванням (Reverse Engineering). ERwin дозволяє створити модель даних шляхом зворотного проектування наявної БД. Після того, як модель створена, можна перемкнутися на інший сервер (модель буде конвертована) і зробити пряме проектування структури БД для іншої СКБД.

Окрім режиму прямого і зворотного проектування ERwin підтримує синхронізацію між логічною моделлю і системним каталогом СКБД упродовж усього життєвого циклу створення ІС.

Для виконання ***прямого проектування*** слід вибрати пункт меню Tools ->Forward Engineer/Schema Generation або клацнути по кнопці  на панелі інструментів Database. У результаті цього буде здійснений вхід у діалог Schema Generation (рис. 1.76).

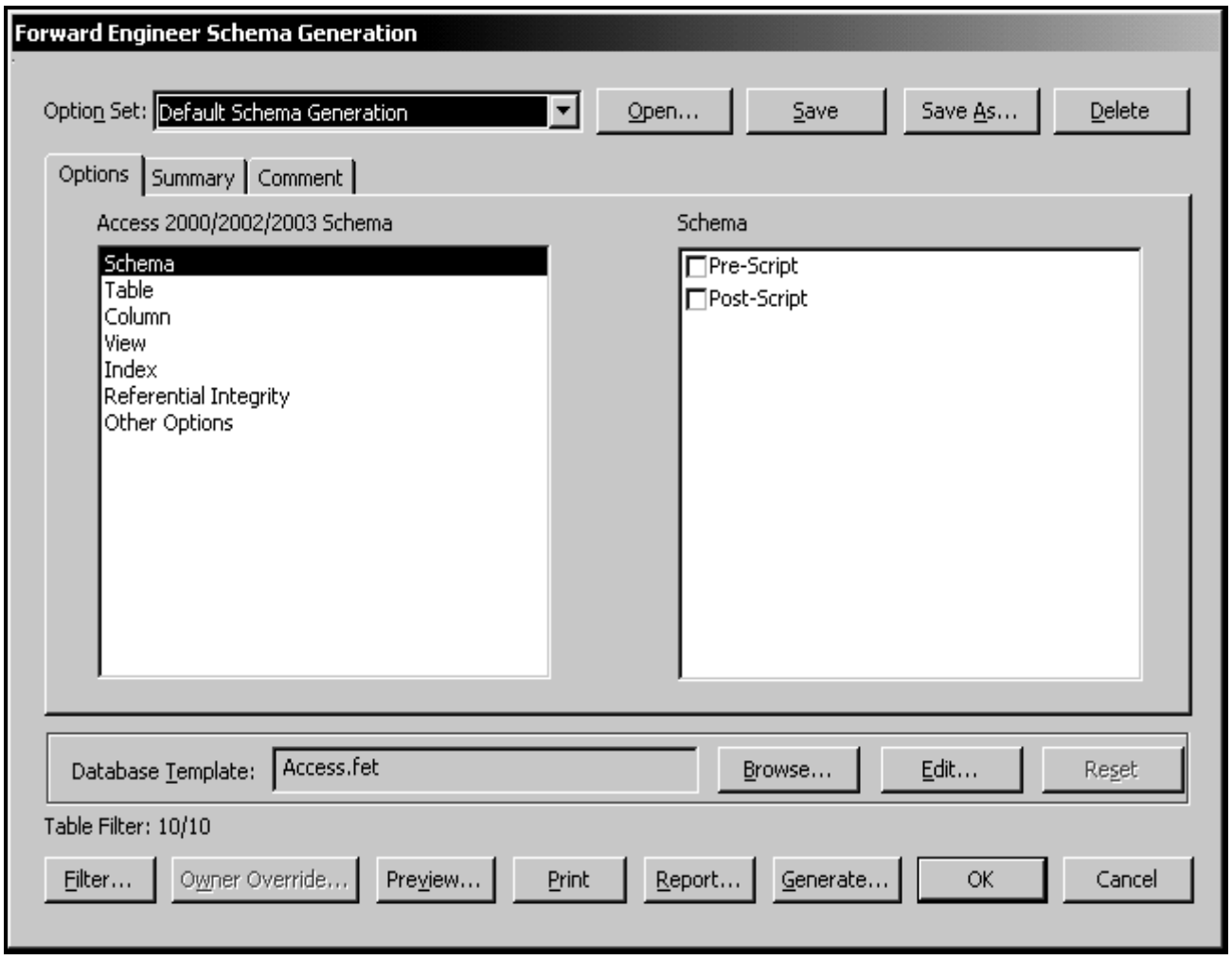


Рис. 1.76. **Вікно прямого проектування схеми бази даних**

Діалог Schema Generation має три закладки:

1. Закладка Options служить для завдання опцій генерації об'єктів БД – тригерів, таблиць, колонок, індексів тощо. Для завдання опцій генерації якого-небудь об'єкту слід вибрати об'єкт в лівому списку закладки, після чого включити відповідну опцію в правому списку.

2. В закладці Summary відображаються усі опції, задані в закладці Options. Список опцій в Summary можна редагувати так само, як і в Options.

3. Закладка Comment дозволяє внести коментар для кожного набору опцій.

Кнопка Preview викликає діалог Schema Generation Preview, в якому відображається програма мовою Visual Basic з використанням DAO, створювана ERwin для генерації системного каталогу СКБД.

Кнопка Print призначена для виводу на друк створюваного ERwin SQL-скрипта.

Кнопка Report зберігає той же скрипт в ERS або SQL текстовому файлі. Ці команди можна надалі редагувати будь-яким текстовим редактором і виконувати за допомогою відповідної утиліти сервера.

Для генерації схеми БД клацніть кнопку Generate. Ви увійдете до діалогу Connection для встановлення сеансу зв'язка з сервером і запуску SQL-скрипта (рис. 1.77).

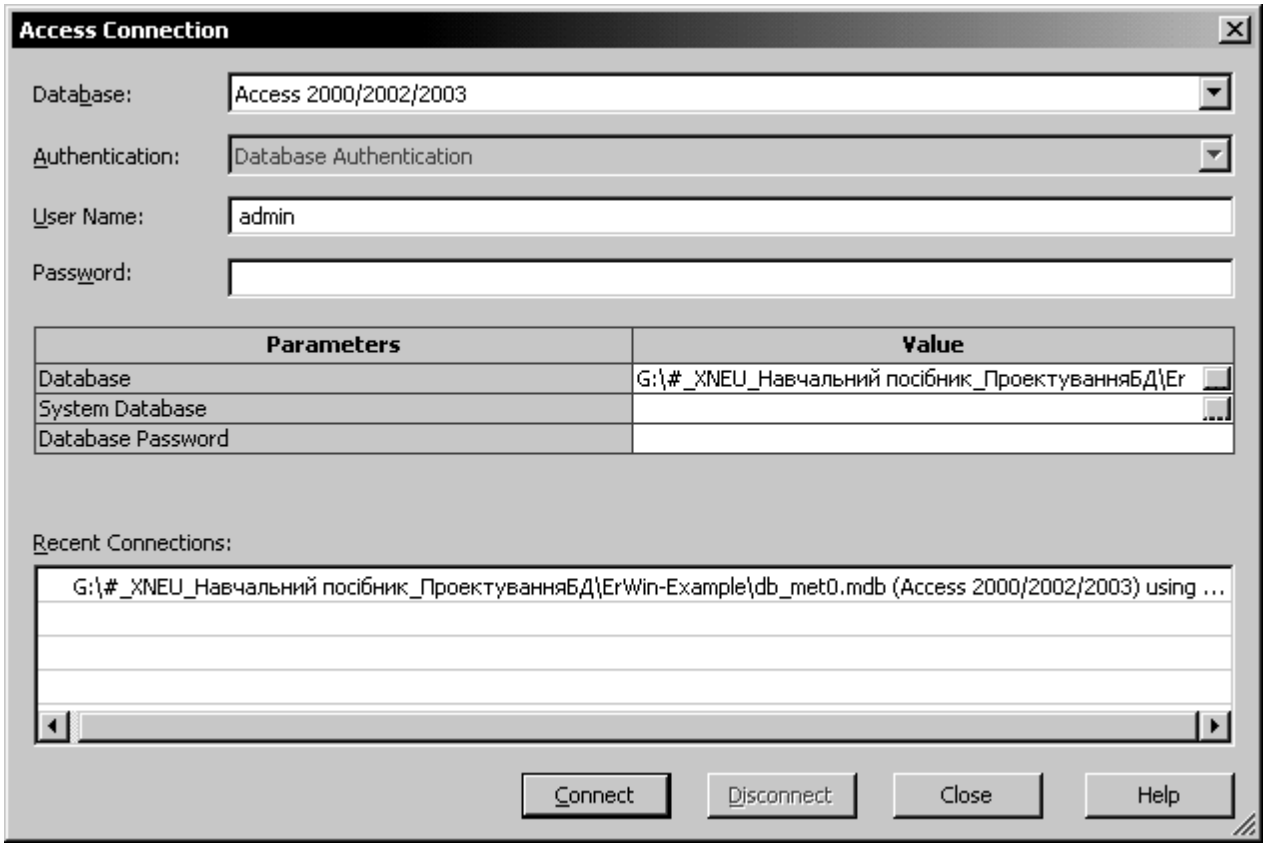


Рис. 1.77. **Вікно вибору сервера бази даних**

Для того, щоб згенерувати базу даних в Access необхідно:

1. Створити порожню базу даних в MS Access.
2. Вибрати створену базу даних на локальному диску комп'ютера у полі Database.
3. Клацнути кнопку Connect.
4. Відбувається запуск SQL-скрипта. За замовчуванням в діалозі Generate Database Schema включена опція Stop If Failure. Це означає, що при першій же помилці виконання скрипта припиняється. Клацнувши

по кнопці Continue, можна продовжити виконання. Кнопка Abort переривається виконання. При вимкненій опції Stop If Failure скрипт виконуватиметься, незважаючи на помилки, що зустрічаються.

Для виконання зворотного проектування слід вибрати пункт меню Tools -> Reverse Engineer.

При цьому виникає діалог Select Template, в якому треба вибрати шаблон діаграми, потім діалог вибору СКБД і задати опції зворотного проектування (рис. 1.78, рис. 1.79).

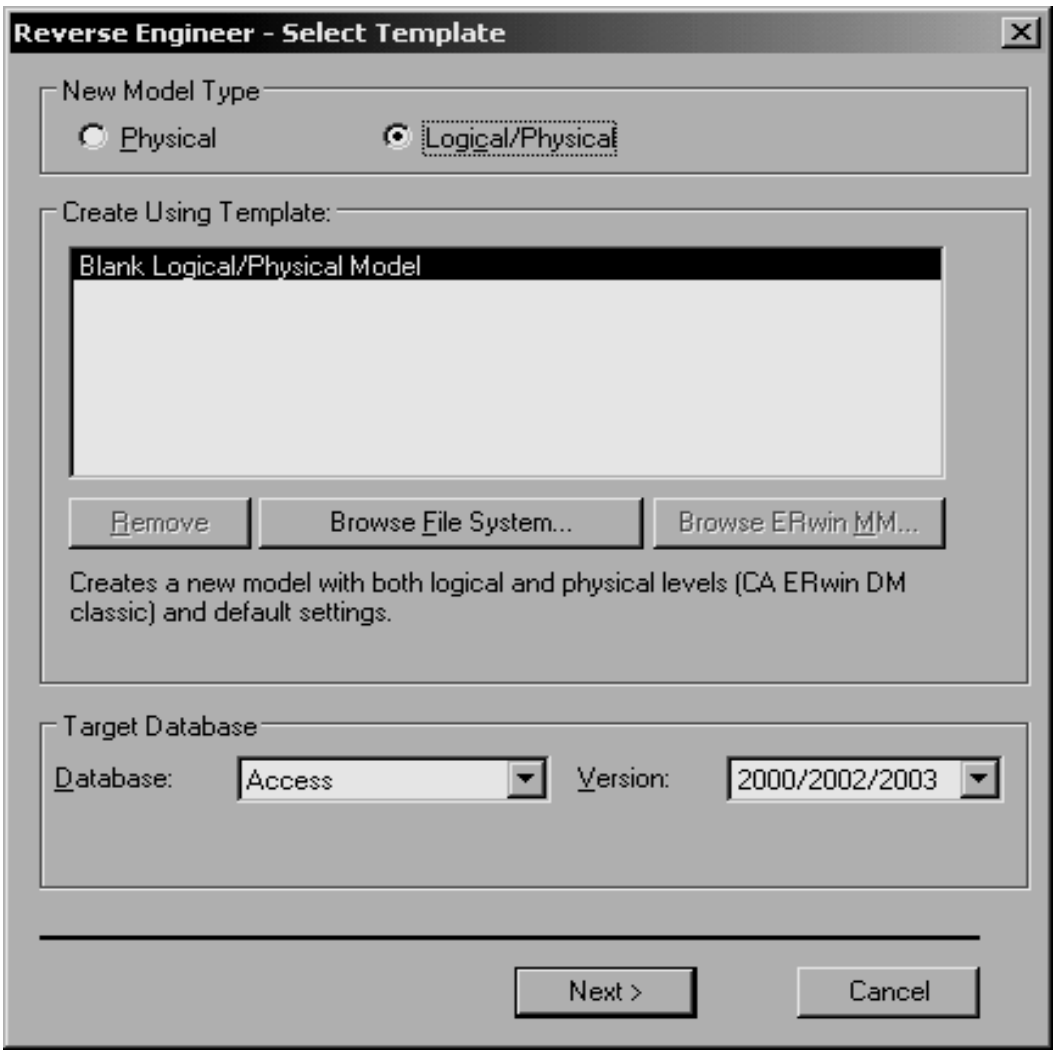


Рис. 1.78. **Вікно зворотного проектування бази даних**

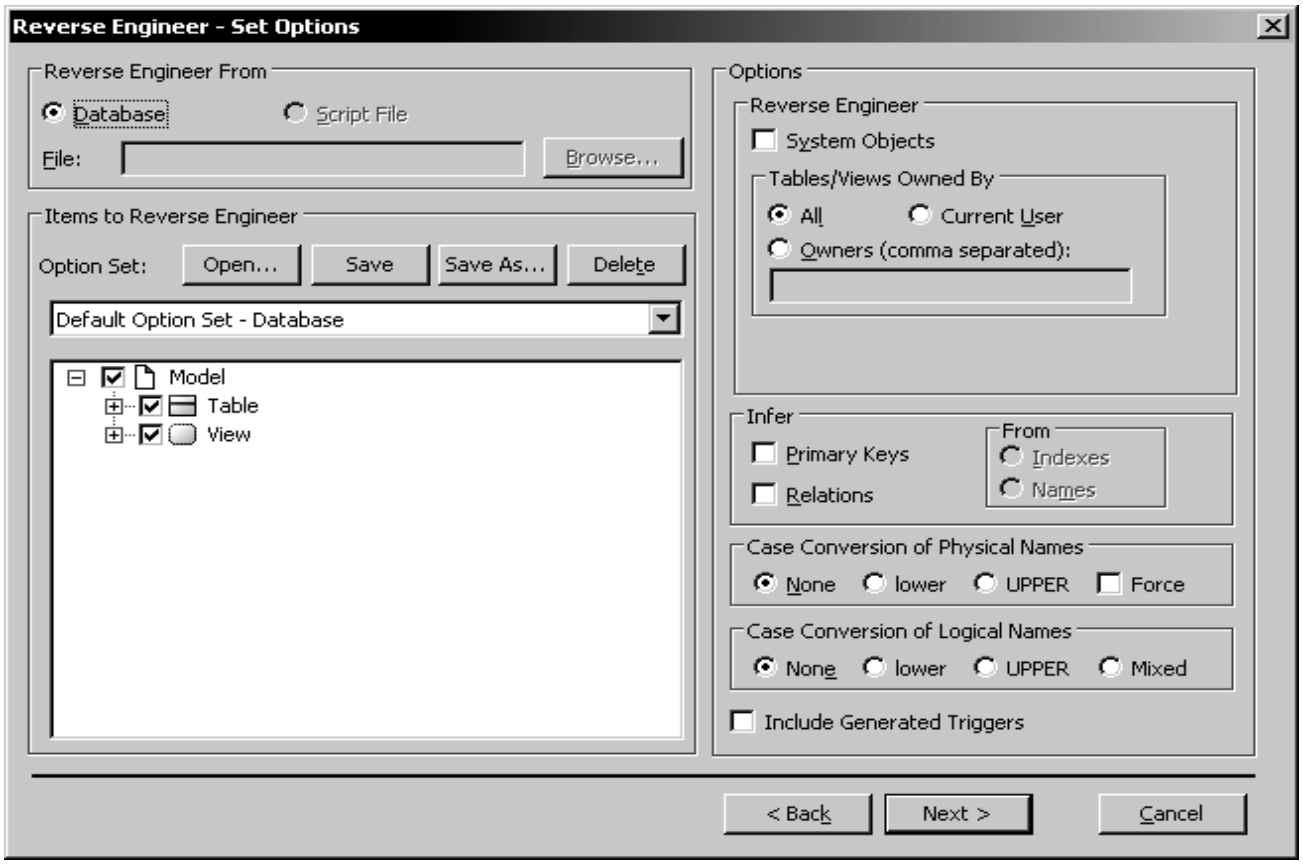


Рис. 1.79. **Вибір опцій зворотного проектування**

Далі вибираємо базу даних і виконуємо підключення (Connect) (рис. 1.80).

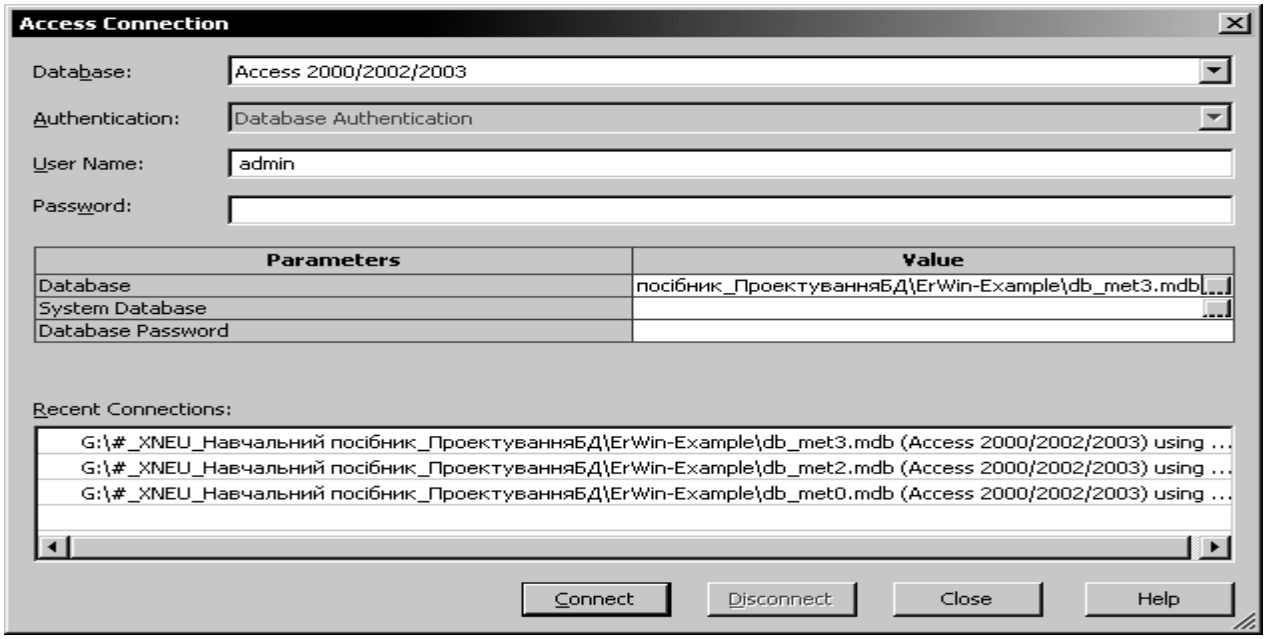


Рис. 1.80. **Підключення до бази даних при зворотному проектуванні**

В результаті отримуємо згенеровану модель.

1.5. Синхронізація системного каталогу БД і поточної моделі

У процесі роботи модель може змінюватися і доповнюватися. З іншого боку, системний каталог БД може редагуватися іншими проектувальниками. В результаті через деякий час після останнього сеансу зворотного проектування можуть виникнути розбіжності між реальним станом системного каталогу і моделлю даних.

Для синхронізації системного каталогу БД і поточної моделі слід вибрати пункт меню Tools ->Complete Compare -> Compare або клацнути кнопку  на панелі інструментів Database. Ви увійдете до діалогу Complete Compare – Set Options (рис. 1.81). Після завдання опцій синхронізації встановлюється сеанс зв'язку з сервером і у діалозі Resolve Differences показується поточний стан моделі (ліворуч) і системного каталогу СКБД (праворуч). Вбудована в AllFusion ERwin Data Modeler (ERwin) потужна технологія автоматизує повну двонаправлену синхронізацію моделі, скриптів і баз даних.

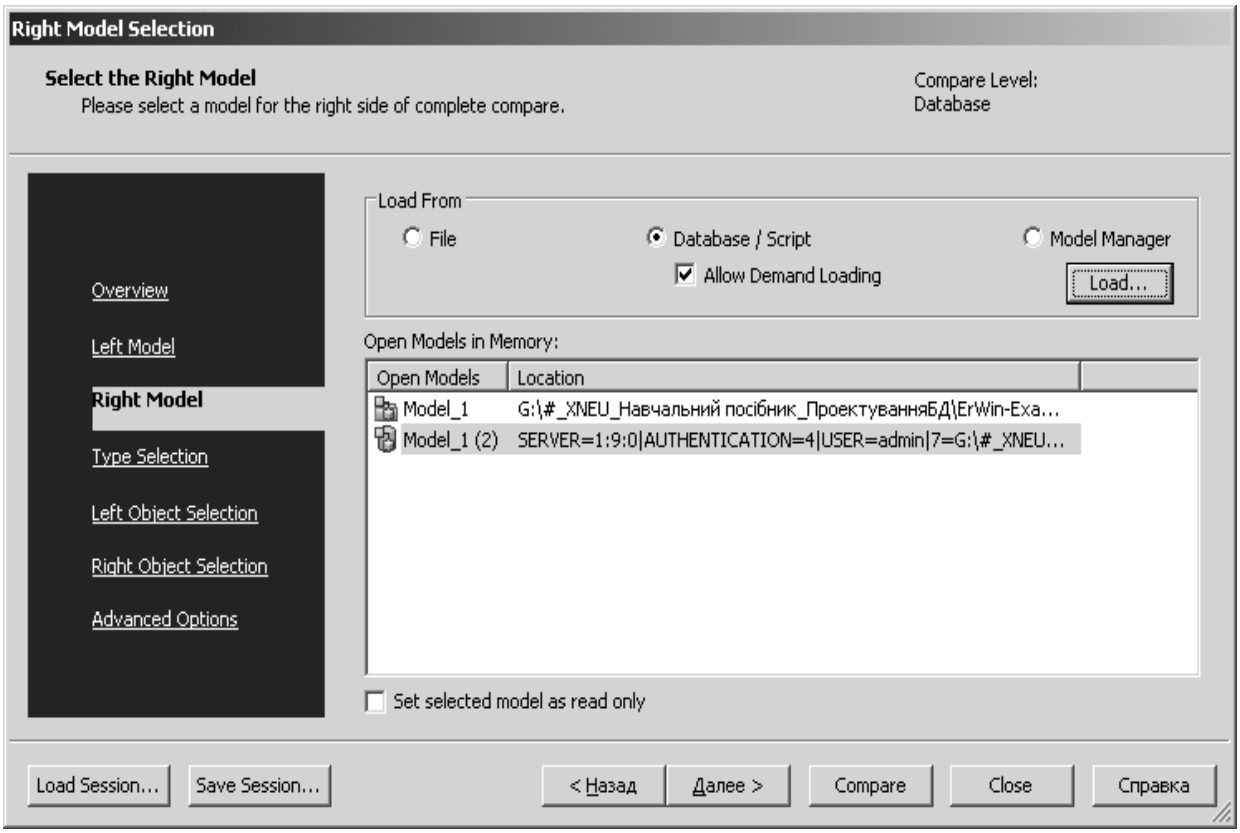


Рис. 1.81.**Вікно вибору бази даних для синхронізації**

При синхронізації, для вибраних користувачем об'єктів, відображаються відмінності, і користувачеві пропонується вибрати, які з виявлених відмінностей і в якому напрямі необхідно внести. При цьому AllFusion ERwin Data Modeler (ERwin) може автоматично згенерувати ALTER-скрипт на зміну (рис. 1.82 – рис. 1.84).

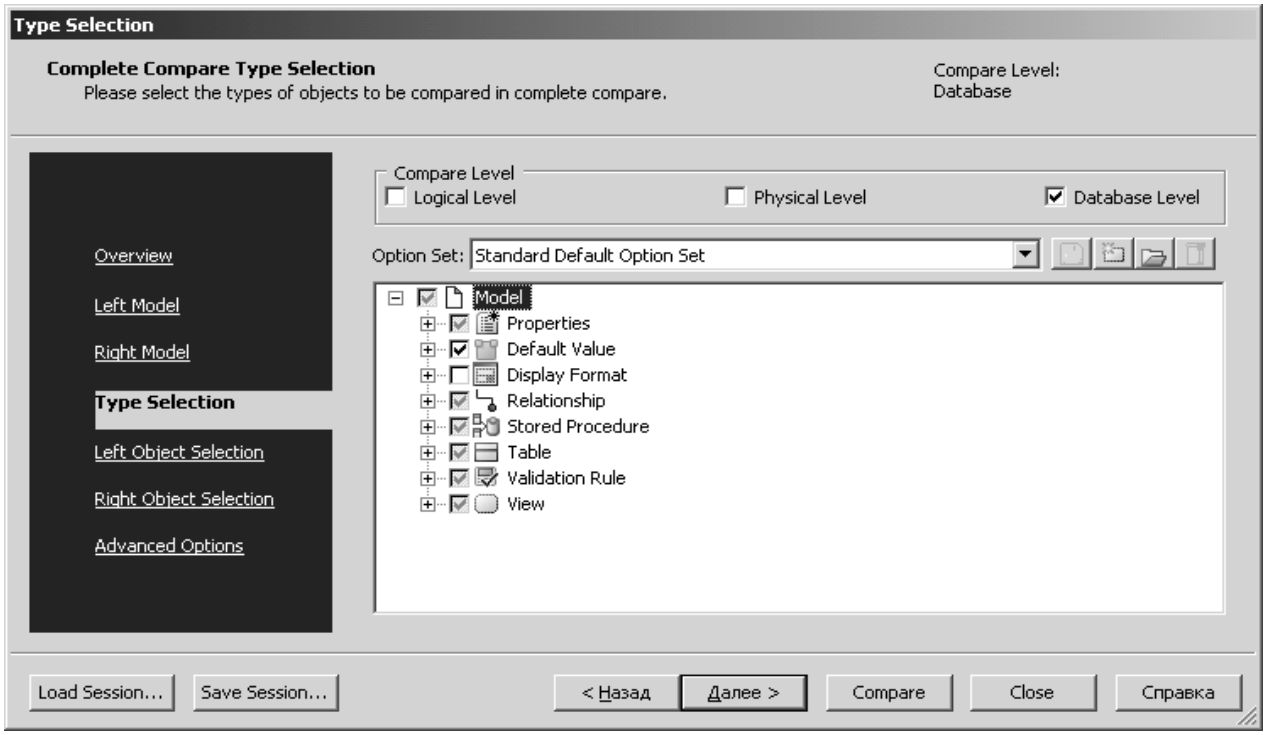


Рис. 1.82. **Вікно вибору типів відмінностей при синхронізації БД**

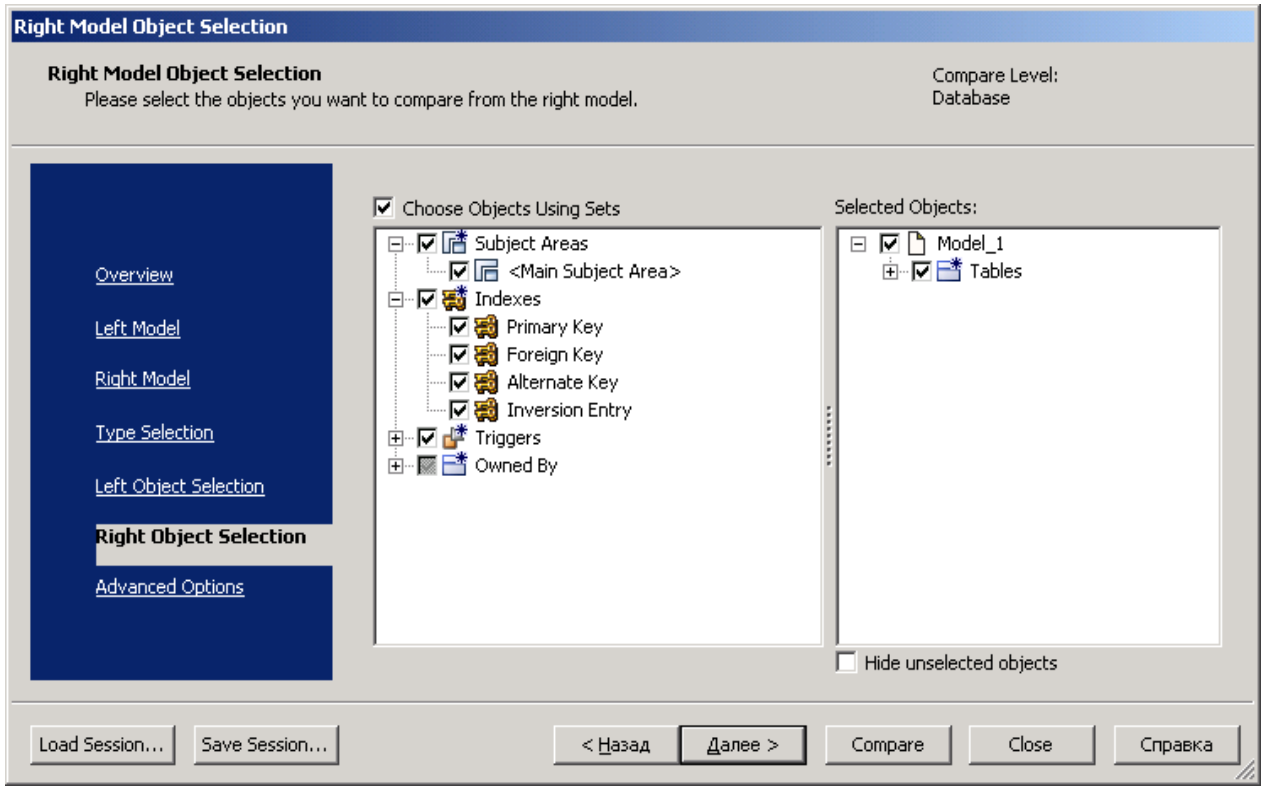


Рис. 1.83. **Вікно вибору об'єктів при синхронізації БД**

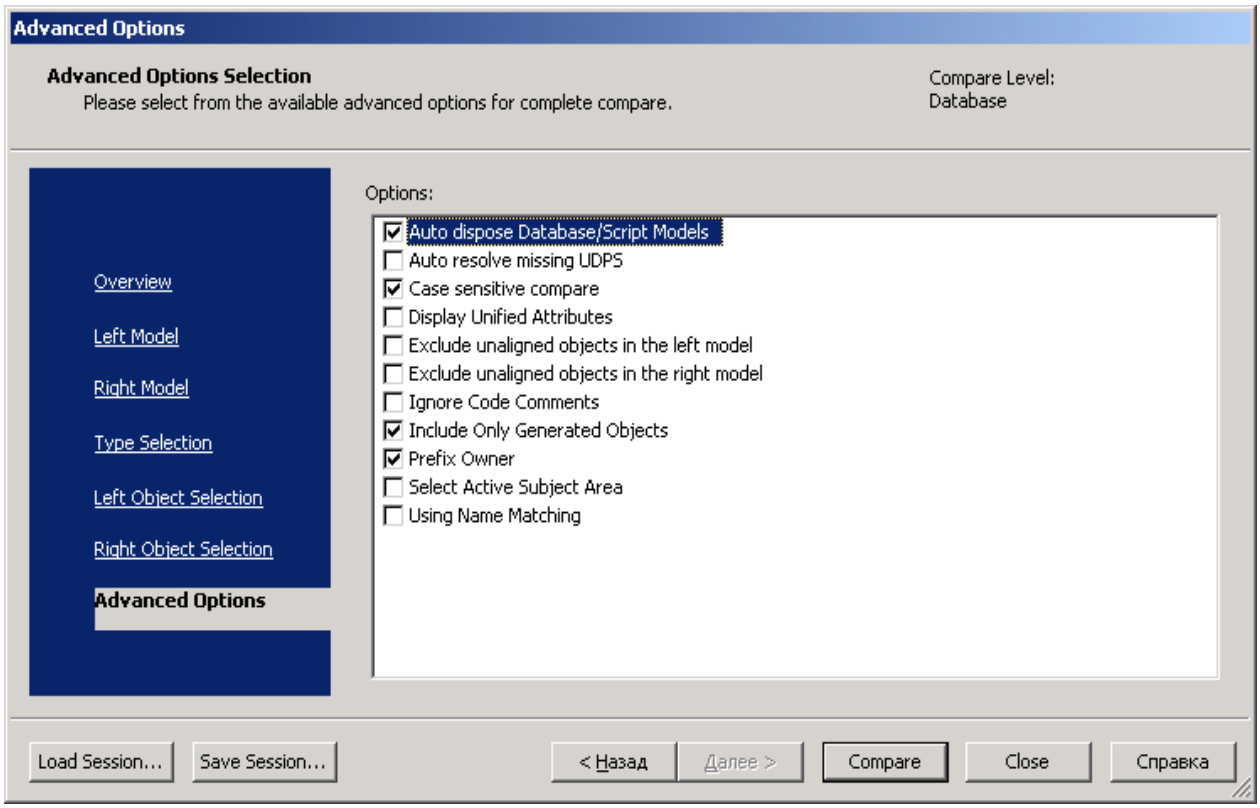


Рис. 1.84. **Вікно встановлення розширених режимів синхронізації БД**

В результаті отримаємо перелік відмінностей у поточній моделі та вибраній базі даних (рис. 1.85).

