**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ХАРЧОВИХ ТЕХНОЛОГІЙ**

**О.М. М'ЯКШИЛО**

# МОДЕЛЮВАННЯ БАЗ ДАНИХ ЗАСОБАМИ CASE - ТЕХНОЛОГІЇ ERWIN

**КОНСПЕКТ ЛЕКЦІЙ**
**з дисципліни "Структурне моделювання систем"**
для студентів спеціальності 6.080400 «Інформаційні управляючі системи та технології»
напряму 0804 "Комп'ютерні науки"
всіх форм навчання

СХВАЛЕНО
на засіданні кафедри
інформаційних систем
Протокол № 12
Від 11. 06.2007р.

**Київ НУХТ 2008 р.**

**М’якшило О.М.** МОДЕЛЮВАННЯ БАЗ ДАНИХ ЗАСОБАМИ CASE - ТЕХНОЛОГІЇ ERWin: Конспект лекцій з дисципліни “Структурне моделювання систем” для студ. спец 6.080400 «Інформаційні управляючі системи та технології» напряму 0804 “Комп’ютерні науки” всіх форм навчання – К.: НУХТ, 2008 — 60 с.

Рецензент: Кондращенко В.Я., д-р техн. наук

О.М. М’ЯКШИЛО, канд.техн.наук

**Видання подається в авторській редакції**

**© О.М. М’якшило, 2008**
**© НУХТ, 2008**

# ЗМІСТ

**ЗМІСТ** ..... **3**
**Модуль 1. Лекції 1-5** ..... **4**
**Лекція 1** ..... **4**
1. Загальна характеристика пакету моделювання ERWin. ..... 4
Питання для самоперевірки ..... 5
**Лекція 2** ..... **6**
2. Створення логічної моделі інформаційної системи на рівні визначень – Definitions model ..... 6
Питання для самоперевірки ..... 7
**Лекція 3** ..... **7**
3. Створення логічної схеми інформаційної системи на рівні атрибутів. ..... 7
Питання для самоперевірки ..... 9
**Лекція 4** ..... **9**
4. Встановлення зв’язків в логічній моделі інформаційної системи ..... 9
Питання для самоперевірки ..... 11
**Лекція 5** ..... **11**
5. Рекурсивний зв’язок та ім’я ролі ..... 11
Питання для самоперевірки ..... 12
**Тест до модуля 1** ..... **13**
**Модуль 2. Лекції 6-9** ..... **15**
**Лекція 6** ..... **15**
6. Зв'язок багато до багатьох та правила його реалізації ..... 15
Питання для самоперевірки ..... 16
**Лекція 7** ..... **16**
7. Встановлення типів залежності об’єктів та ієрархія наслідування. ..... 16
Питання для самоперевірки ..... 18
**Лекція 8** ..... **18**
8. Правила збереження цілісності посилань ..... 18
при транзакціях в базі даних. ..... 18
Питання для самоперевірки ..... 21
**Лекція 9** ..... **21**
9. Нормалізація даних в ERWin ..... 21
Питання для самоперевірки ..... 22
**Тест до модуля 2** ..... **22**
**Модуль 3. Лекції 10-16** ..... **25**
**Лекція 10** ..... **25**
10. Мова формування запитів SQL ..... 25
Питання для самоперевірки ..... 29
**Лекція 11** ..... **29**
11. Оператор вибору SELECT ..... 29
Питання для самоперевірки ..... 31
**Лекція 12** ..... **32**

12. Приклади використання оператора SELECT ..... 32
**Лекція 13** ..... **35**
13.1. Внутрішнє та зовнішнє об'єднання таблиць ..... 35
13.2. Вкладені запити ..... 36
Питання для самоперевірки ..... 39
**Лекція 14** ..... **39**
14. Оператори маніпулювання даними ..... 39
Питання для самоперевірки ..... 41
**Лекція 15** ..... **41**
15. Збережувані процедури та тригери в ERWin. ..... 41
Питання для самоперевірки ..... 44
**Лекція 16** ..... **44**
16. Створення та редагування тригерів в ERWin. ..... 44
Питання для самоперевірки ..... 46
Завдання ..... 46
**Тест до модуля 3** ..... **46**
**ЛІТЕРАТУРА** ..... **47**
**Додаток 1** ..... **47**
Контрольна робота до модуля 3 ..... 47
**Додаток 2** ..... **53**
Типи даних в SQL ..... 53
**Додаток 3** ..... **55**
Приклад побудови логічної моделі бази даних в середовищі ERWin ..... 55

## Модуль 1. Лекції 1-5

1. Загальна характеристика пакету моделювання ERWin.
2. Три рівні логічної моделі даних.
3. Побудова повної атрибутивної моделі.
4. Поняття про зовнішні ключі, альтернативні ключі та інверсні входи.
5. Зв'язки між об'єктами бази даних

## Лекція 1

### 1. Загальна характеристика пакету моделювання ERWin.

Створення сучасних інформаційних систем виливається в складну задачу, вирішення якої потребує використання спеціальних інструментальних засобів моделювання, аналізу та проектування систем.

Реалізацію проектів по створенню інформаційних систем прийнято розподіляти на стадії аналізу, проектування, програмного кодування, тестування та супроводження. Вартість виправлення помилок, виявлених на стадії тестування, у декілька разів перевищує вартість їх виправлення на попередніх стадіях. Щоб знизити рівень помилок при проектуванні, велика увага приділяється попередньому моделюванню системи та детальному аналізу моделі.

До ефективних засобів моделювання інформаційних систем відноситься один з інструментів CASE-технологій розроблений фірмою PLATINUM technology – ERWin. Пізніше права на подальшу розробку і використання пакету було продано фірмі Computer Associates (CA), яка випустила у 2002 році інтегрований пакет AllFusion Modeling Suit 4.1, який вміщує 5 продуктів для забезпечення повного циклу проектування систем. До складу цього пакету увійшов і ERWin 4.1 під назвою AllFusion ERWin Data Modeler. Слід наголосити, що ERWin є засобом не тільки моделювання, дослідження та проектування інформаційних систем, але також засобом генерування системного коду бази даних на сервері, коду клієнтського додатку , або системного коду для локальних баз даних. Для розподілення функцій в ERWin використовується два типи моделі даних: логічна модель та фізична модель. Логічна модель не залежить від типу СКБД і відображає об'єктну декомпозицію предметної області, для якої створюється інформаційна система. Фізична модель враховує характерні ознаки СКБД та структуру використовуваних технічних засобів. На її основі генерується системний код бази даних. Треба відмітити, що кожна логічна модель може мати декілька реалізацій у вигляді фізичних моделей в залежності від наявності СКБД підключених до ERWin.

В курсі розглядаються засоби пакету ERWin для побудови та оптимізації логічної моделі БД інформаційної системи. Вони дозволяють провести декомпозицію предметної області: визначити класи та атрибути об'єктів, встановити зв'язки між об'єктами, провести нормалізацію даних та визначити заходи щодо збереження цілісності даних. Створення фізичної моделі даних та генерація SQL коду системного каталогу БД для обраної СКБД викладається в дисципліні "Автоматизоване проектування комп'ютерних систем", яка є безпосереднім і логічним продовженням даної дисципліни.

Для створення моделей даних в ERWin будемо використовувати графічну мову моделювання IDEF1X, розроблену в свій час для потреб армії США і широко використовувану в державних закладах, фінансових та промислових корпораціях, а також в НАТО і Міжнародному валютному фонді, в якості стандарту. Крім того, ERWin підтримує нотацію IE (Information Engineering), використовувану в промисловості та DM (Dimentional Modeling) – розробка сховищ даних.

## Питання для самоперевірки

1. В чому полягає призначення CASE-технології ERWin ?
2. Які типи моделей даних розглядають в ERWin ?
3. В чому полягає різниця між логічною і фізичною моделлю даних?
4. До складу якого сучасного пакету проектування інформаційних систем входить ERWin ?
5. Які три нотації графічної мови підтримує ERWin ?

# Лекція 2

## 2. Створення логічної моделі інформаційної системи на рівні визначень – Definitions model

ERWin надає засоби, що допомагають розробнику створити наглядну логічну схему (модель) бази даних, провести її дослідження та оптимізацію, узгодити з замовником.

Існує три рівня логічної моделі даних:

- Діаграма об'єкт-зв'язок (ERD);
- Модель даних, що базується на ключах (Key Based model (KB);
- Повна атрибутивна модель (Fully Attributed model (FA).

Діаграма об'єкт-зв'язок являє собою модель даних верхнього рівня. Вона призначена для обговорення структури даних з експертами з боку замовника. При створені моделі на рівні визначень об'єкти бази даних зображують у вигляді прямокутників, в середині яких, записують визначення збережуваних об'єктів, вказують зв'язки між ними і отриману схему узгоджують з замовником, підчас у присутності юриста. Зв'язки між об'єктами встановлюють у відповідності з бізнес-логікою функціонування організації.

Головними компонентами діаграми об'єкт-зв'язок являються об'єкти, або сутності (Entity). .

Кожен об'єкт відображає незалежний обсяг інформації з предметної області. Ім'я об'єкту задається іменником у однині, який відповідає суті одного запису в базі даних. Наприклад: об'єкт називається *Замовник* з атрибутами: *номер замовника, прізвище замовника, адреса замовника.*

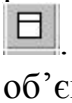
Об'єкт є екземпляром сутності. Сутність має таке ж ім'я, як і об'єкти, що до неї входять. Наприклад, сутність, яка відповідає множині замовників продукції підприємства, називають *Замовник*, тому, що вона має ті ж самі атрибути, що і об'єкт замовник. З цього витікає, що діаграма об'єкт-зв'язок на рівні об'єктів і на рівні сутностей буде мати один і той же вигляд.

З точки зору реляційної бази даних сутності відповідає окрема таблиця, а об'єкту – запис.

Інший важливий компонент діаграми - зв'язки. Зв'язки вказують логічну залежність між даними. Розрізняють наступні види зв'язків: один до одного, один до багатьох, багато до багатьох. Зв'язок багато до багатьох реалізований тільки на рівні логічної моделі.

### 2.1. Внесення об'єкту до моделі

Для створення логічної моделі вибирається рівень *логічна модель* зі списку, що знаходиться з правого боку інструментальної панелі ERWin.

Кнопка на панелі інструментів ERWin Toolbox перемикає курсор в режим побудови діаграми. Зображенню об'єкта на діаграмі відповідає кнопка  . При натисканні цієї кнопки виникає можливість перенесення зображення об'єкту на довільне місце екрану .

Переключення курсору у звичайний режим відбувається за допомогою кнопки .

На панелі інструментів ERWin Toolbox також знаходяться кнопки з зображенням зв'язків. Зв'язки встановлюються між двома сутностями (об'єктами). На рівні діаграми (логічна модель) розрізняють зв'язки типу один до багатьох (ідентифікований та неідентифікований) і багато до багатьох. На рівні фізичної моделі зв'язок багато до багатьох автоматично трансформується у декілька зв'язків один до багатьох. Більш детально проблему визначення та встановлення зв'язків розглянемо у наступних параграфах

### 2.2. Визначення (Definition) та опис об'єктів

Діалогове вікно Entity Editor, в якому задають ознаки об'єкту можна викликати натискуючи праву клавішу миші, коли курсор знаходиться на зображенні об'єкту. В цьому вікні можна задати назву сутності та її характеристики. Кожна сутність повинна бути визначена за допомогою текстового опису в закладці Definition. Закладки Note, Note2, Note3, UPD (User Defined Properties – якості визначені користувачем) слугують для внесення додаткових коментарів і ознак сутності та екземпляру сутності - об'єкту.

Закладка Note дозволяє доповнити додатковими зауваженнями опис об'єкту.

Закладка Note2 дозволяє занотувати деякі важливі запити до описуваного об'єкту.

Закладка Note3 слугує для внесення декількох прикладів даних сутності, що розглядається.

Приклад логічної моделі бази даних на рівні визначень для відділу постачання м'ясокомбінату наведено в додатку 3.

### Питання для самоперевірки

1. Що таке об'єкт, сутність?
2. Чим характеризується об'єкт, сутність?
3. Яким елементам фізичної бази даних відповідає об'єкт, сутність?
4. Яким елементам фізичної бази даних відповідає атрибут?
5. Для чого слугує закладка Note2? Наведіть приклади.
6. Для чого слугує закладка UPD? Наведіть приклади.
7. Які типи зв'язків ви знаєте? Наведіть приклади для кожного типу.

## Лекція 3

### 3. Створення логічної схеми інформаційної системи на рівні атрибутів.

Нагадаємо, що атрибут відображує визначену якість об'єкта. В реляційній базі даних атрибуту відповідає колонка таблиці.

Атрибут характеризується своїм іменем, типом та областю визначення. Атрибут може бути ключовим і не ключовим, може приймати значення NULL чи ні. Ключем (ключовим атрибутом) називають атрибут, який є унікальним

ідентифікатором запису. Ключів може бути декілька. Тоді розрізняють первинний ключ та інші.

Список атрибутів можна сформувати для конкретного об'єкту, якщо клацнути правою клавішею миші, коли її курсор знаходиться на зображенні об'єкту. З'являється діалогове вікно Attribute Editor, в якому задаються назви та ознаки атрибутів. Для визначення атрибутів користуються закладкою Definition. Первинний ключ задається в закладці General, встановленням відповідного прапорця .

Область визначення атрибута називається доменом. Або, інакше, домен - це сукупність значень з яких вибирається значення атрибуту. Кожен атрибут може визначатися лише на одному домені, але на одному домені може бути визначена множина атрибутів. До поняття домен входить не тільки тип даних, але і область існування даних. Домен може бути заданий тільки один раз і використовуватись, як у логічній, так і у фізичній моделі.

Домени дозволяють полегшити роботу з даними, як розробникам на етапі проектування, так і адміністраторам БД на етапі експлуатації системи. На логічному рівні домени можна описати без конкретних фізичних якостей. На фізичному рівні вони автоматично отримують специфічні якості притаманні вибраній СКБД, які можна змінити вручну.

Для створення домену в логічній моделі служить Domain Dictionary Editor. Його можна викликати з меню Edit/ Domain Dictionary при натискуванні на кнопку, яка розташована у верхній лівій частині закладки General вікна Attribute Editor.

ERWin має спеціальний інструмент, який значно полегшує створення нових атрибутів в моделі за рахунок використання доменів. Це вікно викликається, а також закривається гарячою клавішею CTRL+B.

Атрибут чи група атрибутів, які ідентифікують сутність називаються первинним ключем. Атрибути первинного ключа знаходяться на діаграмі в списку атрибутів вище горизонтальної лінії. На діаграмі об'єкт-зв'язок вони відсутні і з'являються вперше на діаграмі (Key Based model (KB).

Вибір первинного ключа не просте діло. Якість вирішення цієї задачі може вплинути на ефективність всієї системи в цілому. Часто декілька атрибутів претендують на роль первинного ключа. Їх називають потенційними ключами (Candidate key).

Ключі можуть бути складними, тобто складатися з декількох атрибутів, які на діаграмі вміщують вище горизонтальної лінії. Для того, щоб стати первинним потенційний ключ повинен задовольняти наступним вимогам:

- Бути унікальним – два записи не повинні мати однакових значень первинного ключа;
- Бути компактним – складний ключ не повинен вміщувати жодного атрибута, вилучення якого не впливає на унікальність;
- Первинний ключ не може мати нульового значення ;
- Значення атрибутів первинного ключа не змінюється впродовж всього часу існування системи (або суттєвості).

Потенційні ключі, що не стали первинними, можуть бути задані в якості альтернативних, тобто ключів за якими можна індексувати базу в разі потреби.

Іноді бізнес правила потребують вибрати в якості індексу атрибут, який не є унікальним для кожного запису, наприклад, *прізвище клієнта.* Такий атрибут називають інверсним входом і відповідно позначають на логічній схемі.

### Внесення первинного ключа

Первинний ключ вибирається з списку атрибутів. Первинний ключ може бути заданий за допомогою закладки General в діалоговому вікні Attribute Editor.

На схемі у зображенні об'єкту виділене спеціальне поле, для первинного ключа. Воно відокремлене від списку атрибутів горизонтальною лінією.

Атрибут, вибраний в якості первинного ключа, можна також перетягнути, за допомогою миші, до поля первинного ключа із списку атрибутів, що знаходяться нижче горизонтальної лінії.

### Питання для самоперевірки

1. Що таке атрибут ?
2. Що таке домен ?
3. На скількох доменах може бути заданий атрибут ?
4. Як задати домен у логічній моделі ?
5. Який інструмент існує для створення атрибута за значенням домену ?
6. Що таке первинний ключ? Які вимоги висуваються до первинного ключа ?
7. Чим можна пояснити необхідність використання первинного ключа?
8. Що таке альтернативний ключ ?
9. Для чого використовують інверсні входи ?
10. Коли виникає необхідність створення складного ключа ?

## Лекція 4

### 4. Встановлення зв'язків в логічній моделі інформаційної системи

Зв'язок це логічне відношення між об'єктами. На логічному рівні можна встановити ідентифікований зв'язок один-до-багатьох, багато-до-багатьох та не ідентифікований зв'язок один-до-багатьох.