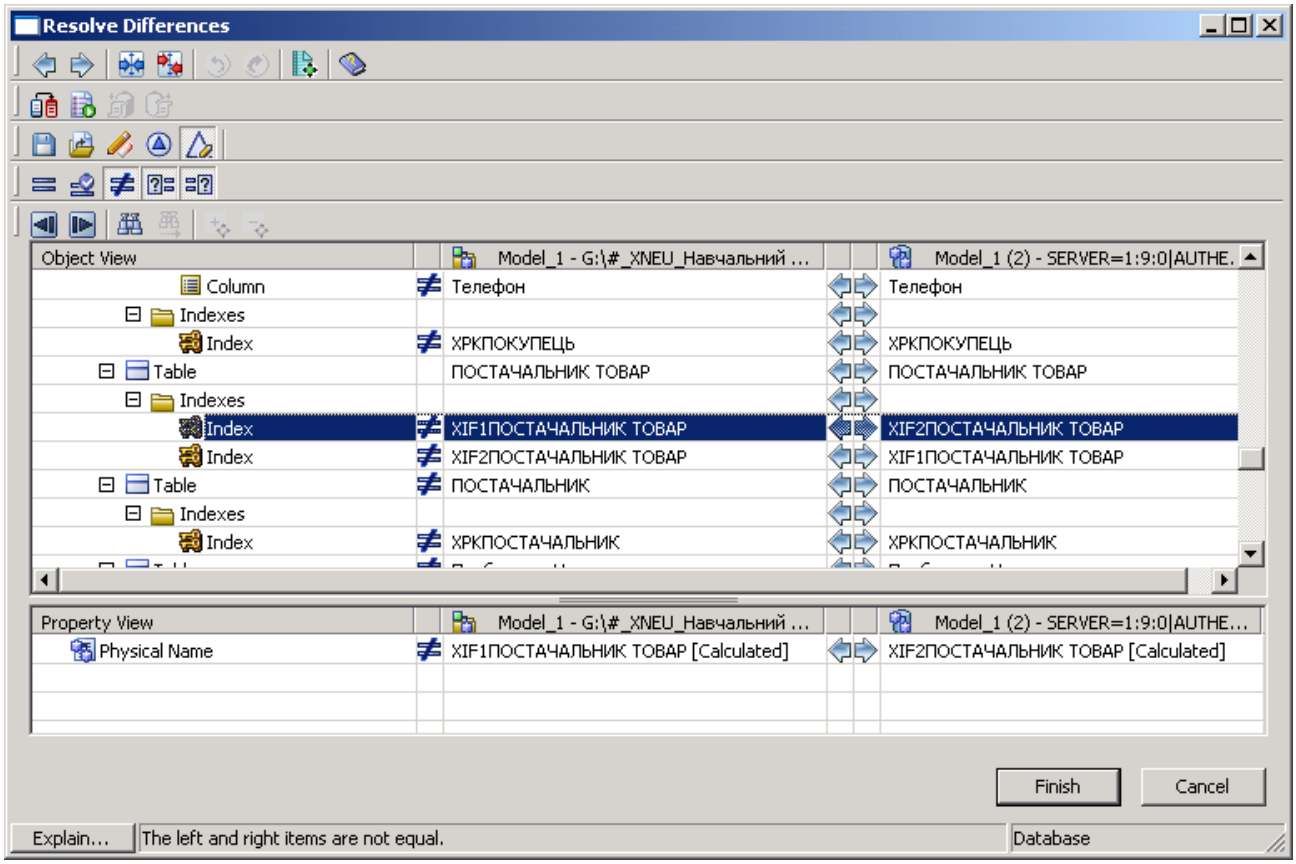


Рис. 1.85. **Вікно з переліком відмінностей моделі та бази даних**

У правій частині діалогу знаходяться кнопки, що дозволяють задати режими синхронізації для кожного об'єкту моделі і бази даних:

1)  – експорт об'єкту з моделі даних у базу даних;

2)  – імпорт об'єкту з моделі даних у базу даних;

Кнопки Match і UnMatch дозволяють створити або відмінити зв'язок моделей з різними іменами. Для того, щоб зв'язати об'єкти, клацніть кнопку Match, потім таблицю в лівому списку, потім таблицю в правому списку.

Крім того є кнопки завдання фільтрів для порівняння і генерації звіту.

1.6. Створення звітів по моделям даних

1.6.1. Створення звітів за допомогою Data Browser

При проектуванні моделі даних вам може знадобитися представити інформацію у вигляді звіту в текстовому форматі.

Для генерації звітів в ERwin є ефективний і простий у використанні інструмент – Data Browser. Він дозволяє виконувати зумовлені звіти (об'єднані по типах), зберігати результати їх виконання, створювати власні звіти, друкувати і експортувати їх в поширені формати. Кожен звіт може бути налагоджений індивідуально, дані в ньому можуть бути відсортовані і відфільтровані.

Перехід до створення звітів здійснюється при натисненні кнопки Data Browser () на панелі інструментів AllFusion ERwin Data Modeler (ERwin) або через меню Tools -> Data Browser (рис. 1.86). У старій версії ERwin це був пункт меню Report Browser.

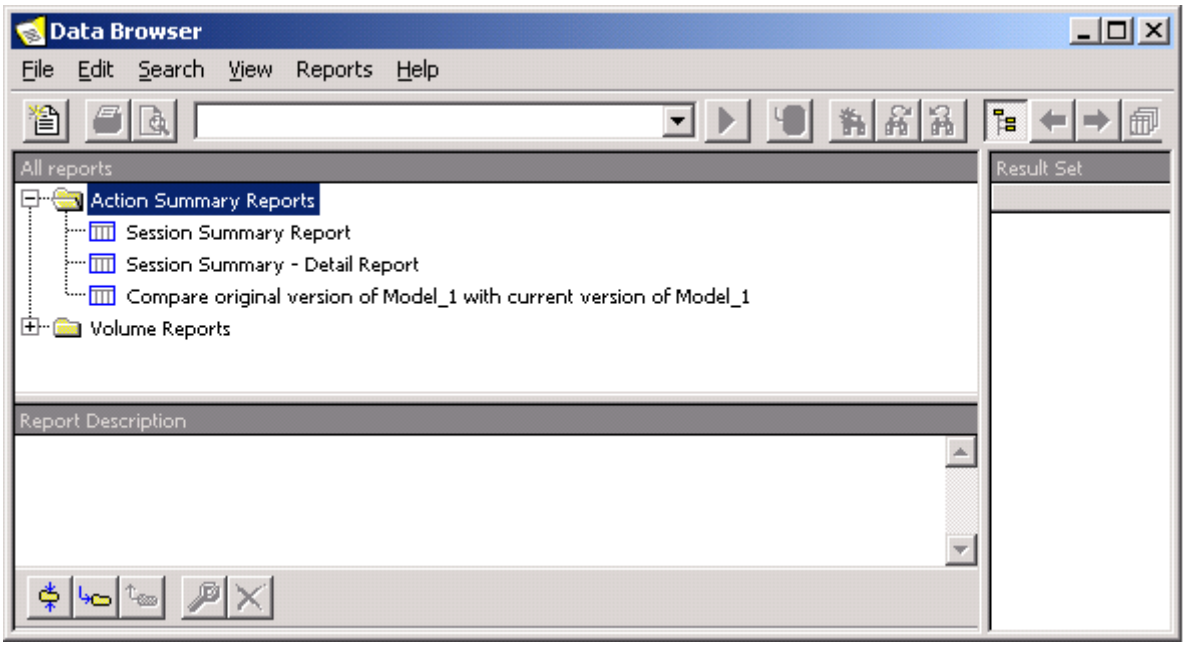


Рис. 1.86. **Вид вікна Data Browser**

Вікно Data Browser має власне меню і панель інструментів (табл. 1.2).

Таблиця 1.2

Кнопки панелі інструментів Data Browser

Кнопки	***Призначення кнопок***
	Створення нового звіту або папки
	Друк звіту
	Перегляд результатів виконання звіту
	Виконання звіту
	Фіксація змін (для редагованого звіту)
	Пошук елементів звіту: завдання умов пошуку, пошук наступного рядка і пошук іншого звіту, що відповідає рядку
	Включення і виключення дерева звітів
	Показати список виконаних звітів у хронологічному порядку
	Перейти до попереднього звіту (при створенні нового звіту на основі рядка існуючого)
	Вибір колонок і сортування виконаного звіту
	Асоціювання рядка звіту з іконкою
	Збереження виконаного звіту у вигляді подання

У верхній лівій частині діалогу розташовано вікно, що відображає дерево звітів. Звіти можуть бути згруповані в папки. Кожен звіт може включати декілька результуючих наборів даних, кожне з яких генерується при черговому виконанні звіту.

Кожен елемент дерева помічений іконкою:

1)  – папка;

2)  – звіт;

3)  – редагований звіт;

4)  – результуючий набір даних;

5)  – подання.

За замовчуванням Data Browser містить заздалегідь визначені звіти, що дозволяють наочно уявити інформацію про основні об'єкти моделі даних, як логічної, так і фізичної. Для виконання звіту досить двічі клацнути по ньому у дереві звітів або клацнути по відповідній кнопці на панелі інструментів. Результат виконання звіту буде відображений в правому вікні діалогу Data Browser. Іконка результуючого набору буде також додана у дерево звітів.

У лівому нижньому вікні Data Browser відображається коментар до звіту (заповнюється при редагуванні властивостей звіту). У нижній частині діалогу міститься додаткова панель інструментів для управління деревом звітів:

Таблиця 1.3

Кнопки додаткової панелі Data Browser

Кнопки	***Призначення кнопок***
	Редагувати виділений звіт
	Видалення звіту
	Показати тільки верхній рівень дерева
	Зробити вибрану папку коренем дерева (показати тільки вибрану гілку дерева)
	Зробити коренем дерева батьківську папку (по відношенню до вибраної)

1.6.2. Створення нового звіту

Для створення нового звіту слід вибрати пункт меню File ->New Report або клацнути кнопку (New Report or Folder – Новий звіт або Папка) на панелі інструментів і вибрати пункт Report. У результаті з'явиться діалог Reports (рис. 1.87).

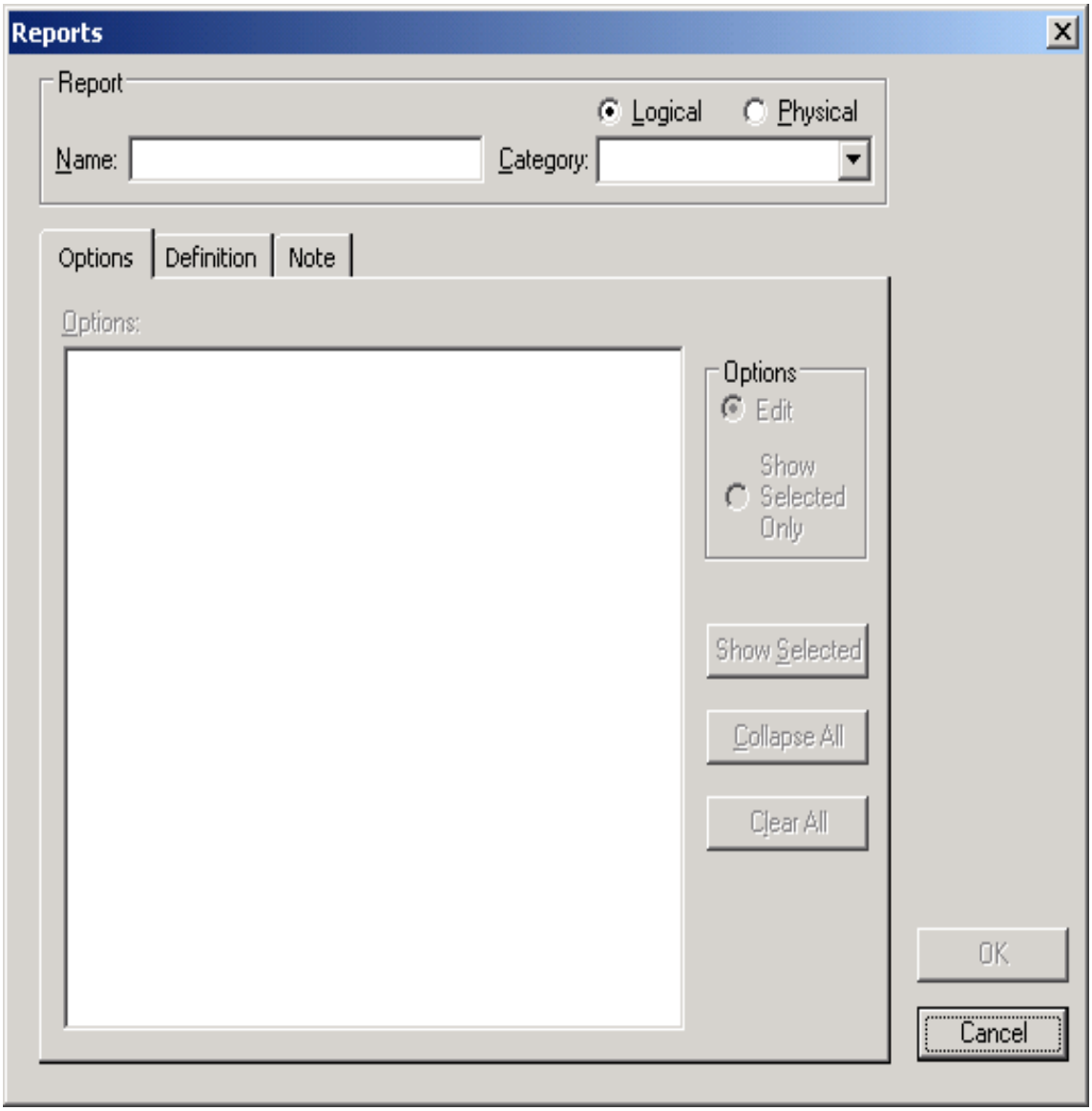


Рис. 1.87. **Вікно створення нового звіту**

У полі Name слід внести ім'я звіту. Категорія звіту (Category) вказує на тип об'єктів моделі, за якими створюватиметься звіт (атрибути, сутності, домени, зв'язки тощо) (рис. 1.88).

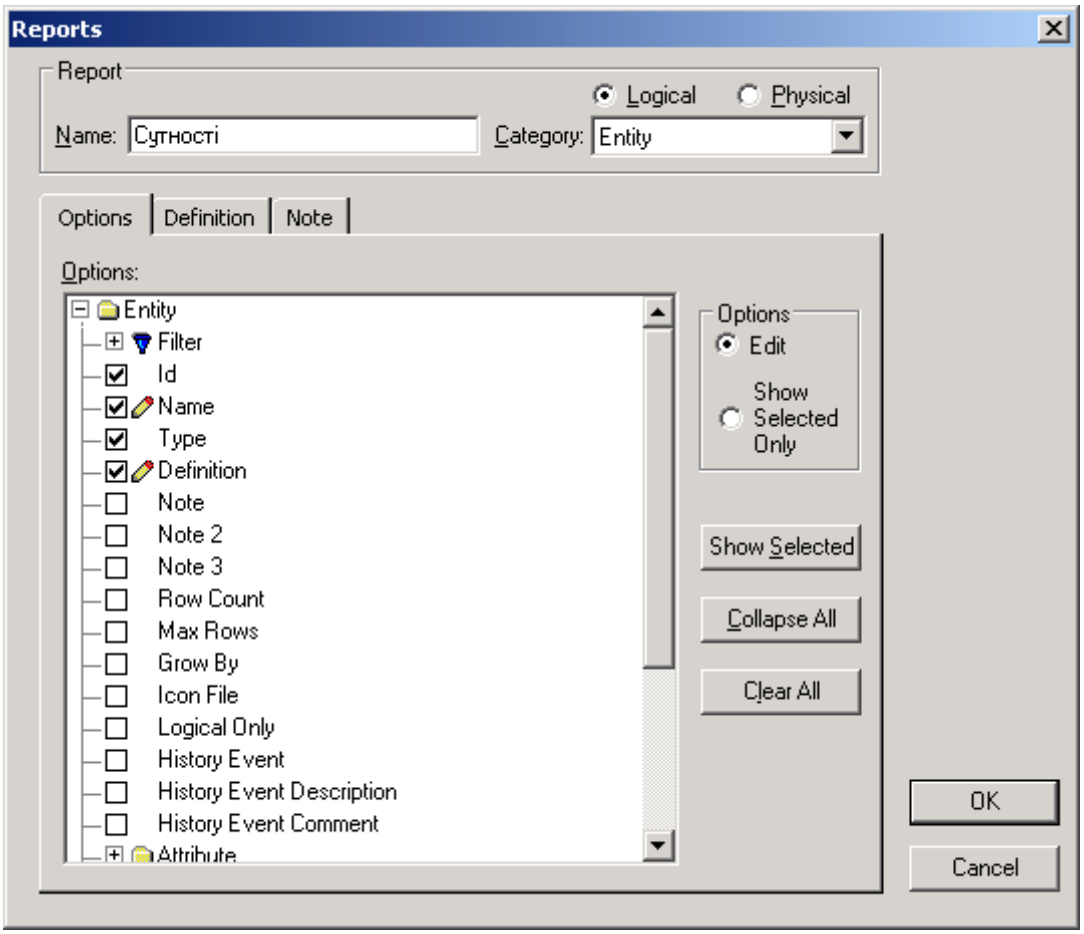


Рис. 1.88. **Вибір об'єктів моделі для звіту**

Закладки Definition і Note служать відповідно для внесення визначення і коментаря до звіту.

Закладка Options відображає інформацію, яка буде включена у звіт. У лівій частині закладки знаходиться ієрархічний список категорій. Папки в цьому списку можуть розкриватися і згортатися. Вікно вибору дозволяє включити відповідний пункт списку в звіт. Значок показує, що відповідну колонку в отриманому звіті можна буде редагувати. Папка з символом  дозволяє вибрати умови фільтрації даних звіту, а з символом  – умови сортування.

Окрім списку, закладка містить такі елементи управління:

група Options – дозволяє вибрати режим відображення елементів у списку – показувати усі можливі або тільки вибрані;

Collapse All – згортає усі теки списку;

Clear All – відміняє усі заздалегідь вибрані опції;

Show Selected – розкриває теки з вибраними опціями.

Після клацання по кнопці ОК, звіт буде доданий в список звітів діалогу Report Browser (рис. 1.89).

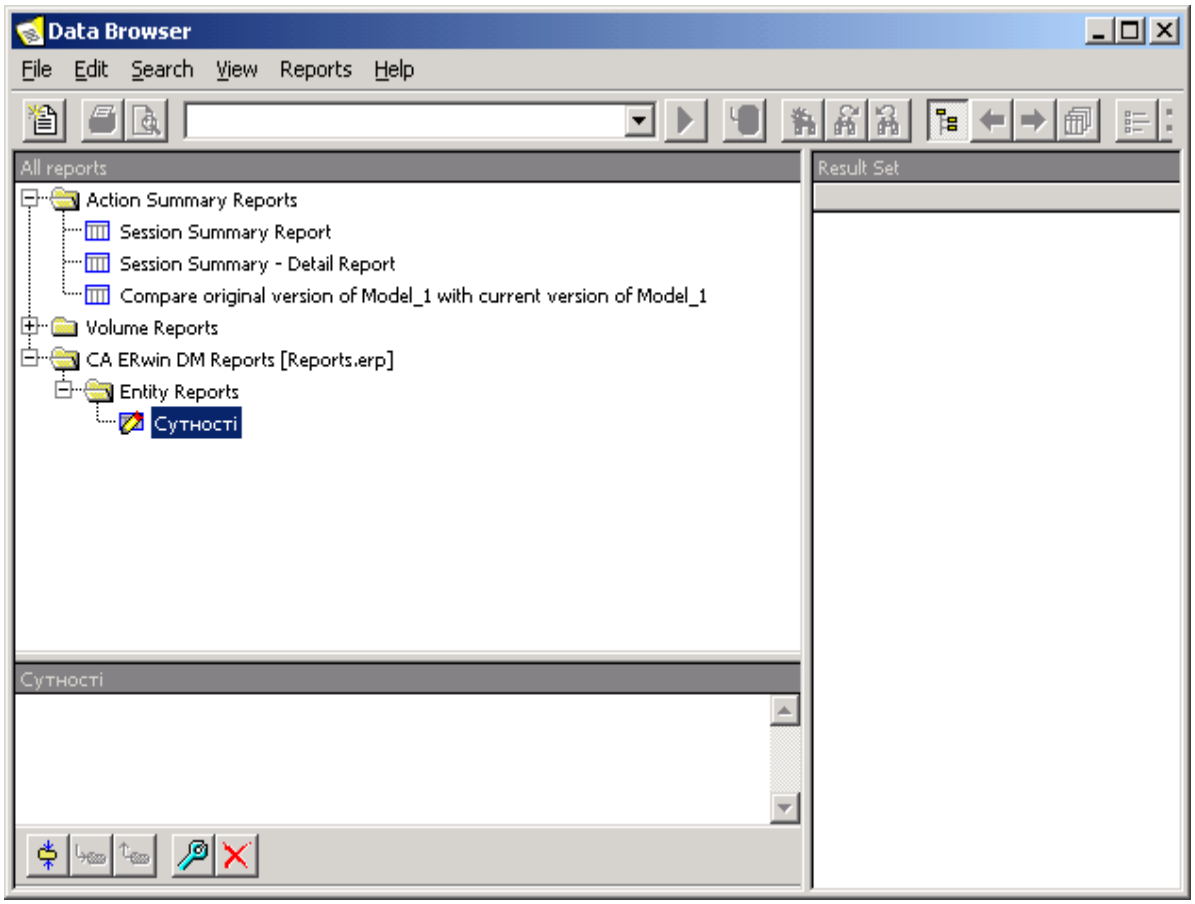


Рис. 1.89. **Результат створення нового звіту**

Для виконання звіту треба або двічі клацнути по його імені у списку, або клацнути по кнопці в палітрі інструментів.

1.6.3. Форматування звіту

Існуючий звіт, у тому числі зумовлений, теж можна змінити за допомогою редактора, якщо в списку клацнути правою кнопкою миші по імені звіту і вибрати в контекстному меню пункт Edit ERwin Report.

Отриманий після виконання звіту результуючий набір даних можна відформатувати, роздрукувати, експортувати або зберегти у вигляді подання.

Для форматування результуючого набору даних слід у списку клацнути правою кнопкою миші по імені набору і вибрати в контекстному меню пункт Edit report format і далі, у діалозі Report Format, можна змінити сортування даних, черговість колонок, зробити колонку невидимою, задати її стиль (рис. 1.90).

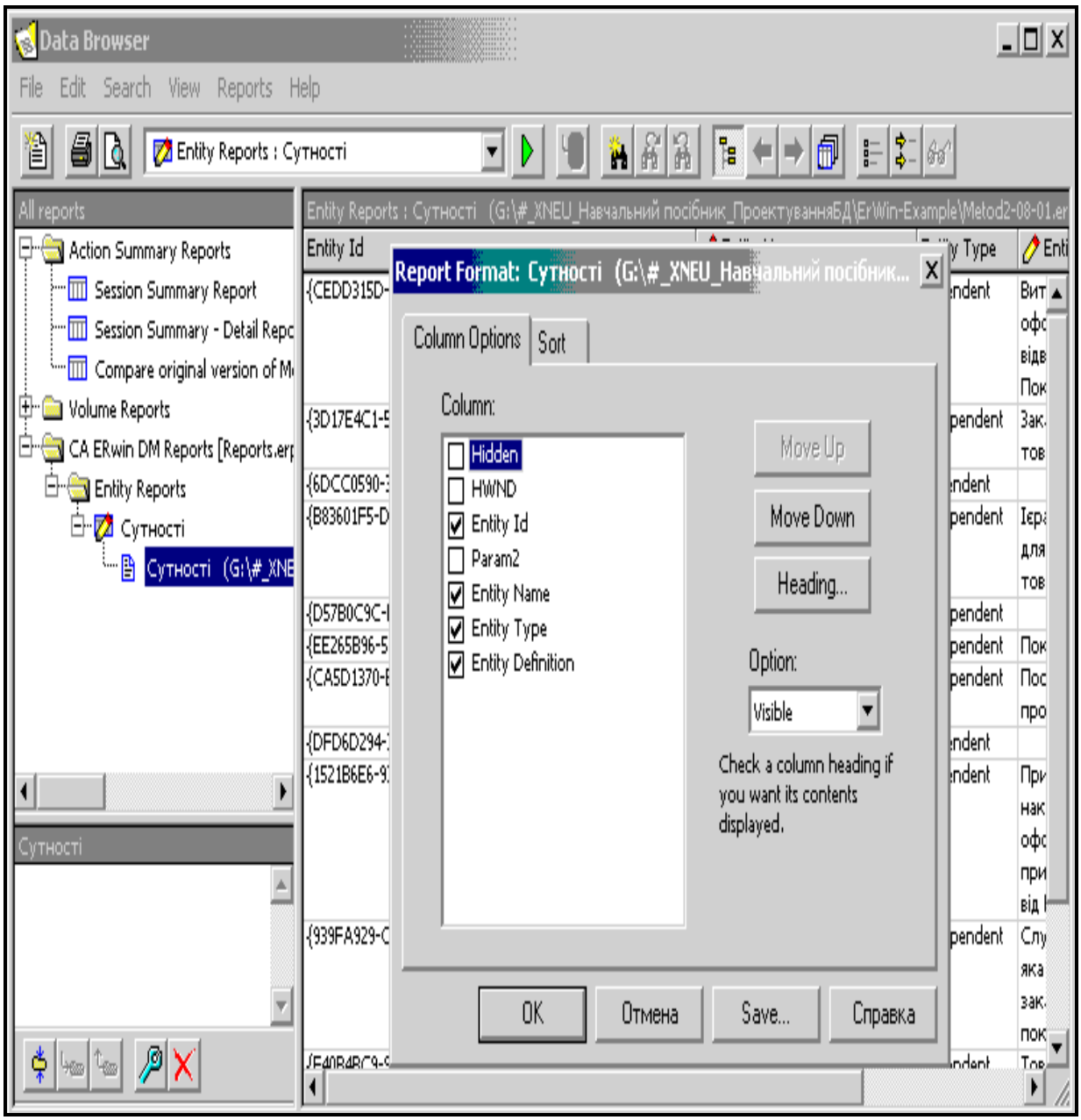


Рис. 1.90. **Вікно форматування звіту**

1.6.4. Експорт звіту

Для експорту результуючого набору даних слід в списку клацнути правою кнопкою миші по імені набору і вибрати у контекстному меню пункт Export result set. Допустимі формати експорту:

1) CSV – текстовий файл;
2) HTLM;
3) DDE – експорт результатів у застосування які підтримують Dynamic Data Exchange (DDE), такі як MS Word або MS Excel;
4) RPTwin – експорт у спеціалізований генератор звітів.

При експорті можна вибрати такі опції (рис. 1.91):

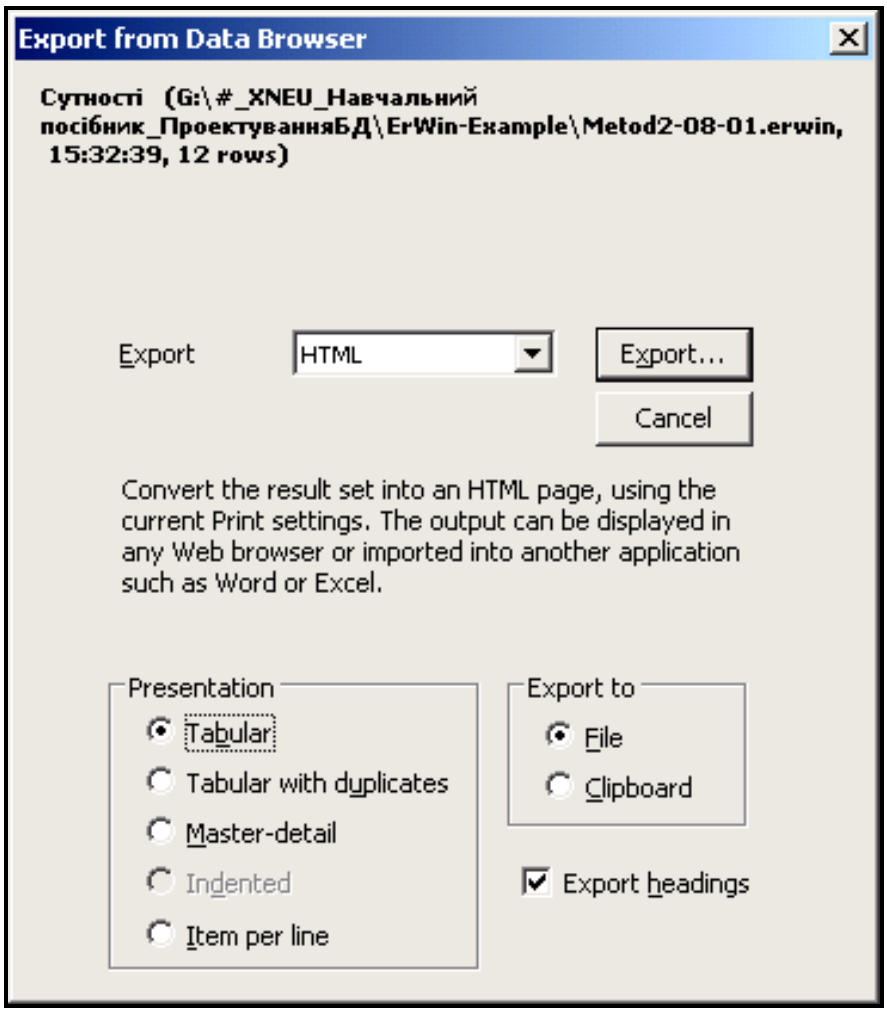


Рис. 1.91. **Вікно вибору опцій звіту**

Tabular – експортуються дані в тому ж форматі, в якому вони відображаються на екрані (рис. 1.92).

Tabular with duplicates – експортує результати, заповнюючи усі елементи таблиці, включаючи ті, які дублюють попередні.

Master-detail – експортує результати у форматі, який використовує комбінацію заголовків і таблиць. Наприклад, для кожної сутності в цьому звіті буде створена своя таблиця атрибутів.

Якщо система не в змозі створити звіт, то видається попередження "No structure detected in report <Назва>".

Indented – експортує результати у вигляді ієрархічного списку.

Item per line – експортує результати, розміщуючи кожен елемент на окремому рядку.

Результати генерації звіту можна експортувати у файл або у буфер обміну.

Сутності

Entity Id	Entity Name	Entity Type	Entity Definition
{CEDD315D-E709-4DE3-B24E-ADDFFF7E7EB7}+00000000	ВитратнаНакладна	Dependent	Витратна накладна оформлюється при відвантаженні товару Покупцю
{3D17E4C1-53ED-4064-9990-716E75103D36}+00000000	ЗАМОВЛЕННЯ	Independent	Заказ на покупку товарів від покупця
{6DCC0590-30E3-489C-90F8-37ED0376DD01}+00000000	ЗАМОВЛЕННЯ ТОВАР	Dependent	
{B83601F5-DDAC-4A2F-8426-C8A3F8B0EA8A}+00000000	КАТАЛОГ	Independent	Ієрархічний каталог для розміщення товарів
{D57B0C9C-BF3A-4115-BCA8-E3FCAE2B5BA9}+00000000	НАКЛАДНА		
{EE265B96-51B6-4CDF-85AF-9BA60A569F19}+00000000	ПОКУПЕЦЬ		Покупець товару
{CA5D1370-E202-43F0-89F0-976E503C7A22}+00000000	ПОСТАЧАЛЬНИК		Постачальник продукції
{DFD6D294-7303-4CE5-ADC9-F09BD9586F76}+00000000	ПОСТАЧАЛЬНИК ТОВАР	Dependent	
{1521B6E6-9397-4033-B5C8-C5CD7ABABA73}+00000000	ПрибутковаНакладна		Прибуткова накладна оформлюється при прийомі товару від Постачальника
{939FA929-C1AC-48DC-8A3D-D40DCD736188}+00000000	СЛУЖБА ДОСТАВКИ	Independent	Служба доставки яка доставляє, згідно заказів, товари покупцям
{F40B4BC9-9B65-4394-B3B7-4B6BB79DCFE8}+00000000	ТОВАР	Dependent	Товар, що продається організацією
{58064B6C-A91B-4990-BABB-EDAA4662A4F2}+00000000	ТОВАР ПрибутковаНакладна		

Рис. 1.92. **Загальний вид звіту у форматі Tabular**

1.6.5. Перевірка моделі на помилки

Особливий інтерес представляє звіт, що дозволяє виявити можливі помилки в створеній моделі. В цьому випадку слід сформувати звіт на базі категорії Model Validation (рис. 1.93).