Ідентифікований зв'язок встановлюється між незалежною (батьківський кінець зв'язку) з залежною (дочірній кінець зв'язку) сутностями. Атрибути первинного ключа батьківської сутності мігрують до області первинного ключа дочірньої сутності. В пакеті ERWin ідентифікований зв'язок показують на діаграмі цільною лінією з товстою крапкою на дочірньому кінці зв'язку. У випадку наявності ідентифікованого зв'язку між сутностями дочірня сутність стає залежною від батьківської, тобто не можна коригувати дочірню сутність довільним чином, а тільки за спеціальними правилами, щоб не порушити

узгодженість (цілісність) даних між батьківською і дочірньою сутностями. Залежна сутність зображується прямокутником з округленими кутами.

При встановленні неідентифікованого зв'язку дочірня сутність залишається незалежною, а атрибути первинного ключа батьківської сутності мігрують до складу не ключових компонентів батьківської сутності. Неідентифікований зв'язок слугує для зв'язування незалежних об'єктів. Наприклад: **Відділ** <*складається*> з **Співробітник**ів.

Екземпляр сутності **Співробітник** може існувати незалежно від будь-якого екземпляру сутності **Відділ,** тобто співробітник може працювати в організації не входячи до складу будь-якого відділу. Неідентифікований зв'язок показується на діаграмі пунктиром.

Кожен зв'язок повинен іменуватися дієсловом або дієслівною фразою (Rilationship Verb Phrases). Ім'я зв'язку відображує деяке обмеження чи бізнес-правило і допомагає читанню діаграми, наприклад:

- Кожен **Клієнт** <*розміщує*> **Замовлення**;
- Кожен **Замовлення** <*виконується*> **Співробітник**ом.

Зв'язок показує, які замовлення розмістив клієнт і який конкретно співробітник їх виконує.

**Встановлення зв'язків**

В палітрі інструментів кнопка відповідає ідентифікованому зв'язку, кнопка зв'язку багато-до-багатьох і кнопка відповідає неідентифікованому зв'язку.

Для створення нового зв'язку слід:

- Встановити курсор миші на потрібній кнопці в палітрі інструментів (ідентифікований чи неідентифікований зв'язок) і натиснути ліву кнопку миші;
- Клацнути спочатку по батьківському, а потім по дочірньому об'єкту.

Форму лінії зв'язку завжди можна змінити за допомогою миші.

Для редагування властивостей зв'язку треба клацнути правою кнопкою миші по зв'язку і вибрати на контекстному меню пункт Relationship Editor.

В закладці General вікна діалогу Relationship Editor можна задати потужність, ім'я та тип зв'язку.

За замовчуванням ім'я зв'язку на діаграмі не показується. Для відображення імені треба в контекстному меню вибрати пункт Display Options/Relationship і потім включити опцію Verb Phrase. Контекстне меню з'являється, якщо клацнути лівою кнопкою миші по будь-якому не зайнятому місцю діаграми.

Потужність зв'язку (Cardinality) – існує для позначення відношення кількості екземплярів батьківського об'єкту до кількості екземплярів дочірньої.

Розрізняють чотири типи потужності:

- Загальний випадок, коли одному екземпляру батьківської сутності відповідають 0,1 чи багато екземплярів дочірньої сутності не відмічають ніяким символом;

- Символом P відмічають випадок, коли одному екземпляру батьківської сутності відповідає 1 чи багато екземплярів дочірньої сутності (виключається нульове значення);
- Символом Z відмічають випадок, коли одному екземпляру батьківської сутності відповідають 0 чи 1 екземпляр дочірньої сутності (виключено множинні значення);
- Цифрою відмічається випадок точної відповідності, коли одному екземпляру з батьківської сутності відповідає заздалегідь задана кількість екземплярів дочірньої сутності.

**Зовнішні ключі**

Зовнішні ключі створюються автоматично, якщо існує зв'язок між сутностями. Атрибути первинного ключа в батьківській сутності переносяться до поля первинного ключа в дочірній сутності з позначкою (FK) після свого імені. Якщо підлегла сутність отримує один і той же зовнішній ключ від декількох батьківських сутностей, то щоб уникнути появи двох атрибутів з однаковими іменами автоматично виконується уніфікація ключа, тобто на діаграмі він показується один раз. Зовнішній ключ потрапляє до поля первинного ключа, якщо зв'язок між сутностями ідентифікований, а якщо неідентифікований, то – до загального списку атрибутів.

Приклади логічної моделі на рівні атрибутів з вказаними ключовими полями та всіма параметрами зв'язків для відділу постачання м'ясокомбінату наведені в додатку 3.

**Питання для самоперевірки**

1. Які типи зв'язків ви знаєте ?
2. Який принцип найменування зв'язків ?
3. Що таке потужність зв'язків?
4. Для зв'язків якого типу встановлюється потужність ?
5. Як створюється зовнішній ключ?
6. Що таке уніфікація зовнішніх ключів?

# Лекція 5

## 5. Рекурсивний зв'язок та ім'я ролі

Рекурсивний зв'язок встановлюється між екземплярами однієї сутності. Розрізняють два типи рекурсивного зв'язку: ієрархічний та мережний.

Ієрархічний зв'язок показує, що існує залежність предок–потомок між записами однієї сутності. Такий зв'язок може бути тільки неідентифікованим, тому що не між усіма записами існує ієрархічна залежність. Таким чином, відбувається посилання в дочірньому записі на ключовий атрибут батьківського запису, який мігрує в якості зовнішнього ключа до області не ключових атрибутів.

Наприклад, Державний Стандарт України (ДСТУ) може мати посилання, як на Міжнародні стандарти (МС), так і на статті власного стандарту. Для того, щоб розрізнити посилання в ДСТУ на різні статті з міжнародного стандарту

(МС) та на власні статті використовують *ім'я ролі,* тобто кожному зовнішньому ключу дають додаткову назву. ВнПос1. ID_D(FK) – реалізація ієрархічного рекурсивного зв'язку. Вн.Пос1– ім'я ролі, що означає внутрішнє посилання, тобто посилання на інший запис цієї ж таблиці .

Схема БД стандартів наведена на рисунку 5.1.

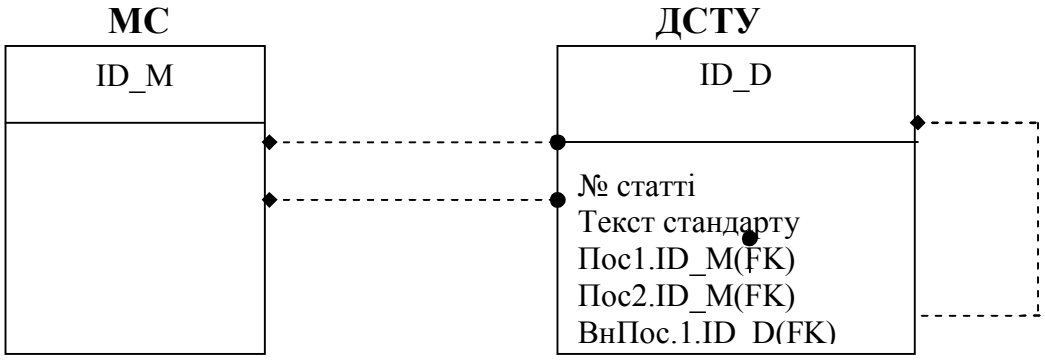



Рис. 5.1. Рекурсивний зв'язок та ім'я ролі в БД стандартів

Інший тип рекурсії – мережний відображає залежність багато до багатьох між екземплярами однієї сутності. Цей зв'язок можна реалізувати за допомогою додаткової сутності.

Наприклад, нехай задана БД перепису населення, в якій знаходяться відомості про членів кожної сім'ї та їх родові відносини між собою. Тобто, одна людина на прізвище Іванчук може бути як батьком так і сином відносно інших членів сім'ї. Для розв'язку такої колізії слід створити додаткову таблицю в якій вказати типи відношень між членами кожної сім'ї (рис. 5.2.).

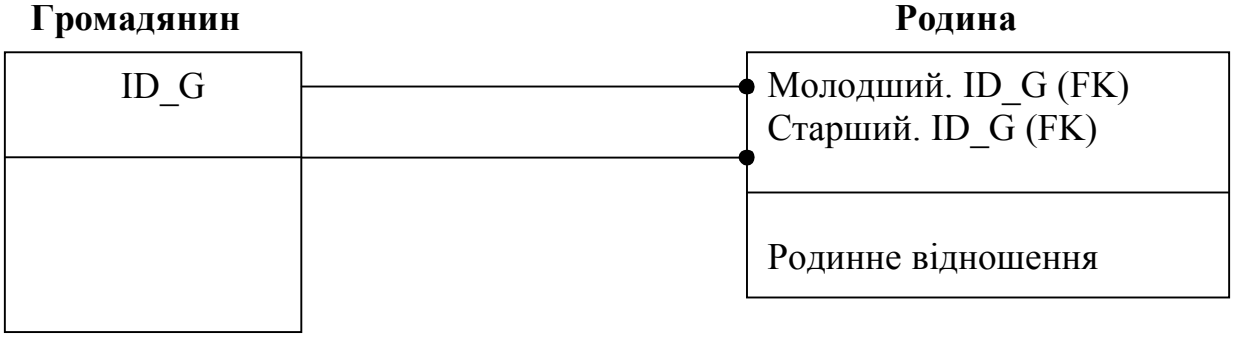



Рис. 5.2. Приклад реалізації мережної рекурсії в БД перепису населення

Як видно з наведених прикладів ім'я ролі широко використовується для реалізації рекурсивного зв'язку. Тобто, ім'я ролі використовують при перенесенні атрибутів в наслідок утворення зв'язків між об'єктами. Ім'я ролі присвоюють атрибуту в таких випадках:

- при наявності декількох однотипних зв'язків між двома сутностями;
- при наявності рекурсивного зв'язку між екземплярами однієї сутності.

**Питання для самоперевірки**

1. Що таке рекурсивний зв'язок ?
2. Які типи рекурсивних зв'язків ви знаєте?

3. Які правила розв’язання рекурсивного зв’язку вам відомі?
4. В яких випадках використовують ім’я ролі ?

## Тест до модуля 1

1. В предметній області сутності відповідає:
а) об’єкт;
б) клас об’єктів;
в) підмножина об’єктів одного класу;
г) сукупність об’єктів;
д) таблиця;
ж) запис.
2. В базі даних об’єкту відповідає:
а) зв’язок;
б) дані;
в) поле;
г) таблиця;
д) запис.
3. В базі даних сутності відповідає:
а) зв’язок;
б) дані;
в) поле;
г) таблиця;
д) запис.
4. В предметній області атрибутом називається:
а) характеристика об’єкту;
б) характеристика класу об’єктів;
в) поле;
г) характеристика бази даних;
д) таблиця;
ж) запис.
5. В базі даних атрибуту відповідає:
а) запис;
б) дані;
в) поле;
г) таблиця;
д) домен.
6. Область визначення атрибуту називається:
а) полем;
б) множиною;
в) даними;
г) доменом;
д) записом.
7. Чим визначається домен ?

а) набором даних в базі даних;
б) розміром даних;
в) обмеженнями на значення даних;
г) типом даних;
д) кількістю полів.
8. Candidate key в середовищі ERWin називаються:
а) альтернативні ключі;
б) зовнішні ключі;
в) унікальні ключі;
г) інверсні входи;
д) індексовані поля.
9. В базі даних інверсні входи можуть бути:
а) зовнішніми ключами;
б) первинними ключами;
в) унікальними ключами;
г) альтернативними ключами;
д) індексованими полями.
10. В базі даних альтернативні ключі можуть бути:
а) зовнішніми ключами;
б) первинними ключами;
в) унікальними ключами;
г) інверсними входами;
д) індексованими полями.
11. В логічній моделі бази даних іменником в однині іменується:
а) атрибут;
б) зв'язок;
в) об'єкт;
г) клас;
д) сутність.
12. В логічній моделі бази даних дієсловом іменується:
а) атрибут;
б) зв'язок;
в) об'єкт;
г) клас;
д) сутність.
13. Зв'язок один до багатьох в логічній моделі бази даних означає, що:
а) одному запису в батьківській таблиці відповідає багато записів в дочірній таблиці;
б) батьківська таблиця пов'язана з багатьма дочірніми таблицями;
в) первинний ключ батьківської таблиці пов'язаний з зовнішніми ключами дочірньої таблиці;
г) екземпляр батьківської сутності пов'язаний з багатьма екземплярами дочірньої сутності;
д) батьківська сутність пов'язана з багатьма дочірніми сутностями.
14. Ідентифікований зв'язок в логічній моделі бази даних означає, що:

а) кожен екземпляр батьківської сутності пов’язаний з багатьма екземплярами дочірньої сутності;

б) кожен екземпляр батьківської сутності пов’язаний хоча б з одним екземпляром дочірньої сутності;

в) батьківська сутність пов’язана хоча б з однією дочірньою сутністю;

г) одному запису в батьківській таблиці відповідає хоча б один запис в дочірній таблиці;

д) первинний ключ батьківської таблиці пов’язаний хоча б з одним зовнішнім ключем дочірньої таблиці.

15. Потужність зв’язку в логічній моделі бази даних показує :

а) скільки дочірніх сутностей зв’язано з батьківською сутністю;

б) скільки екземплярів дочірньої сутності пов’язано з одним екземпляром батьківської сутності;

в) скільки зовнішніх ключів дочірньої сутності пов’язано з первинним ключем батьківської сутності;

г) скільки записів батьківської таблиці пов’язано з одним записом дочірньої таблиці;

д) скільки полів дочірньої таблиці пов’язано з ключовим полем батьківської таблиці.

# Модуль 2. Лекції 6-9

1. Зв’язок багато до багатьох в логічній та фізичній моделі даних.
2. Типи відношень між об’єктами. Поняття про категоріальний зв’язок.
3. Збереження цілісності посилань при транзакціях в базі даних..
4. RI тригери та їх завдання в ERWin.
5. Нормалізація даних.

# Лекція 6

## 6. Зв'язок багато до багатьох та правила його реалізації

Зв'язок багато до багатьох можливий тільки на рівні логічної моделі даних. На рис. 6.1 показаний приклад зв'язку багато до багатьох. Лікар може приймати багато пацієнтів, пацієнт може лікуватися у кількох лікарів. Такий зв'язок позначається безперервною лінією з двома крапками на кінцях.

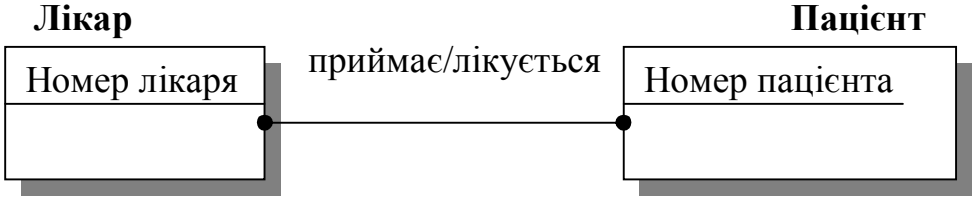



Рис. 6.1. Зв'язок багато до багатьох

Для внесення зв'язку слід встановити курсор на кнопці в палітрі інструментів, клацнути спочатку по одній, а потім по другій сутності.

Зв'язок багато які-до-багатьох повинен іменуватися двома фразами - в обидві сторони (в прикладі "приймає/лікується"). Це полегшує читання

діаграми. Зв'язок на рис. 6.1 слід читати ***"Лікар*** <приймає> ***Пацієнта",*** *а* ***"Пацієнт*** <лікується> у ***Лікаря"***.

При переході до фізичного рівня ERWin автоматично розв’язує зв'язок багато до багатьох, додаючи нову таблицю і встановлюючи два нових зв'язки один до багатьох від старих до нової таблиці (рис.6.2). При цьому ім'я нової таблиці надається автоматично як "Ім'я 1"& "Ім'я 2"

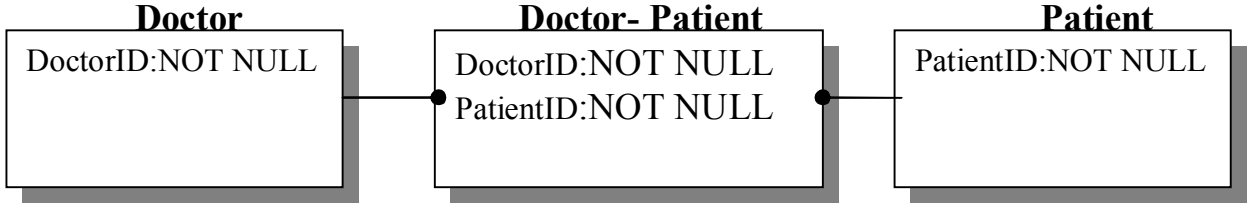



Рис. 6.2. Ілюстрація автоматичного вирішення зв'язку багато до багатьох на рівні фізичної моделі

Автоматичного рішення проблеми зв'язку багато-до-багатьох не завжди виявляється достатнім. В прикладі таблиця *Doctor Patient* має сенс візиту до лікаря, тому її слід перейменувати згідно бізнес-логики у *Visit.* Один и той же пацієнт може багато разів відвідувати лікаря, тому для того, щоб ідентифікувати візит, необхідно у склад первинного ключа таблиці *Visit* додати додаткову колонку, наприклад дату-час відвідування *(Visit Date time),* як показано на рис. 6.3.

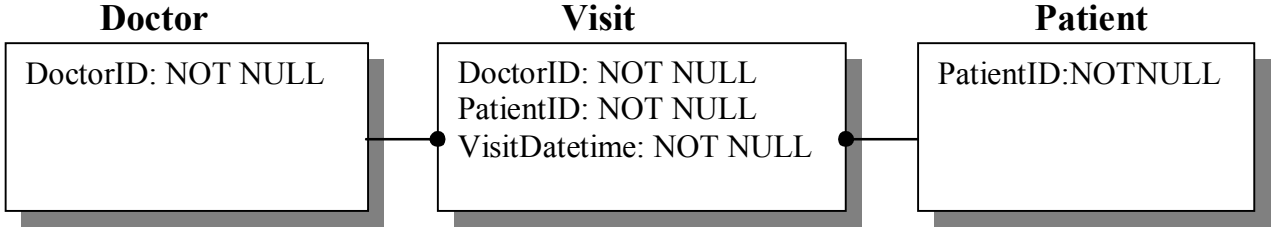



Рис. 6.3. Вирішення зв'язку багато-до-багатьох на рівні фізичної моделі

Слід зауважити, що після внесення додаткової колонки на фізичному рівні на логічному рівні представлення моделі не зміниться, діаграма буде виглядати так, як на рис.6.1.

**Питання для самоперевірки**

1. Що таке зв’язок багато до багатьох в моделі БД?
2. Які правила зображення зв’язку багато до багатьох в логічній моделі БД?
3. Які правила розв’язання зв’язку багато до багатьох в фізичній моделі БД?

# Лекція 7

## 7. Встановлення типів залежності об’єктів та ієрархія наслідування.

Розрізняють декілька типів залежних сутностей (об’єктів):

*Характеристична* – залежна дочірня сутність, яка пов’язана тільки з однією батьківською сутністю і за змістом зберігає інформацію про характеристики батьківської сутності.

*Асоціативна* - сутність, яка пов’язана з декількома батьківськими сутностями. Така сутність вміщує інформацію про зв’язки сутностей.

*Поіменована сутність* - та, що має тільки посилання на батьківські сутності і не має власних атрибутів – частковий випадок асоціативної.

*Категоріальна* – дочірня сутність в ієрархії наслідування.

Ієрархія наслідування (чи ієрархія категорій) – тип об'єднання сутностей, які мають загальні характеристики. Наприклад, в організації працюють службовці, які зайняті повний робочий день та сумісники. Загальні для них атрибути можна помістити у сутність Співробітник, а індивідуальні до сутностей Постійний співробітник та Сумісник. До батьківської сутності треба внести атрибут Тип, що дозволить відрізнити одну категоріальну сутність від іншої.

**Категорії повна та неповна.**

Ієрархії категорій діляться на повні та неповні. У повній категорії одному запису батьківської сутності відповідає запис у будь-якій дочірній сутності. Якщо категорія побудована не повністю, то в батьківській сутності можуть існувати значення атрибуту *Тип*, які не мають відповідних екземплярів у дочірніх сутностях. Наприклад, співробітник, крім вказаних відносин, ще може бути консультантом, або працювати за трудовою угодою. Якщо створити відповідні до цих категорій сутності *Консультант та Контрактник*, то ієрархію можна вважати повною.

Можлива комбінація повної та неповної категорій.

Створення категоріального зв'язку

Для створення категоріального зв'язку слід:

- Встановити курсор миші на кнопці в палітрі інструментів і натиснути ліву кнопку миші;
- Клацнути спочатку по батьківському предку, а потім по потомку;
- Для встановлення другого зв'язку в ієрархії категорій треба спочатку клацнути по символу категорії, а потім по другому потомку.
- Для редагування категорій треба клацнути правою кнопкою миші по символу категорії і вибрати у контекстному меню пункт Subtype Relationship Editor

**Стадії побудови ієрархії наслідування:**

- Виділити сутності з спільними атрибутами;
- Перенести спільні атрибути з дочірніх сутностей до батьківської сутності (якщо не існує батьківської сутності, то треба її створити і перенести до неї спільні для категорії атрибути);

**Створення неповної структури категорій**:

- створити категоріальний зв'язок від нової сутності – батьківської до дочірніх; нову сутність доповнити атрибутом дискримінатором категорії (ТИП);

**Створення повної структури категорій:**

- провести додатковий пошук сутностей, які мають спільні за змістом атрибути з батьківською сутністю. Спільні атрибути переносити до батьківської сутності. Ознаку повної категорії встановити в діалогу Subtype Relationship.

Приклад моделі для відділу постачання м'ясокомбінату наведено в додатку 3.

### Питання для самоперевірки

1. Які характерні ознаки категоріального зв'язку?
2. Якому типу відношень відповідає категоріальний зв'язок?
3. За якими правилами утворюється категоріальний зв'язок?
4. Чим відрізняється повна категорія від неповної? Як вони позначаються на діаграмі?

## Лекція 8

### 8. Правила збереження цілісності посилань при транзакціях в базі даних.

Транзакції – зміни в базі даних (редагування даних, доповнення новими записами, вилучення записів) повинні переводити базу даних з одного цілосного стану до іншого, тобто не рушити логічні зв'язки між даними, не приводити до втрат даних. Для забезпечення цілосності посилань в ERWin передбачені спеціальні процедури, які генеруються автоматично при встановлені, так званих, RI (referential integrity ) - тригерів.

RI-тригери – логічні конструкції, що відображують бізнес правила використання даних і являють собою правила доповнення, оновлення та вилучення (команди INSERT, UPDATE, DELETE мови запитів SQL) .

При наявності ідентифікованого зв'язку існують випадки , коли треба заборонити вилучення батьківського запису, якщо є хоча б один дочірній. В цьому випадку треба вилучити всі дочірні записи, а тільки потім батьківський. Таке правило називається "обмеженням" (Parent Delete RESTRICT). Обмеження з боку дочірньої сутності встановлюється правилом Chirld Delete RESTRICT.

Наприклад, існує батьківська сутність *Замовник* та дочірня сутність *Замовлення,* до якої заносяться всі виконані замовлення. При припиненні відносин між Замовником та Підприємством природно виникає необхідність у вилученні даних про замовника з бази. Але не можна знищувати дані про замовника, коли існують дані відносно виконаних замовлень. У цьому випадку з боку батьківської сутності можна встановити правило Parent Delete RESTRICT.

Правило, коли при вилученні батьківського запису вилучаються всі, пов'язані з ним, дочірні – називається "каскад" (Parent Delete CASCADE). Якщо записи в дочірній сутності за ім'ям ролі пов'язані з підлеглими записами нижчого рівня, то вилучаються і вони. У нашому прикладі про *Замовника* та *Замовлення*, якщо це продиктовано бізнес правилами, доцільно використати правило Parent Delete CASCADE. Тоді при вилученні *Замовник*а вилучаються всі пов'язані з ним *Замовлення*.

При наявності неідентифікованого зв'язку, коли запис в дочірній сутності може приймати значення нуль використовується правило SET NULL. Інший випадок, коли атрибуту зовнішнього ключа встановлюється значення за замовчуванням - SET DEFAULT.

Для обох випадків ідентифікованого і неідентифікованого зв’язку можна задати правило, коли зміни в кожній сутності відбуваються незалежно від інших – NONE.

Аналогічно правилам вилучення запису встановлюються правила доповнення або редагування запису.

**Встановлення правил цілісності посилань**

ERWin автоматично присвоює кожному зв’язку значення цілосності посилань, за замовчуванням, перед тим як помістити зв’язок до логічної схеми.

Якщо бізнес правила вимагають зміни правил цілосності посилань встановлених за замовчуванням, то їх можна задати, використовуючи закладку Rolename/RI Actions Relationship Editor.

Наприклад, для встановлення правила доповнення батьківської сутності новим записом тільки в тому випадку, коли введений хоча б один відповідний запис до дочірньої сутності слід виконати наступні дії:

- Задати потужність ідентифікованого зв’язку між батьківською і дочірньою сутностями “One or more” – один чи більше (тип P);
- Присвоїти дії RI- тригера значення “Parent Insert CASCADE”, для того, щоб при створенні нового запису в батьківській сутності створювався б новий запис у дочірній;
- Присвоїти дії RI- тригера значення “Parent Delete CASCADE”, для того, щоб при вилученні будь-якого запису батьківської сутності вилучалися б всі, пов’язані з ним, записи у дочірній сутності.

Бажана поведінка може бути досягнута наступними діями:

• Задати потужність зв'язку між сутностями *Команда і Гравець,* яка дорівнює "One 0r *тоге"* — 1 або більше (тип P). Передбачається, що ідентифікований зв'язок встановлено.

• Присвоїти дію I-тригера "Parent Insert-CASCADE" для того, щоб при створенні нового рядка в таблиці *Команда* автоматично створювалась хоча б один рядок в дочірній таблиці *Гравець.*

• Присвоїти зв'язку дію I-тригера "Parent Delete-CASCADE" для того, щоб при видаленні рядка із таблиці *Команда* відповідний рядок або рядок із таблиці *Гравець* теж видалялися.

ERWin автоматично присвоює кожному зв'язку значення цілісності посилання, яка встановлюється по замовчуванню, перш ніж додавати її в діаграму. Режими R1, що присвоюється ERWin по замовчуванню (наведені в табл. 2.4), можуть бути змінені в редакторі Referential Integrity Default, який маємо викликати клацнувши на кнопку RI Defaults діалогу Target Server (меню Server/Target Server).

*Таблиця 8.1.* ***Значення RI за замовчуванням та можливі режими для кожного типа зв'язку***

| | Ідентифікований зв'язок | Неідентифікований звязок (Nulls Allowed) | Неідентифікований зв'язок (No Nulls) | Категоріальний зв'язок |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| Child Delete Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |
| Child Delete Режими за замовчуванням | NONE | NONE | NONE | NONE |
| Child Insert Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |
| Child Insert Режими за замовчуванням | RESTRICT | SET NULL | RESTRICT | RESTRICT |
| Child Update Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |
| Child Update Режими за замовчуванням | RESTRICT | SET NULL | RESTRICT | RESTRICT |
| Parent Delete Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |
| Parent Delete Режими за замовчуванням | RESTRICT | SET NULL | RESTRICT | CASCADE |
| Parent Insert Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |
| Parent Insert Режими за замовчуванням | NONE | NONE | NONE | NONE |
| Parent Update Можливі режими | RESTRICT, CASCADE, NONE | RESTRICT, CASCADE, NONE, SET NULL, SET DEFAULT | RESTRICT, CASCADE, NONE, SET DEFAULT | RESTRICT, CASCADE, NONE |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Parent Update Режими за замовчуванням | RESTRICT | SET NULL | RESTRICT | CASCADE |

**Питання для самоперевірки**

1. Що таке транзакції в базі даних?
2. Коли використовуються правила збереження цілосності посилань?
3. Які правила збереження цілосності посилань ви знаєте?
4. Яким чином встановлюються правила цілосності посилань в ERWin?

# Лекція 9

## 9. Нормалізація даних в ERWin

### 9.1. Основні засади нормалізації даних

Нормалізація – процес перевірки і реорганізації сутностей та атрибутів з метою задоволення вимог до реляційної моделі даних. В результаті нормалізації повинна бути утворена структура даних, в якій інформація про кожний факт зберігається тільки один раз. Процес нормалізації – це послідовне приведення структури даних до нормальних форм. Існує шість різновидів нормальних форм. На практиці найчастіше використовують три форми: 1NF, 2NF, 3NF.

Нормальні форми базуються на понятті функціональної залежності. *Атрибут B сутності E функціонально залежить від атрибута A сутності E тоді і тільки тоді, коли кожне значення A в E пов'язане тільки з одним значенням B в E і єдиним чином визначає B.*

Повна функціональна залежність визначається як: *Атрибут B сутності E цілком функціонально залежить від множини атрибутів A сутності E тоді і тільки тоді, коли B функціонально залежить від A і не залежить від підмножини A.*

Тобто, кожен атрибут залежить від первинного ключа, а якщо ключ складний, то залежить від нього цілком, а не від його частини.

**Приведення сутності до першої нормальної форми**

Для приведення сутності до першої нормальної форми слід:

- Розділити складні атрибути на атомарні;
- Створити нову сутність;
- Перенести до неї всі атрибути, які повторюються;
- Вибрати або створити первинний ключ (PK) для нової сутності;
- Встановити ідентифікований зв'язок від попередньої сутності до нової;
- PK попередньої сутності стане зовнішнім ключем (FK) для нової.

**Приведення сутності до другої нормальної форми**

Для приведення сутності до другої нормальної форми слід:

- Виділити атрибути, які залежать тільки від частини первинного ключа, створити нову сутність;
- Перенести до неї всі атрибути, які залежать від частини ключа;

- Встановити ідентифікований зв’язок від попередньої сутності до нової. РК попередньої сутності стане зовнішнім ключем (FK) для нової.

### Приведення сутності до третьої нормальної форми

Для приведення сутності до третьої нормальної форми слід:

- Створити нову сутність і перенести до неї атрибути з однією і тією ж залежністю від не ключового атрибуту;
- Використати атрибут(и), що визначають цю залежність, в якості первинного ключа нової сутності;
- Встановити неідентифікований зв’язок від нової сутності до попередньої.
- ERWin не містить механізму нормалізації БД, але підтримує вимоги до нормалізованої БД за рахунок уніфікації ключів, заборони повторювань назв атрибутів, таблиць та зв’язків.
- Приклад нормалізованої БД відділу постачання м’ясокомбінату наведено в додатку 3.

### Питання для самоперевірки

1. Що таке повна і неповна функціональна залежність?
2. Що таке нормалізація даних Які форми нормалізації даних вам відомі?
3. Коли сутність знаходиться у першій нормальній формі?
4. Коли сутність знаходиться у другій нормальній формі?
5. Коли сутність знаходиться у третій нормальній формі?

## Тест до модуля 2

1. Рекурсивним зв’язком називається:

А) зв’язок між полями однієї сутності;

Б) зв’язок між записами однієї сутності;

В) зв’язок між атрибутами однієї сутності;

Г) зв’язок між екземплярами однієї сутності;

Д) зв’язок між двома сутностями з використанням імені ролі.

2. Рекурсивний зв’язок буває:

А) ієрархічним;

Б) поіменованим;

В) асоціативним;

Г) мережним;

Д) характеристичним.

3. Зв’язок багато до багатьох означає, що:

А) багато записів батьківської сутності пов’язано з багатьма записами дочірньої сутності;

Б) багато записів однієї сутності пов’язано з багатьма записами іншої сутності;

В) багато екземплярів дочірньої сутності пов’язано з багатьма екземплярами батьківської сутності;

Г) багато екземплярів однієї сутності пов’язано з багатьма екземплярами іншої сутності;

Д) багато атрибутів однієї сутності пов’язано з багатьма атрибутами іншої сутності.

4. Ім’я ролі використовується для:

А) розв’язання рекурсивних зв’язків;

Б) розв’язання зв’язку багато до багатьох;

В) розв’язання категоріального зв’язку;

Г) розв’язання декількох однотипних зв’язків по одному ключу;

Д) забезпечення міграції зовнішніх ключів.

5. Мережна рекурсія відповідає зв’язку типа:

А) один до багатьох;

Б) один до одного;

В) багато до багатьох;

Г) один до багатьох не ідентифікований;

Д) один до багатьох ідентифікований.

6. Ієрархічна рекурсія відповідає зв’язку типа:

А) один до багатьох;

Б) один до одного;

В) багато до багатьох;

Г) один до багатьох не ідентифікований;

Д) один до багатьох ідентифікований.

7. Залежну дочірню сутність, яка пов’язана з однією батьківською сутністю і зберігає її характеристики називають:

А) характеристичною;

Б) категоріальною;

В) поіменованою;

Г) асоціативною.

8. Залежну дочірню сутність, яка пов’язана з декількома батьківськими сутностями і зберігає інформацію про зв’язок між ними називають:

А) характеристичною;

Б) категоріальною;

В) поіменованою;

Г) асоціативною.

9. Залежну дочірню сутність, яка пов’язана з декількома батьківськими сутностями, зберігає інформацію про зв’язок між ними і містить тільки ключі батьківських сутностей називають:

А) характеристичною;

Б) категоріальною;

В) поіменованою;

Г) асоціативною.

10. Дочірню сутність в ієрархії наслідування називають:

А) характеристичною;

Б) категоріальною;

В) поіменованою;

Г) асоціативною.

11. Заборона вилучення екземпляра батьківської сутності, якщо є пов’язані з ним екземпляри дочірньої сутності відповідає правилу цілісності посилань:

А) Child Delete- CASCADE ;
Б) Parent Delete- RESTRICT;
В) Child Delete-RESTRICT;
Г) Child Update- CASCADE ;
Д) Parent Update- RESTRICT;
Е) Child Update-RESTRICT;
Є) Child Insert- CASCADE ;
Ж) Parent Insert- RESTRICT;
З) Child Insert-RESTRICT.

12. Видалення всіх екземплярів дочірньої сутності при видаленні зв’язаного з ними екземпляра батьківської сутності відповідає правилу цілісності посилань:

А) Child Delete- CASCADE ;
Б) Parent Delete- RESTRICT;
В) Child Delete-RESTRICT;
Г) Child Update- CASCADE ;
Д) Parent Update- RESTRICT;
Е) Child Update-RESTRICT;
Є) Child Insert- CASCADE ;
Ж) Parent Insert- RESTRICT;
З) Child Insert-RESTRICT.