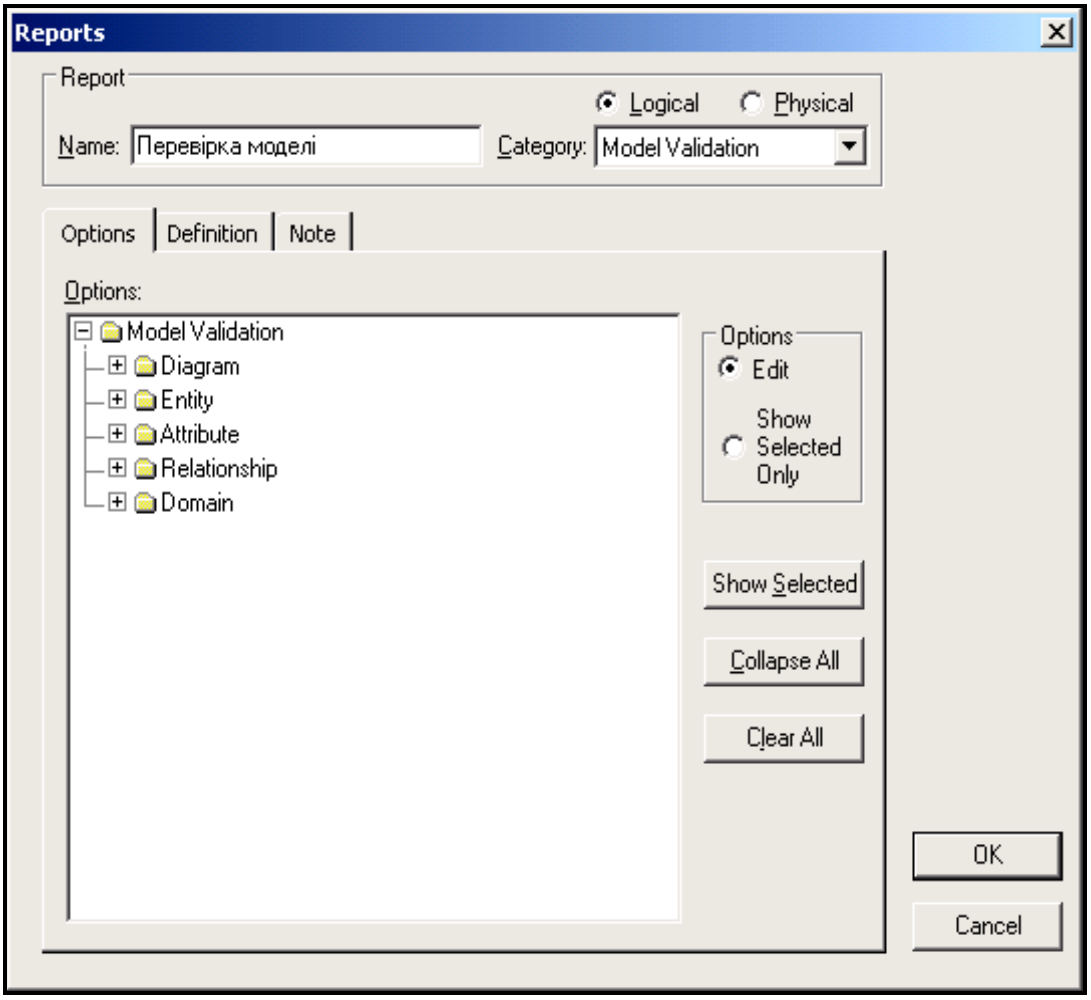


Рис. 1.93. **Формування звіту перевірки моделі**

У вікні Option – встановити (відмітити) ті елементи моделі, які бажано проконтролювати і потім згенерувати звіт.

Перевірку фізичної моделі можна здійснити й іншим способом. Для цього слід натиснути кнопку Validate SQL of current model (рис. 1.94).

В результаті аналізу буде згенерований звіт, в якому вказується, що помилки не виявлені, або перераховуються знайдені помилки (рис. 1.95).

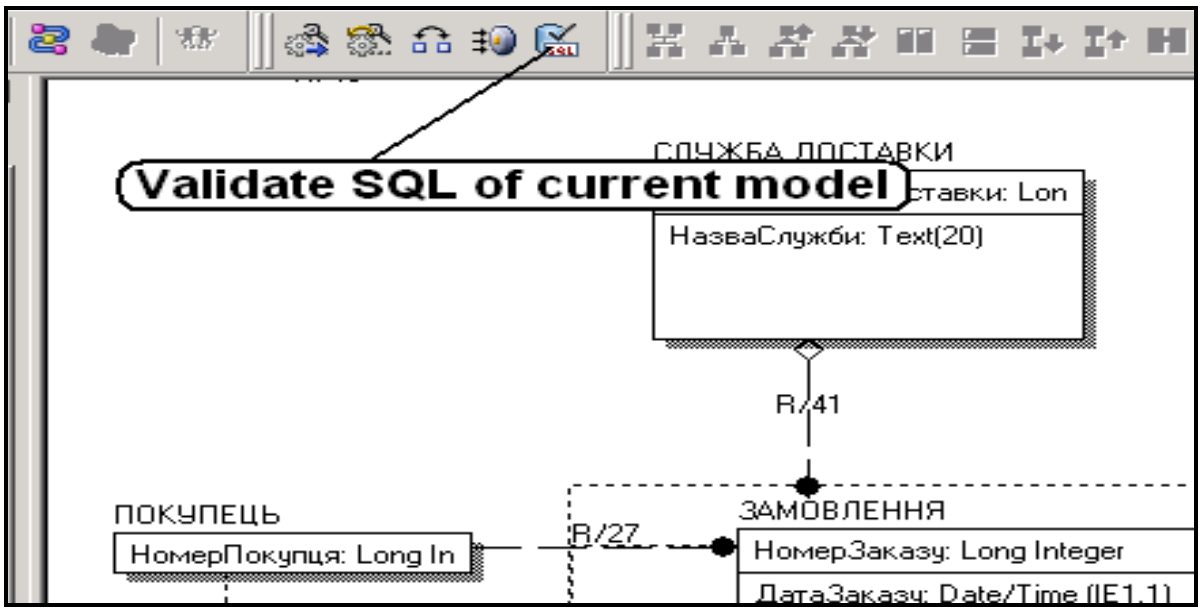


Рис. 1.94. **Вибір перевірки Validate SQL of current model**

Рис. 1.95. **Вид звіту з виявленими помилками у моделі**

Звіт про результати перевірки можна проглянути на екрані або записати його як RTF-файл. Список помилок, що виявляються, залежить від вибраної СКБД. Найбільш розвинена діагностика для ORACLE.

1.6.6. Створення подання для звіту

Після форматування і налаштування результуючого набору даних його можна зберегти в якості іменованого подання що полегшує використання звітів, оскільки усі налаштування досить зробити один раз. Кожен звіт може мати декілька подань. Для створення подання слід встановити фокус у списку на потрібний набір і клацнути по кнопці на панелі інструментів.

У діалозі Save View слід вказати ім'я і визначення подання. Після клацання по кнопці ОК подання додасться у список звітів (рис. 1.96, рис. 1.97).

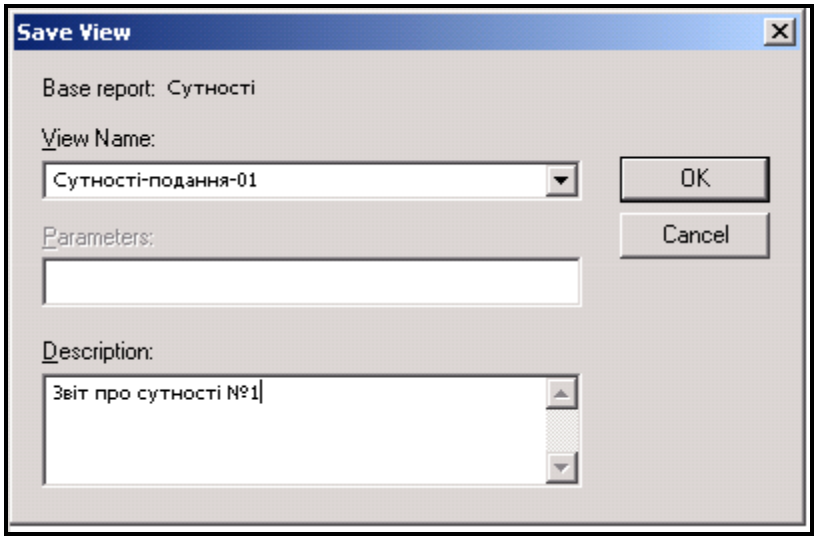


Рис. 1.96. **Вікно збереження подання до звітів**

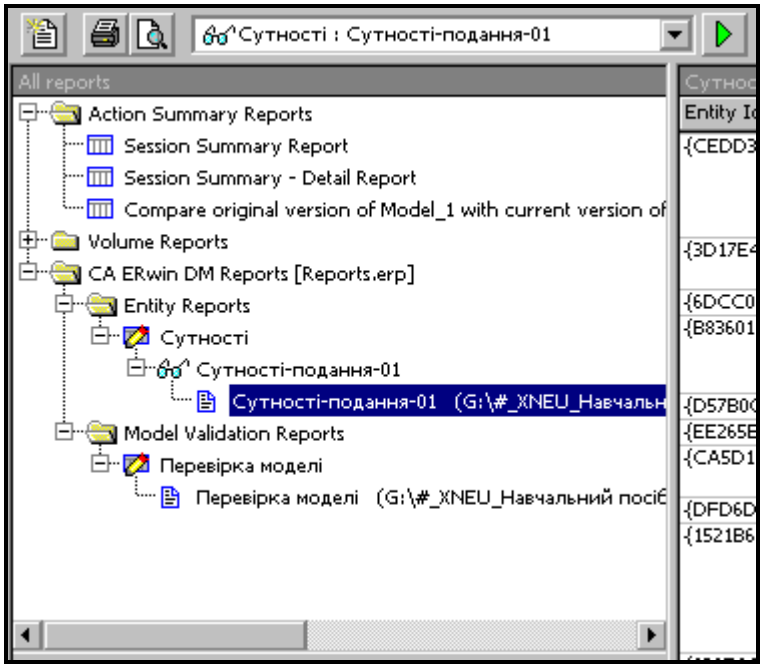


Рис. 1.97. **Вікно з доданим новим звітом**

1.6.7. Створення звітів за допомогою Report Template Builder

Report Template Builder – ще один інструмент генерації звітів, основною перевагою якого є можливість публікації результатів моделювання на Web-сайті.

Діалог Report Templates викликається за допомогою значка  на панелі інструментів або пункту меню Tools ->Report Template Builder (рис. 1.98).

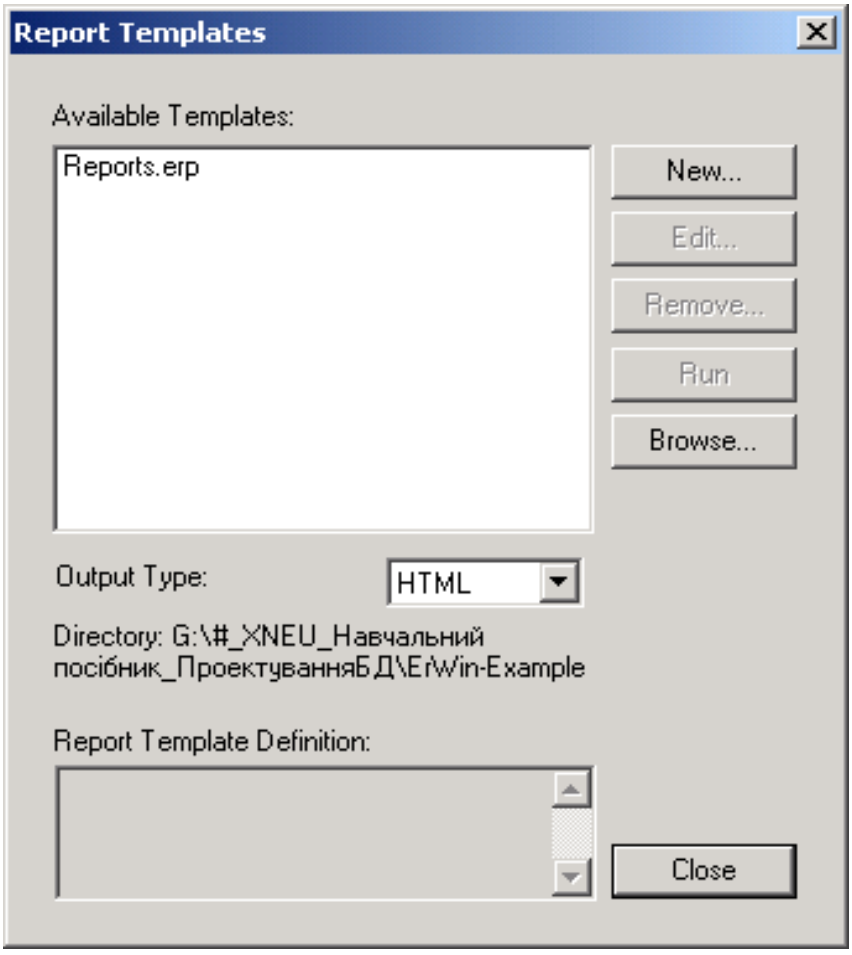


Рис. 1.98. **Вид вікна Report Templates**

Кнопка New служить для створення нового шаблону, кнопка Edit – для редагування існуючого. Список вибору Output Type дозволяє задати формат результату звіту, що буде сгенерований. Кнопка Run дозволяє виконати звіт.

Клацання кнопок New або Edit викликає діалог Report Template Builder (рис. 1.99).

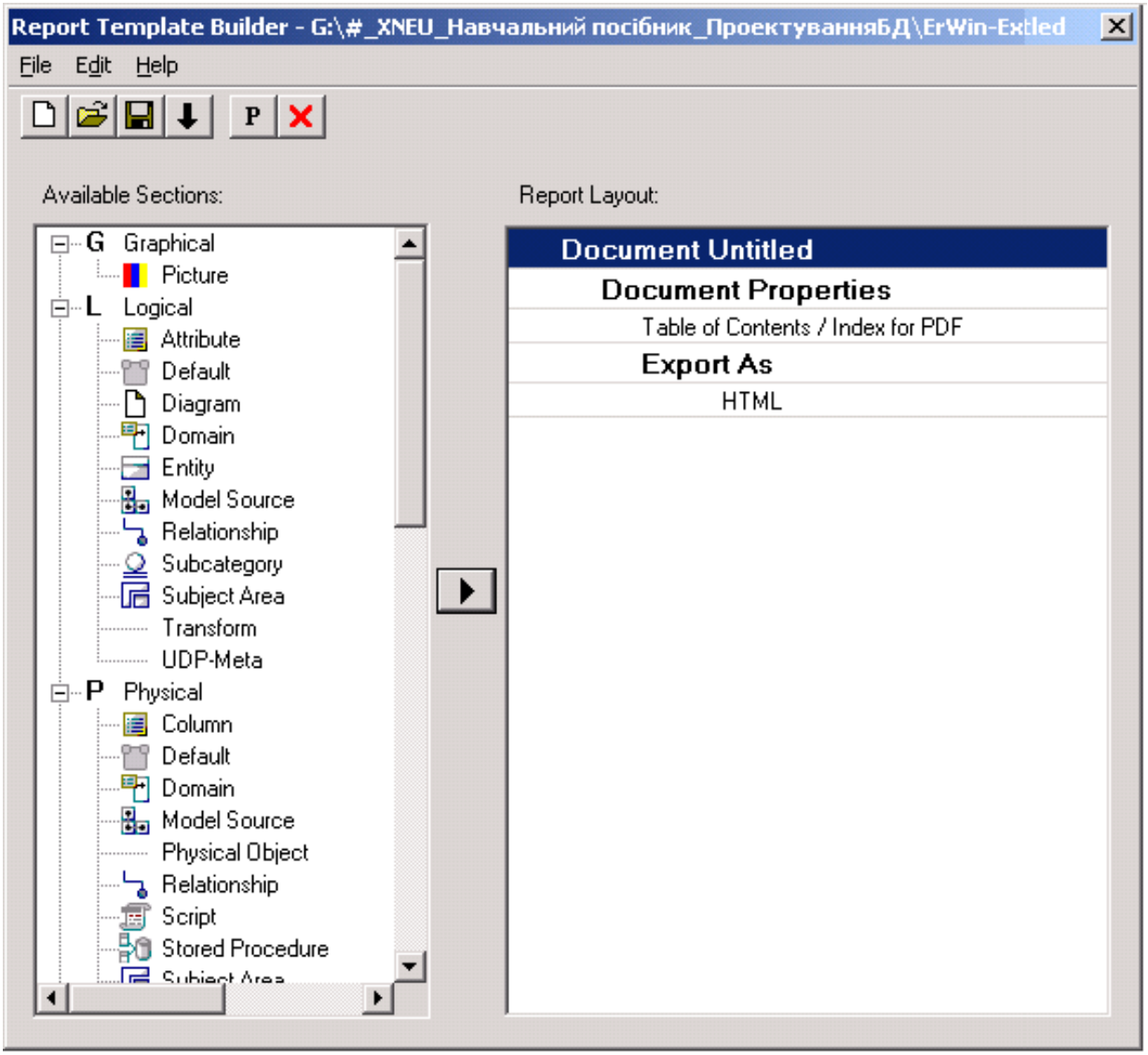


Рис. 1.99. **Вид вікна Report Template Builder**

Діалог Report Template Builder містить два списки і панель інструментів. У частині ліворуч відбивається список можливих секцій звіту, що відповідають типам об'єктів моделі, праворуч – список секцій, включених до звіту.

За замовчуванням до звіту включається тільки ім'я об'єкту.

Для включення інших властивостей необхідно за допомогою меню Edit ->Properties або кнопки P на панелі інструментів викликати діалог Properties для відповідного елементу звіту (рис. 1.100).

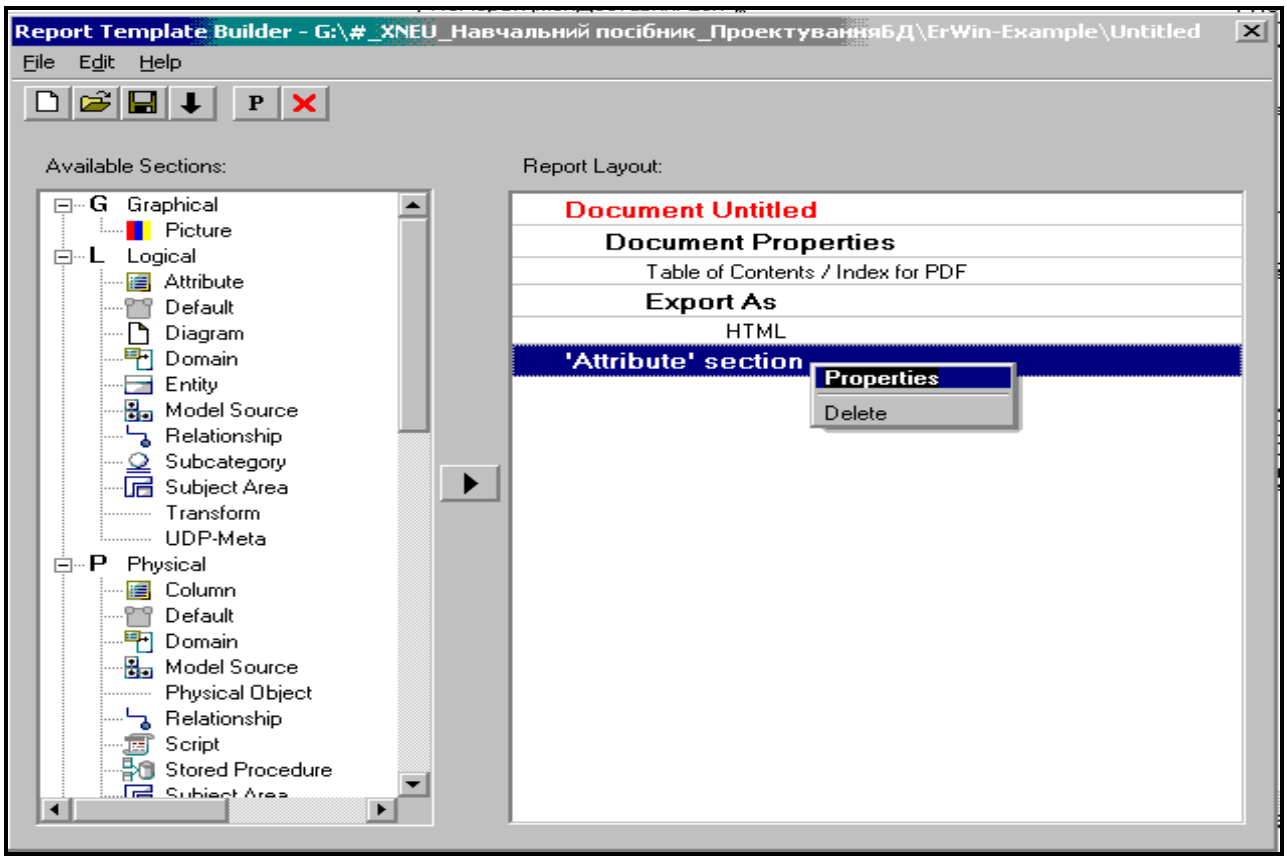


Рис. 1.100. **Виклик діалогового вікна Properties**

Вкладки Properties дозволяють встановити різноманітні опції, що впливають на результати звіту та його оформлення (рис. 1.101).

Для того, щоб додати додаткові властивості для існуючого шаблону необхідно:

1) клацнути мишею "Report Template Builder" або "Report Builder" у меню Tools. Відкриється діалог "Report Templates";

2) вибрати шаблон звіту до якого необхідно внести додаткові властивості. Відкриється діалог "Report Template Builder";

3) вибрати потрібну секцію у "Report Layout" і клацнути мишею "Properties" у меню Edit;

4) вибрати необхідні властивості, розкриваючи у разі потреби дерево "Property Tree". Після закінчення, необхідно закрити вікно і повернутися у діалог "Report Template Builder";

5) вибрати Save у меню Файл;

6) клацнути Run у меню File, в результаті чого буде згенерований файл з відповідним звітом і відкриється програма, яка асоційована з вибраним типом файлу.

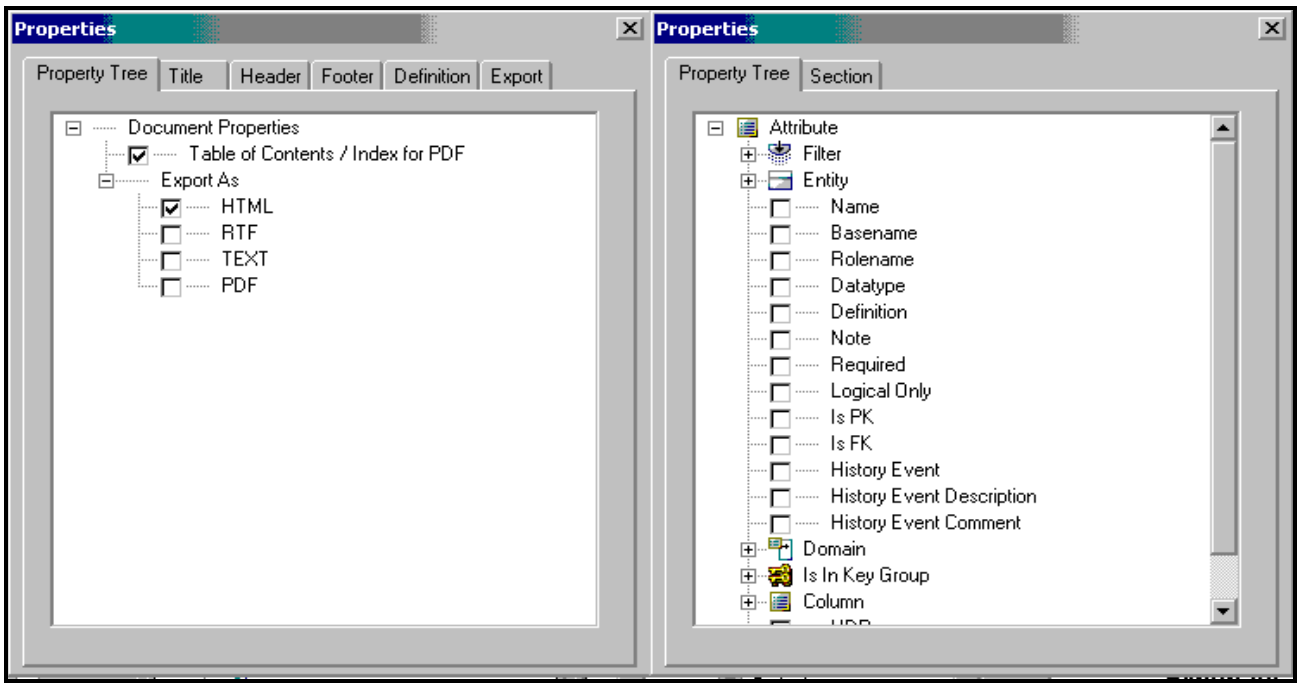


Рис. 1.101. **Вікна Properties заголовка документу і секції атрибутів**

Для генерації звіту клацніть кнопку  .

Питання для самодіагностики

1. Назвіть рівні методології IDEF1X.
2. Із яких моделей складається логічний рівень?
3. Із яких моделей складається фізичний рівень?
4. Що включає діаграма сутність-зв'язок?
5. Сформулюйте вимоги, в яких необхідно переконатися перед початком проектування БД.
6. Що називається моделлю СКБД?
7. Перерахуйте переваги від використання CASE-засоба ERWin.
8. Як викликати діалогове вікно Report Browser?
9. Які кнопки панелі інструментів дозволяють змінити рівень огляду моделі?
10. Як згенерувати схему БД?
11. Яким чином здійснюється вибір сервера для генерації схеми БД?
12. Як додати сутність на діаграму?
13. Як додати категорію у сутність?
14. Назвіть види зв'язків.
15. Як переміщати атрибути усередині сутності?

16. Як додати текст на діаграму?
17. За допомогою якої кнопки на панелі інструментів перемикаються області моделі?
18. Назвіть основні частини ER-діаграми.
19. Мета ER-діаграми.
20. Що є основним компонентом реляційних БД?
21. Що називається сутністю?
22. Сформулюйте принцип іменування сутностей.
23. Що показує взаємозв'язок між сутностями?
24. Назвіть типи логічних взаємозв'язків.
25. Яким чином відображаються логічні взаємозв'язки?
26. Опишіть механізм перевірки адекватності логічної моделі.
27. Що називається первинним ключем?
28. Назвіть принципи, згідно з якими формується первинний ключ.
29. Що називається альтернативним ключем?
30. Що називається інверсійним входом?
31. У якому випадку утворюються зовнішні ключі?
32. Яка мета створення фізичної моделі?
33. Як здійснюється перетворення зв'язків "багато-до-багатьох"?
34. Яке призначення інструменту Report Browser?
35. Назвіть основні елементи вікна Report Browser.
36. Як створити новий звіт?
37. Як зв'язати звіт з іконкою?
38. Як виконати існуючий звіт?
39. Що таке подання звіту і для чого воно призначене?
40. Як зберегти звіт у вигляді подання?
41. Які категорії звітів є присутніми в Report Browser за замовчуванням?
42. Як вибрати умови фільтрації даних звіту?
43. У які формати можна експортувати звіт?
44. Як відредагувати звіт?
45. Що називається результуючим набором?
46. Який тип звіту дозволяє перевірити відсутність помилок у моделі?
47. Опишіть механізм пошуку помилок в моделі за допомогою звітів.

2. Реляційна модель даних

Цілі – ознайомитися з реляційними об'єктами даних та їх властивостями; освоїти поняття цілісності бази даних, насамперед посилальної цілісності; більш детально розібратися з поняттями потенціального, первинного, альтернативного та зовнішнього ключів, їх призначенням та розбіжностями між ними; ознайомитися з правилами зовнішніх ключів та посилальної цілісності; вивчити правила використання Null-значень.

Компетенції, що формуються:

Загальнонаукові – базові знання науково-методичних основ і стандартів в області інформаційних технологій.

Спеціалізовано-професійні – знання основних підходів , методів і технологій створення реляційних моделей баз даних; знання сучасних теорій організації баз даних; знання теоретичних і практичних основ методології та технології моделювання реляційних баз даних.

Основні питання

Реляційні об'єкти даних – розглядаються базові поняття, що є складовими частинами сучасної реляційної моделі даних та їх взаємозв'язок, а саме: відношення, кортеж, атрибут, домен, ключі тощо.

Цілісність реляційних даних – відштовхуючись від основних елементів реляційної моделі ілюструються правила підтримки посилальної цілісності моделі та використання Null-значень.

2.1. Реляційні об'єкти даних

У загальному сенсі модель даних це деяка абстракція, яка призначена для визначення правил структурування даних, процесів динамічної зміни даних та допустимих станів взаємопов'язаних даних.

У рамках реляційної моделі можна виділити три основні складові частини [4; 6]:

1. Структурна частина описує об'єкти, які розглядаються реляційною моделлю. Слід зазначити, що єдиною структурою даних, використовуваною в реляційній моделі, є нормалізовані n-арні відношення.

2. Цілісна частина описує обмеження спеціального виду, які повинні виконуватися для будь-яких відношень у будь-яких реляційних базах даних. Це цілісність сутностей і цілісність зовнішніх ключів.

3. Маніпуляційна частина описує два еквівалентні способи маніпулювання реляційними даними – реляційну алгебру і реляційне числення.

Існує спеціальна термінологія, прийнята в теорії реляційних БД (рис. 2.1)

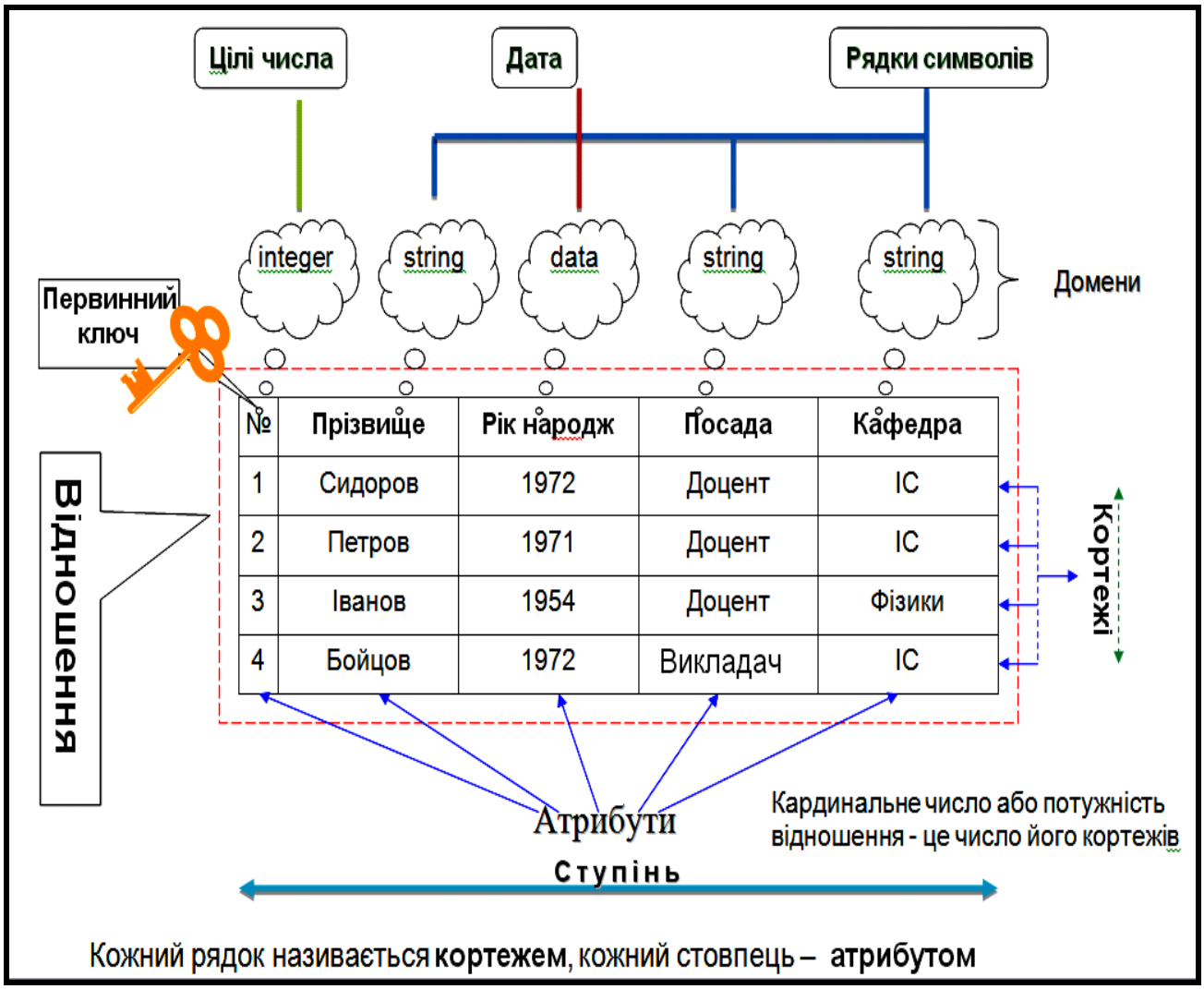


Рис. 2.1. **Реляційні об'єкти даних**

Відношенням називається уся таблиця, що відповідає певним властивостям (про які детальніше буде розказано нижче).