13. Дозволяється доповнення дочірньої сутності новим екземпляром незалежно від наявності зв’язаного екземпляра батьківської сутності, що відповідає правилу цілісності посилань:

А) Child Insert- SET NULL ;
Б) Parent Insert- NONE;
В) Child Insert-SET DEFAULT;
Г) Child Insert- NONE ;
Д) Parent Insert- SET DEFAULT;
Е) Child Update- SET NULL;
Є) Child Insert- CASCADE ;
Ж) Parent Insert- RESTRICT;
З) Child Insert-RESTRICT.

14. Заборона оновлення екземпляра батьківської сутності, якщо є пов’язані з ним екземпляри дочірньої сутності відповідає правилу цілісності посилань:

А) Child Delete- CASCADE ;
Б) Parent Delete- RESTRICT;
В) Child Delete-RESTRICT;
Г) Child Update- CASCADE ;
Д) Parent Update- RESTRICT;

Е) Child Update-RESTRICT;
Є) Child Insert- CASCADE ;
Ж) Parent Insert- RESTRICT;
З) Child Insert-RESTRICT.

15. Встановлення значення по замовчуванню замість атрибута зовнішнього ключа в екземплярах дочірньої сутності при видаленні, пов'язаного з ними, екземпляра батьківської сутності відповідає правилу збереження цілісності посилань:

А) Child Delete- SET NULL ;
Б) Parent Delete- NONE;
В) Child Delete-SET DEFAULT;
Г) Child Delete- NONE ;
Д) Parent Delete- SET DEFAULT;
Е) Child Delete- SET NULL;
Є) Child Delete- CASCADE ;
Ж) Parent Delete- RESTRICT;
З) Child Delete-RESTRICT.

16. Які правила збереження цілісності посилань відповідають не ідентифікованому зв'язку при встановленні опції No Nulls.

А) RESTRICT;
Б) NONE;
В) SET DEFAULT;
Г) CASCADE;
Д) SET NULL.

# Модуль 3. Лекції 10-16

1. Стандарти мови SQL для баз даних з розподіленим доступом.
2. Приклади побудови запитів до бази даних.
3. Моделювання тригерів скриптів та процедур, що зберігаються.

# Лекція 10

## 10. Мова формування запитів SQL
## 10.1. Історія розвитку SQL

В сучасних СКБД звичайно підтримується єдина інтегрована мова, що вміщує всі необхідні засоби для роботи з БД, починаючи від її створення до забезпечення зручного інтерфейсу користувача БД. Стандартною мовою найбільш розповсюджених в теперішній час реляційних СКБД є мова структурованих запитів SQL. SQL - не процедурна мова, оператори якої вказують, які дії повинні виконуватись, а не спосіб їх виконання. Програмний код на SQL в декілька разів менший ніж програмний код написаний на процедурній мові. Це дозволяє виконувати обробку множини записів за один раз на відміну від обробки "запис за записом", що характерно для файлових систем.

*SQL (Structured Query Language) — Структурована Мова Запитів* — стандартна мова запитів в роботі з реляційними БД. Мова SQL з'явилася після реляційної алгебри. ЇЇ прототип було розроблено у кінці 70-х років в компанії IBM Research. Вона була реалізована в першому прототипі реляційної СКБД фірми IBM System R.

Перший міжнародний стандарт мови SQL був прийнятий у 1989 р. (далі будемо називати його SQL/89 або SQL1). Іноді стандарт SQL1 також називають стандартом ANSI/ISO.

В кінці 1992 р. був прийнятий новий міжнародний стандарт мови SQL, який в подальшому будемо називати SQL/92 або SQL2. І він має деякі недоліки, але в той же час є значно точнішим та більш повним, ніж SQL/89. У теперішній час більшість виробників СКБД внесли зміни у свої продукти таким чином, щоб вони у більшій мірі задовольняли стандарту SQL2.

У 1999 році з'явився новий стандарт, який був названий SQL3. Якщо відмінності між стандартами SQL1 та SQL2 здебільшого були кількісними, то стандарт SQL3 має якісні суттєві переробки. У SOL3 були введені нові типи даних, при цьому передбачена можливість завдання складних структурованих типів даних, які точніше відповідають об'єктній орієнтації. Додано розділ, що вводить стандарти на події та тригери, яких раніше не торкались в стандартах, хоча вони давно широко використовувані в промислових СКБД. У стандарті визначені можливості чіткої специфікації тригерів як сукупності події та дії. В межах управління транзакціями сталося повернення до старої моделі транзакції, яка допускає *точки збереження (savepoints)*, і можливість завдання в операторі відката ROOLBACK точок повернення, що дозволяє відкатувати транзакцію не до початку, а у проміжну, раніше збережену точку. Таке рішення збільшує гнучкість реалізації складних алгоритмів обробки інформації.

## 10.2. Структура SQL

На відміну від реляційної алгебри, де були представлені лише операції запитів до БД, SQL є повною мовою, в якій присутні не тільки операції запитів, але і оператори, відповідні мові опису даних (DDL — Data Definition Language). Крім того, мова SQL має оператори, які призначені для управління (адміністрування ) БД.

Відповідні розділи SQL наведені в таблицях 10.1 - 10.6.

*Таблиця 10.1.* ***Оператори визначення даних DDL***

| Оператор | Зміст | Дія |
|---|---|---|
| CREATE TABLE | Створити таблицю | Створює нову таблицю БД |
| DROP TABLE | Видалити таблицю | Видаляє таблицю БД |
| ALTER TABLE | Змінити таблицю | Змінює структуру існуючої таблиці або обмеження цілісності, що визначені для даної таблиці |
| CREATE VIEW | Створити представлення | Створює віртуальну таблицю, що відповідає SQL- запиту |
| ALTER VIEW | Змінити представлення | Змінює раніше створене представлення |

| Оператор | Зміст | Дія |
|---|---|---|
| DROP VIEW | Видалити представлення | Видаляє раніше створене представлення |
| CREATE INDEX | Створити індекс | Створює індекс для деякої таблиці , що забезпечує швидкий доступ за атрибутами, які входять в індекс |
| DROP INDEX | Видалити індекс | Видаляє раніше створений індекс. |

*Таблиця 10.2.* ***Оператори маніпулювання даними***
**Data Manipulation Language (DMP)**

| Оператор | Зміст | Дія |
|---|---|---|
| DELETE | Видалити рядок | Видаляє один або декілька рядків, що відповідають вимогам фільтрації, з базової таблиці. Узгоджується з принципами підтримки цілісності. |
| INSERT | Вставити рядок | Вставляє один рядок в базову таблицю. Дозволені модифікації оператора, при яких відразу декілька рядків можуть бути перенесені з однієї таблиці або запита до базової таблиці |
| UPDATE | Відновити рядок | Відновлює значення одного або декількох стовпців у одному або декількох рядках, що відповідають вимогам фільтрації. |

*Таблиця 10.3.* ***Мова запитів Data Query Language (DQL)***

| Оператор | Зміст | Дія |
|---|---|---|
| SELECT | Вибрати рядок | Оператор, який замінює всі оператори реляційної алгебри та дозволяє сформувати результуюче відношення, що відповідає запиту |

*Таблиця 10.4.* ***Засоби управління транзакціями***

| Оператор | Зміст | Дія |
|---|---|---|
| COMMIT | Завершити транзакцію | Завершити комплексну взаємопов'язану обробку інформації, об'єднану у транзакцію |
| ROLLBACK | Відкатити транзакцію | Відмінити зміни, проведені у процесі виконання транзакції |
| SAVEPOINT | Зберегти проміжну точку виконання транзакції | Збереження проміжного стану БД, відмічання його для подальшого повернення до нього |

*Таблиця 10.5.* ***Засоби адміністрування даних***

| Оператор | Зміст | Дія |
|---|---|---|
| ALTER DATABASE | Змінити БД | Змінити набір головних об'єктів у базі даних, обмежень, що стосуються всієї бази даних |
| ALTER DBAREA | Змінити область зберігання БД | Змінити раніше створену область зберігання |
| ALTER PASSWORD | Змінити пароль | Змінити пароль для всієї бази даних |
| CREATE DATABASE | Створити БД | Створити нову базу даних, визначивши основні параметри для неї |
| CREATE DBAREA | Створити область зберігання | Створити нову область зберігання та зробити її доступною для розміщення даних |
| DROP DATABASE | Видалити БД | Видалити існуючу базу даних (тільки в тому випадку, коли є права на виконання цієї дії) |
| DROP DBAREA | Видалити область зберігання БД | Видалити існуючу область зберігання (якщо в ній в теперішній час відсутні активні дані) |
| GRANT | Надати права | Надати права доступу на ряд дій над деяким об'єктом БД |
| REVOKE | Позбавити прав | Позбавити прав доступу до деякого об’єкту або деяким діям над об’єктом |

*Таблиця 10.6.* ***Програмний SQL***

| Оператор | Зміст | Дія |
|---|---|---|
| DECLARE | Визначає курсор для запиту | Задає деяке ім'я і визначає пов'язаний з ним запит до БД, який відповідає віртуальному набору даних |
| OPEN | Відкрити курсор | Формує віртуальний набір даних, що відповідає опису вказаного курсору та теперішньому стану БД |
| FETCH | Зчитати рядок із множини рядків, визначених курсором | Зчитує черговий рядок, який задається параметром команди з віртуального набору даних, що відповідають відкритому курсору |
| CLOSE | Закрити курсор | Закриває доступ до віртуального набору даних, які відповідають вказаному курсору |
| PREPARE | Підготувати оператор SQL до динамічного виконання | Генерує план виконання запиту, який відповідає заданому оператору SQL |
| EXECUTE | Виконати оператор SQL, що раніше був підготовлений до динамічного виконання | Виконує раніше підготовлений план запиту |

В промислових СКБД набір основних операторів розширений за рахунок включення операторів визначення та запуску збережуваних процедур та операторів визначення тригерів.

**Питання для самоперевірки**

1. З яких частин складається стандарт SQL2 ?
2. В чому полягає відміна сучасного стандарту SQL2 від стандарту SQL1?
3. Що нового з'явилося в стандарті SQL3 порівняно зі стандартом SQL2?
4. Як в сучасному стандарті враховані особливості клієнт-серверних БД?

# Лекція 11

## 11. Оператор вибору SELECT

Мова запитів (Data Query Language) в SQL складається з одного оператора SELECT. Цей єдиний оператор вибору реалізує всі операції реляційної алгебри. Однак один і той же запит може бути реалізований декількома способами, які можуть суттєво відрізнятися часом виконання, хоча будуть формувати однакові результати.

Головною перевагою оператора вибору SELECT в SQL є те, що він в значній мірі звільняє користувача від необхідності створення процедур пошуку даних (маємо на увазі процедури написані на процедурній мові). Синтаксис оператора SELECT має наступний вигляд:

SELECT [ALL [DISTINCT](<Список полів>|*)
FROM <Список таблиць>
[WHERE <Предикат-умова вибору або об'єднання>]
[GROUP BY <Список полів результата>]
[HAVING <Предикат-умова для групи>]
[ORDER BY <Список полів, за якими упорядковано виводжуванні дані>]

*SELECT* – перше слово всіх повідомлень SQL, за допомогою якого проводиться пошук даних в одній або декількох таблицях БД. За ключовим словом SELECT йдуть назви всіх полів (в тому числі вирахуваних), які слід включити в результуючу вибірку. Використовуючи знак * після SELECT, можна автоматично включити в запит всі поля вихідної таблиці.

Ключове слово *ALL* означає, що в результуючий набір рядків включаються всі рядки, що задовільняють умовам запиту. Тобто, в результуючий набір можуть потрапити однакові рядки. І це є порушенням принципів теорії відношень (на відміну від реляційної алгебри, де по замовчуванню передбачається відсутність дублікатів в кожному результуючому відношенні).

*DISTINCT* виключає повторні значення поля в наборі записів, які знайдені за допомогою SQL. DISTINCT, при визначенні дублікату, враховує тільки співпадання *полів* у вибраних записах повідомлення на відміну від DISTINCTROW, який вилучає запис з вибірки при дублюванні всього запису в цілому. Тобто, *DISTINCTROW* – виключити з вибірки дублікати записів.

*FROM* вказує, яка таблиця (або таблиці) вміщує необхідні нам поля.

*JOIN* означає зв'язок між записами, які знаходяться в різних таблицях. JOIN використовується разом с ключовими словами INNER, LEFT і RIGHT, які вказують тип зв'язку між таблицями.

*WHERE* означає умови вибору – критерії, використовувані при відборі записів для включення у вибірку.

За WHERE йде вираз з операцією порівняння. В результуючу вибірку входять тільки ті записи, для яких вираз має значення "ІСТИНА".

*GROUP BY (назва поля, полів)* означає створення групи за вказаним полем (полями).

*HEVING (умова)* створює умову для групи.

*ORDER BY (назва поля, полів)* забезпечує упорядкування за вказаним полем (полями).

Необхідно пам'ятати, що SQL це не програмний продукт, а загальний стандарт для формулювання запитів до баз даних. Хоча більшість версій SQL використовує одні й ті ж самі основні елементи, їх не можна назвати ідентичними. Кожна СКБД має свою версію SQL і слід пам'ятати, що один і той же запит до БД можна реалізувати різними способами.

**Предикати оператора SELECT**

SQL вміщує спеціальні предикати, які використовуються для запису виразів порівняння в реченнях WHERE та HAVING.

Предикат *Like.*

Цей предикат дозволяє використовувати логіку заміни у виразах. За предикатом Like йде рядок шаблонів, який може вміщувати довільну послідовність символів. Символ "%" замінює групу символів, "_" – один символ, "#" – одну довільну цифру. В наступному прикладі використовується слово % Київ %. Цей шаблон дозволяє включати до вибірки будь-який елемент даних, в якому зустрічається слово Київ. Наприклад: Київ Траст, Київські каштани, Київ & Дніпро:

*Where [Фірма] LIKE "% Київ %"*

Предикат Like часто використовують стосовно дат. В наступному прикладі наводиться пошук всіх дат, що відносяться до грудня 2004 року:

*Where [Дата] LIKE "12/%/2004"*

Предикат *In.*

Цей предикат дозволяє визначити, чи знаходиться елемент даних у переліку даних. Наступний вираз має значення "істина", якщо значенням поля "Місто" є одне з трьох міст, що входять до предиката In:

*Where [Місто] In ("Київ", "Харків", "Одеса")*

Порядок слідування елементів всередині дужок на дію предиката In не впливає.

Предикат *Between…And.*

Цей предикат може бути корисним при відборі значення, яке знаходиться всередині визначеного інтервалу. Наприклад, наступний вираз має значення "Істина", якщо значення поля Кількість знаходиться між 100 і 500:

*Where [Кількість] >= 100 And [Кількість] <= 500*

Предикати використовуються в операторах умови WHERE та HAVING.

## Сукупні функції SQL

Сукупні (Агрегативні) функції SQL наведені в таблиці 11.1. Вони автоматично оперують з поточним набором даних при формуванні запиту. В якості аргументу в агрегативних функціях використовується назва поля (колонки), тобто, функція оперує з множиною даних, які знаходяться у стовпчику таблиці. Як аргумент може виступати також вираз, утворений арифметичними операціями над назвами полів. Наприклад, для підрахунку загальної виручки за певний період можна використати наступний вираз:

*Сума= Sum ( [Ціна]*[Кількість]).*

Агрегативні функції використовують аналогічно назвам полів в операторі SELECT. Тобто, їх записують серед переліку полів і формують відповідні заголовки в отриманому, в наслідок виконання запиту, наборі даних. З функціями SUM та AVG можуть використовувати виключно числові поля. З функціями COUNT, MAX та MIN використовують, як числові, так і символьні поля. При виборі, в якості аргументів, символьних полів функції MAX та MIN будуть транслювати їх в еквівалент ASCII коду і обробляти у алфавітному порядку.

*Таблиця 11.1. **Агрегативні функції SQL***

| Назва операції | Операція |
|---|---|
| Avg ( ) | Середнє арифметичне |
| Count ( ) | Кількість записів |
| First ( ) | Значення першого запису |
| Last ( ) | Значення останнього запису |
| Min ( ) | Найменше значення |
| Max ( ) | Найбільше значення |
| Sum ( ) | Сума всіх записів |

Якщо в операторі SELECT використовують операцію групування, то агрегативні функції обробляють дані кожної групи окремо і формують для кожної групи підсумкові значення. В результуючій таблиці буде стільки рядків, скільки утворено груп в таблиці в наслідок операції групування. Кожна група має однакові дані в полі групування.

### Питання для самоперевірки

5. В чому полягає призначення оператора SELECT?
6. Які речення складають оператор SELECT?
7. В яких реченнях оператора SELECT використовують предикати умови?
8. Як забезпечити групування даних вибраних оператором SELECT?
9. Які функції називають агрегативними?
10. Які дані таблиці розглядають в якості аргументів агрегативних функцій?

# Лекція 12

## 12. Приклади використання оператора SELECT

**Приклад 1.** Створити запит з БД відділу збуту молокозаводу, заданої відношенням ЗАМОВЛЕННЯ (код замовлення, назва замовника, дата замовлення, назва продукції, одиниці виміру, кількість, ціна). Вибрати продукцію, яку замовили 12.01.06 р. та вказати кількість замовленої продукції.

Вихідні дані представлені у таблиці Замовлення.

Оператор вибору має сформувати результуючу таблицю, яка за умовою, вміщує поля [Назва продукції ] та [Кількість]. Такі дані можна використати для планування випуску продукції, тому можна сказати, що задача має практичний сенс, хоча значно спрощена відносно реальної для полегшення сприйняття матеріалу.