Відношення характеризується такими поняттями:

Атрибут відповідає стовпцю цієї таблиці, а саме – властивостям об'єктів, відомості про яких зберігаються в ній. У конкретних СКБД атрибути часто називають полями.

Первинний ключ – це атрибут (чи множина атрибутів), значення якого унікально ідентифікують кортежі (записи).

Кортеж відповідає заповненому рядку таблиці. У конкретних СКБД кортежі називають записами.

Ступінь відношення – кількість його атрибутів.

Кардинальне число (потужність відношення) – кількість кортежів у відношенні у нинішній момент часу.

Домен – це загальна сукупність значень, з якої беруться конкретні значення для конкретного атрибуту.

2.1.1. Домени

Початкові поняття про домени були наведені у п. 1.4.6. Домени точніше можна визначити як іменовану множину скалярних значень одного типу. Ці скалярні значення називають скалярами. По суті, це найменша семантична (смислова) одиниця даних. У скалярів немає внутрішньої структури, тобто вони неподільні у цій реляційній моделі.

Наприклад, якщо є атрибут (властивість об'єкту) "ПІБ", він передбачає скаляри, що містять прізвище, ім'я і по батькові. Природно, ці скаляри можна ще розбити на букви, але тоді буде втрачений потрібний сенс. Тобто для цієї моделі найменшими семантичними одиницями даних будуть саме прізвище, ім'я і по батькові.

З доменів беруться значення атрибутів. На практиці домени часто не описують, а задають типом, форматом і іншими властивостями даних. Кожен атрибут має бути визначений на ***єдиному домені.***

Основне призначення доменів – ***обмеження при порівнянні*** різних по сенсу атрибутів.

Наприклад: якщо для атрибутів ***НомерЗаліковоїКнижки*** відношення **Студенти** і ***НомерКабінету*** для відношення **Кабінети** домени задані таким чином:

НомерЗаліковоїКнижки ={51001, 51002, 51003, …, 55150}

НомерКабінету ={101, 102, 103, …, 559},

то система видасть помилку на запит типу: "Вивести усіх студентів, номер залікової книжки яких співпадає з номером кабінету". Якщо ж

домени не визначені, а визначений тільки цілий тип даних для цих атрибутів, то подібний запит виконається, хоча не матиме сенсу [14].

Ще одне можливе застосування доменів – використання їх в спеціальних запитах. Наприклад, "Які відношення у БД включають атрибути, визначені на домені "Номер залікової книжки"? У системі, що підтримує домени, такий запит матиме сенс і результатом його буде список відношень, де використовується Номер залікової книжки (це можуть бути відношення Успішність навчання, Студенти тощо). А в системі, де домени не визначені, реалізувати такого роду запит набагато складніше – якщо через імена атрибутів, то вони можуть не співпадати, а якщо через тип – то вийде багато зайвих відношень, оскільки множина інших атрибутів може мати цілий тип даних.

2.1.2. Відношення

З відношенням пов'язані поняття змінної відношення і значення відношення [29].

Змінна відношення – звичайна змінна, тобто іменований об'єкт, значення якого може змінюватися з часом (по суті – це множина заданих атрибутів цього відношення).

Значення відношення – значення змінної відношення в конкретний момент часу (по суті – це збережені кортежі відношення).

Як визначити поняття відношення?

Відношення R1, визначене на множині доменів D1, D2, ., Dn (необов'язково різних), складається з двох частин: заголовка і тіла.

Заголовок містить фіксовану множину пар {Ai: Di}, де Ai – ім'я атрибуту, Di – ім'я домену.

Тіло містить множину пар {Ai: Zij}, де Ai – ім'я атрибуту, Zij – значення і-го атрибуту в j-му кортежі.

i = 1,2,.n, де n – ступінь відношення

j = 1,2,.m, де m – кардинальне число.

Наприклад, розглянемо відношення **Студенти:**

Заголовок: {НомерЗаліковоїКнижки: Цілий; Прізвище: Текстовий; Ім'я: Текстовий; ДатаНародження: Дата; Адреса: Текстовий; Група: Текстовий}

Тіло: {НомерЗалікової Книжки: 51005; Прізвище: Петров; Ім'я: Петро; По батькові: Петрович; ДатаНародження: 12.05.93; Адреса: Свободи, 11-145; Група: ЭИ-21}

{НомерЗалікової Книжки: 51097; Прізвище: Іванов; Ім'я: Іван; По батькові: Іванович; ДатаНародження: 21.10.94; Адреса: Балакірєва, 20-9; Група: ЭИ-21} і так далі.

Іншим (класичним) визначенням відношення є таке: **Відношення Ri є підмножина декартового добутку множин.** В якості множин розглядається множина доменів, на яких будується відношення.

Для спрощеного опису відношення і його атрибутів часто використовують наступний запис [4]:

Ім'яВідношення (Ім'яАтрибуту1, Ім'яАтрибуту2, ., Ім'яАтрибутуN),

де підкреслюють атрибути, що входять у первинний ключ і де N, – ступінь відношення.

Властивості відношень

1. Немає однакових кортежів. Це випливає з того, що тіло відношення визначене як математична множина кортежів, а множина, за визначенням, не містить однакових елементів.

Як наслідок цієї властивості: у відношенні завжди існує первинний ключ.

2. Кортежі не впорядковані. Це виходить також з того, що тіло відношення визначене як математична множина кортежів. А математична множина за визначенням не впорядкована. Саме тому у відношенні не існує таких понять, як "наступний", "попередній", "другий кортеж" і тому подібне

3. Атрибути не впорядковані. Це випливає з того, що заголовок відношення визначений як математична множина атрибутів. А множина не впорядкована за визначенням. Тобто знову немає понять "Перший атрибут", "наступний атрибут" і тому подібне

4. Усі значення атрибутів неділимі. Це наслідок того, що кожен атрибут визначений на своєму домені, а домен – множина неділимих скалярів.

2.2. Цілісність реляційних даних

У будь-який момент часу будь-яка БД містить деякі певні значення атрибутів, які виражають конкретний стан об'єкту реального світу. Отже, БД потребує визначення ***правил цілісності,*** необхідних для того, щоб дані не вступили в протиріччя з реальним світом. Такі правила цілісності є ***специфічними*** для кожної БД. Це, по суті, інформування СКБД про обмеження реального світу, Наприклад, ім'я – тільки текстове значення, значення ваги, росту – тільки позитивні і тому подібне

Але таких правил цілісності недостатньо – не менш важливо, щоб дані усередині самої БД не суперечили один одному.

Наприклад, у БД Факультет вказали, що Сидоров Петро вчиться у групі ЕІ-5-10, але такої групи на факультеті не існує. Чи для того ж Петра групу вказали вірно, але в якості ПІБ старости його групи написали Ковальова М.О, а насправді старостою є Андрєєв С. В.

Для запобігання подібним ситуаціям існують ***загальні*** правила цілісності реляційних даних. Ці правила пов'язані з первинними і зовнішніми ключами.

2.2.1. Потенційні, первинні та альтернативні ключі

Нехай R – деяке відношення, тоді потенційний ключ K для R це підмножина множини атрибутів R, для яких виконуються такі властивості [4]:

а) **унікальність**, тобто немає двох різних кортежів у поточному значенні змінної відношення R з однаковими значеннями K;

б) **не надмірність**, тобто ніяка підмножина K не має властивість унікальності.

Наприклад, у відношенні **Студенти** потенційним ключем може бути один з атрибутів ***НомерЗаліковоїКнижки***, ***НомерОсобистоїСправи*** або ***СеріяПаспорта+НомерПаспорта***, оскільки кожен з них задовольняє визначенню. Але невірно буде призначити потенційним ключем цього відношення множину декількох з цих атрибутів, оскільки хоча для такої множини виконується властивість унікальності, але не виконується властивість не надмірності.

Потенційні ключі призначені для забезпечення основного механізму адресації на рівні кортежів, тобто за значенням потенційного ключа можна однозначно знайти кортеж.

Як вже було вказано в п.1.4.2, базове відношення може мати декілька потенційних ключів, але один їх них має бути вибраний в якості **первинного ключа**. Інші ж потенційні ключі називатимуться альтернативними.

У будь-якому відношенні має бути первинний ключ, тобто повинен міститися хоч би один потенційний ключ. Це може бути одне поле, а може бути множина декількох або навіть усіх полів відношення (в цьому випадку відношення називається ***повністю ключовим).***

Якщо ж у відношенні немає природних потенційних ключів або вони незручні для використання у рамках створюваної БД, то вводять штучні ключі. Наприклад, у відношенні можна ввести поле КодВикладача, щоб не використовувати номер паспорта, номер особистої справи або табельний номер. Часто такі штучні ключі вводять для забезпечення зв'язку двох або більше відношень з метою підвищення ефективності обробки запитів до БД і зменшення об'єму даних, що зберігаються.

2.2.2. Зовнішні ключі

Зовнішній ключ (п.1.3.3) існує для забезпечення несуперечності даних усередині БД, тобто значення зовнішнього ключа не може бути таким, якого немає серед значень первинного ключа зв'язаного з ним відношення (таблиці).

У реляційній БД передбачений зв'язок зовнішнього ключа не лише з первинним, але і з будь-яким іншим потенційним ключем, тобто з альтернативним. Але не можна створювати дублюючі зв'язки – і з первинним, і з альтернативним, щоб уникнути надмірності даних.

Нехай R2 – базове відношення деякої БД. Тоді ***зовнішній ключ*** FK (foreign key) відношення R2 – ця підмножина множини атрибутів R2, така, що:

а) існує базове відношення R1, що містить потенційний ключ CK (candidate key);

б) кожне значення FK в поточному значенні R2, завжди співпадає зі значенням CK деякого кортежу в поточному значенні відношення R1.

Деякі зауваження стосовно цього визначення [36]:

1. Кожне значення зовнішнього ключа повинно бути значенням відповідного потенційного ключа, проте, зворотне не потрібно, тобто потенційний ключ, що відповідає зовнішньому ключу може містити значення, які в даний момент не є значенням зовнішнього ключа. Наприклад, може існувати запис про групу, в яку доки ніхто із студентів не зарахований.

2. Зовнішній ключ буде складеним тоді і тільки тоді, коли відповідний потенційний ключ також буде складеним. Аналогічно – зовнішній ключ буде простим (що складається з одного атрибуту) тоді і тільки тоді, коли відповідний потенційний ключ – простий.

3. Кожен атрибут, що входить у зовнішній ключ має бути визначений на тому ж домені, що і відповідний атрибут відповідного потенційного ключа.

4. R1 і R2 не обов'язково різні.

З поняттям зовнішнього ключа зв'язується ще ряд термінів:

можна сказати, що значення зовнішнього ключа є посиланням до кортежу, що містить відповідне значення потенційного ключа. Цей кортеж називається **посилальним (цільовим) кортежем**, а відношення, що його містить – **посилальним (цільовим)**. Відношення, що містить зовнішній ключ називається **відношенням, що посилається**.

Із зовнішніми ключами пов'язане одне з основних правил цілісності, а саме правило посилальної цілісності [10]: база даних не повинна містити неузгоджених значень зовнішнього ключа (неузгоджені значення – такі значення, яких немає для потенційного ключа у посилальному відношенні). По суті, це правило еквівалентне визначенню зовнішнього ключа.

Правила зовнішніх ключів.

Правило посилальної цілісності має на увазі стан БД в конкретний момент часу. Але як уникнути тимчасових некоректних ситуацій, які можуть виникнути при оновленні даних у БД?

Найпростіший шлях – заборонити будь-які операції, приведені до порушення правила посилальної цілісності.

Але при оновленні БД не уникнути тимчасового порушення цілісності (наприклад, розформовується група або на початку навчального року перейменовується група). Тому необхідно ввести

компенсуючи операції**,** які "виправлять" це тимчасове порушення цілісності.

Такі компенсуючи операції пов'язані із зовнішніми ключами, тому для будь-якого зовнішнього ключа при створенні зв'язку необхідно відповісти на два питання:

1. Що повинне статися при спробі ***видалити*** об'єкт посилання зовнішнього ключа?

2. Що повинне статися при спробі ***змінити (відновити)*** значення потенційного ключа, на який є посилання?

Варіанти вирішення цієї проблеми були розглянуті у п. 1.4.7.

2.2.3. Null-значення

Ускладнення при забезпеченні цілісності даних можуть бути викликані невизначеними або відсутніми значеннями. Наприклад, у БД з творів мистецтва – невідомий автор картини, у БД Школа деякі діти – сироти (немає батьків) і тому подібне

Для вирішення проблем відсутності значень, Е. Кодд запропонував ввести спеціальні мітки, названі ним Null-значеннями, які визначив так: якщо цей кортеж має Null-значення якогось атрибуту, то це означає, що в ньому значення атрибуту відсутнє.

Це не те ж, що числовий 0 або пробіл, це взагалі не значення, а тільки мітка – позначення відсутності будь-якого значення.

Більшість сучасних реляційних СКБД підтримують Null-значення.

З Null-значеннями пов'язане друге правило цілісності – **правило цілісності об'єктів**:

Жоден елемент первинного ключа базового відношення не може бути Null-значенням.

Це правило пояснюється наступним: кортежі відношень відповідають об'єктам реального світу; за визначенням, ці об'єкти помітні, тобто деяким чином їх можна розпізнати; первинні ключі виконують функцію унікальної ідентифікації об'єктів; якщо неможливо ідентифікувати об'єкт, то не можна сказати чи існує він взагалі.

Відносно цього правила можна сказати таке:

1) це правило торкається тільки базових відношень (тобто не обчислюваних, не похідних);

2) правило застосовується тільки для первинних ключів, а для альтернативних ключів Null-значення можуть бути заборонені або дозволені.

Застосування Null-значень для зовнішніх ключів:

1) Коли немає даних, тобто немає відповідного кортежу в посилальному відношенні (наприклад, немає даних про батьків учня);

2) При каскадному видаленні (наприклад, на факультеті розформували одну з груп, а студентів цієї групи розподіляють в інші, але доки точно не відомо в які. Тоді при каскадному видаленні цієї групи з таблиці Групи віддаляться також усі студенти цієї групи з таблиці Студенти. Таке видалення буде помилковим, з точки зору здорового глузду, тому у студентів такої групи тимчасово треба замінити значення атрибуту КодГрупи у відношенні Студенти на Null-значення.

Деякі розробники БД намагаються уникати Null-значень, застосовуючи замість цього значення за замовчуванням. Але у багатьох сучасних СКБД, підтримується Null-значення.

Питання для самодіагностики

1. Поясніть, чому на Ваш погляд, спрощення процесу накладення обмежень цілісності (одна з цілей нормалізації) пов'язана з потенційними ключами.
2. Поясніть смисл понять: відношення, змінна відношення, значення відношення, заголовок відношення, тіло відношення, атрибут, первинний ключ, ступінь відношення, домен, кортеж, кардинальне число.
3. Які реляційні об'єкти даних пов'язані із заголовком відношення?
4. Які реляційні об'єкти даних пов'язані з тілом відношення?
5. Що загального і в чому відмінності понять "Змінна відношення" і "змінна" (мовою програмування)?
6. Доведіть, що твердження "у відношенні завжди існує первинний ключ" є наслідком властивості відношення, в якому говориться про те, що немає однакових кортежів.
7. Як ви розумієте, що таке: правила цілісності, потенційний ключ, первинний ключ, альтернативний ключ, зовнішній ключ, правило посилальної цілісності, вторинний ключ.

8. Що таке Null-значення?

9. В чому відмінність первинного і потенційного ключа?

10. Чи може не існувати у відношенні первинного, потенційного або альтернативного ключа?

11. Чи може первинний ключ одночасно бути зовнішнім ключем? Якщо ні – доведіть, якщо так – наведіть приклади.

12. Якщо в СКБД не передбачені компенсуючи операції для зовнішніх ключів, до чого це призведе?

13. Що треба зробити, щоб у СКБД замість Null-значень використати значення за замовчуванням?

3. Реляційна алгебра

Цілі – познайомитись с типами мов, які використовуються для виконання операцій на відношеннях; освоїти основні операції реляційної алгебри аналогічні традиційним операціям над множинами; освоїти власне реляційні та додаткові операції реляційної алгебри.

Компетенції, що формуються:

Загальнонаукові – базові знання в області фундаментальної та прикладної математики та уміння їх застосовувати в професійній діяльності.

Професійні – ґрунтовна математична підготовка з теоретичних і методичних основ інформаційних технологій для використання математичного апарату під час вирішення прикладних завдань в області інформаційних систем.

Спеціалізовано-професійні – здатність до математичного та логічного мислення, знання основних понять, ідей і методів фундаментальної математики.

Основні питання

Теоретичні мови виконання операцій над відношеннями – аналізуються два основі підходи до теоретичних мов запитів до бази даних, які базуються на математичних основах, запропонованих Е. Коддом: реляційна алгебра та реляційне числення, що є еквівалентними один одному за своїми виразними можливостями.

Основні операції реляційної алгебри – дається пояснення сенсу основних операцій реляційної алгебри як тим, що є аналогічними традиційним операціям над множинами, так і власне реляційним та додатковим операціям.

3.1. Теоретичні мови виконання операцій над відношеннями

Операції, які виконуються над відношеннями, можна поділити на дві групи. Першу групу складають операції над множинами, до яких відносяться операції об'єднання, перетину, різниці, ділення, декартового добутку.

Другу групу складають спеціальні операції над відношеннями, до яких, зокрема, відносять операції проекції, з'єднання, вибору.

У реляційних СКБД для виконання операцій над відношеннями використовуються дві групи мов, що мають в якості своєї математичної основи, теоретичні мови запитів запропоновані Е. Коддом:

1) реляційна алгебра;
2) реляційне числення.

Ці мови представляють мінімальні можливості реальних мов маніпулювання даними відповідно до реляційної моделі і еквівалентні один одному своїми виразними можливостями. Існують не дуже складні правила перетворення запитів між ними.

У реляційної алгебрі операнди і результати усіх дій є відношеннями. Мови реляційної алгебри є процедурними, оскільки відношення, що є результатом запиту до реляційної БД, обчислюється при виконанні послідовності реляційних операторів, які застосовуються до відношення. Оператори складаються з операндів, у ролі яких виступають відношення, і реляційних операцій. Результатом реляційної операції є відношення.

Мови числення, на відміну від реляційної алгебри, є непроцедурними (описовими, або декларативними) і дозволяють висловлювати запити за допомогою предиката першого порядку (висловлювання у вигляді функції), якому повинні задовольняти кортежі або домени відношень. Запит до БД, виконаний з використанням подібної мови, містить лише інформацію про бажаний результат. Для цих мов характерна наявність наборів правил для запису запитів [4].

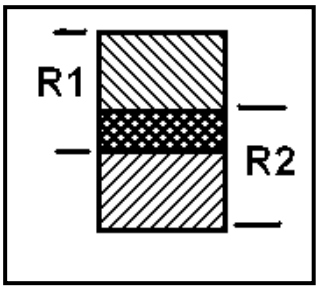
У реалізаціях конкретних реляційних СКБД зараз не використовується в чистому вигляді ні реляційна алгебра, ні реляційне числення. Фактичним стандартом доступу до реляційних даних стала мова SQL (Structured Query Language) [4; 23]. Мова SQL є сумішшю операторів реляційної алгебри і виразів реляційного числення, яка використовує синтаксис, близький до фраз англійської мови з розширеними додатковими можливостями, відсутніми в реляційної алгебрі і реляційному численні. Взагалі, мова доступу до даних називається реляційно-повною, якщо вона по виразній силі не поступається реляційній алгебрі (чи, що те ж саме, реляційному численню), тобто будь-який оператор реляційної алгебри може бути виражений засобами цієї мови. Саме таким і є мова SQL. Проте для глибшого розуміння тих можливостей, які надає мова SQL, необхідно абсолютно ясно уявляти той математичний апарат, а саме реляційну алгебру, на основі якої будується мова запитів SQL.

3.2. Основні операції реляційної алгебри

Реляційна алгебра, на основі якої створена мова SQL і яку визначив Кодд, складається з 8 операцій. Їх можна розділити на дві групи: реляційні операції, аналогічні традиційним операціям над множинами, і власне реляційні операції.

Для усіх реляційних операцій потрібне виконання ***властивості замкнутості***, тобто результатом кожної операції над відношеннями повинно бути відношення.

3.2.1. Реляційні операції, аналогічні традиційним операціям над множинами



Об'єднання. Результатом об'єднання відношень R1 і R2 є відношення R3, що містить усе кортежі, які належать хоч би одному з R1 і R2.

На відміну від об'єднання множин, результатом є не множина кортежів, а саме відношення. Тому кортежі мають бути однорідні, тобто об'єднувані відношення мають бути ***сумісні за типом***. Це означає, що:

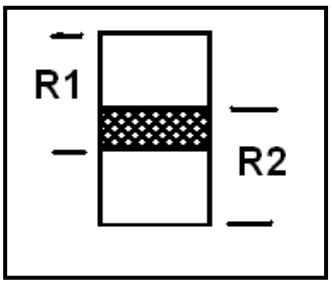
a) кожне з них має одну і ту ж множину атрибутів;
b) відповідні атрибути визначені на одному і тому ж домені.

Синтаксис: **R1 UNION R2**

Математичний запис: **REZ = $R_1 \cup R_2$**

Наприклад: припустимо, у БД університету є окремі відношення ***ШтатніСпівробітники*** і ***Сумісники.*** Ці дві таблиці можна об'єднати і в результаті отримати відношення, що містить усі дані на співробітників університету, як штатних так і сумісників.

Перетин. Результатом перетину відношень R1 і R2 є відношення R3, що містить кортежі, які належать і R1, і R2. Для цієї операції також повинна виконуватися умова сумісності за типом.

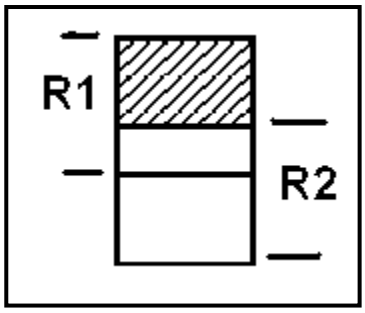


Синтаксис: **R1 INTERSECT R2**

Математичний запис: **REZ = R1 ∩ R2**

Наприклад: припустимо, у БД університету окремі штатні співробітники можуть ще працювати за сумісництвом. Тоді за допомогою цієї операції можна знайти тих штатних співробітників, які ще працюють за сумісництвом.

Віднімання (Різниця). Результатом віднімання відношення R2 з відношення R1 є відношення R3, усі кортежі якого належать R1 і не належать R2. Умова сумісності за типом також повинна виконуватися.



Синтаксис: **R1 MINUS R2**

Математичний запис: **REZ = R1 \ R2**

Наприклад, з тих же відношень ***ШтатніСпівробітники*** і ***Сумісники*** бази даних університету можна з'ясувати, які штатні співробітники не є одночасно сумісниками.

Добуток (декартовий). Результатом добутку відношень R1 і R2 є відношення R3, що містить усі можливі кортежі, які є поєднанням двох кортежів, що належать відповідно відношенням R1 і R2.

Синтаксис: **R1 TIMES R2**

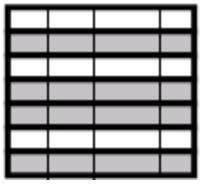
R_1	R_2	R_3	
a	x	a	x
b	y	a	y
c		b	x
		b	y
		c	x
		c	y

Математичний запис: **REZ = R1 x R2**

Результатом є відношення, а не множина пар кортежів, як це буває при добутку над множинами. Первинні кортежі зчіплюються, тобто утворюють в результаті новий кортеж.

Наприклад, якщо виконати добуток відношень ***Студенти*** і ***Дисципліни***, то отримаємо в результаті відношення, в якому міститиметься інформація про те, що кожен студент вивчає кожну дисципліну.

3.2.2 Власне реляційні операції



Вибірка (операція обмеження). Результатом вибірки, застосованої до відношення R1, є відношення R2, що містить усі кортежі відношення R1, що задовольняють певним умовам.

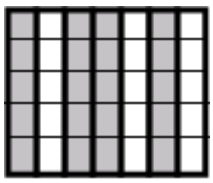
Можна сказати, що це "горизонтальна" підмножина початкового відношення.

Синтаксис: **R1 WHERE xθy** (θ (тета) – будь-який умовний оператор).

Наприклад, з відношення Студенти (НомерЗалікової Книжки, Прізвище, Ім'я, По батькові, КодГрупи) вивести дані про тих студентів, які навчаються у групі ЭИ-31.

Математичний запис: **REZ = sB = "b" (R)**

що означає селекцію (вибір) з відношення R тих кортежів, для яких значення атрибуту B рівне " b".



Проекція. Результатом проекції, застосованої до відношення R1, є відношення R2, що містить усі кортежі R1 після виключення з нього деяких атрибутів. Такі кортежі називаються підкортежами.

Можна сказати, що це "вертикальна" підмножина початкового відношення.

Спеціального синтаксису для позначення операції проекції немає. В якості математичного запису часто використовують такий:

Математичний запис: **REZ = $\pi_{I1, I2, \ldots, Im}$ (R)**,

де **IJ –** атрибути відношення R, задані іменами або номерами (цілими числами з діапазону [1, kR]).

Звернемо увагу на окремі випадки:

1) можливо вказати список усіх атрибутів початкового відношення – це тотожна проекція;

2) можливо вказати порожній список атрибутів – це нульова проекція.

Наприклад, з відношення ***Студенти*** (НомерЗалікової Книжки, Прізвище, Ім'я, По батькові, ДатаНародження, Адреса, КодГрупи) вивести для усіх студентів інформацію тільки про прізвище, ім'я, по батькові та групі.

З'єднання. Результатом з'єднання відношень R1 і R2 є відношення R3, кортежі якого є зчеплення двох кортежів (що належать відповідно R1 і R2), що мають загальне значення для одного або декількох загальних атрибутів R1 і R2.

Причому ці загальні значення в результуючому відношенні з'являються тільки один раз.

Синтаксис: **R1 JOIN R2.**

З' єднання має властивості:

асоціативність: (R1 JOIN R2) JOIN R3 = R1 JOIN (R2 JOIN R3);

комутативність: R1 JOIN R2 = R2 JOIN R1.

Ця операція має декілька різновидів, але найпоширеніше – природне з'єднання. Прикладом природного з'єднання може служити з'єднання відношень Студенти і Групи по атрибуту Група. В результаті вийде відношення, що містить інформацію про студентів і для кожного студента, – про його групу. Або приклад з'єднання двох відношень R1 та R2 за атрибутом Цех (рис. 3.1).

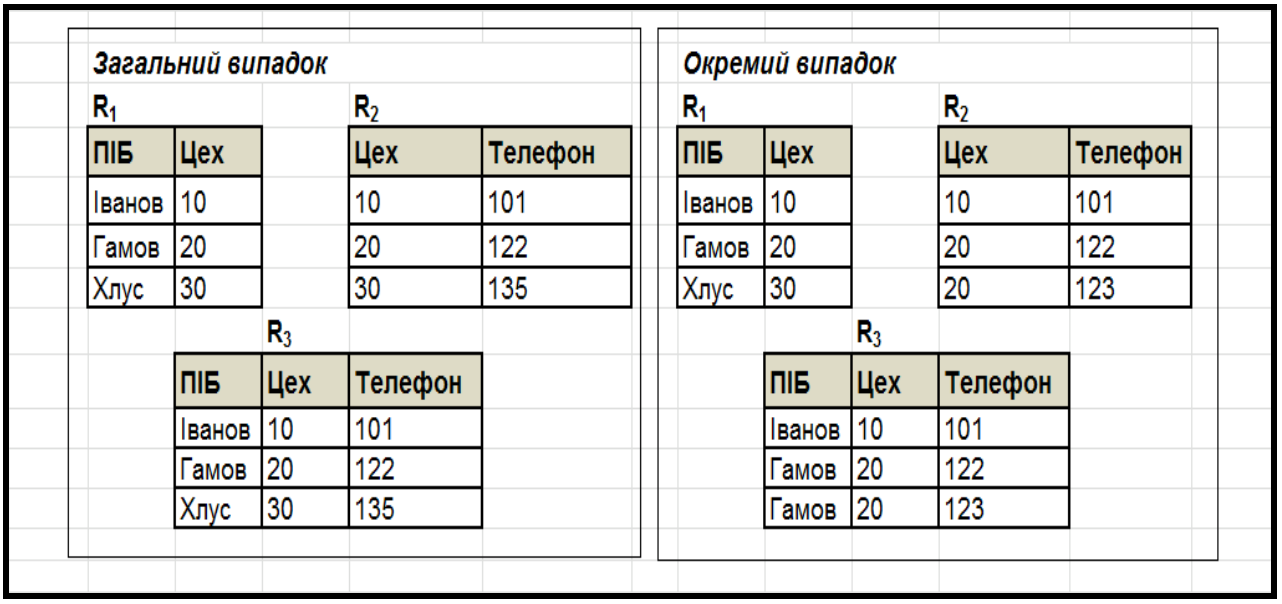


Рис. 3.1. **Приклади еквіз'єднання**

Існує ще **θ(тета)** – з'єднання. Воно призначене для випадків, коли два відношення з'єднуються на основі деяких умов (xθy), відмінних від еквівалентності.

В цьому випадку можна записати: (R1 TIMES R2) WHERE xθy, тобто поєднання добутку і вибірки.

Наприклад, з'єднання (R1 TIMES R2) WHERE R1.Цех < R2.Цех дасть такий результат (рис. 3.2).

R_3

ПІБ	Цех	Телефон
Іванов	10	122
Іванов	10	122
Гамов	20	135

Рис. 3.2. **Приклад тетаз'єднання**

Ділення. Нехай відношення ***R***, що називається діленим, містить атрибути ***(A1, A2,...,An)***. Відношення ***S*** – дільник містить підмножину атрибутів **A**: ***(A1,A2,...,Ak) (k<n)***. Результуюче відношення ***C*** визначене на атрибутах відношення ***R***, яких немає в ***S***, тобто ***Ak+1,Ak+2,...,An.*** Кортежі включаються в результуюче відношення **C** тільки у тому випадку, якщо його декартовий добуток з відношенням ***S*** міститься в діленому ***R***. Наприклад, ділення відношення ***ЕкзаменаційнаВідомість*** на відношення ***Результати*** (рис. 3.3).

Синтаксис: R1 DIVIDE BY R2.

ЕКЗ.ВІДОМІСТЬ

ПРІЗВИЩЕ	ПРЕДМЕТ	ОЦІНКА
Данилов	Історія	11
Данилов	Фізика	11
Сидоров	Історія	9
Сидоров	Фізика	9
Петров	Історія	9
Петров	Фізика	11

:

РЕЗУЛЬТАТИ

ПРЕДМЕТ	ОЦІНКА
Історія	11
Фізика	9

=

СТУДЕНТИ

ПРІЗВИЩА
Данилов
Сидоров

Рис. 3.3. **Приклад операції ділення**

3.2.3. Додаткові операції реляційної алгебри

До восьми основних операцій реляційної алгебри були додані деякі інші в якості додаткових. Їх зручно використовувати для практичних цілей, хоча можна реалізувати через основні. Найвдаліше основний набір доповнили дві операції: розширення і підбиття підсумків [4].

Операція розширення

З її допомогою з початкового відношення створюється нове відношення, яке містить додатковий атрибут, значення якого отримані за допомогою деяких скалярних обчислень.

EXTEND <Ім'я відношення> ADD (<скалярний вираз>) AS <Ім'я нового атрибуту>

Приклади:

1. Нехай є відношення **СПІВРОБІТНИКИ** (ТабельнийНомер, Прізвище, Ім'я, По батькові, ДатаПрийняття). Для кожного співробітника вивести стаж роботи.

EXTEND СПІВРОБІТНИКИ ADD (year(date()) – year (ДатаПрийняття)) AS Стаж.

2. Окремий випадок – додавання нового атрибуту, заповненого однаковими значеннями: у відношення СекціїКружки бази даних Факультет додати атрибут МісцеПроведення, заповнивши його однаковими значеннями для усіх секцій – "ОЦ-411".

EXTEND СекціїКружки ADD ("ОЦ-411") AS МісцеПроведення.

3. Окремий випадок – перейменування атрибуту Адреса відношення Студенти в АдресаРеєстрації.

EXTEND Студенти ADD (Адреса) AS АдресаРеєстрації.

Операція підбиття підсумків

Можна сказати, що якщо операція розширення забезпечує можливість "горизонтального" обчислення, то операція підбиття підсумків забезпечує можливість "вертикального" обчислення. Іншими словами ця операція дає можливість виконати групові операції по атрибутах (підрахувати кількість, суму записів і тому подібне).