*Таблиця **ЗАМОВЛЕННЯ***

| Код замовлення | Замовник | Дата | Назва продукції | Од_виміру | Кількість | Ціна за од., грн. |
|---|---|---|---|---|---|---|
| 1 | “Фуршет” | 12.01.06 | Молоко 1л., жирн. 2,5% | Уп. | 500 | 2,7 |
| 2 | “Фуршет” | 12.01.06 | Кефір 0,5 л., жирн. 2,5% | Уп | 500 | 1,2 |
| 3 | “Фуршет” | 13.01.06 | Кефір 1 л., жирн. 2,5% | Уп | 200 | 3,2 |
| 4 | “Фуршет” | 12.01.06 | Сметана 0,5л.,жирн. 20% | Уп | 150 | 3,9 |
| 5 | “Сільпо” | 12.01.06 | Молоко 1л., жирн. 2,5% | Уп | 100 | 2,7 |
| 6 | “Сільпо” | 13.01.06 | Кефір 0,5 л., жирн. 2,5% | Уп | 200 | 1,2 |
| 7 | “Сільпо” | 12.01.06 | Сметана 0,2л.,жирн. 15% | Уп | 200 | 2,2 |
| 8 | “Біла” | 13.01.06 | Кефір 0,5 л., жирн. 2,5% | Уп | 400 | 1,2 |
| 9 | “Біла” | 13.01.06 | Кефір 1 л., жирн. 2,5% | Уп | 300 | 3,2 |
| 10 | “Біла” | 13.01.06 | Сметана 0,2л.,жирн. 15% | Уп | 500 | 2,2 |
| 11 | “Біла” | 13.01.06 | Молоко 1л., жирн. 2,5% | Уп | 300 | 2,7 |

Отже запит має вигляд:

```
SELECT [Назва продукції ] , [Од_виміру], [Кількість]
    FROM ЗАМОВЛЕННЯ
    WHERE  [Дата] = 12.01.06
```

В результаті отримаємо таблицю:

| Назва продукції | Од_виміру | Кількість |
|---|---|---|
| Молоко 1л., жирн. 2,5% | Уп. | 500 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 500 |
| Сметана 0,5л.,жирн. 20% | Уп | 150 |
| Молоко 1л., жирн. 2,5% | Уп | 100 |
| Сметана 0,2л.,жирн. 15% | Уп | 200 |

**Приклад 2.** Використовуючи дані з прикладу 1 вибрати продукцію, жирність якої складає 2,5%.

Текст запиту:

*SELECT DISTINCT [Назва продукції ]*
*FROM ЗАМОВЛЕННЯ*
*WHERE [Назва продукції ] LAKE %жирність2,5%.*

Результат:

| Назва продукції |
|---|
| Молоко 1л., жирн. 2,5% |
| Кефір 0,5 л., жирн. 2,5% |
| Кефір 1 л., жирн. 2,5% |

Завдяки слову DISTINCT вилучено повторення рядків результуючої таблиці.

**Приклад 3.** Використовуючи дані з прикладу1, підрахувати кількість продукції кожного виду, замовленої 12.01.06.

Текст запиту:

*SELECT [Назва продукції ] , [Од_виміру], Sum[Кількість] As [Загальна кількість]*
*FROM ЗАМОВЛЕННЯ*
*WHERE [Дата] = 12.01.06*
*GROUP BY [Назва продукції ]*

Результат запиту має наступний вигляд:

| Назва продукції | Од_виміру | Загальна кількість |
|---|---|---|
| Молоко 1л., жирн. 2,5% | Уп. | 600 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 500 |
| Сметана 0,5л.,жирн. 20% | Уп | 150 |
| Сметана 0,2л.,жирн. 15% | Уп | 200 |

Для формування результуючого запиту використали агрегативну функцію Sum, а для заголовок поля сформували використовуючи псевдонім після слова AS.

**Приклад 4**. Використовуючи дані з прикладу1, підрахувати кількість продукції кожного виду, замовленої на кожну вказану дату.

Текст запиту:

SELECT [Назва продукції ], [Од_виміру], [Дата], Sum[Кількість], As [Загальна кількість]
FROM ЗАМОВЛЕННЯ
GROUP BY [Дата], [Назва продукції ]

Результат запиту має наступний вигляд:

| Назва продукції | Од_виміру | Дата | Загальна кількість |
|---|---|---|---|
| Молоко 1л., жирн. 2,5% | Уп. | 12.01.06 | 600 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 12.01.06 | 500 |

| | | | |
|---|---|---|---|
| Сметана 0,5л.,жирн. 20% | Уп | 12.01.06 | 150 |
| Сметана 0,2л.,жирн. 15% | Уп | 12.01.06 | 200 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 13.01.06 | 600 |
| Кефір 1 л., жирн. 2,5% | Уп | 13.01.06 | 5 00 |
| Сметана 0,2л.,жирн. 15% | Уп | 13.01.06 | 500 |
| Молоко 1л., жирн. 2,5% | Уп | 13.01.06 | 300 |

**Приклад 5.** Використовуючи дані з прикладу1, підрахувати на яку суму реалізовано продукції кожного виду на кожну вказану дату.

Текст запиту:

*SELECT [Назва продукції ], [Од_виміру], [Дата],*
*Sum([Кількість]*[Ціна]) As [Сума]*
*FROM ЗАМОВЛЕННЯ*
*GROUP BY [Дата] , [Назва продукції ]*

Результат запиту

| Назва продукції | Од_виміру | Дата | Сума |
|---|---|---|---|
| Молоко 1л., жирн. 2,5% | Уп. | 12.01.06 | 1620 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 12.01.06 | 600 |
| Сметана 0,5л.,жирн. 20% | Уп | 12.01.06 | 585 |
| Сметана 0,2л.,жирн. 15% | Уп | 12.01.06 | 440 |
| Кефір 0,5 л., жирн. 2,5% | Уп | 13.01.06 | 720 |
| Кефір 1 л., жирн. 2,5% | Уп | 13.01.06 | 1600 |
| Сметана 0,2л.,жирн. 15% | Уп | 13.01.06 | 1100 |
| Молоко 1л., жирн. 2,5% | Уп | 13.01.06 | 810 |

**Приклад 6.** Використовуючи дані з прикладу 1, визначити яка продукція дала максимальну виручку на кожну вказану дату.

Для цього підрахуємо загальну суму по кожному виду продукції, групуючи дані по даті та назві продукції.

Запит1

*SELECT [Назва продукції], [Од_виміру], [Дата],*
*Sum([Кількість]*[Ціна]) As [Сума]*
*FROM ЗАМОВЛЕННЯ*
*GROUP BY [Дата] , [Назва продукції ]*

Другий запит допоможе нам з підрахованих сум вибрати найбільші по кожному виду продукції:

*SELEKT Max([Сума])*
*FROM Запит1*

Результат виконання запитів

| Назва продукції | Од_виміру | Дата | Сума |
|---|---|---|---|
| Молоко 1л., жирн. 2,5% | Уп. | 12.01.06 | 1620 |
| Кефір 1 л., жирн. 2,5% | Уп | 13.01.06 | 1600 |

# Лекція 13

## 13.1. Внутрішнє та зовнішнє об'єднання таблиць

В попередньому параграфі ми розглядали приклади вибірки даних з однієї таблиці. Однак, в реальній базі даних кожна таблиця зберігає дані про об'єкти одного класу і для наведення відповідності між цими об'єктами, в межах однієї предметної області, створюють зв'язки між таблицями. Яким же чином умова зв'язку між таблицями відображена в мові SQL? В стандарті SQL1 умову зв'язку кортежів з різних відношень задавали в реченні WHERE, шляхом порівняння значень ключів. Наприклад: З бази даних заданої відношеннями ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, ціна, сума) вибрати замовників, що замовляли продукцію не пізніше 31.12.2005 р. Текст оператора має наступний вигляд:

*SELECT DISTINCT ЗАМОВНИК.назва замовника*
*FROM ЗАМОВНИК , ЗАМОВЛЕННЯ*
*WHERE ЗАМОВНИК. Код замовника = ЗАМОВЛЕННЯ.код замовника*
*AND ЗАМОВЛЕННЯ.дата < 31.12.2005*

Зв'язок між таблицями вказано в реченні *WHERE ЗАМОВНИК. Код замовника = ЗАМОВЛЕННЯ.код замовника.* В цьому випадку до результуючої таблиці потрапляють кортежі, які відповідають умові зчеплення. Для наведеного прикладу такої умови досить. Таке об'єднання відношень називається внутрішнім. Інакше внутрішнє об'єднання можна записати наступним чином:

*SELECT DISTINCT ЗАМОВНИК.назва замовника*
*FROM ЗАМОВНИК INNER JOIN ЗАМОВЛЕННЯ ON ЗАМОВНИК. Код замовника = ЗАМОВЛЕННЯ.код замовника*
*WHERE ЗАМОВЛЕННЯ.дата < 31.12.2005*

Тобто, в реченні FROM використовують оператор JOIN з службовим словом INNER. Це зроблено для того , щоб уніфікувати операцію об'єднання відношень. В такому випадку вид об'єднання визначається службовими словами. В SQL2 передбачені стандарти, що описують різні варіанти об'єднання відношень, які на відміну від внутрішнього об'єднання, називаються зовнішніми. Загалом, в SQL2 синтаксис речення FROM має наступний вигляд:

FROM <перелік вихідних таблиць> | <вираз природного об'єднання> | <вираз об'єднання> | <вираз перехресного об'єднання > | <вираз запита на об'єднання>,

де <перелік вихідних таблиць>::= <назва_таблиці1> [<синонім таблиці1>] […] [, <назва_таблиціN> [<синонім таблиціN>] ] ;

<вираз природного об'єднання> ::= <назва_таблиці1> NATURAL { INNER | FULL [OUTER] | LEFT [OUTER] | RIGHT [OUTER] } JOIN <назва_таблиці2> ;

<вираз перехресного об'єднання > ::= <назва_таблиці1> CROSS JOIN <назва_таблиці2> ;

<вираз запита на об’єднання> ::= <назва_таблиці1> UNION JOIN
<назва_таблиці2> ;
<вираз об’єднання> ::= <назва_таблиці1> {INNER | FULL [OUTER]
| LEFT [OUTER] | RIGHT [OUTER]} JOIN {ON <умова об’єднання> | [USING (список стовпчиків )]} <назва_таблиці2> ;

Слова LEFT/RIGHT використовують у випадках, коли необхідно мати всі записи з однієї таблиці і показати зв’язані з нею записи іншої таблиці.

Для виділення головної таблиці оператор JOIN доповнюється службовими словами RIGHT або LEFT, які вказують праворуч чи ліворуч від оператора JOIN знаходиться головна таблиця. Наприклад, не всі замовники вже зробили замовлення, хоча, на основі угоди в таблицю ЗАМОВНИК дані про них занесено. Виникає наступна задача – з таблиці ЗАМОВНИК вибрати замовників, які замовляли продукцію у 2005 році, але ще не зробили замовлення у 2006 році.

В мові SQL2, використовуючи оператор JOIN запит має вигляд:

*SELECT DISTINCT ЗАМОВНИК.назва замовника*
*FROM ЗАМОВНИК LEFT JOIN ЗАМОВЛЕННЯ ON*
*ЗАМОВНИК. Код замовника = ЗАМОВЛЕННЯ. Код замовника*
*WHERE Замовлення.дата < 31.12.2005)*

Слово CROSS використовують, коли необхідно мати у вибірці всі варіанти об’єднання записів з обох таблиць, тобто кожний рядок з однієї таблиці об’єднується з кожним рядком другої таблиці. Таке об’єднання відповідає операції розширеного декартового добутку з реляційної алгебри.

Операція запиту на об’єднання еквівалентна операції теоретико-множинного об’єднання реляційної алгебри. При цьому необхідно підтримувати еквівалентність схем вихідних відношень. Запит на об’єднання виконують за наступною схемою:

*SELECT - запит*
*UNION*
*SELECT - запит*
*UNION*
*SELECT - запит*

Всі запити, використовувані в запиті на об’єднання не повинні містити виразів, тобто, обчислюваних полів.

## 13.2. Вкладені запити

В SQL існує можливість вкладання запитів один в одного. Як правило, внутрішній запит, який називають SubQuery, генерує значення, яке перевіряється в предикаті умови зовнішнього запита ( в реченнях WHERE або HAVING ). В частині FROM оператора SELECT дозволено примінять синоніми до назви таблиці, якщо при виконанні запита з підзапитом маємо використовувати два екземпляра однієї таблиці – один у запиті, другий у підзапиті. Синоніми створюють з використанням ключового слова AS, яке можна опустити. В якості синонімів використовують літери, літери з цифрами, або сукупність літер коротшу ніж назва таблиці.

FROM ЗАМОВНИК AS R1, ЗАМОВЛЕННЯ AS R2 або
FROM ЗАМОВНИК R1, ЗАМОВЛЕННЯ R2

Розглянемо вкладені запити на прикладі БД відділу збуту молокозаводу.

**Приклад 1** . В базі даних заданій відношенням Замовлення визначити дату максимального замовлення за кількістю.

*SELECT дата, кількість*
*FROM ЗАМОВЛЕННЯ*
*WHERE кількість = (SELECT MAX(кількість) FROM ЗАМОВЛЕННЯ)*

**Приклад 2.** Визначити дати оборуток, коли кількість замовленого товару перевищила середнє значення та підрахувати величину перевищення.

*SELECT дата, кількість, кількість - (SELECT AVG(кількість) FROM ЗАМОВЛЕННЯ) AS перевищення*
*FROM ЗАМОВЛЕННЯ*
*WHERE кількість > (SELECT AVG(кількість) FROM ЗАМОВЛЕННЯ )*

**Приклад 3.** Визначити замовників, які зробили замовлення на максимальну кількість продукції.

*SELECT ЗАМОВНИК. назва*
*FROM ЗАМОВНИК NATURAL INNER JOIN ЗАМОВЛЕННЯ*
*WHERE ЗАМОВЛЕННЯ.кількість = (SELECT MAX (ЗАМОВЛЕННЯ. кількість) FROM ЗАМОВЛЕННЯ)*

За рахунок використання слова *NATURAL* ми можемо не використовувати вираз порівняння для кодів зв'язку між таблицями, тому що це слово передбачає наявність у різних таблицях полів для зв'язку з однаковими назвами.

Разом з підзапитом часто використовують предикат EXISTS або NOT EXISTS. Предикат EXISTS має значення ІСТИНА, коли результат виконання підзапиту не порожній і навпаки, NOT EXISTS має значення ІСТИНА, коли результат виконання підзапиту порожній.

**Приклад 4.** В базі даних, заданій відношеннями ЗАМОВНИК та ЗАМОВЛЕННЯ, вибрати замовників, що не брали на нашому підприємстві молоко у листопаді 2006 року та підрахувати сумарну виручку від реалізації продукції цим замовникам. У вибірку включити тільки тих замовників, для кого сума реалізації перевищує 5000 грн.

*SELECT ЗАМОВНИК. Назва замовника, SUM (ЗАМОВЛЕННЯ. Кількість * ЗАМОВЛЕННЯ.Ціна) as Сума*
*FROM ЗАМОВНИК NATURAL INNER JOIN ЗАМОВЛЕННЯ*
*WHERE ЗАМОВЛЕННЯ.дата LIKE "%11.2006"*
*NOT EXISTS*
*( SELECT ЗАМОВНИК. Назва замовника*
*FROM ЗАМОВНИК NATURAL INNER JOIN ЗАМОВЛЕННЯ WHERE ЗАМОВЛЕННЯ.Дата LIKE "%11.2006" AND ЗАМОВЛЕННЯ. Назва продукції LAKE "молоко%")*
*GROUP BY (Назва замовника)*
*HAVING Сума > 5000*

В стандарті SQL2 оператори порівняння розширено до багатократних порівнянь з використанням ключових слів ANY та ALL. Таке розширення використовують для порівняння вибраного стовпчика таблиці зі стовпчиком даних поверненим підзапитом.

Ключове слово ANY використовують тоді, коли хоча б один рядок у результаті виконання підзапита відповідає умові.

**Приклад 5.** У БД молокозаводу заданій відношеннями ЗАМОВНИК та ЗАМОВЛЕННЯ вибрати назви замовників, які хоча б один раз у 2006 р. взяли будь-яку продукцію більше, ніж на 1000 грн.

*SELECT R1.Назва замовника*
*FROM ЗАМОВНИК R1, ЗАМОВЛЕННЯ R2*
*WHERE YEAR(R2.Дата) =”2006” AND 1000< ANY*
*( SELECT (Кількість*Ціна) AS Вартість*
*FROM ЗАМОВЛЕННЯ R3*
*WHERE R2.Код замовника =*
*R3. Код замовника )*

Ключове слово ALL вимагає, щоб предикат порівняння мав би значення “Істина” під час порівняння з усіма рядками підзапиту.

**Приклад 6.** У БД молокозаводу заданій відношеннями ЗАМОВНИК ЗАМОВЛЕННЯ та ПРОДУКЦІЯ вибрати назви замовників, які замовляли весь асортимент продукції у 2006 р.

*SELECT DISTINCT R1. назва замовника*
*FROM ЗАМОВНИК AS R1 NATURAL INNER JOIN ЗАМОВЛЕННЯ AS R2*
*WHERE YEAR(R2.Дата) =”2006” AND R2. код продукції =*
*ALL ( SELECT R3. код продукції FROM ЗАМОВЛЕННЯ AS R3 NATURAL RIGHT JOIN ПРОДУКЦІЯ WHERE R2. код замовника = R3. код замовника)*

В даному запиті відбувається подвійне використання таблиці ЗАМОВЛЕННЯ: в основному запиті для вибірки замовників (синонім R2), які замовляли продукцію у 2006 році і в підзапиті (синонім R3), для виявлення замовлень по всіх видах продукції. Для того, щоб зв’язати ці два запити в підзапиті використано умову R2. код замовника = R3. код замовника.

Цю задачу можна вирішити використовуючи операцію групування і підзапит у реченні HAVING. В даному рішенні відбувається підрахунок кількості видів продукції, що випускає завод. Аналогічний підрахунок відбувається по замовленнях, які згруповані за замовником. Слово Distinct використовують для того, щоб не враховувати одну й ту ж продукцію по декілька разів.

*SELECT DISTINCT R1. назва замовника*
*FROM ЗАМОВНИК AS R1 NATURAL INNER JOIN ЗАМОВЛЕННЯ AS R2*
*WHERE YEAR(R2.Дата) =”2006”*
*GROUP BY (R1. назва замовника )*
*HAVING COUNT (DISTINCT R2. код продукції)= (SELECT COUNT (R3.код продукції) FROM ПРОДУКЦІЯ AS R3*

**Питання для самоперевірки**

1. Які способи об'єднання таблиць в операторі SELECT ви знаєте?
2. Чим відрізняється зовнішнє об'єднання таблиць від внутрішнього?
3. Які види зовнішнього об'єднання таблиць ви знаєте?
4. За якої умови запит на об'єднання (UNION) буде коректним?
5. Якій операції реляційної алгебри відповідає перехресне об'єднання таблиць?
6. В яких випадках використовують вкладені запити?
7. В яких реченнях SQL є можливість використання вкладеного запиту?
8. Які допоміжні слова використовують перед вкладеним запитом? Що означає кожне з них?

# Лекція 14

## 14. Оператори маніпулювання даними

До операцій маніпулювання даними відносяться три операції: вставки нового запису, видалення запису (записів) та коригування запису (записів), яким відповідають оператори INSERT, DELETE та UPDATA, відповідно.

Всі оператори маніпулювання даними дозволяють змінювати дані тільки однієї таблиці.

Оператор введення даних має наступний синтаксис:

*INSERT INTO ім'я_таблиці [(<список стовпців>)] VALUES (<список значень>)*

Подібний синтаксис дозволяє ввести значення тільки одного запису. В тому випадку, коли список стовпців не задано введення відбувається у всі стовпці таблиці. Якщо у переліку стовпців вказано не всі стовпці таблиці, то значення не вказаних стовпців автоматично встановлюється в NULL.

В набір даних також можуть бути введені функції та вирази. Єдиним обмеженням на ці функції та вирази є те, що вони повинні бути визначені на момент введення даних. Наприклад, до бібліотечного формуляра, заданого таблицею *Exemplar (INV, ISBN, YES_NO, NUM_READER, DATE_IN, DATE_OUT)* маємо внести дані про інвентарний номер екземпляру книжки, бібліотечний шифр, відмітку про наявність на полиці, номер читача, дату видачі та дату повернення книги. Причому для введення поточної дати використаємо функцію Get Date(), яка автоматично зчитує системну дату, а для введення дати повернення функцію DateADD (d. GetDate(),14), яка з поточної дати виділяє день та додає до виділеного значення 14 днів і таким чином формує дату повернення книги. Запишемо команду для книги з інвентарним номером 1872 та відповідними значеннями інших даних:

*INSERT INTO Exemplar (INV, ISBN, YES_NO, NUM_READER, DATE_IN, DATE_OUT)*

*VALUES ( 1872, '5-88782-290-2', NO, 344, GetDate(),DateADD (d. GetDate(),14))*

Оператор введення даних дозволяє ввести одночасно декілька рядків таблиці, якщо їх можна вибрати з іншої таблиці. Для цього слід використати підзапит, заданий оператором SELECT.

Наприклад, можна внести всіх студентів, як читачів бібліотеки, до таблиці READER одним оператором, зробивши вибірку даних про них з таблиці STUDENT:

*INSERT INTO READER (NAME_READER, ADRESS, HOOM_PHONE, BIRTH_DAY)*

*SELECT (NAME_STUDENT, ADRESS, HOOM_PHONE, BIRTH_DAY)*

*FROM STUDENT*

Оператор видалення даних дозволяє знищити один або декілька рядків таблиці у відповідності з умовами видалення. Синтаксис оператора DELETE наступний:

*DELETE FROM ім'я_таблиці [WHERE <умови_відбору>]*

Якщо умови відбору не задано, то з таблиці видаляються всі рядки, хоча сама таблиця залишається. Наприклад, для очищення таблиці продажу продукції за поточний місяць запишемо:

*DELETE FROM Продаж_поточний.*

Для видалення всіх замовлень від магазину "Фуршет" з таблиці ЗАМОВЛЕННЯ, яку ми розглядали в розділі 6.8, маємо написати вираз:

*DELETE FROM ЗАМОВЛЕННЯ WERE Замовник =''Фуршет''*

В частині *WERE* може знаходитись вбудований запит. Наприклад , з бази даних заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, ціна, сума) видалити замовників, що замовляли продукцію не пізніше 31.12.2005 р. Текст оператора має наступний вигляд:

*DELETE FROM ЗАМОВНИК*

*WERE код замовника IN (SELECT DISTINCT код замовника*

*FROM ЗАМОВНИК RIGHT JOIN ЗАМОВЛЕННЯ ON*

*ЗАМОВНИК. Код замовника = ЗАМОВЛЕННЯ.код замовника*

*WHERE Замовлення.дата < 31.12.2005)*

В наведеному прикладі умова видалення замовника пов'язана з терміном замовлення, тому для видалення використовується вкладений запит, який дозволяє вибрати замовлення, що надійшли до початку 2006 року і,за рахунок зв'язку між таблицями, визначити код замовника для вилучення. Оскільки замовлень у одного замовника може бути декілька, тобто, код замовника в результаті виконання підзапиту може повторюватись декілька разів, використовуємо слово DISTINCT, щоб уникнути повторень. Для відображення зв'язку між таблицями слугують слова RIGHT JOIN. Це вказує на те, що для кожного замовлення слід знайти відповідний запис у таблиці ЗАМОВНИК.

Для уникнення втрат даних на операцію DELETE можуть накладатися деякі обмеження. Взагалі, всі операції маніпулювання даними проводяться у відповідності з правилами збереження цілісності даних, але особливу увагу приділяють операціям вилучення даних та модифікації. В наведеному вище прикладі ми вилучили замовників з таблиці ЗАМОВНИК не зважаючи на те, що відповідні замовлення в таблиці ЗАМОВЛЕННЯ залишились. Для реальної бази даних така операція не припустима, тому що порушує цілісність даних,

тобто логічну узгодженість даних. В такому випадку, в залежності від правила збереження цілісності, система запропонує спочатку вилучити дані з дочірньої таблиці, або автоматично вилучить ЗАМОВЛЕННЯ сама. Про засоби підтримки цілісності даних і правила збереження цілісності даних було викладено в розділі 8. Слід пам'ятати, що не всі операції можуть бути виконані системою, хоча вираз написано правильно.

Операція оновлення використовується тоді коли маємо відкоригувати дані, що зберігаються в таблиці. Операція оновлення має наступний синтаксис:

*UPDATE ім'я_таблиці*
*SET ім'я_стовпця = нове_значення*
*[WHERE умова_відбору ]*

Частина *WHERE* не є обов'язковою, як і в операторі *DELETE.* Вона дозволяє вибрати рядки, яких стосується операція модифікування. Якщо умову відбору не задано, то операцію модифікування буде застосовано до всіх рядків таблиці. В операторі *UPDATE* теж дозволено використання вкладених запитів. Наприклад: В базі даних заданій відношеннями СТУДЕНТ (код_студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код_екзамену, код_студента, дата, дисципліна, оцінка) замінити оцінку з дисципліни "Моделювання систем" студенту Бойко на "4".

Оператор оновлення, що відповідає умові задачі, має вигляд:

*UPDATE ЗВЕДЕНА ВІДОМІСТЬ*
*SET Оцінка= "4"*
*WHERE дисципліна = "Моделювання систем" AND код_студента IN (SELECT код_студента FROM ЗВЕДЕНА ВІДОМІСТЬ JOIN СТУДЕНТ ON*
*ЗВЕДЕНА ВІДОМІСТЬ. код_студента = СТУДЕНТ.код_студента*
*WHERE СТУДЕНТ.ПІП = "Бойко")*

### Питання для самоперевірки

1. В чому полягає дія оператора Insert ?
2. В чому полягає дія оператора Update ?
3. В чому полягає дія оператора Delete ?
4. Які з операторів маніпулювання даними можуть використовувати підзапити ?
5. Який предикат використовують для включення підзапиту в умову вибірки ?
6. В якому випадку використовують підзапит у операторі Insert ?

## Лекція 15

### 15. Збережувані процедури та тригери в ERWin. Загальні відомості

Збережувана процедура є фрагментом SQL коду на вибірку або маніпулювання даними, який зберігається разом з базою даних і слугує для реалізації найбільш поширених запитів клієнтів. Розглянемо можливості ERWin в частині побудови збережуваних процедур і особливо їх різновиду - тригерів.

Виходячи з синтаксису, можна сказати що збережуваною процедурою називається набір попередньо відкомпільованих команд SQL, який розташовано на сервері бази даних. Збережувана процедура може бути запущена з клієнтського додатку або з іншої процедури.

Тригером називається збережувана процедура виконувана автоматично як реакція на подію. В якості події виступає вилучення, заміна або вставка рядка таблиці бази даних. Тобто, тригер повідомляє СКБД які дії слід виконати для обслуговування команд DELETE, UPDATE, INSERT .

В ERWin розглядають три види тригерів: тригери цілісності посилань, тригери зв'язку, тригери для сутності (таблиці).

**Тригери цілісності посилань (RI - тригери).**

Тригер цілісності посилань - особливий вид тригера використовуваний для підтримки цілісності даних у зв'язаних таблицях , яка може порушитись під час змін в одній з них. Якщо рядок в одній зі зв'язаних таблиць вилучено, змінено або вставлено, то RI-тригер повідомляє СКБД, що слід робити з тими рядками (рядком) в іншій таблиці, в яких значення зовнішнього ключа співпадає зі значенням первинного ключа вилученого, зміненого або вставленого рядка.

Приклад: В базі даних "Бібліотека" існують таблиці КНИГА (шифр книги, Назва, Автор, Рік видання, Видавництво, Кількість_примірників, Вид доступу), КНИГА_АБОНЕМЕНТ (Код абонемента, Шифр книги, Примітка) та ЧИТ_ЗАЛ_КНИГА (Код чит_зал, Шифр книги, Примітка). Тут атрибут "Шифр книги" є первинним ключем таблиці КНИГА та зовнішнім ключем таблиць АБОНЕМЕНТ і ЧИТ_ЗАЛ_КНИГА. Таблиці знаходяться в ієрархічній залежності і пов'язані категоріальним зв'язком за категорією Вид доступу. Таблиця КНИГА - батьківська, таблиці АБОНЕМЕНТ і ЧИТ_ЗАЛ_КНИГА - дочірні. При введенні нової книги в таблицю КНИГА бажано, щоб нове значення поля "Шифр книги" автоматично з'являлось також в таблицях АБОНЕМЕНТ і ЧИТ_ЗАЛ_КНИГА. Таку функціональність визначено правилом CHILD INSERT CASCADE і підтримано відповідним RI тригером.

Для генерації тригерів ERWin використовує механізм шаблонів - спеціальних скриптів побудованих з використанням макрокоманд. Під час генерації SQL коду тригера замість макрокоманд мають бути підставлені імена таблиць, колонок, змінні та інші фрагменти коду, що відповідають синтаксису обраної СКБД.

Розглянемо шаблон RI-тригера, що реалізує правило ON PARENT DELETE RESTRICT [2] .

```
/* ERWin Builtin %DateTime */
/*% Parent % Verb Phrase % Child ON PARENT DELETE RESTRICT */
SELECT Count(*) into numrose
FROM % Cild
WHERE
/*%JOIN FK PK (%Child, : %% Old, "=","and");*/
%JOIN FK PK (%Child, : % Old, "=","and");
```

*IF (numrose >0)*
*Then*
*Raise_application_error (*
*20001,*
*'Cannot DELETE % Parent becose %Child exists');*
*END IF;*

При генерації схеми СКБД для Oracl 8 згенерований тригер має вигляд:

*Create trigger ID_CUSTOMER after DELETE on CUSTOMER for eash row*
*-ERWin Builtin Tue Jan 15 15:30:12 2006*
*-DELETE trigger on CUSTOMER*
*declare numrows INTEGER*
*begin*
*/* ERWin Builtin Tue Jan 15 15:30:12 2006*/*
**/CUSTOMER розміщуєORDER ON PARENT DELETE RESTRICT */*
*SELECT Count(*) into numrose*
*FROM ORDER*
*WHERE*
*/*%JOIN FK PK (%Child, : %% Old, "=","and");*/*
*ORDER..CUSTOMERID =:old.CUSTOMERID;*
*IF (numrose >0)*
*Then*
*Raise_application_error (*
*20001,*
*'Cannot DELETE CUSTOMER becoseORDER exists');*
*END IF;*

## Правила перевизначення тригерів.

ERWin використовує для генерації коду тригера на мові SQL , по замовчуванню, вбудовані шаблони тригерів цілісності посилань, які автоматично присвоюються кожному зв'язку. Шаблони тригерів цілісності посилань генеровані ERWin за замовчуванням можна змінювати.

Перевизначення типу цілісності посилань.

Для кожної комбінації правил цілісності посилань ERWin дозволяє створити перевизначений шаблон і використати цей шаблон замість шаблона, що використовується за замовчуванням , для всіх зв'язків діаграми, яким був присвоєний цей тип правила цілісності посилань. Нові шаблони будуть використовувані замість стандартних, якщо при генерації SQL коду включена опція RI Type Override.

Перевизначення шаблона тригера для зв'язку.

Можна перевизначити шаблон, заданий за замовчуванням, для деякого конкретного зв'язку. Для цього використовуємо діалог Relationship Template Editor, в якому можна задати шаблони Relationship Override, що використовуються замість стандартних шаблонів ( а також замість шаблонів RI Type Override, якщо вони є). Шаблони Relationship Override мають

використовуватись замість стандартних шаблонів, якщо при генерації схеми включено опцію Relationship Override.

Перевизначення шаблона тригера для сутності.

ERWin дозволяє створювати власні тригери Entity Override для будь-якої сутності моделі бази даних. Вони використовуються замість стандартних шаблонів, а також замість шаблонів RI Type Override і Relationship Override, якщо при генерації SQL коду включена опція Entity Override .

**Питання для самоперевірки**

1. Що таке збережувана процедура?
2. Чим характеризується тип процедур, який називають тригером?
3. Які види тригерів ви знаєте?
4. Яке призначення RI тригерів?
5. Які правила перевизначення тригерів існують в ERWin?

# Лекція 16

## 16. Створення та редагування тригерів в ERWin.

Для перегляду тригера у контекстному меню кожної таблиці є пункт Trigger. Вибір цього пункту меню дозволяє відкрити вікно SQL коду тригера - Table Trigger Viewer. Дві кнопки Table Trigger таTrigger Template відкривають вікна для редагування тригера таблиці та тригера цілісності посилань відповідно.

Table Trigger Editor дозволяє:

- Вибрати таблицю для редагування.
- Задати команду ( Delete, Insert, Update), на яку має реагувати тригер.
- Визначити до (Befor) чи після (After) виконання команди має запускатися тригер.
- Задати ім’я нового тригера ( якщо ім’я не задане, то використовується тригер цілісності посилань).
- Вибрати колонки таблиці (атрибути), яких стосується дія тригера.
- Зробити посилання на попереднє значення атрибута (опція "old") та на нове (опція "new") при оновленні запису. Цій дії відповідає макрос % Ref Clause в шаблоні тригера.
- Вказати ім’я батьківської таблиці, дочірньої таблиці, ім’я зв’язку (Verb Phrase) та правило цілісності посилань.

Для автоматизації створення нового шаблону тригера використовують режим Trigger Toolbox, запускаючи його з вікна Table Trigger Editor .

У Trigger Toolbox представлено п’ять типів макросів: табличні, зв’язку, атрибутів, обмежень та макроси загального призначення. Тут також наведено опис кожного макросу та приклад його використання.

Для включення макросу до тригера слід вибрати його зі списку макросів і подвійним натискуванням лівої клавіші миші відправити до Triger Editor у поле Template Code. У полі Template Code сформований шаблон тригера можна передивитись на мові SQL обраної СКБД.

Розглянемо створення шаблону тригера для сутності Customer, який відслідковує зміну даних в полі Customer Name і реєструє в таблиці Sequriti дату зміни, старе ім'я, нове ім'я клієнта, та особу, що внесла зміни.