SUMMARIZE <Ім'я відношення> BY (<список імен атрибутів>)
ADD <групова операція> AS <Ім'я поля для підсумкового значення>

Груповими операціями можуть бути:

SUM (<ім'я атрибуту>) – сума числових значень;

COUNT (<ім'я атрибуту>) – кількість значень;

AVG (<ім'я атрибуту>) – середнє значення;

MIN (<ім'я атрибуту>) – мінімальне значення;

MAX (<ім'я атрибуту>) – максимальне значення.

Приклади:

1. Є відношення ***Студенти*** (НомерЗаліковоїКнижки, Прізвище, Ім'я, По батькові, КодГрупи). Підрахувати кількість студентів в кожній групі.

SUMMARIZE Студенти BY (КодГрупи) ADD COUNT (НомерЗаліковоїКнижки) AS КількістьСтудентів.

2. Є відношення **Заняття** (КодГрупи, Дисципліна, Викладач). Підрахувати кількість груп, які вивчають більше 10 дисциплін.

SUMMARIZE Заняття BY (КодГрупи) ADD COUNT(Дисципліна) AS N WHERE N>10.

3.2.4. Операції модифікації (оновлення)

До операцій оновлення відносяться операції вставки, зміни і видалення.

Зауваження: *Не слід ототожнювати синтаксис цих операцій реляційної алгебри з синтаксисом запитів, що виконують аналогічні дії в мові SQL!*

Вставка: INSERT (<реляційне вираз або список атрибутів>) INTO <Ім'я відношення>

Приклади:

1. Вставити новий запис у відношення Студенти.

INSERT (НомерЗалікової Книжки=041112; Прізвище="Іванов"; Ім'я = "Іван") INTO Студенти

2. У відношення Студенти31Групи вставити ті записи з відношення Студенти, які відповідають студентам групи ЭІ-31.

INSERT (Студенти WHERE КодГрупи = "ЭІ-31") INTO Студенти31групи.

Зміна: UPDATE <Ім'я відношення> <реляційне вираження>333

Приклад: У відношенні Студенти змінити групи "ЭІ-31" на "ЭІ-41".

UPDATE Студенти WHERE КодГрупи="ЭІ-31" КодГрупи :="ЭІ-41".

Видалення: DELETE <реляційне вираження>

Приклад: З відношення Студенти видалити усіх студентів групи ЭІ-35.

DELETE Студенти WHERE КодГрупи="ЭІ-35".

Основна мета реляційної алгебри – забезпечити ***запис виразів***, які не обов'язково можуть бути запитами.

В якості прикладів можна привести такі ***застосування виразів***:

1) визначення області вибірки (WHERE);

2) визначення області оновлення (тобто даних для вставки, зміни або видалення);

3) визначення правил цілісності;

4) визначення правил безпеки (тобто даних, для яких здійснюється контроль доступу).

Тобто, можна сказати, що вирази означають символічний запис намірів користувача БД.

Мову називають реляційно-повною, якщо його можливості відповідають можливостям реляційних операцій алгебри.

Питання для самодіагностики

1. Що таке реляційна алгебра?
2. Які реляційні операції ви знаєте? Наведіть приклади.
3. Як ви розумієте сенс терміну "реляційно повна мова"
4. У БД Факультет містяться відношення

Студенти (НомерЗалікової Книжки, Прізвище, Ім'я, По батькові, ДатаНародження, Адреса, Телефон, КодГрупи),

Стіпендіанти (№Залікової Книжки, Прізвище, Ім'я, По батькові, КодГрупи, РозмірСтипендії)

Групи (КодГрупи, Спеціальність, Курс)

Запишіть в реляційних виразах такі запити:

А. Отримати список студентів, що одержують стипендію.

Б. Отримати список студентів, які не одержують стипендію.

В. Отримати список студентів, у яких немає телефону.

Г. Отримати список прізвищ та імен студентів із вказівкою для них усіх відомостей про групи, в яких вони вчаться.

Д. Отримати список студентів із вказівкою віку.

Е. Отримати кількість груп кожної спеціальності.

4. Теорія нормалізації реляційної моделі даних

Цілі – ознайомитися з поняттям функціональних залежностей у відношеннях та їх значенню при виконанні процедури нормалізації; ознайомитись з поняттями першої, другої та третьої нормальної форми та правилами перетворення базового відношення від однієї форми до іншої; розібрати особливості нормальних форм більш високого порядку; навести перелік основних кроків які потрібно виконати для нормалізації початкового відношення.

Компетенції, що формуються:

Загальнонаукові – базові знання науково-методичних основ в області інформаційних технологій, уміння їх застосовувати під час розробки та інтеграції систем.

Спеціалізовано-професійні – знання загальнометодологічних принципів проектування схеми реляційної бази даних.

Спеціалізовано-професійні – знання сучасних теорій організації баз даних, методів і технологій їх розробки.

Основні питання

Функціональні залежності – наводиться поняття функціональної залежності; правила їх виводу та процедура отримання множини неприведених функціональних залежностей.

Нормалізація відношень – розглядається сенс та співвідношення нормальних форм, поняття "декомпозиції без втрат" та перелік кроків, які необхідно виконати для здійснення переходу від однієї форми до іншої.

Нормальні форми більш високого порядку – розкриваються поняття багатозначної залежності та залежності по з'єднанню і на їх основі пояснюється сенс четвертої та п’ятої нормальних форм.

Підсумкова схема процедури нормалізації – наводиться узагальнений перелік кроків, які необхідно виконати для здійснення процесу нормалізації відношень.

Денормалізація відношень – наводяться певні міркування стосовно можливого виконання зворотної процедури нормалізації, а саме денормалізації, з метою підвищення ефективності виконання запитів до бази даних.

Один з перших етапів проектування бази даних пов'язаний з побудовою ***логічної структури БД*** (п.1.1). Іншими словами, треба вирішити питання – які базові відношення, і з якими атрибутами слід задати. Але наступним кроком на цьому шляху є ***нормалізація відношень***, яка дозволяє усунути дублювання, забезпечує несуперечність збережених даних і зменшує трудовитрати на ведення БД.

Процес нормалізації полягає в розбитті (декомпозиції) початкових відношень БД на простіші. При цьому на кожній ступені цього процесу схеми стосунків наводяться до нормальних форм. Для кожного ступеня нормалізації є набори обмежень, яким повинні задовольняти відношення в реляційній базі даних.

Спочатку розглянемо основні поняття, необхідні для обговорення питань нормалізації відношень.

4.1. Функціональні залежності

4.1.1. Поняття функціональної залежності

Згадаємо, що будь-яке відношення розглядається як змінна, що набуває певних значень в певні моменти часу.

Нехай R – змінна відношення, X, Y – довільні підмножини множини усіх атрибутів R. ***Y функціонально залежить від X*** тоді і тільки тоді, коли для будь-якого допустимого значення R кожне значення X пов'язане тільки з одним значенням Y.

Позначається: **X→Y.** Кажуть, що: "X функціонально визначає Y" або "Y функціонально залежить від X".

Ліва частина виразу називається ***детермінантом (детермінантою)*** функціональної залежності (ФЗ), права – ***залежною частиною*** ФЗ.

Наприклад [5; 24; 27], у відношенні **Студенти** (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, Адреса, КодГрупи) існують такі ФЗ

{НомерЗалікової Книжки} → {Прізвище, І'мя, По батькові}

{НомерЗалікової Книжки} → {Адреса, КодГрупи}

{НомерЗалікової Книжки, Прізвище, І'мя, По батькові} → {Адреса, КодГрупи}

Це лише деякі ФЗ, з яких можна зробити висновок, що якщо детермінант містить первинний ключ, то множина усіх інших атрибутів відношення функціонально залежить від нього.

Інший приклад: відношення **Кафедри** (КодКафедри, НазваКафедри, Кабінет, Телефон) існують ФЗ

{КодКафедри} → {Кабінет, Телефон}

{НазваКафедри} → {Кабінет, Телефон}

Таким чином, аналогічний висновок можна зробити не лише для первинних ключів, але і для альтернативних, тобто для усіх потенційних ключів.

Множина атрибутів відношення, яке містить в якості підмножини потенційний ключ називається ***суперключем*** цього відношення.

Розглянемо ще один приклад [5; 24]: якщо в те ж відношення Студенти додати атрибут СтаростаГрупи, то з'являться такі ФЗ:

{КодГрупи} → {СтаростаГрупи}

{СтаростаГрупи} → {КодГрупи}

(причому, ні атрибут КодГрупи, ні атрибут СтаростаГрупи не є потенційними ключами)

В цьому випадку є надмірність даних, яка може привести до введення помилкових відомостей (користувач випадково може ввести в якості старости деякої Групи не того студента, який насправді є старостою, але система не видасть помилку).

Фактично, якщо у відношенні є ФЗ, в якій детермінант не є суперключем, то відношення надмірне.

Існують такі ФЗ, які враховуються тільки формально, оскільки вони завжди існують і маються на увазі самим визначенням ФЗ. Це тривіальні ФЗ.

Тривіальна Функціональна залежність – це така ФЗ, залежна частина якої є підмножиною детермінанта.

Наприклад,

{НомерЗалікової Книжки, Прізвище, І'мя } → {Прізвище, І'мя }

{КодГрупи, Курс} → {Курс}

Такі тривіальні ФЗ не розглядаються при нормалізації, але все таки вони існують і завжди формально враховуються.

4.1.2. Правила виводу функціональних залежностей

З деякої множини ФЗ конкретного відношення можна отримати похідні ФЗ. Наприклад, з ФЗ відношення Студенти

{НомерЗалікової Книжки, Прізвище, І'мя } → {Адреса, Телефон}

можна отримати такі ФЗ:

{НомерЗалікової Книжки, Прізвище, І'мя } → {Адреса}

{НомерЗалікової Книжки, Прізвище, І'мя } → {Телефон}

Нехай S – деяка множина ФЗ. Тоді множину усіх ФЗ, яку можна отримати з S називається ***замиканням*** множини S і позначається S+.

Щоб отримати замикання деякої множини ФЗ, потрібні правила виведення ФЗ. Такі правила виводу сформулював Армстронг (Швеція), тому їх називають правилами Армстронга (чи аксіомами Армстронга) [4].

Позначимо за **A, B, C** довільні підмножини множини атрибутів заданого відношення **R**, а записом **AB** означатиме об'єднання **A** та **B**.

Правила виведення ФЗ Армстронга :

1. ***Рефлексивність***: якщо B є підмножиною A, те A → B;
2. ***Доповнення***: якщо A → B, то AC → BC;
3. ***Транзитивність***: якщо A → B і B → C, то A → C.

Кожне з цих правил може бути доведене на основі визначення ФЗ, а перше правило – це визначення тривіальної ФЗ.

Ці правила ***повні***, оскільки їх вистачає для виведення замикання (тобто усіх ФЗ) із початкової множини ФЗ.

Вони також ***вичерпні***, оскільки ніякі додаткові ФЗ не можуть бути виведені з початкової множини ФЗ.

Але з цих правил для спрощення практичного виведення ФЗ можна отримати декілька додаткових правил (наслідків) [4]:

4. ***Самовизначення***: A → A;
5. ***Декомпозиція***: якщо A → BC, то A → B і A → C;
6. ***Об'єднання***: якщо A → B і A → C, то A → BC;

7. ***Композиція***: якщо A → B і C → D, то AC → BD.

Приклад: розглянемо відношення Групи (КодГрупи, Спеціальність, Курс, Староста).

Як початкову множину ФЗ візьмемо множину з таких двох ФЗ [24]:

(1) {КодГрупи} → {Спеціальність, Курс}

(2) {КодГрупи} → {Староста}

Виведемо замикання цієї множини ФЗ.

За правилом 1 можна записати усі тривіальні залежності:

(3) {Спеціальність, Курс} → {Спеціальність}

(4) {Спеціальність, Курс} → {Курс}

За правилом 2:

(5) {КодГрупи, Староста} → {Спеціальність, Курс, Староста}

(6) {КодГрупи, Спеціальність} → {Староста, Спеціальність}

(7) {КодГрупи, Курс} → {Староста, Курс}

(8) {КодГрупи, Спеціальність, Курс} → {Староста, Спеціальність, Курс}

За правилом 3 безпосередньо нічого не виведемо.

За правилом 4:

(9) {КодГрупи} → {КодГрупи}

(10) {Спеціальність} → {Спеціальність}

і так далі (11) (12)

За правилом 5:

(13) {КодГрупи} → {Спеціальність}

(14) {КодГрупи} → {Курс}

За правилом 6:

(15) {КодГрупи} → {Спеціальність, Курс, Староста}

За правилом 7 безпосередньо нічого не виведемо.

Проте, знання цих правил і вміння ними користуватися не забезпечують виведення замикання множини ФЗ, оскільки не існує чіткого алгоритму такого виводу.

4.1.3. Неприведені функціональні залежності

Функціональна залежність називається ***неприведеною ліворуч***, якщо жоден атрибут не може бути опущений із її детермінанта без порушення залежності (іншими словами, детермінант не надмірний) [26].

Наприклад: ФЗ {НомерЗаліковоїКнижки, Прізвище, І'мя } → {Адреса} приведена, оскільки з детермінанта можна виключити атрибути без порушення ФЗ.

А ось ФЗ {КодГрупи, Дисципліна} → {Викладач} відношення Заняття (КодГрупи, Дисципліна, Викладач) неприведена, оскільки з детермінанта не можна виключити ні атрибут КодГрупи, ні атрибут Дисципліна без порушення залежності.

Множина ФЗ називається ***неприведеною*** тоді і тільки тоді, коли виконуються три властивості:

1) залежна частина кожної ФЗ з цієї множини містить тільки один атрибут;
2) кожна ФЗ з цієї множини є неприведеною ліворуч;
3) жодна ФЗ з цієї множини не може бути опущена.

Наприклад, розглянемо множину ФЗ відношення Групи з прикладу у попередньому підрозділі 4.1.2.

(1) {КодГрупи} → {Спеціальність, Курс}
(2) {КодГрупи} → {Староста}
(3) {Спеціальність, Курс} → {Спеціальність}
(4) {Спеціальність, Курс} → {Курс}
(5) {КодГрупи, Староста} → {Спеціальність, Курс, Староста}
(6) {КодГрупи, Спеціальність} → {Староста, Спеціальність}
(7) {КодГрупи, Курс} → {Староста, Курс}
(8) {КодГрупи, Спеціальність, Курс} → {Староста, Спеціальність, Курс}
(9) {КодГрупи} → {КодГрупи}
(10) {Спеціальність} → {Спеціальність}
(11) {Курс} → {Курс}
(12) {Староста} → {Староста}
(13) {КодГрупи} → {Спеціальність}
(14) {КодГрупи} → {Курс}
(15) {КодГрупи} → {Спеціальність, Курс, Староста}

По першій властивості виключаємо з цієї множини такі ФЗ: 1, 5, 6, 7, 8, 15.
По другій властивості виключаємо наступні ФЗ: 3, 4.
По третій властивості виключаємо: 9, 10, 11, 12.
Таким чином залишилися:

(2) {КодГрупи} → {Староста}

(13) {КодГрупи} → {Спеціальність}

(14) {КодГрупи} → {Курс}

Отримана множина ФЗ неприведена. Можна зробити висновок, що отримана множина ФЗ виражає той факт, що відношення Групи містить атрибути КодГрупи, Староста, Спеціальність, Курс, і КодГрупи – первинний ключ.

В якості висновку можна сказати, якщо T є неприведеною множиною ФЗ, яка еквівалентна множині S, то перевірка виконання функціональних залежностей із множини T автоматично забезпечить виконання усіх функціональних залежностей з множини S [4].

4.2. Нормалізація відношень

Уся теорія нормалізації відношень являє собою формалізований процес загальних правил, які повинні виконувати розробники БД з метою створення такої схеми бази, яка б відповідала критеріям не надмірності, не суперечливості та незалежності даних. Ці правила, також, можуть бути автоматизовані для реалізації на ЕОМ.

4.2.1. Огляд нормальних форм

Процес нормалізації відношень заснований на концепції нормальних форм (НФ). Кажуть, що відношення знаходиться в деякій НФ, якщо воно задовольняє заданому набору умов. Відомо декілька НФ (рис. 4.1).

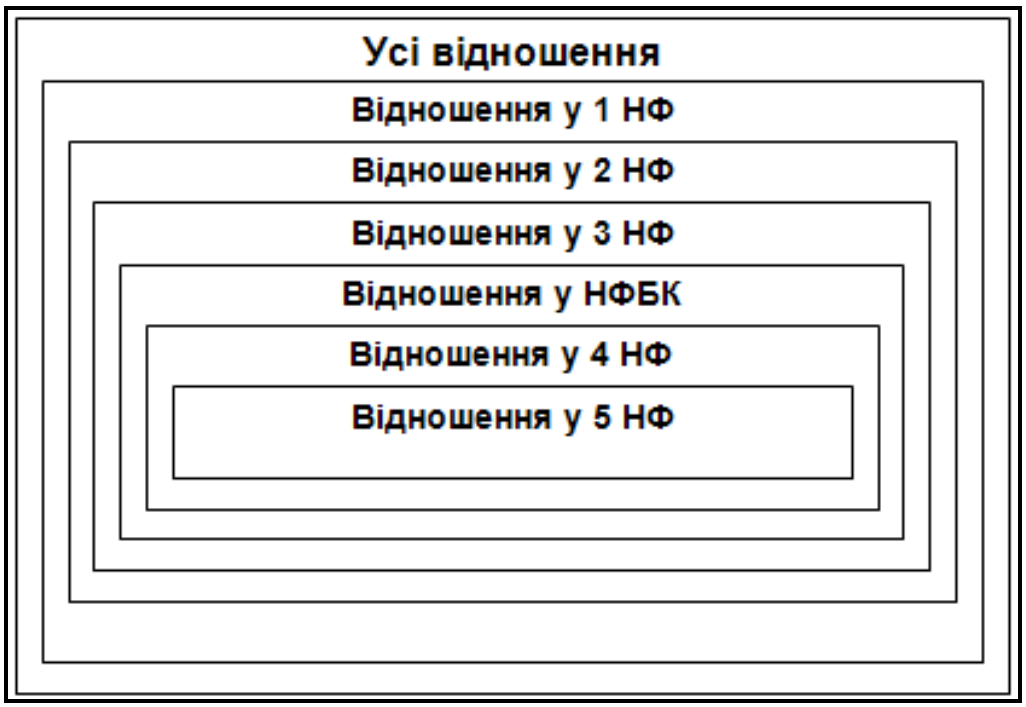


Рис. 4.1 **Співвідношення нормальних форм**

З рис. 4.1 видно, що усі умови, необхідні для деякої НФ, повинні виконуватися і для усіх подальших НФ.

Перші три НФ були визначені Коддом, наступна – нормальна форма Бойса-Кодда (НФБК) – Бойсом і Коддом, 4 і 5НФ були визначені Фейгіном.

Виникає питання, чи можна продовжити нормалізацію далі, отримати 6, 7 і так далі НФ? Дійсно, існують додаткові НФ, але 5НФ вважається в деякому розумінні остаточною. А для практичного проектування достатньою 3НФ.

4.2.2. Декомпозиція без втрат

Нормалізація використовує операцію декомпозиції. Річ у тому, що процедура нормалізації передбачає розбиття відношення на інші, тобто його ***декомпозицію***. Причому ця декомпозиція повинна статися ***без втрат*** інформації з первинного відношення. Можна сказати, що декомпозиція має бути ***оборотною***.

Наприклад, розглянемо вже відоме відношення **Студенти** (НомерЗаліковоїКнижки, Прізвище, І'мя, По батькові, КодГрупи, Адреса, Телефон).

Виконаємо декомпозицію на два відношення:

Студенти_1 (НомерЗалікової Книжки, КодГрупи, Прізвище, І'мя, По батькові) та

Студенти_2 (КодГрупи, Адреса, Телефон)

При такій декомпозиції інформація втрачається – для кожної Групи буде виведені адреса і телефон тільки першого за списком студента у цій групі. При з'єднанні отриманих відношень неможливо відновити повністю первинне відношення.

Якщо ж виконати декомпозицію по-іншому:

Студенти_3 (НомерЗалікової Книжки, Прізвище, І'мя, По батькові) і

Студенти_4 (НомерЗалікової Книжки, КодГрупи, Адреса, Телефон),

то отримаємо декомпозицію без втрат. Можна з'єднати отримані два відношення, отримавши початкове.

За своєю суттю декомпозиція – це операція ***проекції*** реляційної алгебри, тому кожне отримане при декомпозиції відношення називають проекціями первинного.

Виникає питання: яких умов потрібно дотримуватися для того, щоб проекції первинного відношення при зворотному з'єднанні гарантували отримання початкового відношення? На це питання дає відповідь **теорема Хеза (Heath)** [4]:

нехай **R (A, B, C)** – відношення, де **A, B, C** – підмножини множини його атрибутів;

якщо у **R** існує ФЗ **A** → **B**, то **R** дорівнює з'єднанню його проекцій **{A,B}** та **{A,C}**.

Слід звернути увагу, що теорема не затверджує "тоді і тільки тоді".

4.2.3. Перша, друга та третя нормальні форми

Відношення знаходиться в **1НФ** тоді і тільки тоді, коли усі домени, що в ньому використовуються, містять тільки скалярні значення. Іншими словами: Відношення знаходиться в **1НФ** тоді і тільки тоді, коли значення усіх полів неділимі.

Наприклад, якщо у відношенні є поле ПІБ, відношення не знаходиться в 1НФ, оскільки значення цього поля можна розділити на Прізвище, І'мя і По батькові.

Відношення знаходиться в **2НФ** тоді і тільки тоді, коли воно знаходиться у **1НФ** і кожен неключовий атрибут ***неприведено залежить*** від ***первинного ключа.***

Або по-іншому – якщо кожен його атрибут, що не є основним атрибутом, ***функціонально повно*** залежить від первинного ключа цього відношення.

Наприклад, розглянемо відношення **Успішність** (НомерЗаліковоїКнижки, Прізвище, І'мя, По батькові, Дисципліна, Оцінка).

Якщо первинним ключем тут призначити НомерЗаліковоїКнижки, то від цього ключа не буде залежати атрибут Дисципліна. Тобто в цьому випадку відношення не знаходиться у 2НФ.

Можна тоді в якості первинного ключа узяти множину атрибутів {НомерЗаліковоїКнижки, Дисципліна}. В цьому випадку від такого ключа залежать усі атрибути, але атрибути Прізвище, І'мя, По батькові залежать приведено, оскільки з детермінанта наступної ФЗ можна прибрати атрибут Дисципліна без порушення залежності :

{НомерЗаліковоїКнижки, Дисципліна} → {Прізвище, І'мя, По батькові}

І при такому первинному ключі відношення не знаходиться в 2НФ.

Щоб отримати відношення в 2НФ зробимо декомпозицію. Можна це зробити за теоремою Хеза.

Відношення Успішність задовольняє таким ФЗ:

{НомерЗаліковоїКнижки} → {Прізвище, І'мя, По батькові} **(A → B).**

Поза цією ФЗ залишилася наступна множина атрибутів : {Дисципліна, Оцінка} **(C).**

Тоді за теоремою Хеза відношення Успішність дорівнює з'єднанню його проекцій з такими множинами атрибутів :

{НомерЗаліковоїКнижки, Прізвище, І'мя, По батькові} ({A, B}).

{НомерЗаліковоїКнижки, Дисципліна, Оцінка} ({A, C})

Тобто, декомпозиція без втрат можлива на відношеннях:

Студенти (НомерЗаліковоїКнижки, Прізвище, І'мя, По батькові),

Успішність1 (НомерЗаліковоїКнижки, Дисципліна, Оцінка).

Відношення знаходиться в **3НФ** тоді і тільки тоді, коли воно знаходиться у **2НФ** і кожен неключовий атрибут нетранзитивно залежить від первинного ключа. Іншими словами: відношення знаходиться в 3НФ тоді і тільки тоді, коли кожен кортеж відношення складається зі значення первинного ключа, яке ідентифікує деякий об'єкт, і набору взаємно незалежних (чи порожніх) значень атрибутів, що описують цей об'єкт.

Наприклад, додамо у відношення Студенти атрибут СтаростаГрупи. Тоді у відношенні будуть такі ФЗ:

{НомерЗалікової Книжки} → {КодГрупи},

{КодГрупи} → {СтаростаГрупи}.

Тобто атрибут СтаростаГрупи залежить від первинного ключа (НомерЗалікової Книжки) транзитивно через атрибут КодГрупи, а не безпосередньо.

Значить, це відношення не знаходиться у 3НФ.

Проведемо декомпозицію. В даному випадку за теоремою Хеза може бути два варіанти декомпозиції.

1 варіант заснований на ФЗ {НомерЗалікової Книжки}→{Прізвище, І'мя, По батькові, КодГрупи}. У результаті отримаємо такі проекції:
{НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи} і
{НомерЗалікової Книжки, СтаростаГрупи}

2 варіант заснований на ФЗ {КодГрупи}→{СтаростаГрупи}. В результаті отримаємо такі проекції:
{КодГрупи, СтаростаГрупи},
{КодГрупи, НомерЗалікової Книжки, Прізвище, І'мя, По батькові}.

Підходить другий варіант, оскільки при першому варіанті можливі помилки при оновленні даних (користувач може ввести для конкретного студента старосту невірно, і система не видасть помилки). Така ситуація говорить про те, що теорема Хеза не завжди є вдалим і єдиним способом для вибору проекцій при декомпозиції. У результаті при декомпозиції на дві проекції:

Групи (КодГрупи, СтаростаГрупи) і

Студенти1 (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи) отримаємо відношення в 3НФ.

Таким чином, якщо відношення не знаходиться ні у 2НФ, ні у 3НФ, існує надмірність, яка призводить до так званих аномалій оновлення, тобто порушенні цілісності при вставці, видаленні або зміні даних.

Наприклад, розглянемо відношення, яке не знаходиться ні у 2НФ, ні у 3НФ.

Успішність (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи, СтаростаГрупи, Дисципліна, Оцінка).

Які можуть статися аномалії при оновленні даних в такому відношенні? Наприклад, такі:

1) При додаванні нового студента вірно вказавши Групу, в якій він вчиться, може бути помилково вказана староста Групи.

2) При видаленні одного із записів буде видалена не лише інформація про оцінку відповідного студента, але й інформація про те, в якій Групі він навчається.

3) При зміні старости Групи необхідно вручну змінити його для усіх записів студентів цієї ж Групи – при цьому можна помилитися.

Для запобігання подібним аномаліям, для цього відношення потрібно провести декомпозицію. Це зручно зробити, керуючись функціональними залежностями, які можна виявити у цьому відношенні:

{НомерЗалікової Книжки} –> {Прізвище, І'мя, По батькові, КодГрупи},

{КодГрупи}–>{СтаростаГрупи},

{НомерЗалікової Книжки, Дисципліна}–>{Оцінка}.

Як один із варіантів, первинне відношення Успішність можна розбити на три наступні проекції:

Студент (НомерЗалікової Книжки,Прізвище,І'мя,По батькові, КодГрупи),
Групи (КодГрупи, СтаростаГрупи),
Успішніть1 (НомерЗалікової Книжки, Дисципліна, Оцінка).

Усі отримані відношення знаходяться у 2НФ та 3НФ.

Але чи без втрат проведена ця декомпозиція? Так, оскільки після з'єднання цих трьох відношень отримаємо первинне.

Отримані відношення є ***незалежними проекціями***. Проекції R1 і R2 відношення R незалежні тоді і тільки тоді, коли виконуються дві умови:

1) кожна ФЗ відносно R є логічним наслідком ФЗ в проекціях R1 і R2;

2) загальні атрибути проекцій R1 і R2 утворюють потенційний ключ хоч би для однієї з них.

Для приведених трьох проекцій перша умова виконується, оскільки усі їх ФЗ забезпечують ФЗ первинного відношення.

Виконується і друга умова: для проекцій Студент і Групи загальний атрибут – КодГрупи, він первинний ключ для відношення Групи; для Студент і Успішність1 загальний атрибут – НомерЗалікової Книжки, він первинний ключ для Студент.

Якщо ж у для відношень, що були отримані у результаті декомпозиції якась з цих двох умов не виконується, то ці проекції не будуть незалежними.

Відношення, яке не може бути піддане декомпозиції з отриманням незалежних проекцій, називається **атомарним**.

Але це не означає, що будь-яке неатомарне відношення може бути розбите на атомарні відношення. І не завжди є сенс такого розбиття.

Наприклад, відношення

Студенти (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи) атомарно.

А відношення **Студенти2** (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи, Адреса, ДомашнійТелефон) неатомарно, оскільки домашній телефон залежить від адреси, тому відношення Студенти2 можна розбити на такі незалежні проекції:

Студенти3 (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, Адреса) і

АдресаТелефони (Адреса, ДомашнийТелефон). Але в цьому немає сенсу.

4.2.4. Нормальна форма Бойса-Кодда

При визначенні 3НФ робилося допущення про те, що відношення має тільки один потенційний ключ, який і є первинним. Але визначення 3НФ не зовсім підходить для відношень з наступними властивостями:

1) відношення має два або більше потенційних ключа;
2) потенційні ключі складні;
3) потенційні ключі перекриваються, тобто мають хоч би один загальний атрибут.

Для відношень, що мають ці властивості даного раніше визначення 3НФ недостатньо, тому Бойс і Кодд ввели визначення НФ Бойса-Кодда (НФБК).

Відношення знаходиться в **НФБК** тоді і тільки тоді, коли кожна нетривіальна і неприведена ліворуч ФЗ має **потенційний ключ в якості детермінанта**.

На практиці відношення з комбінацією перерахованих властивостей зустрічаються украй рідко, а для відношень без цих властивостей 3НФ і НФБК еквівалентні.

Можна помітити, що визначення НФБК простіше, ніж 3НФ, оскільки в нім немає згадки 2НФ і не використовується поняття транзитивної залежності.

Наприклад, розглянемо відношення з двома потенційними ключами, що перекриваються [4].

Успішність (НомерЗалікової Книжки, КодДисципліни, НазваДисципліни, Оцінка).

Потенційні ключі: {НомерЗалікової Книжки, КодДисципліни} і {НомерЗалікової Книжки, НазваДисципліни}.

Це відношення не в НФБК, оскільки у ньому є такі ФЗ:

КодДисципліни → НазваДисципліни

НазваДисципліни → КодДисципліни

А детермінанти цих ФЗ не є потенційними ключами.

Це відношення також не у 3НФ, оскільки не знаходиться у 2НФ, тому що атрибут НазваДисципліни залежить тільки від частини первинного ключа, а саме КодДисциплини. Як наслідок, цьому відношенню властива надмірність.

Можлива декомпозиція на два відношення:

Дисципліна (КодДисципліни, НазваДисципліни),

Успішність1 (НомерЗалікової Книжки, КодДисципліни, Оцінка).

Розглянемо приклад відношення, яке знаходиться у 3НФ, але не знаходиться в НФБК. Нехай у БД є таке відношення:

Лекції (КодСпеціальності, НазваСпеціальності, Дисципліна, Лектор) (рис.4.2):

КодСпеціальності	НазваСпеціальності	Дисципліна	Лектор
401	Інф.Управл.Системи	ООП	Щербаков О.В.
407	Комп.Екол.Моніторінг	ООП	Парфьонов Ю.Е.
401	Інф.Управл.Системи	ОБДЗ	Тарасов О.В.
407	Комп.Екол.Моніторінг	ОБДЗ	Федько В.В.
…			

Рис. 4.2 **Загальний вигляд відношення Лекції**

Введемо таке обмеження: кожну дисципліну у конкретної спеціальності веде тільки один Лектор, але одну і ту ж дисципліну можуть читати різні викладачі.

У цьому відношенні можна виділити наступні ФЗ:
{КодСпеціальності} -> {НазваСпеціальності},
{НазваСпеціальності} -> {КодСпеціальності},
{КодСпеціальності, Дисципліна} -> {Лектор),
{НазваСпеціальності, Дисципліна} -> {Лектор).

А у якості первинного ключа можуть виступати складені ключі КодСпеціальності+Дисципліна або НазваСпеціальності+Дисципліна.

Це відношення знаходиться у 2НФ, оскільки воно не містить неключових атрибутів, залежних від частини складеного первинного ключа.

Крім того, відношення не містить залежних один від одного неключових атрибутів, оскільки неключовий атрибут всього один – Лектор, отже, відношення Лекції знаходиться у 3НФ.

Проте з аналізу цього відношення можна зробити висновок, що дані зберігаються у ньому з надмірністю – при зміні найменування спеціальності, це найменування треба змінити в усіх кортежах, де воно зустрічається. Для усунення цієї аномалії потрібно розбити початкове відношення Лекції, наприклад, на такі:

Спеціальності (КодСпеціальності, НазваСпеціальності)
Лекції1 (КодСпеціальності, Дисципліна, Лектор).
Ця декомпозиція переведе початкове відношення у НФБК.

4.3. Нормальні форми більш високого порядку

4.3.1. Багатозначні залежності

Для пояснення 4НФ необхідно ввести поняття багатозначної залежності (БЗ). Нехай A, B, C – довільні підмножини множини атрибутів відношення R. Тоді B ***багатозначно залежить*** від A тоді і тільки тоді, коли множина значень B, що відповідає заданій парі значень (значення A, значення C) відношення R, залежить тільки від A, але не залежить від C.

Позначається: A ⏩ B

Кажуть, що "A багатозначно визначає B" або "B багатозначно залежить від A" або "A подвійна стрілка B".