*Create trigger % Trigger Name*
*%Fire %Actions ("or")*
*on %TableName*
*%RefClause*
*%Scope (- "для кожного рядка"* - підтримує тільки СКБД Oracl )
*/*ERWin bultin %DateTime*/*
*/*Default body for %Trigger Name*/*
*begin*
*Insert into Sequrity (OldName, NewName, UserUpdate, UpdateDate)*
*Values (:old1.CustomerName, new1.CustomerName, User, Sysdate);*
*End;*
*/*

Макрос %Fire може приймати одне з двох значень: After або Befor. Макрос % Actions під час генерації коду тригера замінюється на одну з команд: Delete, Insert або Update. Макрос %RefClause генерує вказівку на ім'я змінних, що вміщують старе та нове значення оновленого поля.

В середовищі Oracl 8 тригер має вигляд:

*Create trigger SequrWrite*
*Befor update of*
*CustomerName*
*On Customer*
*REFERENSING OLD AS old1 NEW AS new1*
*For each row*
*/*ERWin Builtin Tue Jan 26 21:30:00 2004*/*
*/*default body for Secur Write*/*
*begin*
*Insert into Securiti (oldName, New Name, UserUpdate, UpdateDate)*
*Values (:old1. CustomerName, :new1. CustomerName, User, Sysdate);*
*End;*
*/*

Trigger Template Editor призначено для перевизначення типу цілісності посилань, тобто для зміни чи доповнення стандартного шаблону. Для зміни шаблону маємо:

- Вибрати змінюваний тип цілісності посилань та натискуванням кнопки Detach відокремити його від стандартного шаблону тригера.
- Задати ім'я нового шаблону тригера у полі Template Name.
- За допомогою Trigger Toolbox створити шаблон нового тригера та натискуванням кнопки Add добавити його у список User Override.
- Ім'я нового шаблону кнопкою Attach під'єднати до коригованого типу цілісності посилань.

**Питання для самоперевірки**

1. Яке призначення тригерів для сутності?
2. Який макрос визначає момент запуску тригера?
3. Який макрос відповідає за подію на яку реагує тригер?
4. Який макрос визначає змінні, в яких зберігаються нове та старе значення скоригованих даних?

**Завдання**

1. Створити, за допомогою макросів ERWin, шаблон тригера для вилучення запису з таблиці БД, змодельованої в ході лабораторних робіт, який записував би час, вміст старого запису і особу, що вилучила запис у окрему таблицю.
2. Створити, за допомогою макросів ERWin, шаблон тригера для оновлення запису з таблиці БД, змодельованої в ході лабораторних робіт, який записував би час, назву оновлених полів і особу, що оновила запис у окрему таблицю.
3. Створити, за допомогою макросів ERWin, шаблон тригера для доповнення запису до таблиці БД, змодельованої в ході лабораторних робіт, який записував би час, назву таблиці і особу, що доповнила БД у окрему таблицю.

## Тест до модуля 3

1. В якій нормальній формі знаходиться сутність, якщо атрибути вміщують тільки атомарні значення, кожний не ключовий атрибут повністю залежить від первинного ключа і ніякий не ключовий атрибут не залежить від іншого не ключового атрибуту:

A) 1NF
B) 2NF
C) 3NF
D) 3 NF- Бойса-Кода
E) 4NF
F) 5NF

2. В якій нормальній формі знаходиться сутність, якщо атрибути вміщують тільки атомарні значення:

A) 1NF
B) 2NF
C) 3NF
D) 3 NF- Бойса-Кода
E) 4NF
F) 5NF

3. В якій нормальній формі знаходиться сутність, якщо атрибути вміщують тільки атомарні значення , кожний не ключовий атрибут повністю залежить від первинного ключа.

A) 1NF
B) 2NF
C) 3NF

D) 3 NF- Бойса-Кода
E) 4NF
F) 5NF

4. Виберіть нормальну форму, приведення до якої усуває транзитивну залежність між атрибутами сутності:

A) 1NF
B) 2NF
C) 3NF
D) 3 NF- Бойса-Кода
E) 4NF
F) 5NF

5. Які дані називаються атомарними:

A) ті, що не повторюються в одному стовпчику;
B) ті, що не повторюються в одному рядку;
C) ті, що відповідають одному факту;
D) ті, що не повторюються в одній комірці;
E) ті, що не повторюються в базі даних;
F) ті, що не повторюються в одній таблиці.

6. Нормалізація структури бази даних проводиться для:

A) Усунення протиріч в базі даних;
B) Спрощення адміністрування БД;
C) Зменшення таблиць БД;
D) Зменшення часу виконання запиту до БД;
E) Зменшення зв'язків між даними;
F) Зменшення обсягу даних в БД.

# ЛІТЕРАТУРА

1. Карпова Т.. Базы данных. Модели, разработка, реализация. СПб.: Питер, 2001. – 304 с.

2. Маклаков С.В. CASE–средства разработки информационных систем. – М. : Диалог – МИФИ, 2001 – 256 с.

3. Мартин Дж. Организация баз данных в вычислительных системах. М. Мир, 1978 – 615 с.

4. Мякшило О.М, Загоровська Л.Г. Моделювання систем: Лабораторний практикум. (частина 2) для студентів спец. 6.080400 "Інформаційні управляючі системи і технології" ден. форми навчання ./ Уклад.:– К. : УДУХТ, 2000 – 20 с.

# Додаток 1

**Контрольна робота до модуля 3**
**Завдання 1**

1. Створити запит для БД, заданої відношенням НАКЛАДНА (код накладної, номер накладної, дата, вантажовідправник) та РЯДОК НАКЛАДНОЇ(код рядка, код накладної, номер рядка, назва товару, ціна за

один., кількість), в якому реалізувати вибірку накладних за поточну дату і підрахувати виручку за проданий товар. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

2. Створити запит для БД, заданої відношенням НАКЛАДНА (код накладної, номер накладної, дата, вантажовідправник) та РЯДОК НАКЛАДНОЇ(код рядка, код накладної, номер рядка, назва товару, ціна за один., кількість), в якому реалізувати вибірку накладних за назвами товару і підрахувати кількість кожного виду товару проданого за поточну дату. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

3. Створити запит для БД, заданої відношенням НАКЛАДНА (код накладної, номер накладної, дата, вантажовідправник) та РЯДОК НАКЛАДНОЇ (код рядка, код накладної, номер рядка, назва товару, ціна за один., кількість), в якому реалізувати вибірку накладних за назвами товару і підрахувати суму по кожному виду товару проданого за поточну дату. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

4. Створити запит для БД, заданої відношенням ПОСТАЧАЛЬНИК (код постачальника, назва постачальника, адреса, реквізити), реалізувати вибірку постачальників, що знаходяться у київській області. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

5. Створити запит для БД, заданої відношенням ПОСТАЧАЛЬНИК (код постачальника, назва постачальника, адреса, реквізити) та ДОГОВІР (код договору, код постачальника, дата укладання, назва сировини, кількість, сума, термін поставки), реалізувати вибірку постачальників, договори з якими укладено не раніше 2005 року. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

6. Створити запит для БД, заданої відношенням ПОСТАЧАЛЬНИК (код постачальника, назва постачальника, адреса, реквізити) та ДОГОВІР (код договору, код постачальника, дата укладання, назва сировини, кількість, сума, термін поставки), реалізувати вибірку постачальників, договори з якими укладено не раніше 2005 року і на суму більшу за 10000 грн. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

7. Створити запит для БД, заданої відношенням ПОСТАЧАЛЬНИК (код постачальника, назва постачальника, адреса, реквізити) та ДОГОВІР (код договору, код постачальника, дата укладання, назва сировини, кількість, сума, термін поставки), реалізувати вибірку постачальників, договори з якими укладено не раніше 2001 року і не пізніше 2005 року. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

8. Створити запит для БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки), підрахувати суму замовлень по кожному замовнику за 2005 рік. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

9. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), вибрати регіони, в яких сума продажу була максимальною за 4 квартал 2005 року, згрупувати по регіонах. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

10. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), вибрати регіони, в яких сума продажу кожного виду продукції була мінімальною за 4 квартал 2005 року, відсортувати по регіонах. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

11. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), вибрати продукцію яка користувалась найбільшим попитом у 2005 році (у грошовому вимірі), відсортувати по регіонах. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

12. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), вибрати регіони де був реалізований більш ніж один вид продукції у 2005 році, згрупувати по регіонах. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

13. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), підрахувати кількість продаж кожного виду продукції у різних регіонах за 2005 рік (у кількісному вимірі), відсортувати по регіонах. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

14. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), підрахувати кількість продаж по кожному виду продукції у 2005 році. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

15. Створити запит для БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума), підрахувати кількість продаж по кожному виду продукції у 2005 році у Київській області. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

16. Створити запит для БД, заданої відношенням ОБЛАДНАННЯ (код обладнання, назва обладнання) та ЖУРНАЛ РЕМОНТІВ (код ремонту, код обладнання, дата, опис ремонту, виконавець), в якому реалізувати групування ремонтних робіт за назвами обладнання і підрахувати кількість ремонтів кожного виду обладнання за 2005 рік. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

17. Створити запит для БД, заданої відношенням ОБЛАДНАННЯ (код обладнання, назва обладнання) та ЖУРНАЛ РЕМОНТІВ (код ремонту, код обладнання, дата, опис ремонту, виконавець), в якому реалізувати групування ремонтних робіт за виконавцями і підрахувати кількість ремонтів на кожного виконавця за 2005 рік. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

18. Створити запит для БД, заданої відношенням ОБЛАДНАННЯ (код обладнання, назва обладнання) та ЖУРНАЛ РЕМОНТІВ (код ремонту, код обладнання, дата, опис ремонту, виконавець), в якому реалізувати вибірку ремонтних робіт з назвами обладнання і підрахувати кількість ремонтів за 2005 рік. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

19. Створити запит для БД, заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка), в якому реалізувати вибірку студентів, що успішно здали сесію. Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

20. Створити запит для БД, заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка), в якому реалізувати вибірку студентів, що успішно здали сесію (відсортувати по групах). Побудувати схему БД. Записати SQL код запит. Навести приклад результуючої таблиці.

21. Створити запит для БД, заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка), в якому реалізувати вибірку студентів, що успішно здали сесію (відсортувати по дисциплінах). Побудувати схему БД. Записати SQL код запит. Навести приклад результуючої таблиці.

22. Створити запит для БД, заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка), в якому реалізувати вибірку студентів, що мають двійки за сесію (відсортувати по групах). Побудувати схему БД. Записати SQL код запит. Навести приклад результуючої таблиці.

23. Створити запит для БД, заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка), в якому вибрати студентів, що мають більше 2 двійок (відсортувати по групах). Побудувати схему БД. Записати SQL код запита. Навести приклад результуючої таблиці.

## Завдання 2

1. З бази даних заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка) видалити студентів (з відношення СТУДЕНТ), що по трьом дисциплінам мають незадовільні оцінки.

2. З бази даних заданої відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка) видалити студентів (з відношення СТУДЕНТ), що мають середній бал за сесію нижче “4”.

3. У базу даних задану відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка) ввести відмінний результат здачі іспиту з дисципліни “Моделювання систем” студенткою Івановою Іриною. Вважати, що діє правило збереження цілісності посилань Parent Insert Cascade, а код екзамену – поле типа AutoInc.

4. В базі даних заданій відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка) замінити оцінку з дисципліни “Моделювання систем” студенту Бойко на “4”.

5. В БД заданої відношенням Замовник (код замовника, назва замовника, адреса, реквізити) замінити адресу замовника “Фуршет” на наступну: “Київ-145, Богоутівська, 17”.

6. В базу даних задану відношенням Замовник (код замовника, назва замовника, адреса, реквізити) ввести назву замовника: “Фуршет”, адресу :“Київ-145, Богоутівська, 17” та реквізити: “Аваль, МФО 2656002163, ІД 0032210013”.

7. З бази даних заданої відношенням Замовник (код замовника, назва замовника, адреса, реквізити) видалити замовника “Фуршет”

8. З БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки) видалити замовників, що замовляли продукцію не пізніше 31.12.2004 р.

9. З БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки) видалити замовлення, що датовані не пізніше 31.12.2004 р.

10. В БД, задану відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, ціна, термін поставки) вставити замовлення від магазину “Фуршет” на поточну дату, продукції “Ковбаса с/к “Віденська”, в кількості 100 кг., по ціні 50 грн./кг., з терміном поставки 10 днів.

11. В БД, заданій відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, ціна, термін поставки) збільшити термін поставки в замовленні магазину “Фуршет” від 15.10.2005, продукції “Ковбаса с/к “Віденська” на 5 днів.

12. В БД, заданій відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон,

кількість, сума) поміняти дані в полях “кількість” та “сума” для продукції “молоко “Буренка” за 3 квартал 2005 року у м. Києві.

13. В БД, заданій відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума) ввести дані в поля “кількість” та “сума” для продукції “молоко “Ласуня” за 2 квартал 2005 року у м. Києві. Задача вирішується у дві команди.

14. В БД, задану відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума) ввести дані про новий вид продукції “Актимель” за 3 квартал 2005р. в місті Києві, кількість - 20000 пакетів, а сума - 40000 грн. Врахувати, що діє правило збереження цілісності посилань Parent Insert Cascade, а код продажу – поле типа AutoInc.

15. З БД, заданої відношенням ОБЛАДНАННЯ (код обладнання, назва обладнання) та ЖУРНАЛ РЕМОНТІВ (код ремонту, код обладнання, дата, опис ремонту, виконавець) видалити дані про ремонти виконані не пізніше 31.12.2004 року.

16. З БД, заданої відношенням ОБЛАДНАННЯ (код обладнання, назва обладнання) та ЖУРНАЛ РЕМОНТІВ (код ремонту, код обладнання, дата, опис ремонту, виконавець) видалити дані про ремонти обладнання з назвою “Холодильна камера №1”, оскільки її демонтовано. Врахувати, що діє правило обмеження цілісності посилань Parent Delete Cascade.

17. З БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума) вилучити дані про продукцію “молоко “Ласуня” за 2 квартал 2005 року у м. Києві.

18. З БД, заданої відношенням ПРОДУКЦІЯ (код продукції, назва продукції) та ПРОДАЖ (код продажу, код продукції, період, рік, регіон, кількість, сума) вилучити дані про продукцію “молоко “Ласуня”, оскільки дослідження попиту на неї більше проводитись не будуть. Врахувати, що діє правило обмеження цілісності посилань Parent Delete Cascade.

19. З БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки) видалити замовника “Сільпо” та пов’язані з ним замовлення, враховуючи, що діє правило Parent Delete Restrict.

20. З БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки) видалити всі замовлення на продукцію “Ковбаса ліверна” та замовників її, враховуючи, що діє правило Child Delete Cascade та Parent Delete Restrict.

21. З БД, заданої відношенням ЗАМОВНИК (код замовника, назва замовника, адреса, реквізити) та ЗАМОВЛЕННЯ (код замовлення, код замовника, дата замовлення, назва продукції, кількість, сума, термін поставки) видалити всіх замовників на продукцію “Ковбаса ліверна”, якщо вони не замовляли іншої продукції.

22. В базі даних заданій відношенням СТУДЕНТ (код студента, № зал_кн, ПІП, група) та ЗВЕДЕНА ВІДОМІСТЬ (код екзамену, код студента, дата, дисципліна, оцінка) замінити оцінку з дисципліни “Моделювання систем” студенту Бойко на “4” і вказати нову дату здачі іспиту.

# Додаток 2

### Типи даних в SQL

У мові SQL/89 підтримуються наступні типи даних:

CHARACTER(n) або CHAR(n) — символьні рядки постійної довжини у n символів. При завданні даного типу під кожне значення завжди відводиться n символів, і якщо реальне значення займає менше, ніж *n* символів, то СКБД автоматично доповнює недостаючи символи проміжками.

NUMERIC[(*n,m*)] — точні числа, тут *n* — загальна кількість цифр в числі, *m* — кількість цифр ліворуч від десяткової крапки.

DECIMAL[(*n,m*)] —точні числа, тут *n* — загальна кількість цифр в числі, *m* — кількість цифр ліворуч від десяткової крапки.

DEC[(*n,m*)] — теж саме, що і DECIMAL[(n,m)].

INTEGER або INT — цілі числа.

SMALLINT — цілі числа меншого діапазону.

Не зважаючи на те, що в стандарті SQL1 не визначено точно, що розуміють під типом INT і SMALLINT (це віддано на відкуп реалізації), вказано тільки співвідношення між цими типами даних, в більшості реалізацій тип даних INTEGER відповідає цілим числам, що зберігаються у чотирьох байтах, а SMALLINT — відповідає цілим числам, що зберігаються у двох байтах. Вибір одного з цих типів визначається розміром числа.

FLOAT[(n)] — числа більшої точності, які зберігаються у формі з плаваючою крапкою. Тут n — число байтів, зарезервованих під зберігання одного числа. Діапазон чисел визначається конкретною реалізацією.

REAL — дійсний тип чисел, який відповідає числам с плаваючою комою, меншої точності, ніж FLOAT.

DOUBLE PRECISION — специфікує тип даних з визначеною в реалізації точністю більшою, ніж визначена в реалізації точність для REAL.

У стандарті SQL92 додані наступні типи даних:

VARCHAR(n) — рядки символів змінної довжини.

NCHAR(N) — рядки локалізованих символів постійної довжини.

NCHAR VARYING(n) — рядки локалізованих символів змінної довжини.

BIT(n) — рядок бітів постійної довжини.