Спрощено можна сказати так: A багатозначно визначає B, якщо для кожного значення A не існує ***єдиного значення*** *B*, що відповідає їй, тобто не існує ФЗ A $\rightarrow$ B, але кожне значення A визначає множину значень B, що їй відповідають.

По суті ФЗ – це окремий випадок БЗ.

Для пояснення суті багатозначної залежності розглянемо приклад [24]:

Нехай є відношення **ДВК** (Дисципліна, Викладач, Кабінет) (рис. 4.3).

Дисципліна	Викладач	Кабинет
ООП	Щербаков О.В.	ОЦ-14
ООП	Щербаков О.В.	ОЦ-411
ООП	Парфьонов Ю.Е.	ОЦ-14
ООП	Парфьонов Ю.Е.	ОЦ-411
КПП	Парфьонов Ю.Е.	ОЦ-4
КПП	Парфьонов Ю.Е.	ОЦ-5
КПП	Поляков А.О.	ОЦ-4
КПП	Поляков А.О.	ОЦ-5
…		

Рис. 4.3. **Загальний вид відношення ДВК**

Введемо наступні обмеження:

1) кожній дисципліні може відповідати будь-яка кількість викладачів і будь-яка кількість кабінетів;

2) викладачі і кабінети не залежать один від одного (тобто незалежно від того, хто викладає цю дисципліну, для цієї дисципліни використовується один і той же набір кабінетів);

3) конкретний викладач і конкретний кабінет можуть бути пов'язані з будь-якою кількістю дисциплін.

В цьому випадку первинний ключ відношення – множина усіх атрибутів. Крім того це відношення знаходиться у:

1) 1НФ – оскільки атрибути неділимі (вважатимемо, що замість ПІБ викладача використовується деякий код викладача, наприклад, табельний номер, але для зручності в прикладі використовуватимемо ПІБ).

2) 2НФ – оскільки неключових атрибутів взагалі немає.

3) 3НФ – з тієї ж причини.

4) НФБК – оскільки в ньому немає декількох потенційних ключів, що перекриваються.

Отже, можна зробити висновок, що повністю ключове відношення (якщо воно знаходиться у 1НФ) знаходиться і у 2НФ, 3НФ та в НФБК.

Базуючись на обмеженні №2 можна стверджувати, що якщо існують два кортежі:

{Дисципліна1, Викладач1, Кабинет1} та

{Дисципліна1, Викладач2, Кабинет2},

то існують і кортежі

{Дисципліна1, Викладач1, Кабинет2} та

{Дисципліна1, Викладач2, Кабинет1}.

Наприклад, якщо існують кортежі:

{КПП, Парфьонов Ю.Е., ОЦ-5} та { КПП, Поляков А.О., ОЦ-4},

то існують і кортежі

{КПП, Парфьонов Ю.Е., ОЦ-4} та {КПП, Поляков А.О., ОЦ-5}

В цьому випадку відношення явно надмірне і це може призвести до аномалій оновлення. Наприклад, для додавання інформації про те, що Дисципліна ООП вестиметься новим викладачем Лосєвим М. Ю. необхідно створити стільки нових кортежів, скільки кабінетів підходять

для цієї дисципліни. А при цьому помилково можна ввести кортежі не для усіх кабінетів або, навпаки, із зайвими кабінетами.

Існування подібних проблем викликане, як правило, незалежністю атрибутів, у даному випадку Викладач і Кабінет. Для позбавлення від надмірності і аномалій оновлення можна замінити відношення ДПК двома його проекціями:

ДВ (Дисципліна, Викладач) і **ДК** (Дисципліна, Кабінет)

Обидві проекції повністю ключові, отже, знаходяться у НФБК.

Крім того, це декомпозиція без втрат, оскільки при з'єднанні цих двох відношень по атрибуту Дисципліна отримаємо первинне відношення ДПК.

Виконати процес декомпозиції відношення ДПК ґрунтуючись на ФЗ неможливо, оскільки усі ФЗ у відношенні тривіальні.

Але проведену декомпозицію можна зробити на основі БЗ, оскільки у відношенні їх дві:

Дисципліна ⏩ Викладач

Дисципліна ⏩Кабінет

Математичне обґрунтування такої декомпозиції було зроблено Фейгіним у 1971 році саме за допомогою поняття багатозначних залежностей.

Перша з наведених залежностей означає, що хоча для кожної дисципліни не існує ***єдиного*** викладача (тобто ФЗ Дисципліна -> Викладач є неправильною), але кожна Дисципліна визначає ***множину*** викладачів, що їй відповідають.

Друга означає, що хоча для кожної дисципліни не існує ***єдиного*** кабінету (не вірна ФЗ Дисципліна -> Кабінет), але кожна Дисципліна визначає ***множину*** кабінетів, що їй відповідають.

По цих БЗ і можна провести декомпозицію.

Можна помітити, що для відношення ДПК багатозначна залежність Дисципліна ⏩ Викладач виконується тоді і тільки тоді, коли виконується БЗ Дисципліна ⏩ Кабінет.

Для подібних відношень це виконується завжди, тобто в узагальненому виді можна сформулювати ***правило багатозначних залежностей:***

Для R (A, B, C) існує A ⏩ B тоді і тільки тоді, коли існує A ⏩ C.

Таким чином, БЗ утворюють пари і їх зазвичай представляють так: A ⏩B | C.

Для ДПК можна записати Дисципліни ⏩ Викладачі | Кабінети.

Теорема Фейгіна. Нехай A, B, C – підмножини множини атрибутів відношення R. Відношення R дорівнюватиме з'єднанню його проекцій {A, B} та {A, C} тоді і тільки тоді, коли для відношення R виконується БЗ A ⏩B | C.

4.3.2. Четверта нормальна форма

Відношення R знаходиться в **4НФ** тоді і тільки тоді, коли

1) якщо існують такі підмножини A і B множини атрибутів R, що виконується нетривіальна БЗ A ⏩ B, то усі інші атрибути відношення R функціонально залежать від атрибуту A.

Інше формулювання:

2) Відношення R знаходиться в 4НФ якщо воно знаходиться в **НФБК** і усі **БЗ** відношення R фактично **є ФЗ** від потенційних ключів.

Відношення ДПК з попереднього прикладу не знаходиться в 4НФ, оскільки (по першому формулюванню) :

1) Існує БЗ Дисципліна ⏩ Викладач, але при цьому атрибут Кабінет не залежить функціонально від атрибуту Дисципліна.

2) Існує БЗ Дисципліна ⏩ Кабінет, але при цьому атрибут Викладач не залежить функціонально від атрибуту Дисципліна.

Але обидві проекції ДП і ДК знаходяться в 4НФ, оскільки в кожній з них існує по одній БЗ, і інших атрибутів (що не входять у ці БЗ) немає.

Фейгін довів також, що 4НФ завжди може бути отримана, тобто будь-яке відношення, що містить БЗ, може бути піддане декомпозиції без втрат на набір відношень у 4НФ.

4.3.3. Залежність з'єднання

Не завжди можна провести декомпозицію без втрат одного відношення на дві проекції. Якщо це неможливо, але можливо розбити відношення три або більше за проекції, таке відношення називають **n-декомпозуємим**, де **n** – кількість проекцій, на які можна розбити відношення без втрат.

Наприклад, розглянемо відношення **ЛДС** (Література, Дисципліна, Спеціальність) [24] яке є повністю ключове. Це відношення виражає інформацію про те, що деяка наявна у бібліотеці книга підходить для деякої дисципліни, що вивчається на певній Спеціальності (рис. 4.4).

Література	**Дисципліна**	**Спеціальність**
Хомоненко А. и др. Базы данных	ОБДЗ	Інформ.управл.системи
Хомоненко А. и др. Базы данных	ІС	Комп.еколог.моніторинг
Хомоненко А. и др. Базы данных	ІС	Інформ.управл.системи
Дейт К. Введение в системы баз данных	ОБДЗ	Інформ.управл.системи
Дейт К. Введение в системы баз данных	ІС	Інформ.управл.системи

Рис. 4.4. **Загальний вигляд відношення ЛДС**

Виконаємо декомпозицію на три проекції: ЛД, ДС, ЛС (рис. 4.5.).

ЛД	
Література	**Дисципліна**
Хомоненко	ОБДЗ
Хомоненко	ІС
Дейт	ОБДЗ
Дейт	ІС

ДС	
Дисципліна	**Спеціальність**
ОБДЗ	Інформ. управл. системи
ІС	Комп. еколог. моніторинг
ІС	Інформ. управл. системи

ЛС	
Література	**Спеціальність**
Хомоненко	Інформ. управл. системи
Хомоненко	Комп. еколог. моніторинг
Дейт	Інформ. управл. системи

Рис. 4.5. **Декомпозиція відношення ЛДС на три відношення**

З'єднаємо назад тільки дві проекції, ЛД та ДС по атрибуту Дисципліна, і порівняємо кортежі отриманого відношення з кортежами первинного відношення ЛДС (рис. 4.6.):

Література	Дисципліна	Спеціальність	
Хомоненко	ОБД3	Інформ.управл.системи	- є
Хомоненко	ІС	Комп.еколог.моніторинг	- є
Хомоненко	ІС	Інформ.управл.системи	- є
Дейт	ОБД3	Інформ.управл.системи	- є
Дейт	ІС	Комп.еколог.моніторинг	- нема!
Дейт	ІС	Інформ.управл.системи	- є

Рис. 4.6. **Результат зворотного з'єднання двох відношень**

В результаті отримали зайвий кортеж. Аналогічна ситуація буде і при з'єднанні ЛД і ЛС, а також ДС і ЛС. А якщо з'єднати усі три проекції, то отримаємо саме первинне відношення ЛДС.

Таким чином, можна зробити висновок, що відношення ЛДС 3-декомпозуємо.

У зв'язку з такими n-декомпозиціями вводиться поняття залежності з'єднання (ЗЗ), на основі якого потім визначається 5НФ.

Нехай R – відношення, A, B, ., Z – довільні підмножини множини атрибутів R. **Відношення R задовольняє залежності з'єднання *(A, B, ., Z)** тоді і тільки тоді, коли воно рівносильне з'єднанню своїх проекцій з підмножинами атрибутів A, B, ., Z.

Для нашого прикладу відношення ЛДС задовольняє залежності з'єднання

*(ЛД, ДС, ЛС) чи

*({Література, Дисципліна},{Дисципліна, Спеціальність},{Література, Спеціальність}).

Як ФЗ є окремим випадком БЗ, так і БЗ є окремим випадком ЗЗ.

4.3.4. П'ята нормальна форма

Відношення R знаходиться в **5НФ**, яка також називається ***проекційно-сполучною НФ***, тоді і тільки тоді, коли кожна ЗЗ відносно R мається на увазі потенційними ключами відношення R.

Інше визначення 5НФ таке: таблиця знаходиться у 5НФ, якщо вона знаходиться в 4НФ і будь-яка багатозначна залежність з'єднання у неї є тривіальною.

П'ята нормальна форма більшою мірою є теоретичним дослідженням і практично не застосовується при реальному проектуванні баз даних. Це пов'язано із складністю визначення самої наявності залежностей "проекції – з'єднання", оскільки твердження про наявність такої залежності має бути зроблене для усіх можливих станів БД.

4.4. Підсумкова схема процедури нормалізації

Таким чином, розглянувши базові питання, пов'язані з нормалізацією відношень, можна зробити такий висновок – основна мета цієї процедури така [24; 28]:

1. Виключення надмірності.
2. Усунення аномалій оновлення.
3. Проектування макету даних, який відповідав би реальному світу, був інтуїтивно зрозумілий і служив основою для подальшого розвитку.
4. Спрощення процесу накладення обмежень цілісності. Ця мета пов'язана з потенційними ключами, тобто якщо дотримуватися умови унікальності потенційних ключів і організовувати зв'язки тільки через них, то ця мета буде досягнута.

Нехай є відношення R, яке знаходиться в 1НФ. Також відомі усі обмеження цього відношення, тобто ключі, ФЗ, МЗ і ЗЗ. Тоді основна ідея нормалізації відношення R полягає в декомпозиції без втрат, принципи якої полягають у такому:

1) послідовно відношення R наводиться до набору менших відношень, який еквівалентний відношенню R, але більш прийнятний;

2) кожен етап цього процесу складається з розбиття на проекції відношень, отриманих на попередньому етапі;

3) при цьому усе задані залежності (ФЗ, МЗ, ЗС) використовуються на кожному кроці для вибору проекцій наступного етапу.

Таким чином можна сформулювати перелік основних правил, на які спирається процедура нормалізації:

1. Відношення в 1НФ слід розбити на проекції для виключення усіх приведених ФЗ. В результаті отримаємо набір відношень в 2НФ.

2. Відношення в 2НФ слід розбити на проекції для виключення усіх транзитивних ФЗ. В результаті отримуємо набір відношень в 3НФ.

3. Відношення в 3НФ слід розбити на проекції для виключення усіх ФЗ, в яких детермінанти не є потенційними ключами. В результаті з'являється набір відношень у НФБК.

Примітка. Три правила можна сконцентрувати в одному: "початкове відношення слід розбити на проекції для виключення усіх ФЗ, в яких детермінанти не є потенційними ключами".

4. Відношення у НФБК слід розбити на проекції для виключення усіх БЗ, які не є ФЗ. В результаті – набір відношень у 4НФ. (На практиці такі БЗ зазвичай виключаються перед першим етапом, тобто відділяються незалежні Групи, що повторюються)

5. Відношення в 4НФ слід розбити на проекції для виключення усіх ЗЗ, які не маються на увазі потенційними ключами (якщо такі ЗЗ можна виявити). В результаті – набір відношень в 5НФ.

До основних правил можна додати такі доповнення:

1. Процес розбиття на проекції має бути виконаний без втрат і зберігаючи залежності початкових даних.

2. Бувають випадки, коли зручніше міняти послідовність процедури нормалізації.

3. Хоча ідеї нормалізації важливі і найбільш прийнятні для проектування БД, вони не є універсальним засобом з наступних причин:

- окрім ФЗ, МЗ і ЗС можуть існувати і інші типи залежностей і обмежень – специфічні для кожної БД;
- декомпозиція може бути не унікальною, тобто можуть існувати різні її варіанти і критерій вибору найкращого з них іноді зовсім не очевидні;
- не всяку надмірність даних можна усунути розбиттям на проекції.

Тим не менш, незважаючи на ці зауваження, можна сказати, що методика проектування БД, яка базується на нормалізації відношень,

дозволяє створити несуперечливий, нормалізований, дієздатний макет БД.

4.5. Денормалізація відношень

Для усунення аномалій при виконанні операцій по модифікації даних відповідно до теорії реляційних баз даних необхідно, щоб будь-яка база даних була нормалізована хоч би до третьої нормальної форми, що дозволяє мінімізувати надмірність бази даних і забезпечує найбільшу теоретично доступну гнучкість.

Проте дуже часто нормалізація вступає в протиріччя з вимогами ефективності роботи з базою даних. В результаті нормалізації цілісні таблиці розбиваються на пов'язані посиланнями набори таблиць, що значно знижує ефективність виконання запитів. Тому, якщо ефективність виявляється важливіша, ніж гнучкість і об'єм БД, може проводитися денормалізація – тобто зворотне перетворення БД, при якому пов'язані посиланнями таблиці об'єднуються для більш ефективного доступу. Природно, що при денормалізації можливе виникнення аномалій у базі даних.

Денормалізація – це не результат поганого проектування бази даних, це навмисне порушення нормальних форм, для збільшення продуктивності системи [34].

У яких випадках денормалізація може бути виправдана? Відповіддю на це питання можуть служити наступні міркування:

1. Якщо при виконанні запитів до бази даних дуже часто виникає необхідність у з'єднанні великої кількості таблиць, то краще об'єднувати в одну декілька таблиць, що мають невеликий розмір, містять рідко змінювану (як говорять, умовно-постійну, або нормативно-довідкову) інформацію, причому інформацію, за смислом тісно пов'язану між собою.

2. Запити, в яких здійснюється розрахунок окремих показників, на основі інформації, що зберігається у базі даних, особливо при використанні угрупувань та агрегатних функцій (COUNT, MAX, SUM тощо) дуже часто виконуються достатньо довго. Внаслідок цього, введення додаткового стовпця, який би містив значення, які б заздалегідь розраховувалися, дозволяє значно заощадити час при виконанні запиту, проте вимагає своєчасної зміни даних в цьому стовпці.

3. Якщо у базі даних є таблиці з великою кількістю записів, що містять поля великого розміру, наприклад Blob, то значно збільшити швидкість виконання запитів до такої таблиці можна, якщо винести такі поля в окрему таблицю і встановити з нею зв'язок типу 1:1 з початковою. Сенс такого розділення полягає в тому, що інформація, що міститься в таких полях рідко буває потрібна при виконанні звичайних запитів і як наслідок, значно зменшивши розмір початкової таблиці, швидкість обробки може зрости дуже сильно.

Проте необхідно відмітити, що проводити денормалізацію якщо і необхідно, то тільки у тому випадку, коли повністю і правильно виконана нормалізація БД і визначені ті "вузькі місця" в схемі, які можуть знижувати ефективність виконання запитів. Технологія проведення денормалізації дуже специфічна у кожному конкретному випадку. Детальніше ознайомитися з цими питаннями можна в роботах [20; 25; 32; 33].

Питання для самодіагностики

1. Для яких відношень зручніше було б поміняти описану в правилах послідовність нормалізації. Наведіть приклади.
2. Чи можна побудувати теорію нормалізації, ґрунтуючись не на операціях проекції і з'єднання, а на яких-небудь інших реляційних операціях? Якщо ні – обґрунтуйте, якщо так – наведіть приклад таких операцій.
3. Поясніть сенс понять: відношення, змінна відношення, значення відношення, заголовок відношення, тіло відношення, атрибут, первинний ключ, ступінь відношення, домен, кортеж, кардинальне число.
4. Які реляційні об'єкти даних пов'язані із заголовком відношення?
5. Які реляційні об'єкти даних пов'язані із тілом відношення?
6. Що загального і в чому відмінності понять "Змінна відношення" і "змінна" (у мові програмування)?
7. Доведіть, що твердження "у відношенні завжди існує первинний ключ" є наслідком властивості відношення, в якому говориться про те, що немає однакових кортежів.

8. Як ви розумієте, що таке: правила цілісності, потенціальний ключ, первинний ключ, альтернативний ключ, зовнішній ключ, правило посилальної цілісності, вторинний ключ.

9. Що таке Null-значення?

10. В чому відмінність первинного і потенційного ключа?

11. Чи може не існувати у відношенні первинного, потенційного або альтернативного ключа?

12. Чи може первинний ключ одночасно бути зовнішнім ключем? Якщо ні – доведіть, якщо так – наведіть приклади.

13. Що таке реляційна алгебра?

14. Які реляційні операції ви знаєте? Наведіть приклади.

15. Як ви розумієте смисл терміну "реляційно повна мова".

16. У БД Факультет містяться відношення:

Студенти (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, ДатаНародження, Адреса, Телефон, КодГрупи),

Стіпендіанти (НомерЗалікової Книжки, Прізвище, І'мя, По батькові, КодГрупи, РозмірСтипендії)

Групи (КодГрупи, Спеціальність, Курс)

Запишіть в реляційних виразах наступні запити:

А) Отримати список студентів, що одержують стипендію.

Б) Отримати список студентів, які не одержують стипендію.

В) Отримати список студентів, у яких немає телефону.

Г) Отримати список прізвищ і імен студентів із вказівкою для них усіх відомостей по Групах, в яких вони вчаться.

Д) Отримати список студентів із вказівкою віку.

Е) Отримати кількість Груп кожної спеціальності.

17. Поясніть сенс понять: функціональна залежність, тривіальна функціональна залежність, неприведена функціональна залежність.

18. В чому смисл аксіом Армстронга?

19. Що таке "декомпозиція без втрат"?

20. Дайте визначення поняттям: перша нормальна форма, друга і так далі

21. Поясніть, що таке "аномалії оновлення"? Наведіть приклади.

22. Що таке багатозначна залежність? Наведіть приклади.

5. Приклад нормалізації відношення

Мета – ознайомитися на конкретному прикладі з процедурою нормалізації початкового відношення.

Компетенції, що формуються:

Спеціалізовано-професійні – знання загальнометодологічних принципів проектування схеми реляційної бази даних.

Спеціалізовано-професійні – знання основних підходів, методів і технологій створення реляційних моделей баз даних; знання сучасних теорій організації баз даних; знання теоретичних і практичних основ методології та технології моделювання реляційних баз даних.

Основні питання

Опис предметної області "Реалізація хлібобулочних виробів" – наводиться короткий опис предметної області та будується словик даних до неї.

Визначення обмежень та залежностей між атрибутами – визначається перелік обмежень на окремі атрибути та формується множина функціональних залежностей, які присутні у наведеній предметній області.

Формування початкового відношення для аналізованої предметної області – наведений приклад формування початкового (базового) відношення та знаходження його первинного ключа.

Нормалізація початкового відношення – безпосередньо наводиться покрокова процедура нормалізації початкового відношення згідно з визначеною схемою.

Розглянемо приклад нормалізації відношення, яке характеризує реалізацію Хлібобулочних виробів, що випускаються різними виробниками, через торгові точки міста [18].

5.1 Опис предметної області "Реалізація хлібобулочних виробів"

Існує ряд підприємств, що випускають різноманітну Хлібобулочну продукцію. Кожен вид продукції випускається у відповідності в державним стандартом (ДСТУ). Щодня, вироблена продукція

відпускається, згідно з укладеними договорами на постачання продукції, фірмам-реалізаторам, які безпосередньо продають Хлібобулочні вироби населенню. За отриману продукцію, фірма-реалізатор перераховує відповідні грошові кошти на банківський розрахунковий рахунок фірми-виробника.

Для описаної предметної області побудуємо словник даних (табл. 5.1.).

Таблиця 5.1

Словник даних до предметної області

№	Найменування елемента	Ідентифікатор	Тип та довжина	Призначення елемента
1	Постачальник	Postavshik	VARCHAR(30)	Найменування фірми-виробника ХБВ
2	Реалізатор	Realizator	VARCHAR(30)	Найменування фірми-реалізатора ХБВ
3	№ Договору	No_Dogovora	VARCHAR(6)	Номер договору на постачання ХБВ
4	Банк Постачальника	Bank_Postav	VARCHAR(30)	Найменування банку Постачальника
5	Р/Р Постачальника	RS_Postav	NUMBER(8,0)	Номер розрахункового рахунку Постачальника у банку
6	МФО БанкаПостачальника	MFO_Banka_Postav	NUMBER(6,0)	МФО банку Постачальника
7	Товар	Tovar	VARCHAR(30)	Найменування ХБВ
8	ДСТУ на товар	DSTU_Tovar	VARCHAR(10)	Номер державного стандарту на виробництво ХБИ відповідного типу
9	Дата постачання	Data_Postavki	DATE	Дата постачання ХБВ фірмі-реалізатору
10	Кількість	Kolvo	INTEGER	Об'єм в шт. Хлібобулочної продукції відповідного виду, що поставляється

5.2. Визначення обмежень та залежностей між атрибутами

В рамках визначеного переліку атрибутів, необхідно виявити обмеження, яким повинні задовольняти значення цих атрибутів. Це можуть бути обмеження на обов’язковість чи не обов’язковість значення,

на умови, яким повинно відповідати значення, на співвідношення між значеннями різних атрибутів тощо.

Аналізуючи атрибути, включені в словник даних, можна навести наступний перелік обмежень на їх можливі значення (табл. 5.2).

Таблиця 5.2

Обмеження на значення атрибутів предметної області

№	Найменування елемента	Обмеження
1	Постачальник	Поле повинно бути обов'язковим (NOT NULL)
2	Реалізатор	Поле повинно бути обов'язковим (NOT NULL)
3	№ Договору	Поле повинно бути обов'язковим (NOT NULL)
4	Банк Постачальника	Поле повинно бути обов'язковим (NOT NULL)
5	Р/Р Постачальника	Поле повинно бути обов'язковим (NOT NULL)
6	МФО БанкаПостачальника	Поле повинно бути обов'язковим (NOT NULL)
7	Товар	Поле повинно бути обов'язковим (NOT NULL)
8	ДСТУ на товар	Поле повинно бути обов'язковим (NOT NULL). Значення повинно мати не менш 4 цифр
9	Дата постачання	Поле повинно бути обов'язковим (NOT NULL). Значення введеної дати, не може бути більше поточної. Рік введеної дати, повинен дорівнювати поточному року.
10	Кількість	Поле повинно бути обов'язковим (NOT NULL). Значення позитивне.

Окрім виявлення обмежень на значення атрибутів, дуже важливу, навіть ключову, роль відіграють функціональні залежності, які існують (або можуть потенційно існувати) в рамках початкового відношення. Саме базуючись на виявлених функціональних залежностях проводять процес нормалізації відношень.

Виходячи з аналізу предметної області, можна навести такі функціональні залежності між атрибутами або групами атрибутів.

- ***Номер договору*** залежить від значень атрибутів **Постачальник** і **Реалізатор**;
- ***Номер ДСТУ*** залежить від виду хлібобулочного виробу (**Товару**);
- ***Розрахунковий рахунок, Банк і МФО*** залежать від **Постачальника**;
- ***МФО банку*** залежить від **Банку**;

До речі, на одному й тому ж переліку атрибутів аналогічної предметної області можна виявити і зовсім протилежні функціональні залежності. Наприклад можна сказати, що саме **Номер договору** визначає значення атрибутів ***Постачальник*** *і* ***Реалізатор,*** або ***Постачальник*** залежить від **Розрахункового рахунку** та **Банку** тощо. Але в даному прикладі будемо виходити з означених вище функціональних залежностей.

5.3. Формування початкового відношення для аналізованої предметної області

Початкове відношення визначимо у вигляді такої таблиці (рис. 5.1):

Це відношення відповідає початковому опису предметної області і відображає такі факти:

1. Існує ряд підприємств, що випускають Хлібобулочну продукцію (Постачальник). Назви постачальників відрізняються один від одного.
2. Є множина фірм, що займаються реалізацією Хлібобулочної продукції (Реалізатор). Назви фірм-реалізаторів відрізняються.
3. У Постачальників укладені угоди на постачання продукції з Реалізаторами, які перераховують гроші за отриманий товар на розрахунковий рахунок Постачальника, що знаходиться у відповідному банку.
4. Щодня Постачальники поставляють Реалізаторам різноманітну Хлібобулочну продукцію (Товар), що випускається відповідно до вимог державного стандарту (ДСТУ) у певній кількості.

Це відношення знаходиться в 1НФ, оскільки усі значення в ньому атомарні і воно не містить кортежів, що повторюються.

Оскільки відношення знаходиться у 1НФ, воно повинно мати первинний ключ, значення якого однозначно визначає кожен кортеж. Для цього відношення первинний ключ є складеним і включає такі атрибути: ***Постачальник, Реалізатор, Товар, Дата продажу***.

Аналізуючи приведене відношення визначимо залежності між атрибутами або Групами атрибутів у ньому.

Від первинного ключа залежить тільки значення атрибуту Кількість, яке характеризує об'єм закупленого Реалізатором у Постачальника Товару за конкретну Дату. Інші атрибути не залежать від первинного ключа,

ПОСТАЧАННЯ ХЛІБОБУЛОЧНИХ ВИРОБІВ

Постачальник	Реалізатор	№Договору	Р/Р Постачал.	Банк Постачал.	МФО БП	Товар	ДСТУ на товар	Дата постач	Кіль-кість
АТ "Кулінічі"	ТФ"Барс"	12-091	26000001	Синтез	851300	Дарницький	4583	10.10	200
АТ "Кулінічі"	ТФ"Булка"	045-11	26000001	Синтез	851300	Нарізний	4583	10.10	250
ЗАТ "Хліб"	ТФ"Булка"	23-011	26010234	Ексімбанк	851301	Нарізний	4583	10.10	350
ЗАТ "Хліб"	ТФ"Булка"	23-011	26010234	Ексімбанк	851301	Сімейка	4585	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	114-01	26010234	Ексімбанк	851300	Сімейка	4585	10.10	350
ТОВ "Салтівський"	ТФ"Сдоба"	14-235	26000234	Синтез	851300	Білий	4583	10.10	245
АТ "Кулінічі"	ТФ"Барс"	12-091	26000001	Синтез	851300	Нарізний	4583	11.10	500
ЗАТ "Хліб"	ТФ"Булка"	23-011	26010234	Ексімбанк	851301	Український	4583	11.10	150
ТОВ "Салтівський"	ТФ"Сдоба"	14-235	26000234	Синтез	851300	Ржаний	4583	11.10	200
ТОВ "Салтівський"	ТФ"Булка"	045-13	26000234	Синтез	851300	Завиток	4585	12.10	300
ЗАТ "Хліб"	ТФ"Сдоба"	114-01	26010234	Ексімбанк	851301	Сімейка	4585	12.10	120
ТОВ "Салтівський"	ТФ"Сдоба"	14-235	26000234	Синтез	851300	Завиток	4585	12.10	310

Рис. 5.1. **Загальний вигляд початкового відношення**

проте існують ще наступні Функціональні залежності, визначені у постановці завдання:

1) {Постачальник, Товар, Реалізатор, Дата постачання} -> {Кількість}

2) {Постачальник, Реалізатор} -> {Номер договору}

3) {Товар} -> {ДСТУ на товар}

4) {Постачальник} -> {Р/Р у банку, Банк, МФО}

5) {Банк} -> {МФО}

5.4. Нормалізація початкового відношення

Для перетворення цього відношення з 1НФ в 2НФ, необхідно декомпозувати відношенні так, щоб в усіх отриманих відношеннях кожен його атрибут, що не є основним атрибутом, функціонально повно залежав від первинного ключа цього відношення. Для цього усі детермінанти виявлених функціональних залежностей повинні бути первинними ключами.

Таким чином початкове відношення в 1НФ "**ПОСТАЧАННЯ ХЛІБОБУЛОЧНИХ ВИРОБІВ**", перетвориться в 4 відношення в 2НФ:

1) **ЗАКУПІВЛІ** (**Постачальник, Товар, Реалізатор, Дата постачання**, Кількість).

2) **ДОГОВІР НА ПОСТАЧАННЯ** (**Постачальник, Реалізатор**, Номер договору).

3) **СТАНДАРТ НА ТОВАР** (**Товар**, ДСТУ на товар).

4) **БАНКІВСЬКІ РЕКВІЗИТИ ПОСТАЧАЛЬНИКА** (**Постачальник**, Р/Р у банку, Банк, МФО).

Цей набір відношень знаходиться у 2НФ, але не в 3НФ, оскільки в результаті аналізу початкового відношення була виявлена ще одна Функціональна залежність {Банк} -> {МФО}, а отже відношення "БАНКІВСЬКІ РЕКВІЗИТИ ПОСТАЧАЛЬНИКА" містить транзитивну залежність {Постачальник} -> { Банк} -> {МФО}.

Для перетворення відношення у 3НФ необхідно позбавитися від виявленої транзитивної залежності. Для цього відношення "БАНКІВСЬКІ РЕКВІЗИТИ ПОСТАЧАЛЬНИКА" декомпозуємо на два відношення внаслідок чого початкове відношення буде представлене у 3НФ:

1) **ЗАКУПІВЛІ** (**Постачальник, Товар, Реалізатор, Дата постачання**, Кількість).

2) **ДОГОВІР НА ПОСТАЧАННЯ** (***Постачальник, Реалізатор***, Номер договору).

3) **СТАНДАРТ НА ТОВАР** (**Товар**, ДСТУ на товар).

4) **РОЗРАХУНКОВИЙ РАХУНОК ПОСТАЧАЛЬНИКА** (**Постачальник**, Р/Р у банку, Банк).

5) **МФО БАНКУ** (**Банк**, МФО).

Отримані відношення, в цьому випадку, матимуть такий вигляд (рис. 5.2. – рис. 5.6) :

ЗАКУПІВЛІ

Постачальник	Реалізатор	Товар	Дата. постач	Кіль-кість
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200
АТ "Кулінічі"	ТФ"Булка"	Нарізний	10.10	250
ЗАТ "Хліб"	ТФ"Булка"	Нарізний	10.10	350
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
АТ "Кулінічі"	ТФ"Барс"	Нарізний	11.10	500
ЗАТ "Хліб"	ТФ"Булка"	Український	11.10	150
ТОВ "Салтівський"	ТФ"Сдоба"	Ржаний	11.10	200
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310

Рис. 5.2. **Загальний вигляд таблиці ЗАКУПІВЛІ**

ДОГОВОР НА ПОСТАЧАННЯ

Постачальник	Реалізатор	№Договору
АТ "Кулінічі"	ТФ"Барс"	12-091
АТ "Кулінічі"	ТФ"Булка"	045-11
ЗАТ "Хліб"	ТФ"Булка"	23-011
ЗАТ "Хліб"	ТФ"Сдоба"	114-01
ТОВ "Салтівський"	ТФ"Булка"	045-13
ТОВ "Салтівський"	ТФ"Сдоба"	14-235

Рис. 5.3. **Загальний вигляд таблиці ДОГОВІР НА ПОСТАЧАННЯ**

СТАНДАРТ НА ТОВАР

Товар	ДСТУ на товар
Білий	4583
Дарницький	4583
Завиток	4585
Нарізний	4583
Ржаний	4583
Сімейка	4585
Український	4583

Рис. 5.4. **Загальний вигляд таблиці СТАНДАРТ НА ТОВАР**

РОЗРАХУНКОВИЙ РАХУНОК ПОСТАЧАЛЬНИКА

Постачальник	Р/Р Постач.	Банк Постач.
АТ "Кулінічі"	26000001	Синтез
ЗАТ "Хліб"	26010234	Ексімбанк
ТОВ "Салтівський"	26000234	Синтез

Рис. 5.5. **Таблиця РОЗРАХУНКОВИЙ РАХУНОК ПОСТАЧАЛЬНИКА**

МФО БАНКУ

Банк Поставщ.	МФО БП
Синтез	851300
Ексімбанк	851301

Рис. 5.6. **Загальний вигляд таблиці СТАНДАРТ НА ТОВАР**

Отриманий набір відношень знаходиться у 3НФ. Він також знаходиться і НФБК, оскільки в отриманих відношеннях немає ключів, що перекриваються.

Припустимо, що у відношенні ДОГОВІР НА ПОСТАЧАННЯ для Реалізатора необхідно зберігати код його організації (код ЕДРПОУ). Значення цього коду є унікальним для будь-якої організації (рис. 5.7).

ДОГОВІР НА ПОСТАЧАННЯ			
Постачальник	**Реалізатор**	**Код ЕДРПОУ Реалізатора**	**№Договору**
АТ "Кулінічі"	ТФ"Барс"	23476092	12-091
АТ "Кулінічі"	ТФ"Булка"	31512733	045-11
ЗАТ "Хліб"	ТФ"Булка"	31512733	23-011
ЗАТ "Хліб"	ТФ"Сдоба"	30011211	114-01
ТОВ "Салтівський"	ТФ"Булка"	31512733	045-13
ТОВ "Салтівський"	ТФ"Сдоба"	30011211	14-235

Рис. 5.7. **Модифікований вигляд таблиці ДОГОВІР НА ПОСТАЧАННЯ**

Тоді відношення ДОГОВІР НА ПОСТАЧАННЯ буде містити наступні функціональні залежності:

- {Постачальник, Реалізатор} -> {№Договору}
- {Постачальник, Код ЕДРПОУ Реалізатора} -> {№Договору }
- { Реалізатор } -> { Код ЕДРПОУ Реалізатора }
- { Код ЕДРПОУ Реалізатора } -> { Реалізатор }

Можливими ключами відношення, в цьому випадку, будуть такі:

- (Постачальник, Реалізатор)
- (Постачальник, Код ЕДРПОУ Реалізатора)

Ключі є такими, що перекриваються і який би ключ не був вибраний в якості первинного, існуватиме функціональна залежність частини первинного ключа від неключового атрибуту (чи { Реалізатор } -> { Код ЕДРПОУ Реалізатора }, чи { Код ЕДРПОУ Реалізатора } -> { Реалізатор }). Отже таке відношення не знаходиться у НФБК і його необхідно декомпозуват на два, наприклад: **ДОГОВІР НА ПОСТАЧАННЯ_1** і **РЕАЛІЗАТОР** (рис. 5.8, рис. 5.9):

ДОГОВІР НА ПОСТАЧАННЯ_1		
Постачальник	**Код ЕДРПОУ Реалізатора**	**№Договору**
АТ "Кулінічі"	23476092	12-091
АТ "Кулінічі"	31512733	045-11
ЗАТ "Хліб"	31512733	23-011
ЗАТ "Хліб"	30011211	114-01
ТОВ "Салтівський"	31512733	045-13
ТОВ "Салтівський"	30011211	14-235

Рис. 5.8. **Загальний вигляд таблиці ДОГОВІР НА ПОСТАЧАННЯ_1**

РЕАЛІЗАТОР	
Код ЕДРПОУ Реалізатора	**Реалізатор**
23476092	ТФ"Барс"
31512733	ТФ"Булка"
30011211	ТФ"Сдоба"

Рис. 5.9. **Загальний вигляд таблиці РЕАЛІЗАТОР**

Примітка. Можливе і інше розбиття початкового відношення.

Отримані відношення знаходяться:

- у 2НФ, оскільки усі неключові атрибути неприведено залежать від первинного ключа;
- у 3НФ, оскільки в них відсутні транзитивні залежності;
- у НФБК, внаслідок того, що усунені ключі, що перекриваються, і відсутні функціональні залежності частини первинного ключа від неключових атрибутів.

Припустимо, що кожен Реалізатор має декілька Кіосків для реалізації Хлібобулочних виробів і парк автомобілів, для доставки продукції від Постачальників до місця реалізації. Тоді відношення Реалізатор, можна представити у вигляді відношення "ВІДОМОСТІ ПРО РЕАЛІЗАТОРА" (рис. 5.10).

ВІДОМОСТІ ПРО РЕАЛІЗАТОРА			
Код ЕДРПОУ Реалізатора	**Реалізатор**	**Кіоски реалізації**	**Автотранспорт**
23476092	ТФ"Барс"	Кіоск "Вогник"	АХ-1827
23476092	ТФ"Барс"	Кіоск "Вогник"	АХ-2256
23476092	ТФ"Барс"	Кіоск "Завиток"	АХ-1827
23476092	ТФ"Барс"	Кіоск "Завиток"	АХ-2256
23476092	ТФ"Барс"	Кіоск "Хліб"	АХ-1827
23476092	ТФ"Барс"	Кіоск "Хліб"	АХ-2256
31512733	ТФ"Булка"	Кіоск "Бублик"	ХА-1515
31512733	ТФ"Булка"	Кіоск "Бублик"	АХ-0011
31512733	ТФ"Булка"	Кіоск "Булочки"	ХА-1515
31512733	ТФ"Булка"	Кіоск "Булочки"	АХ-0011
30011211	ТФ"Сдоба"	Кіоск "Світанок"	ХА-2345
30011211	ТФ"Сдоба"	Кіоск "Світанок"	ХА-4576
30011211	ТФ"Сдоба"	Кіоск "Хліб"	ХА-2345
30011211	ТФ"Сдоба"	Кіоск "Хліб"	ХА-4576
30011211	ТФ"Сдоба"	Кіоск "Печиво"	ХА-2345
30011211	ТФ"Сдоба"	Кіоск "Печиво"	ХА-4576

Рис. 5.10. **Загальний вигляд таблиці ВІДОМОСТІ ПРО РЕАЛІЗАТОРА**

У цьому відношенні існують наступні Функціональні (ФЗ) і багатозначні (БЗ) залежності:

- ФЗ { Код ЕДРПОУ Реалізатора} -> {Реалізатор}
- ФЗ { Реалізатор } -> { Код ЕДРПОУ Реалізатора }
- МЗ { Реалізатор } >> {Кіоски реалізації} | {Автотранспорт}
- МЗ { Код ЕДРПОУ Реалізатора } >> {Кіоски реалізації} | {Автотранспорт}

На першому кроці, для декомпозиції відношення "ВІДОМОСТІ ПРО РЕАЛІЗАТОРА" скористаємося теоремою Хеза. Оскільки існує ФЗ {Код ЕДРПОУ Реалізатора} -> {Реалізатор}, те розіб'ємо відношення на два:

1) **РЕАЛІЗАТОР** (**Код ЕДРПОУ Реалізатора**, Реалізатор);
2) **ДОСТАВКА РЕАЛІЗАТОРА** (**Код ЕДРПОУ Реалізатора, Кіоски реалізації, Автотранспорт**).

Отримана декомпозиція залишає БЗ у відношенні "ДОСТАВКА РЕАЛІЗАТОРА", яку можна усунути, скориставшись теоремою Фейгина, розбивши його на два відношення:

3) **КІОСКИ РЕАЛІЗАТОРА** (**Код ЕДРПОУ Реалізатора, Кіоски реалізації**);
4) **АВТОТРАНСПОРТ РЕАЛІЗАТОРА** (**Код ЕДРПОУ Реалізатора, Автотранспорт**).

Таким чином, приведення початкового відношення "ВІДОМОСТІ ПРО РЕАЛІЗАТОРА" до 4НФ з метою усунення багатозначних залежностей дозволило отримати наступні відношення (рис. 5.11. – рис. 5.13.):

РЕАЛІЗАТОР

Код ЕДРПОУ Реалізатора	Реалізатор
23476092	ТФ"Барс"
31512733	ТФ"Булка"
30011211	ТФ"Сдоба"

Рис. 5.11. **Загальний вигляд таблиці РЕАЛІЗАТОР (до 4НФ)**

КІОСКИ РЕАЛІЗАТОРА

Код ЕДРПОУ Реалізатора	Кіоски реалізації
23476092	Кіоск "Вогник"
23476092	Кіоск "Завиток"
23476092	Кіоск "Хліб"
31512733	Кіоск "Бублик"
31512733	Кіоск "Булочки"
30011211	Кіоск "Світанок"
30011211	Кіоск "Хліб"
30011211	Кіоск "Печиво"

Рис. 5.12. **Вигляд таблиці КІОСКИ РЕАЛІЗАТОРА (до 4НФ)**

АВТОТРАНСПОРТ РЕАЛІЗАТОРА

Код ЕДРПОУ Реалізатора	Автотранспорт
23476092	АХ-1827
23476092	АХ-2256
31512733	ХА-1515
31512733	АХ-0011
30011211	ХА-2345
30011211	ХА-4576

Рис. 5.13. **Вигляд таблиці АВТОТРАНСТПОРТ РЕАЛІЗАТОРА (до 4НФ)**

Висновки: Таким чином в результаті проведених операцій, початкове відношення було нормалізоване до третьої нормальної форми. Крім того були проілюстровані необхідні кроки, які дозволять

перевести відношення у нормальну форму Бойса-Кодда та четверту нормальну форму.

6. Приклад виконання операцій реляційної алгебри

Мета – ознайомитися на конкретних прикладах з виконанням основних операцій реляційної алгебри.

Компетенції, що формуються:

Загальнонаукові – базові знання в області фундаментальної та прикладної математики та уміння їх застосовувати в професійній діяльності.

Професійні – ґрунтовна математична підготовка з теоретичних і методичних основ інформаційних технологій для використання математичного апарату під час вирішення прикладних завдань в області інформаційних систем.

Спеціалізовано-професійні – здатність до математичного та логічного мислення, знання основних понять, ідей і методів фундаментальної математики.

Проілюструємо на прикладах виконання основних операцій реляційної алгебри на відношеннях з предметної області по закупівлях Хлібобулочних виробів .

Нехай у БД існують такі відношення (рис. 6.1, рис. 6.2):

ЗАКУПІВЛІ_1

Постачальник	Реалізатор	Товар	Дата постач	Кількість
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310

Рис. 6.1. **Загальний вигляд таблиці ЗАКУПІВЛІ_1**

ЗАКУПІВЛІ_2

Постачальник	Реалізатор	Товар	Дата постач	Кількість
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
АТ "Кулінічі"	ТФ"Барс"	Нарізний	11.10	500
ЗАТ "Хліб"	ТФ"Булка"	Український	11.10	150
ТОВ "Салтівський"	ТФ"Сдоба"	Ржаний	11.10	200
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300

Рис. 6.2. **Загальний вигляд таблиці ЗАКУПІВЛІ_2**

6.1. Операція об'єднання

Об'єднання двох відношень ЗАКУПІВЛІ_1 та ЗАКУПІВЛІ_2 дає нове відношення, що містить усі кортежі, які належать хоч би одному з ЗАКУПІВЛІ_1 та ЗАКУПІВЛІ_2.

Результатом операції об'єднання REZ=ЗАКУПІВЛІ_1 **UNION** ЗАКУПІВЛІ_2 буде наступне відношення ЗАКУПІВЛІ_ОБ (рис. 6.3):

ЗАКУПІВЛІ_ОБ

Постачальник	Реалізатор	Товар	Дата постач	Кількість
АТ "Кулінічі"	***ТФ"Барс"***	***Дарницький***	***10.10***	***200***
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350
ТОВ "Салтівський"	***ТФ"Сдоба"***	***Білий***	***10.10***	***245***
АТ "Кулінічі"	ТФ"Барс"	Нарізний	11.10	500
ЗАТ "Хліб"	ТФ"Булка"	Український	11.10	150
ТОВ "Салтівський"	ТФ"Сдоба"	Ржаний	11.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120
ТОВ "Салтівський"	***ТФ"Булка"***	***Завиток***	***12.10***	***300***
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310

Рис. 6.3. **Загальний вигляд таблиці ЗАКУПІВЛІ_ОБ**

Кортежі результуючого відношення які помічені курсивом, дублювалися, але згідно з вимогами, що пред'являються до реляційної моделі, дублювання має бути усунене.

6.2. Операція перетину

Перетин двох відношень ЗАКУПІВЛІ_1 та ЗАКУПІВЛІ_2 дає нове відношення, що містить усі кортежі, які належать як відношенню ЗАКУПІВЛІ_1 так і відношенню ЗАКУПІВЛІ_2.

Результатом операції перетину REZ=ЗАКУПІВЛІ_1 **INTERSECT** ЗАКУПІВЛІ_2 буде наступне відношення ЗАКУПІВЛІ_ПЕР (рис. 6.4.):

ЗАКУПІВЛІ_ПЕР

Постачальник	Реалізатор	Товар	Дата постач	Кількість
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300

Рис. 6.4. **Загальний вигляд таблиці ЗАКУПІВЛІ_ПЕР**

У результуюче відношення потрапляють лише ті кортежі, які містяться як у першому, так і у другому відношенні.

6.3. Операція віднімання

Віднімання (або різниця) двох відношень ЗАКУПІВЛІ_1 та ЗАКУПІВЛІ_2 дає нове відношення, що містить усі кортежі, які належать першому відношенню ЗАКУПІВЛІ_1 і не належать другому – ЗАКУПІВЛІ_2.

Результатом операції віднімання REZ=ЗАКУПІВЛІ_1 **MINUS** ЗАКУПІВЛІ_2 буде наступне відношення ЗАКУПІВЛІ_ВІД (рис.6.5):

ЗАКУПІВЛІ_ВІД

Постачальник	Реалізатор	Товар	Дата постач	Кількість
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310

Рис. 6.5. **Загальний вигляд таблиці ЗАКУПІВЛІ_ВІД**

Результат міститиме тільки ті кортежі, які існують у першому відношенні, але відсутні в другому.

6.4. Операція декартового добутку

Для ілюстрації операції декартового добутку відношень з метою мінімізації одержуваного результату, візьмемо ще одне відношення МФО_БАНКА (рис. 6.6):

Декартів добуток двох відношень ЗАКУПІВЛІ_ВІД та МФО_БАНКУ дає нове відношення ЗАКУПІВЛІ_ВІД_МФО, що містить усі можливі кортежі, які є поєднанням двох кортежів, що належать відповідно відношенням ЗАКУПІВЛІ_ВІД та МФО_БАНКУ. Оскільки перше відношення мало потужність чотири, а друге – два, то потужність результуючого відношення буде дорівнювати 4 x 2 = 8. Ступінь результуючого відношення буде дорівнювати сумі ступенів відношень ЗАКУПІВЛІ_ВІД та МФО_БАНКУ, тобто 5 + 2 = 7.

МФО БАНКУ

Банк Поставщ.	МФО БП
Синтез	851300
Ексімбанк	851301

Рис. 6.6. **Загальний вигляд таблиці МФО_БАНКА**

Результатом операції добутку REZ=ЗАКУПІВЛІ_ВІД **TIMES** МФО_БАНКУ буде наступний відношення ЗАКУПІВЛІ_ВІД_МФО (рис. 6.7):

ЗАКУПІВЛІ_ВІД_МФО

Постачальник	Реалізатор	Товар	Дата постач	Кількість	Банк Поставщ.	МФО БП
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200	Синтез	851300
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350	Синтез	851300
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120	Синтез	851300
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310	Синтез	851300
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200	Ексімбанк	851301
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350	Ексімбанк	851301
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120	Ексімбанк	851301
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310	Ексімбанк	851301

Рис. 6.7. **Загальний вигляд таблиці ЗАКУПІВЛІ_ВІД_МФО**

Тобто, результуюче відношення містить всілякі комбінації кортежів першого і другого відношення.

6.5. Операція вибірка

Вибірка з відношення ЗАКУПІВЛІ_1 за певними умовами у результаті дає нове відношення, що містить усі кортежі початкового відношення ЗАКУПІВЛІ_1, що задовольняють цим самим умовам, наприклад, де кількість реалізованого товару більше 200. Можна сказати, що ця "горизонтальна" підмножина початкового відношення.

Результатом операції вибірка REZ=ЗАКУПІВЛІ_1 **WHERE** Кількість > 200 буде наступне відношення ЗАКУПІВЛІ_ВИБ (рис. 6.8):

ЗАКУПІВЛІ_ВИБ

Постачальник	Реалізатор	Товар	Дата постач	Кількість
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310

Рис. 6.8. **Загальний вигляд таблиці ЗАКУПІВЛІ_ВИБ**

Тобто у результат попадуть ті кортежі початкового відношення для яких виконується задана умова.

6.6. Операція проекція

Проекція застосована до відношення ЗАКУПІВЛІ_1 у результаті дає нове відношення, що містить усі кортежі початкового відношення, але тільки певні атрибути, які були визначені в операції. Можна сказати, що це "вертикальна" підмножина початкового відношення.

Результатом операції проекція REZ=$\pi_{1,3,4}$ (ЗАКУПІВЛІ_1) буде таке відношення ЗАКУПІВЛІ_ПРЦ (рис. 6.9):

ЗАКУПІВЛІ_ПРЦ

Постачальник	Товар	Дата постач
АТ "Кулінічі"	Дарницький	10.10
ЗАТ "Хліб"	***Сімейка***	***10.10***
ТОВ "Салтівський"	Білий	10.10
ТОВ "Салтівський"	***Завиток***	***12.10***
ЗАТ "Хліб"	Сімейка	12.10

Рис. 6.9. **Загальний вигляд таблиці ЗАКУПІВЛІ_ПРЦ**

Кортежі результуючого відношення які помічені жирним похилим шрифтом, дублювалися, але згідно з вимогами, що пред'являються до реляційної моделі, дублювання має бути усунене.

6.7. Операція з'єднання

Для ілюстрації операції З'єднання введемо в розгляд відношення РОЗРАХУНКОВИЙ_РАХУНОК_ПОСТАЧАЛЬНИКА (рис. 6.10):

З'єднання відношення ЗАКУПІВЛІ_1 та відношення РОЗРАХУНКОВИЙ_РАХУНОК_ПОСТАЧАЛЬНИКА за загальним атрибутом "Постачальник", дає нове відношення ЗАКУПІВЛІ_ЗДН , кортежі якого є зчепленням двох кортежів (що належать відповідно ЗАКУПІВЛІ_1 і РОЗРАХУНКОВИЙ_РАХУНОК_ПОСТАЧАЛЬНИКА), та мають однакове значення загального атрибуту "Постачальник" .

Причому ці загальні значення в результуючому відношенні з'являються тільки один раз.

РОЗРАХУНКОВИЙ_РАХУНОК_ПОСТАЧАЛЬНИКА

Постачальник	Р/Р Поставщ.	Банк Постач.
АТ "Кулінічі"	26000001	Сінтез
ЗАТ "Хліб"	26010234	Ексімбанк
ТОВ "Салтівський"	26000234	Сінтез

Рис. 6.10. **Загальний вигляд таблиці Р/Р ПОСТАЧАЛЬНИКА**

Можна сказати, що кортежі, які мають однакові значення загальних атрибутів "склеюються". Тоді відношення ЗАКУПІВЛІ_ЗДН, що є результатом виконання операції

REZ = ЗАКУПІВЛІ_1 **JOIN** РОЗРАХУНКОВИЙ_РАХУНОК_ПОСТАЧАЛЬНИКА

буде мати такий вигляд (рис. 6.11):

ЗАКУПІВЛІ_ЗДН

Постачальник	Реалізатор	Товар	Дата постач	Кіль-кість	Р/Р Пос-тавщ.	Банк Постач.
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200	26000001	Синтез
ЗАТ "Хліб"	ТФ"Булка"	Сімейка	10.10	200	26010234	Ексімбанк
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	10.10	350	26010234	Ексімбанк
ТОВ"Салтівський"	ТФ"Сдоба"	Білий	10.10	245	26000234	Синтез
ТОВ "Салтівський"	ТФ"Булка"	Завиток	12.10	300	26000234	Синтез
ЗАТ "Хліб"	ТФ"Сдоба"	Сімейка	12.10	120	26010234	Ексімбанк
ТОВ "Салтівський"	ТФ"Сдоба"	Завиток	12.10	310	26000234	Синтез

Рис. 6.11. **Загальний вигляд таблиці ЗАКУПІВЛІ_ЗДН**

6.8. Операція ділення

Нехай відношення ЗАКУПІВЛІ має такі кортежі (рис. 6.12):

ЗАКУПІВЛІ

Постачальник	Реалізатор	Товар	Дата постач	Кіль-кість
АТ "Кулінічі"	ТФ"Барс"	Дарницький	10.10	200
АТ "Кулінічі"	ТФ"Булка"	Нарізний	10.10	250
ЗАТ "Хліб"	ТФ"Булка"	Нарізний	10.10	350
ЗАТ "Хліб"	***ТФ"Булка"***	***Сімейка***	***10.10***	***200***
ЗАТ "Хліб"	***ТФ"Сдоба"***	***Сімейка***	***10.10***	***350***
ТОВ "Салтівський"	ТФ"Сдоба"	Білий	10.10	245
ТОВ "Салтівський"	***ТФ"Сдоба"***	***Сімейка***	***10.10***	***350***
АТ "Кулінічі"	ТФ"Барс"	Нарізний	11.10	500
ЗАТ "Хліб"	ТФ"Булка"	Український	11.10	150
ТОВ "Салтівський"	ТФ"Сдоба"	Ржаний	11.10	200
ЗАТ "Хліб"	***ТФ"Сдоба"***	***Сімейка***	***12.10***	***120***
ТОВ "Салтівський"	***ТФ"Булка"***	***Завиток***	***12.10***	***300***
ТОВ "Салтівський"	***ТФ"Сдоба"***	***Завиток***	***12.10***	***310***

Рис. 6.12. **Модифікований вигляд таблиці ЗАКУПІВЛІ**

Введемо у БД відношення ЗДОБНІ_ВИРОБИ, що складається з одного атрибуту Товар (рис. 6.13):

ЗДОБНІ_ВИРОБИ

Товар
Сімейка
Завиток

Рис. 6.13. **Загальний вигляд таблиці ЗДОБНІ_ВИРОБИ**

Операція ділення відношення ЗАКУПІВЛІ на відношення ЗДОБНІ_ВИРОБИ дає нове відношення ЗАКУПІВЛІ_ДІЛ, яке містить тільки ті атрибути відношення ЗАКУПІВЛІ, які відсутні у відношенні ЗДОБНІ_ВИРОБИ, і тільки ті кортежі, декартовий добуток яких з відношенням ЗДОБНІ_ВИРОБИ існує у відношенні ЗАКУПІВЛІ.

Тоді результатом виконання операції ділення
REZ= ЗАКУПІВЛІ **DIVIDE BY** ЗДОБНІ_ВИРОБИ, буде таке відношення (рис. 6.14):

ЗАКУПІВЛІ_ДІЛ

Постачальник	Реалізатор	Дата постач	Кіль-кість
ЗАТ "Хліб"	ТФ"Булка"	10.10	200
ЗАТ "Хліб"	ТФ"Сдоба"	10.10	350
ТОВ "Салтівський" "	ТФ"Сдоба"	10.10	350
ЗАТ "Хліб"	ТФ"Сдоба"	12.10	120
ТОВ "Салтівський"	ТФ"Булка"	12.10	300
ТОВ "Салтівський"	ТФ"Сдоба"	12.10	310

Рис. 6.14 **Загальний вигляд таблиці ЗАКУПІВЛІ_ДІЛ**

Висновки: Таким чином в результаті виконаних дій, були проілюстровані базові операції реляційної алгебри над відношеннями. Результатом такої операції є теж відношення.

Тобто, і операндами, і результатом операцій реляційної алгебри є відношення.

Тести

Питання 1. Які завдання відносяться до завдань обробки даних?

a) завдання з великим об'ємом складних обчислень;

b) завдання обліку кадрового складу організації;

c) завдання бухгалтерського обліку;

d) рішення систем лінійних рівнянь.

Питання 2. Що є елементом логічного запису?

a) прості змінні;

b) елементи масиву;

c) файли;

d) поля.

Питання 3. З чого складається логічний запис?

a) з простих змінних і полів;

b) з елементів масиву і змінних;

c) з полів;

d) з простих змінних.

Питання 4. Що таке логічний файл?

a) сукупність полів;

b) сукупність логічних записів;

c) сукупність екземплярів логічних записів;

d) набір даних в зовнішній пам'яті ЕОМ.

Питання 5. Які поняття не використовуються при описі логічного файлу?

a) екземпляр запису;

b) поле;

c) логічний запис;

d) масив.

Питання 6. З яких основних етапів складається рішення задачі обробки даних?

a) проведення складних математичних обчислень;

b) занесення даних в зовнішню пам'ять;

c) читання даних із зовнішньої пам'яті;

d) пошук необхідних даних;

Питання 7. У якому вигляді видаються інтегровані дані?

a) окремий файл;

b) набір окремих файлів;

c) набір екземплярів записів одного типу;

d) набір екземплярів записів різних типів і зв'язків між ними.

Питання 8. Що таке база даних?

a) сукупність екземплярів запису одного типу;

b) сукупність екземплярів записів різних типів;

c) сукупність екземплярів записів різних типів і зв'язків (відношень) між ними;

d) пойменована сукупність логічних записів.

Питання 9. Відмітити основні властивості бази даних.

a) відсутність дублювання;

b) мінімальна надмірність;

c) мінімальний час рішення усіх задач;

d) використовується для…

Питання 10. До чого приведе відсутність логічної і фізичної незалежності даних?

a) до необхідності зміни застосовних програм при зміні фізичного представлення бази даних;

b) до більшої достовірності даних;

c) до можливої зміни фізичного представлення даних при зміні прикладних програм;

d) до ефективнішої взаємодії користувачів з базою даних.

Питання 11. Основні цілі забезпечення логічної та фізичної цілісності бази даних?

a) захист від неправильних дій прикладного програміста;

b) захист від неправильних дій адміністратора баз даних;

c) захист від можливих помилок введення даних;

d) захист від машинних збоїв;

e) захист від можливої появи невідповідності між даними після виконання операцій видалення і коригування.

Питання 12. Що таке концептуальна модель?

a) інтегровані дані;

b) база даних;

c) узагальнене уявлення користувачів про дані;

d) опис представлення даних в пам'яті комп'ютера.

Питання 13. Як співвідносяться поняття інформаційно-логічної моделі і узагальненого концептуального представлення?

a) одне й теж;

b) це різні поняття;

c) узагальнене концептуальне представлення є частиною інформаційно-логічної моделі;

d) інформаційно-логічна модель є частиною узагальненого концептуального представлення.

Питання 14. Який зв'язок між логічною моделлю бази даних і СКБД?

a) це не пов'язані поняття;

b) логічна модель бази даних використовує специфікації СКБД;

c) СКБД відображає логічну модель бази даних в структуру збереження;

d) логічна модель бази даних описує структуру збереження даних системою керування базами даних.

Питання 15. Як називаються рівні архітектури бази даних?

a) нижній;

b) зовнішній;

c) концептуальний;

d) внутрішній;

e) верхній;

Питання 16. Основні етапи проектування бази даних:

a) вивчення предметної області;

b) проектування узагальненого концептуального представлення;

c) проектування концептуального представлення, що специфіковане до моделі цих СКБД (логічної моделі);

d) розробка застосовних програм.

Питання 17. З яких етапів складається перша стадія концептуального проектування?

a) вивчення предметної області;

b) проектування узагальненого концептуального представлення;

c) проектування концептуального представлення, що специфіковане до моделі цих СКБД (логічної моделі);

d) проектування представлення даних в пам'яті комп'ютера (структур зберігання);

e) розробка застосовних програм.

Питання 18. Як називається поняття, що використовують для опису сутності?

a) властивість;

b) атрибут;

c) об'єкт;

d) екземпляр.

Питання 19. Чим відрізняються поняття властивість і атрибут?

a) одне й теж;

b) атрибут ця властивість, що набуває конкретних значень;

c) властивість використовується для опису атрибуту;

d) атрибут описує конкретну властивість.

Питання 20. Що таке клас сутностей?

a) набір екземплярів сутностей;

b) сукупність сутностей з однаковими властивостями;

c) сукупність атрибутів;

d) сукупність сутностей з однаковими значеннями атрибутів.

Питання 21. Чим визначається існування зв'язку між сутностями?

a) функціональними взаємовідносинами між сутностями;

b) інформаційними зв'язками між сутностями;

c) інформаційними потребами користувача;

d) властивостями сутностей.

Питання 22. Які бувають типи зв'язків?

a) один до одного;

b) один до багатьом;

c) багато до багатьом;

d) багато до одного.

Питання23. Для чого використовується ER-діаграма?

a) графічне представлення концептуальної моделі;

b) графічне представлення сутностей та зв'язків між ними;

c) графічне уявлення узагальненого представлення користувача про дані;

d) графічне представлення усіх сутностей;

e) графічне представлення зв'язків.

Питання 24. Як на ER-діаграмі представляються способи реалізації зв'язків?

a) не представляються;

b) у вигляді адресних посилань;

c) представляються на логічному рівні;

d) представляються на фізичному рівні;

Питання 25. Який порядок дій при побудові концептуальної моделі?

a) визначення сутностей, визначення атрибутів, встановлення зв'язків;

b) визначення атрибутів, визначення сутностей, встановлення зв'язків;

c) вибір зв'язків, визначення сутностей, визначення атрибутів;

d) вибір екземплярів сутностей, встановлення зв'язків між екземплярами.

Питання 26. Які прийоми використовуються при об'єднанні локальних моделей?

a) відкидаються деякі ідентичні елементи;

b) зливаються ідентичні елементи;

c) локальні моделі об'єднуються по нових зв'язках;
d) локальні моделі об'єднуються по наявних в них зв'язках;
e) подібні типи сутностей узагальнюються;
f) подібні типи сутностей виключаються;
g) вводяться нові сутності.

Питання 27. Навіщо потрібні обмеження цілісності?
a) для забезпечення правильного введення даних у базу даних;
b) для забезпечення достовірної інформації у базі даних;
c) для перевірки правильності роботи застосовних програм;
d) для зменшення помилок при пошуку даних.

Питання 28. Які поняття використовуються при описі даних в термінах моделі даних?
a) атрибут;
b) сутність;
c) поле;
d) запис;
e) рядок;
f) екземпляр запису;
g) файл.

Питання 29. Як зберігаються фізичні записи в пам'яті при використанні індексації?
a) впорядковані по значенням ключа;
b) у вигляді неврегульованої послідовності;
c) у вигляді списку;
d) використовується додатковий файл.

Питання 30. Як видається сутність у реляційній моделі?
a) рядком таблиці;
b) стовпцем таблиці;
c) таблицею;
d) набором таблиць;

Питання 31. Які етапи створення бази даних підтримуються засобами автоматизованого проектування?

a) розробка ER-діаграми;
b) розробка програм створення структури бази даних;
c) розробка інтерфейсу користувача;
d) розробка застосовних програм.

Питання 32. Чому відповідає поняття "Схема відношення"?

a) двовимірній таблиці;
b) опису структури конкретної таблиці;
c) опису структури будь-якої таблиці;
d) множині значень в таблиці.

Питання 33. Що відповідає імені атрибуту в схемі відношення?

a) множина значень певного типу даних;
b) домен;
c) кортеж;
d) множина значень різних типів даних.

Питання 34. Що таке ключ відношення?

a) підмножина атрибутів, таких що будь-які два кортежі відношення не співпадають по значеннях цієї підмножини;

b) мінімальна підмножина атрибутів, таких що будь-які два кортежі відношення не співпадають по значеннях цієї підмножини;

c) максимальна підмножина атрибутів, таких що будь-які два кортежі відношення не співпадають по значеннях цієї підмножини;

d) множина усіх атрибутів.

Питання 35. Що називається реляційною моделлю бази даних?

a) сукупність схем відношень, використовуваних для представлення концептуальної моделі;

b) сукупність відношень, що реалізує концептуальну модель;
c) поточні значення відношень;
d) модель даних реляційної СКБД.

Питання 36. Що називається об'єднанням відношень?

a) множина кортежів, що належать одному або іншому відношенню, або їм обом;

b) множина кортежів, що належать одному або іншому відношенню;

c) множина кортежів, що належать обом відношенням;

d) множина кортежів, одна частина якої представляє кортеж з першого відношення, друга частина – кортежу з другого відношення.

Питання 37. Що називається різницею відношень?

a) множина кортежів, які представляють кортежі з першого відношення за мінусом тих значень, які входять в кортежі другого відношення;

b) множина кортежів, що належать першому відношенню, але що не належать другому відношенню;

c) множина кортежів відношення, яке виходить із першого відношення видаленням атрибутів другого відношення;

d) множина атрибутів, яка виходить з першого відношення видаленням атрибутів другого відношення.

Питання 38. Що є результатом операції "декартового добутку" двох відношень?

a) схема відношення, складена з двох схем відношень;

b) нове відношення з схемою відношення, складеною з двох початкових схем відношень;

c) множина всіляких кортежів, перша частина яких представляє кортежі першого відношення, друга частина – кортежі другого відношення;

d) множина кортежів, що отримуються додаванням до кортежів першого відношення кортежів з відповідного рядка другого відношення.

Питання 39. Якщо ступінь відношень, що беруть участь в операції "декартовий добуток" дорівнює відповідно k1 і k2, то чому дорівнює ступінь отриманого відношення?

a) k1+ k2;

b) k1*k2;

c) k1 - k2;

d) k1* (k1+k2).

Питання 40. За допомогою якої операції вибираються потрібні стовпці таблиці?

a) селекція;

b) проекція;

c) декартовий добуток;

d) різниця.

Питання 41. За допомогою якої операції вибираються потрібні кортежі відношення?

a) проекція;

b) декартовий добуток;

c) різниця;

d) селекція.

Питання 42. Для чого потрібні операції з'єднання?

a) для "склеювання" таблиць;

b) для переходу від значень атрибутів в одній таблиці до таких же значень атрибутів в іншій таблиці;

c) для об'єднання таблиць із співпадаючими значеннями одного або декількох атрибутів;

d) для реалізації вибірки даних на основі використання двох таблиць, пов'язаних загальними атрибутами.

Питання 43. Які операції повинна підтримувати реляційна система?

a) пошук;

b) додавання;

c) видалення;

d) селекція;

e) проекція;

f) з'єднання.

Питання 44. З чим пов'язано основне дублювання інформації в реляційній базі даних?

a) з повторенням однакових рядків в одній таблиці;

b) з повторенням однакових стовпців в одній таблиці;

c) з повторенням однакових значень атрибутів в одній таблиці;

d) з повторенням однакових значень атрибуту в різних таблицях.

Питання 45. Які аномалії необхідно усунути при проектуванні реляційної бази даних?

a) створення;

b) видалення;

c) оновлення;

d) включення;

e) виключення.

Питання 46. Як здійснюється вибір раціональних схем відношень?

a) шляхом нормалізації;

b) шляхом послідовного перетворення відношень до ряду нормальних форм;

c) шляхом об'єднання схем відношень;

d) шляхом декомпозиції схем відношень.

Питання 47. Що таке нормалізація?

a) послідовне перетворення відношень до ряду нормальних форм;

b) певне об'єднання схем відношень;

c) певна декомпозиція схем відношень;

d) перетворення відношень з використанням операцій реляційної алгебри.

Питання 48. Що таке X функціонально визначає Y?

a) кожне значення множини X пов'язане з одним значенням множини Y;

b) якщо два кортежі співпадають по значеннях X, то вони співпадають по значеннях Y;

c) Y є функцією X;

d) Y залежить від X;

e) кожне значення множини Y пов'язане з одним значенням множини X;

f) якщо два кортежі співпадають по значеннях Y, то вони співпадають по значеннях X.

Питання 49. Що характеризують функціональні залежності?

a) схему відношення;

b) усі можливі значення відношення;

c) усі можливі значення рядків відношення;

d) відношення як змінну.

Питання 50. Що таке фізична модель даних?

a) внутрішній рівень бази даних;

b) концептуальна модель, специфікована в термінах СКБД;

c) структура пам'яті комп'ютера;

d) відображення концептуальної моделі бази даних у фізичну організацію даних.

Питання 51. Які властивості повинні мати декомпозиції при нормалізації?

a) збереження функціональних залежностей

b) з'єднання без втрат

c) розбиття без втрат

d) збереження ключа

Питання 52. За яких умов відношення знаходиться в другій нормальній формі?

a) якщо воно знаходиться в першій нормальній формі і кожен не ключовий атрибут залежить від усього первинного ключа;

b) якщо воно знаходиться в першій нормальній формі і кожен не ключовий атрибут залежить від частини первинного ключа;

c) якщо воно знаходиться в першій нормальній формі і кожен не ключовий атрибут не залежить від первинного ключа;

d) якщо воно знаходиться в першій нормальній формі і кожен не ключовий атрибут не залежить частково від первинного ключа.

Питання 53. За яких умов відношення знаходиться в третій нормальній формі?

a) якщо воно знаходиться в другій нормальній формі і кожен не ключовий атрибут залежить від усього первинного ключа;

b) якщо воно знаходиться в другій нормальній формі і кожен не ключовий атрибут нетранзитивно залежить від частини первинного ключа;

c) якщо воно знаходиться в другій нормальній формі і кожен не ключовий атрибут нетранзитивно залежить від первинного ключа;

d) якщо воно знаходиться в другій нормальній формі і кожен не ключовий атрибут не залежить від частини первинного ключа.

Питання 54. Які з перерахованих залежностей можуть існують у відношенні ЕКЗАМЕНИ (Код студента, Прізвище, Код екзамену, Предмет, Дата, Оцінка), та за яких умов? Відповідь обґрунтуйте.

a) Код студента →Прізвище;
b) Предмет→Дата;
c) Код іспиту →Дата;
d) Код студента→Оцінка.

Питання 55. Які з перерахованих залежностей не існують у відношенні ЕКЗАМЕНИ (Код студента, Прізвище, Код екзамену, Предмет, Дата, Оцінка)? Відповідь обгрунтуйте.

a) Код студента, Прізвище→Прізвище
b) Предмет→Дата
c) Код іспиту →Дата
d) Код студента→Оцінка
e) Код студента, Предмет→Оцінка

Питання 56. Які вимоги повинні виконуватися для підтримки цілісності даних в реляційних СКБД?

a) унікальність будь-якого кортежу відношення;
b) наявність у будь-якого відношення первинного ключа;

c) для кожного значення зовнішнього ключа у відношенні, що посилається, повинен існувати кортеж з таким же значенням первинного ключа у відношенні, на яке посилаються;

d) для кожного значення первинного ключа у відношенні, що посилається, повинен існувати кортеж з таким же значенням зовнішнього ключа у відношенні, на яке посилаються.

Питання 57. Що таке фізична модель даних?

a) внутрішній рівень бази даних;
b) концептуальна модель, специфікована в термінах СКБД;
c) структура пам'яті комп'ютера;

d) відображення концептуальної моделі бази даних у фізичну організацію даних.

Відповіді на тести

№ Питання	Відповідь	№ Питання	Відповідь
1	c, d	30	c
2	d	31	a, b, c
3	c	32	b
4	c	33	a, b
5	d	34	b
6	b, c, d	35	a
7	d	36	a
8	c	37	b
9	b, d	38	b
10	a, c	39	a
11	c, d, e	40	b
12	c	41	d
13	a	42	a, b, c, d
14	b, c	43	d, e, f
15	b, c, d	44	c
16	a, b, c	45	b, c, d
17	a, b	46	a, b, d
18	a, b	47	a, c
19	b	48	a, b
20	b	49	b, c, d
21	a, b, d	50	a, d
22	a, b, c, d	51	a, b
23	a, b, c	52	a, d
24	a	53	c
25	a	54	a, b, c (умови?)
26	b, c, e, g	55	d (умови?)
27	a, b	56	a, b, c
28	c, d, f, g	57	a, d
29	b, d		

Глосарій

Альтернативний ключ – ключ-кандидат у відношенні.

Аномалія введення – проблема, що виникає у невірно спроектованих відношеннях, коли користувачеві не вдається ввести дані через те, що значення усіх стовпців первинного ключа недоступні.

Аномалія видалення – проблема, що виникає в невірно спроектованих відношеннях, коли користувач випадково втрачає потрібні дані; видалення частини первинного ключа викликає видалення цілого рядка відношення.

Аномалія модифікації – проблема, що виникає в невірно спроектованих відношеннях, коли зайві дубльовані дані не оновлюються погоджено, і дані, які повинні мати однакові значення, такими не є.

Аномалія оновлення – те ж, що аномалія модифікації.

Атрибут – атрибут є типом характеристики, пов'язаної з множиною реальних або абстрактних предметів (людей, місць, подій і так далі); стовпець у відношенні.

Атрибут неключовий – будь-який атрибут, що не є частиною первинного ключа сутності. Неключові атрибути можуть входити в інверсійний вхід (вторинний ключ) і/або альтернативний ключ, а також можуть бути зовнішніми ключами.

Багатозначна залежність – залежність між трьома атрибутами, коли атрибутом A визначається кінцева множина значень атрибуту B і кінцева множина значень атрибуту C, але атрибути B і C один від одного не залежать.

База даних (1) – це реалізована за допомогою комп'ютера інформаційна структура (модель), що відображає стани об'єктів і їх відношення.

База даних (2) – зарезервований об'єм пам'яті на одному або більше пристроях зберігання інформації, використовуваних для зберігання даних і визначень об'єктів, наприклад, таблиць і індексів.

Базова таблиця – таблиця, дані якої фізично зберігаються у базі даних.

Базове відношення – відношення, яке на фізичному рівні представлене таблицею у базі даних.

Бізнес-обмеження – обмеження, засноване на предметній області.

Бізнес-правило – обмеження цілісності, засноване на предметній області, а не витікає з реляційної теорії

Бінарний зв'язок – зв'язок, в якому рівно один екземпляр батьківської сутності відповідає 0,1 або більше екземплярам дочірньої. У IDEF1X ідентифікуючі, неідентифікуючі зв'язки і зв'язки підтипу є бінарними зв'язками.

Булевий вираз – вираз, результатом якого може бути або значення True, або False.

Валідації правила – правила перевірки допустимих значень.

Вертикальне розділення – розподіл стовпців таблиці по декількох таблицях для підвищення ефективності зчитування інформації.

Взаємовиключне відношення – відношення, в якому екземпляр сутності A можна зв'язати або з екземпляром сутності B, або з екземпляром сутності C, але не з обома.

Визначник – теж що детермінант.

Відношення – Опис структури двомірної таблиці, що складається з атрибутів (стовпців) і кортежів (рядків).

Відношення "багато до багатьох" – відношення між двома сутностями, в якому один екземпляр сутності A може бути пов'язаний з нулем, одним або декількома екземплярами сутності B, а екземпляр сутності B може бути пов'язаний з нулем, одним або декількома екземплярами сутності A.

Відношення "один до багатьох" – відношення між двома сутностями, в якому екземпляр сутності A може бути пов'язаний з нулем, одним або декількома екземплярами сутності B, а екземпляр сутності B можна зв'язати, саме більше, з одним екземпляром сутності A.

Відношення "один до одного" – відношення між двома сутностями, в якому екземпляр сутності A може бути пов'язаний, саме більше, з одним екземпляром сутності B, а екземпляр сутності B можна зв'язати максимум з одним екземпляром сутності A.

Віртуальна таблиця – таблиця реляційної бази даних, існуюча тільки в оперативній пам'яті.

Внутрішнє з'єднання – еквіз'єднання.

Внутрішнє обмеження – обмеження, що визначає фізичну структуру бази даних.

Вторинний ключ – теж, що вхід інверсійний.

Вхід інверсійний (вторинний ключ) – атрибут (атрибути), який(які) не визначають унікальним чином екземпляр сутності, але часто використовуються для звернення до екземплярів сутностей. ERwin генерує неунікальні індекси для усіх інверсійних входів.

Головне відношення – відношення, первинний ключ якого зберігається в іншому відношенні, що бере участь в зв'язку.

Горизонтальне розділення – розбиття рядків таблиці на декілька таблиць з метою підвищення ефективності зчитування інформації.

Група, що повторюється – атрибут, що має по декілька значень в кожному рядку відношення.

Дані відношення – дані, що описують відношення між двома сутностями, а не кожну сутність окремо.

Декартів добуток – реляційна операція, що виконує з'єднання кожного запису одного набору записів з кожним записом іншого набору.

Декларативна цілісність – метод визначення обмежень цілісності, при якому обмеження явно оголошуються при визначенні таблиць.

Декомпозиція без втрат – можливість розділити відношення так, щоб при об'єднанні відношень, що вийшли в результаті цього, інформація не була загублена.

Детермінант (визначник) – атрибут, від якого функціонально залежить інший атрибут.

Дискримінатор – значення атрибуту в екземплярі загального батька визначає, до якого з можливих підтипів належить цей екземпляр. Цей атрибут прийнято називати дискримінатором. Наприклад, значення атрибуту "стать" в екземплярі сутності "службовець" визначає, до якого з можливих підтипів (чоловік-службовець або жінка-службовець) належить цей екземпляр.

Діаграма відношень сутностей – демонструє відношення між сутностями в середовищі бази даних.

Домен – сукупність значень, з яких беруться значення атрибутів. Кожен атрибут може бути визначений тільки на одному домені, але на кожному домені може бути визначена множина атрибутів. У поняття домена входить не лише тип даних, але і область значень даних. У ERwin домен може бути визначений тільки один раз і використовуватися як в логічній, так і у фізичній моделі .

Домени сумісних типів – домени, для яких допускається логічне порівняння.

Еквіз'єднання – з'єднання, засноване на відповідності однакових значень.

Екземпляр (відношення) – відношення, що містить рядки значень даних.

Екземпляр (сутності) – реальний прояв сутності, представлений значеннями її атрибутів.

Заголовок відношення – визначення атрибуту і домена, що розміщується у верхній частині відношення.

Замикання – принцип, що декларує, що результатом усіх операцій над відношенням є відношення, над яким можна виконувати інші операції.

Запис – фізичне представлення кортежу.

Запити – об'єкти, які служать для витягання даних з таблиць і надання їх користувачеві в зручному вигляді. За допомогою запитів виконують такі операції, як відбір даних, їх сортування і фільтрацію. За допомогою запитів можна виконувати перетворення даних по заданому алгоритму, створювати нові таблиці, виконувати автоматичне наповнення таблиць даними, імпортованими з інших джерел, виконувати прості обчислення в таблицях і багато що інше.

Звичайна сутність – сутність, яка може існувати незалежно від її участі в зв'язку.

Зв'язок – функціональна залежність між об'єктами. У реляційних базах даних між таблицями встановлюються зв'язки по ключах, один з яких в головній (батьківській) таблиці – первинний, другий – зовнішній

ключ – в зовнішній (дочірній) таблиці і як правило, первинним не є і утворює зв'язок "один до багатьох" (1:М). У разі первинного зовнішнього ключа зв'язок між таблицями має тип "один до одного" (1:1). Інформація про зв'язки зберігається у базі даних.

Зв'язок визначений – відношення між сутностями, в якому кожен екземпляр батьківської сутності пов'язаний з 0,1 або більше екземплярами дочірньої сутності і кожен екземпляр дочірньої сутності пов'язаний з 0 або 1 екземплярами батьківської сутності.

Зв'язок ідентифікуючий – зв'язок, в якому екземпляр дочірньої сутності ідентифікується за допомогою свого відношення до батьківської сутності. Атрибути первинного ключа батьківської сутності стають атрибутами первинного ключа дочірньої.

Зв'язок невизначений – зв'язки між батьківською і дочірньою суттю і зв'язки підтипу вважаються певними зв'язками, оскільки вони точно визначають, яким чином екземпляри однієї сутності пов'язані з екземплярами інший. Проте на початкових етапах розробки моделі часто буває корисне завдання "невизначених" зв'язків між двома сутностями. Невизначений зв'язок, який називають також зв'язком "багато до багатьох", – відношення між двома сутностями, при якому кожен екземпляр першої сутності пов'язаний з 0,1 або більше екземплярами другої сутності і кожен екземпляр другої сутності пов'язаний з 0,1 або більше екземплярами першої сутності.

Зв'язок неідентифікуючий – зв'язок, в якому екземпляр дочірньої сутності не ідентифікується за допомогою її відношення до батьківської сутності. Атрибути первинного ключа батьківської сутності стають неключовими атрибутами дочірньої.

Зв'язок підтипу – зв'язком підтипу (інша назва – категоріальний зв'язок) називають зв'язок між суттю підтипу і її груповим батьком. Зв'язок підтипу завжди зв'язує один екземпляр групового батька з 0 або одним екземпляром підтипу.

З'єднання – операція реляційної алгебри, в якій два відношення об'єднуються по відповідності рядків на основі значень стовпців двох таблиць. Відношенням відповідності зазвичай є комбінація "первинний ключ зовнішній ключ".

Зовнішнє відношення – відношення, що одержує зовнішній ключ від іншого учасника зв'язку.

Зовнішнє з'єднання – з'єднання, що повертає усі записи, присутні у внутрішньому з'єднанні, а також усі записи, присутні в одному або в обох інших учасниках з'єднання.

Зовнішній ключ – стовпець або комбінація стовпців, аналогічних стовпцям первинного ключа деякої таблиці тієї ж самої бази даних.

Ідентифікатор сутності – атрибут або комбінація атрибутів, значення яких однозначно визначають кожен екземпляр сутності.

Ім'я ролі – нове ім'я, що привласнюється зовнішньому ключу. Ім'я ролі використовується для вказівки, що домен зовнішнього ключа є підмножиною домена атрибуту батьківської сутності і виконує певну функцію (чи роль) по суті.

Індекс – об'єкт СКБД, призначений для пошуку даних. Він подібний до змісту книги, який вказує на усі номери сторінок, присвячених конкретній темі. Індекс містить відсортовану по колонці або декільком колонкам інформацію і вказує на рядки, в яких зберігається конкретне значення колонки.

Інструментальний засіб CASE – програмний засіб, використовуваний для підтримки процесу проектування і розробки інформаційних систем прикладного програмного забезпечення.

Кардинальність – називається також "Потужність зв'язку". Відношення числа екземплярів батьківської сутності до числа екземплярів дочірньої. У IDEF1X кардинальність бінарних зв'язків рівна 1: M, де M може дорівнювати: 0,1 або більше – позначається пропусками; 1 або більше – позначається буквою "P"; 0 або 1 – позначається буквою "Z"; рівно M – де M – деяке число

Каскадне оновлення – автоматичне оновлення сутностей в зовнішньому відношенні при зміні відповідної сутності головного відношення.

Каталог – те ж, що словник даних.

Керованість словником даних – властивість реляційних баз даних, що означає, що усім операціям звернення до даних, що зберігаються, передує звернення до словника; це допомагає визначити,

чи існують прошені елементи даних, і чи має користувач повноваження доступу на виконання запрошеної операції.

Ключ альтернативний – 1) атрибут, який унікальним чином ідентифікує екземпляр сутності; 2) якщо правилу 1 задовольняє більше, ніж один атрибут (група атрибутів), то альтернативним ключем називаються ті атрибути або групи атрибутів, які не були вибрані в якості первинного ключа. ERwin генерує унікальний індекс для кожного альтернативного ключа.

Ключ зовнішній – атрибут, що мігрував від батьківської сутності до дочірньої через зв'язок.

Ключ первинний – 1) атрибут (атрибути), який(які) унікальним чином ідентифікують екземпляр сутності; 2) якщо більше, ніж один атрибут (група атрибутів) задовольняють правилу 1, то первинний ключ вибирається з цього списку кандидатів, виходячи з того, яким видається його значення для бізнесу в якості ідентифікатора. У ідеалі первинні ключі не повинні мінятися з часом і мають бути як можна меншого розміру. ERwin генерує унікальний індекс для кожного первинного ключа.

Ключа зовнішнього міграція – ситуація, при якій ключ батьківської сутності автоматично з'являється в ключі дочірньої сутності зі значком (FK), що означає зовнішній ключ.

Ключ-кандидат – один або декілька атрибутів, що унікально ідентифікують відношення.

Ключовий елемент таблиці (ключ) – таке її поле (простий ключ) або строкове вираження, утворене зі значень декількох полів (складений ключ), по якому можна визначити значення інших полів для однієї або декількох записів таблиці. На практиці для використання ключів створюються індекси – службова інформація, що містить впорядковані відомості про ключові значення. У реляційній теорії і концептуальній моделі поняття "ключ" застосовується для атрибутів відношення або сутності.

Кортеж – рядок у відношенні.

Метадані – дані про дані; дані, що зберігаються в словнику даних.

Модель даних – формальний спосіб опису відношень між сутностями у базі даних для системи керування базами даних. Іншими

словами – концептуальний опис предметної області в термінах вибраної моделі.

Нормалізація – процес розміщення атрибутів в таблицях, що виключає проблеми, характерні для невірно організованої бази даних.

Нормальна форма – критерії проекту, яким повинне відповідати відношення.

Null – значення Null, "невідоме" значення, що показує, що інформація в конкретному полі відсутня. Дозвіл на можливість існування значення Null може задаватися для окремих полів таблиці.

Обмеження – правило, якому повинен слідувати деякий елемент у базі даних.

Обмеження домена – правило, яке вимагає, щоб усі значення атрибуту відповідали вказаному домену.

Обмеження на рівні домена – обмеження цілісності, що визначає діапазон допустимих для домена значень.

Обмеження на рівні сутності – обмеження цілісності, що гарантує дійсність сутностей, що моделюються в системі.

Обмеження цілісності – правило, що підтримує цілісність даних.

Обмеження, визначене для бази даних – обмеження цілісності, що посилається на множину відношень.

Однорідність по стовпцях – властивість відношення, визначальна, що усі значення цього стовпця беруться з одного і того ж домена.

Первинний ключ – головний ключовий елемент, що однозначно ідентифікує рядок в таблиці. Можуть також існувати альтернативний і унікальний ключі, службовці також для ідентифікації рядків в таблиці.

Подання – об'єкт БД, дані в якому не зберігаються постійно, як в таблиці, а формуються динамічно при зверненні до нього. Подання не може існувати саме по собі, а визначається тільки в термінах однієї або декількох таблиць, що дозволяє розробникові БД забезпечити кожному користувачеві або групі користувачів свій погляд на дані, що вирішує проблеми простоти використання і безпеки даних.

Поле – представлення атрибуту у базі даних на фізичному рівні.

Посилальна цілісність – обмеження відношення, визначальне, що кожне значення зовнішнього ключа, відмінне від NULL, повинне посилатися на існуюче значення деякого первинного ключа.

Потужність відношення – число кортежів (рядків) у відношенні.

Потужність зв'язку – максимальне число екземплярів сутності, які можуть брати участь у відношенні.

Предикат – логічний вираз, по якому відбираються дані.

Предметна область – частина реального світу, що моделюється застосуванням бази даних.

Природне з'єднання – особливий випадок еквіз'єднання; при цьому з'єднання засноване на операції рівності, в нім беруть участь усі загальні і тільки один набір загальних полів включається в набір результатів.

Проектування зворотне – процес генерації логічної моделі з фізичної бази даних.

Проектування пряме – процес генерації фізичної моделі (схеми бази даних) з логічної моделі даних.

Простий ключ -- ключ-кандидат, що складається з одного атрибуту.

Реальна сутність – сутність, що моделює об'єкт або подію реального світу.

Реляційна база даних – база даних, в якій єдиними структурами даних є відношення.

Рівень логічний – представлення і моделювання предметів безпосередньо з реального світу.

Рівень фізичний – інформація, що відноситься до моделі, яка визначається залежно від бази даних і СКБД; наприклад, таблиці, колонки, типи даних і так далі

Розмірність зв'язку – число (кількість) учасників зв'язку.

CASE-технологія – сукупність методологій аналізу, проектування, розробки і супроводу складних систем, підтримувана комплексом засобів автоматизації.

Система керування базами даних (СКБД) – комплекс програмних і мовних засобів, необхідних для створення і модифікації бази даних, додавання, модифікації, видалення, пошуку і відбору інформації, представлення інформації на екрані і в друкарському виді, розмежування прав доступу до інформації, виконання інших операцій з базою.

Складена сутність – сутність, що представляє відношення між двома іншими сутностями.

Складений ідентифікатор – ідентифікатор сутності, що складається зі значень декількох атрибутів.

Складений ключ – ключ-кандидат, що складається з двох або більше за атрибути.

Слабка сутність – сутність, яка не може існувати у базі даних, якщо немає пов'язаного з нею екземпляра іншої сутності.

Словник даних – Сховище даних, що описують структурні елементи бази даних.

Ступінь (розмірність) відношення – число (кількість) стовпців у відношенні.

Сутність – Набір реальних або абстрактних предметів (людей, місць, подій і так далі), що мають загальні атрибути або характеристики.

Сутність залежна – сутність, екземпляри якої не можуть бути унікальним чином ідентифіковані, якщо не визначений її зв'язок з іншою суттю або сутностями.

Сутність незалежна – сутність, екземпляри якої можуть бути унікальним чином ідентифіковані без визначення її зв'язку з іншою суттю.

Схема – загальний логічний план бази даних – структура бази даних. Як правило, будується на основі файлу скрипта, написаного на DDL (мові визначення даних). DDL складається з операторів CREATE TABLE, CREATE INDEX і інших.

Таблиця бази даних – регулярна структура, яка складається з однотипних рядків (записів), розбитих на стовпці (поля). Відношення в реляційній базі даних.

Тетаз'єднання – теоретично: будь-яке з'єднання, засноване на операторові порівняння; як правило, відмінному від оператора рівності.

Тимчасова таблиця – базова таблиця, існуюча тільки в оперативній пам'яті впродовж конкретного сеансу роботи з базою даних.

Тіло відношення – кортежі, що становлять відношення.

Транзитивна залежність – група функціональних залежностей, коли атрибутом A визначається атрибут B, який, у свою чергу, визначає атрибут C. Тому вірно і те, що атрибутом A визначається атрибут C.

Тригер – процедура (іменований блок коду SQL), яка виконується автоматично при звершенні певної події.

Тризначна логіка – набір логічних таблиць істинності зі значеннями true (істина), false (брехня) і unknown (невизначеність).

Трирівнева (трьохсхемна) архітектура – три погляди на базу даних: через фізичну схему, через логічну схему і через призначені для користувача представлення даних.

Унарний зв'язок – зв'язок відношення з самим собою.

Файл – сукупність пов'язаних записів, що зберігаються в зовнішній пам'яті комп'ютера і розглядаються як єдине ціле. Зазвичай файл однозначно ідентифікується вказівкою імені файлу, його розширення і шляхи доступу до файлу.

Фізична схема – фізичні структури зберігання інформації бази даних.

Функціональна залежність – відношення між двома атрибутами, таке, що у будь-який момент часу кожному унікальному значенню атрибуту A у базі даних відповідає тільки одне значення атрибуту B.

Цілісність даних – правила, використовувані у базі даних, які гарантують, що дані збережені у базі, навіть якщо не є точними, принаймні правдоподібні.

Використана література

1. Автоматизированные информационные технологии в экономике : учебник / под ред. проф. Г. А. Титоренко. – М. : Компьютер, ЮНИТИ, 1999. – 400 с.

2. Ахмадеев И. А. Базы даннях : учебн. пособ. / И. А. Ахмадеев, А. Х. Хайруллин, С. Ю. Юрасов ; под общей редакцией проф. И. А. Ахмадеева. – Набережные Челны : Камский государственный политехнический институт, 2004. – 237 с.

3. Бази даних у питаннях і відповідях : навч. посіб. / укл. В. В. Чубук, Р. М. Чен, Л. А. Павленко та ін. – Х. : Вид. ХНЕУ, 2004. – 288 с.

4. Дейт К. Дж. Введение в системы баз данных / Дж. К. Дейт. – 8-е изд. – М. :Издательский дом "Вильямс", 2005. – 1328 с.

5. Карпова Т. С. Базы данных. Модели, разработка, реализация : учебник / Т. С Карпова. – СПб. : Питер, 2001. – 302 с.

6. Когаловский М. Р. Энциклопедия технологий баз данных (Эволюция технологий. Технологии и стандарты. Инфраструктура. Терминология) / М. Р. Когаловский. – М. : Финансы и статистика, 2002. – 836 с.

7. Конноли Т. Базы данных: проектирование, реализация и сопровождение : учебн. пособ. / Т. Конноли ; пер. с англ. – М. : Издательский дом "Вильямс", 2000. – 1120 с.

8. Маклаков С. В. BPwin и ERwin. CASE–средства разработки информационных систем / С. В. Маклаков. – М. : ДИАЛОГ-МИФИ, 1999. – 256 с.

9. Марков А. С. Базы данных. Введение в теорию и методологію : учебник / А. С. Марков, К. Ю. Лисовский. – М. : Финансы и статистика, 2006. – 512 с.

10. Мартьянова А. Е. Базы данных и знаний : учебн. пособ. – 2-е изд. / А. Е. Мартьянова. – Астрахань : Изд. АГТУ, 2009. – 291 с.

11. Методология IDEF1X. Информационное моделирование. – М. : МетаТехнология, 1993. – 120 с.

12. Мінухін С. В. Методи і моделі проектування на основі сучасних CASE-засобів : навч. посібн. / С. В. Мінухін, О. М. Беседовський, С. В. Знахур. – Х. : Вид. ХНЕУ, 2008. – 272 с.

13. Применение CASE-средств BPwin и ERwin для проектирования информационных систем. Компьютерный практикум / В. Д. Сапунцов, М. А. Лысенко, Ф. Я. Султанов и др. ; под ред. В. Д. Сапунцова. – М. : РГУ нефти и газа, 2000. – 53 с.

14. Пушников А. Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель даннях : учебн. пособ. / А. Ю. Пушников. – Уфа : Изд-е Башкирского ун-та, 1999. – 108 с.

15. Роб П. Системы баз данных: проектирование, реализация и управление / П. Роб, К. Коронер ; пер. с англ. – СПб. : БХВ-Петербург, 2004. – 1040 с.

16. Робоча програма навчальної дисципліни "Організація баз даних та знань" / укл. М. Ю. Лосєв, О. В. Тарасов, В. В. Федько. – Х. : Вид. ХНЕУ, 2010. – 67 с.

17. Рыбанов А. А. Инструментальные средства автоматизированного проектирования баз данных : учебн. пособ. и варианты заданий к лабораторным работам по дисциплине "Базы данных" / А. А. Рыбанов. – ВолгГТУ, Волгоград, 2007. – 96 с.

18. Федько В. В. Лабораторний практикум "Основи баз даних та знань" з модуля навчальної дисципліни "Організація баз даних та знань" : навчально-практичний посібник / укл. В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Х. : Вид. ХНЕУ, 2011. – 192 с.

19. Харрингтон Д. Р. Проектирование реляционных баз данных / Д. Р. Харрингтон. – М. : Лори, 2006. – 241 с.

20. Энсор Д. Oracle. Проектирование баз данных / Д. Энсор, Й. Стивенсон. – К. : BHV, 1999. – 560 с.

21. Chen P. P.-S. The Entity-Relationship Model - Toward a Unified View of Data // ACM TODS. – March 1976.– 1, № 1.

22. Date C.J. A Note on One-to-One Relationships // Relational Database Writings: 1985-1989. – Reading, Mass.: Addison-Wesley, 1990.

23. International Organization for Standardization (ISO): Information Technology – Database Languages – SQL, Document ISO / IEC 9075:1999.

24. Базы данных [Электронный ресурс]. – Режим доступа : http://any-book.org/download/ 35918.html.

25. Денормализация БД. Зачем? Когда? Как? [Электронный ресурс]. – Режим доступа : http://habrahabr.ru/blogs/mysql/64524/.

26. Зеленков Ю. А. Введение в базы данных [Электронный ресурс]. – Режим доступа : http://www.mstu.edu.ru/study/materials/ zelenkov/toc.html.

27. Карпова Т. С. Базы данных. Модели, разработка, реализация [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/department/ database/dbmdi/.

28. Кириллов В. В. Основы проектирования реляционных баз данных [Электронный ресурс]. – Режим доступа : http://citforum.ru/database/ dbguide/index.shtml.

29. Кузнецов С. М. Базы данных. Вводный курс [Электронный ресурс]. – Режим доступа : http://citforum.univ.kiev.ua/database/ advanced_intro/.

30. Принципы организации баз данных. Технологии баз данных: SQL, T-SQL, PL/SQL, реляционные БД [Электронный ресурс]. – Режим доступа : http://datasql.ru/ baseworkbd/2.htm.

31. Рубцов С. В. Методология структурного анализа и проектирования [Электронный ресурс]. – Режим доступа : http://www.cfin.ru/rubtsov/ RSV/SADT/SADT.htm.

32. Туманов В. Е. Проектирование хранилищ данных для приложений систем деловой осведомленности (Business Intelligence Systems) [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/department/ database/bispowerd/21/3.html.

33. Туманов В. Е. Основы проектирования реляционных баз данных [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/department/ database/rdbdev/ 10/3.html.

34. CA ERWIN® DATA MODELER. Managing the business of database design and data modeling [Электронный ресурс]. – Режим доступа : http://arcserve.com/us/ sitecore/content/Home/products/detail/CA-ERwin-Data-Modeler.aspx.

35. IDEF1X Data Modeling Method. [Электронный ресурс]. – Режим доступа : http://www.idef.com/IDEF1x.htm.

36. SQL.RU Client/Server Technologies/ Понимание SQL (Understanding SQL) [Электронный ресурс]. – Режим доступа : http://www.sql.ru/docs/sql/u_sql/.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Тарасов Олександр Васильович
Федько Віктор Васильович
Лосєв Михайло Юрійович

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ. ПРОЕКТУВАННЯ БАЗ ДАНИХ

Навчально-практичний посібник для самостійної підготовки студентів

Частина 1

Відповідальний за випуск **Пономаренко В. С.**

Відповідальний редактор **Сєдова Л. М.**

Редактор **Пушкар І. П.**

Коректор **Бутенко В. О.**

План 2011 р. Поз. № 60-П.
Підп. до друку Формат 60×90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 12,5. Обл.-вид. арк. 15,63. Тираж прим. Зам. №

Видавець і виготівник – видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***

Тарасов О. В.
Федько В. В.
Лосєв М. Ю.

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ. ПРОЕКТУВАННЯ БАЗ ДАНИХ

Навчально-практичний посібник
для самостійної підготовки студентів

Частина 1