BIT VARYING(n) — рядок бітів змінної довжини.

DATE (n) — календарна дата.

TIMESTAMP (точність) — дата та час.

INTERVAL — часовий інтервал.

До SQL2 введено ряд стандартних вбудованих функцій:

BIT_LEXGTH (рядок) — кількість бітів у рядку;

CAST (значення AS тип даних) — значення, перетворене у заданий тип даних;

CHAR_LENGTH (рядок) — довжина рядку символів;

CONVERT (рядок USING функція) — рядок, перетворений у відповідності з функцією, що вказана;

CURRENT_DATE — поточна дата;

CURRENT_TIME (точність) — поточний час з вказаною точністю;

CURRENT_TIMESTAMP (точність) — поточна дата та час з вказаною точністю;

LOWER (рядок) — рядок, перетворений до нижнього регістру;

OCTED_LENGTH (рядок) — число байтів у рядку символів;

POSITION (перший рядок IN другий рядок) — позиція, з якої починається входження першого рядка до другого;

SUBSTRING (рядок FROM n FOR довжина) — частина рядка, що починається з n-го символу і який має вказану довжину;

TRANSLATE (рядок USING функція) — рядок, що перетворений зі застосуванням вказаної функції;

TRIM (BOTH символ FROM рядок) — рядок, у якому видалені всі перші та останні символи;

TRIM (LEADING символ FROM рядок ) — рядок, у якому видалені всі перші вказані символи;

TRIM (TRAILING символ FROM рядок) — рядок, у якому видалені останні вказані символи;

UPPER (рядок) — рядок, перетворений до верхнього регістру.

# Додаток 3

**Приклад побудови логічної моделі бази даних в середовищі ERWin**

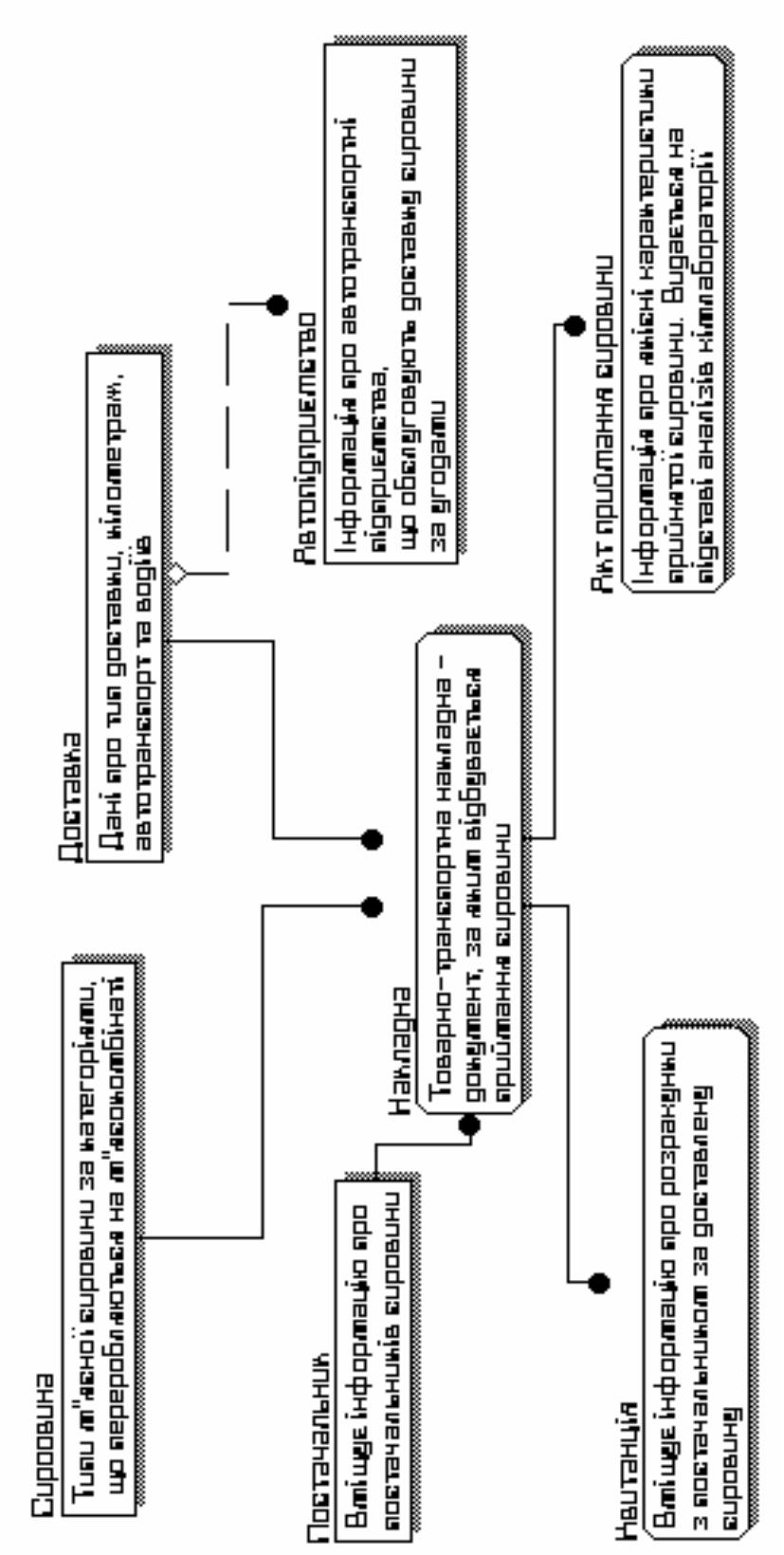



Рис.1 Діаграма об’єкт-зв’язок бази даних відділу постачання на м’ясокомбінаті (Definitions)

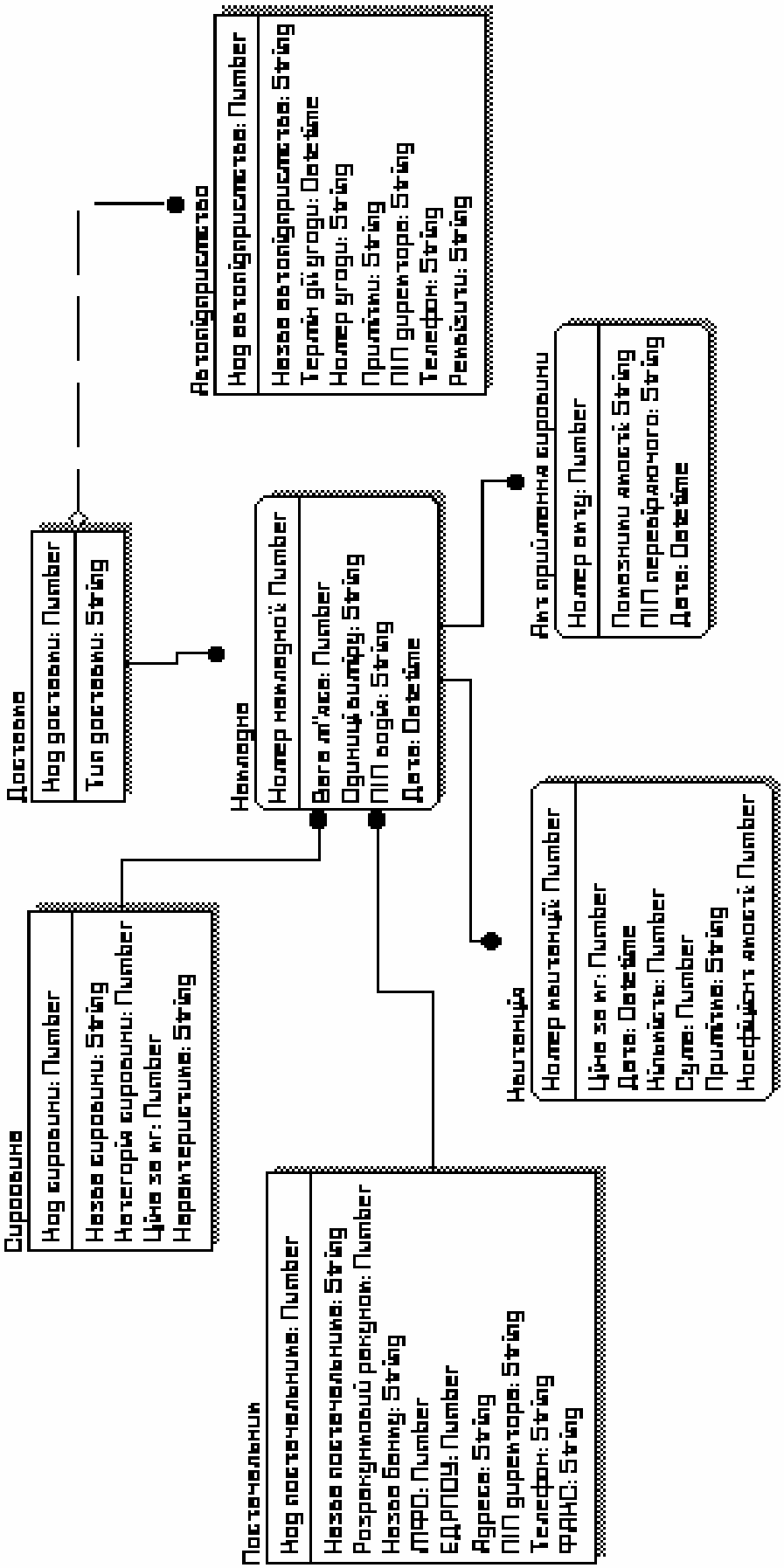



Рис. 2 Повна атрибутивна модель бази даних відділу постачання м’ясокомбінату

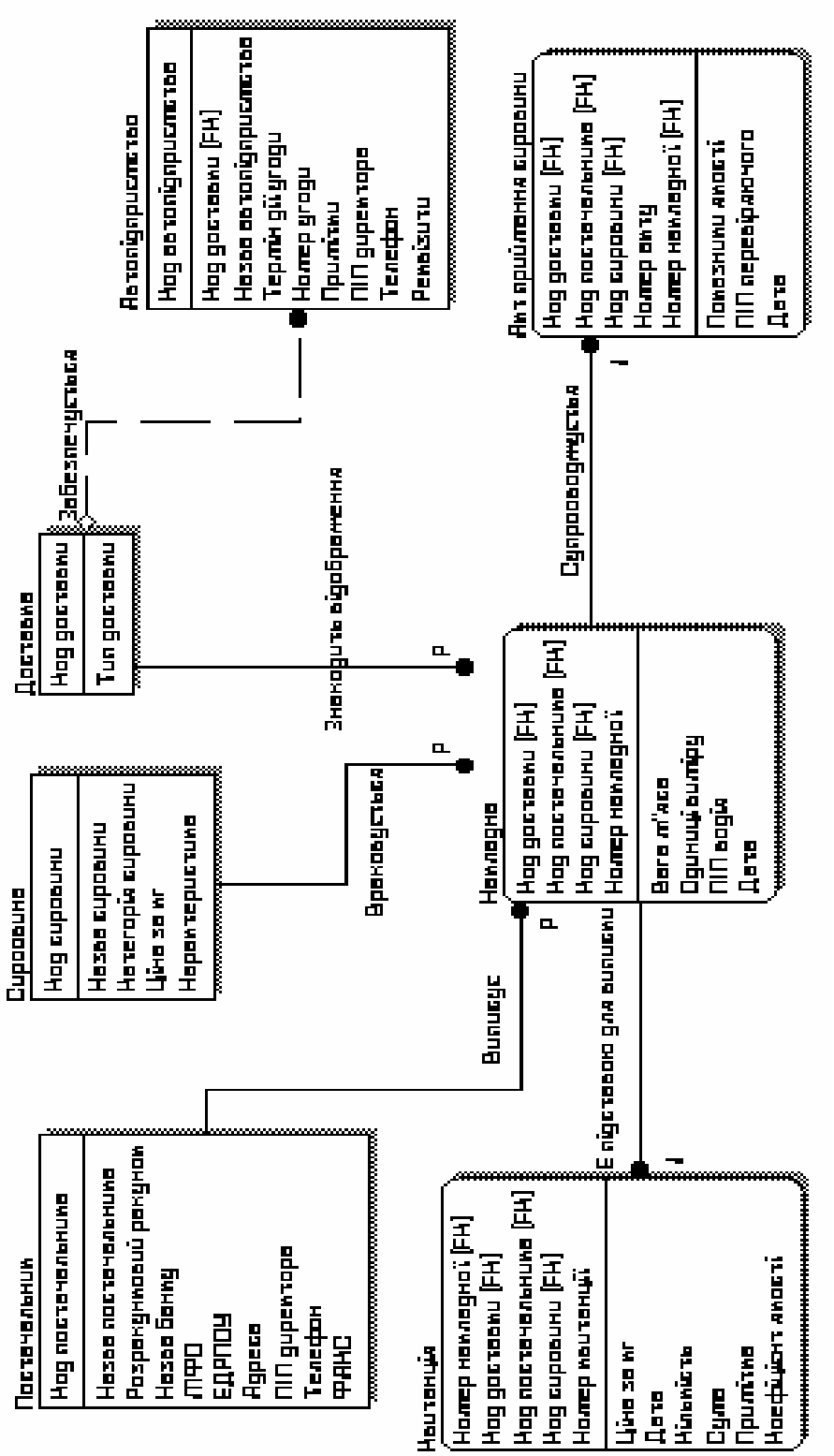



Рис.3 Атрибутивна модель бази даних відділу постачання м’ясокомбінату з поіменованими зв’язками та їх потужністю

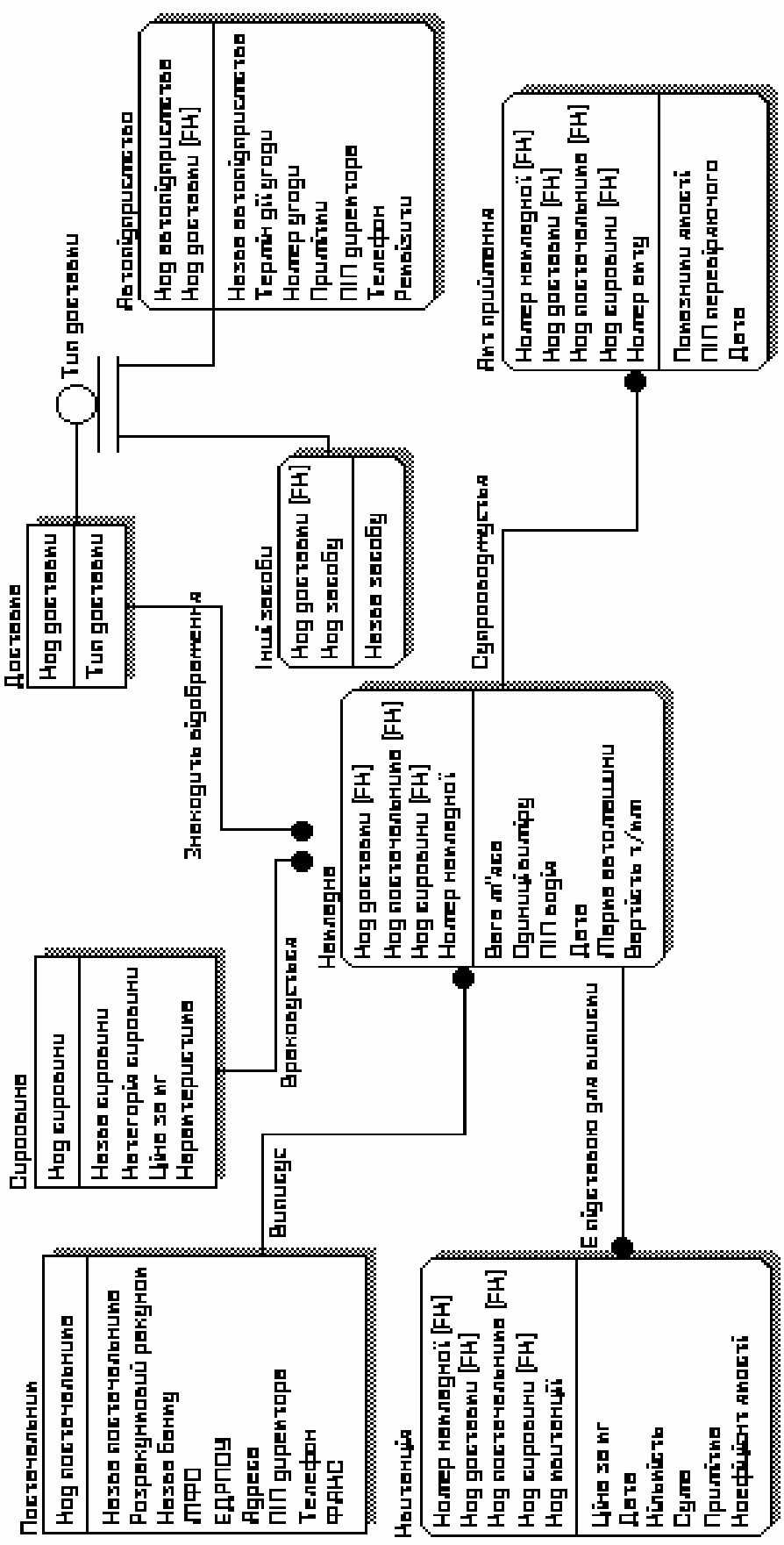



Рис .4. Атрибутивна модель відділу постачання м’ясокомбінату з категоріальним зв’язком

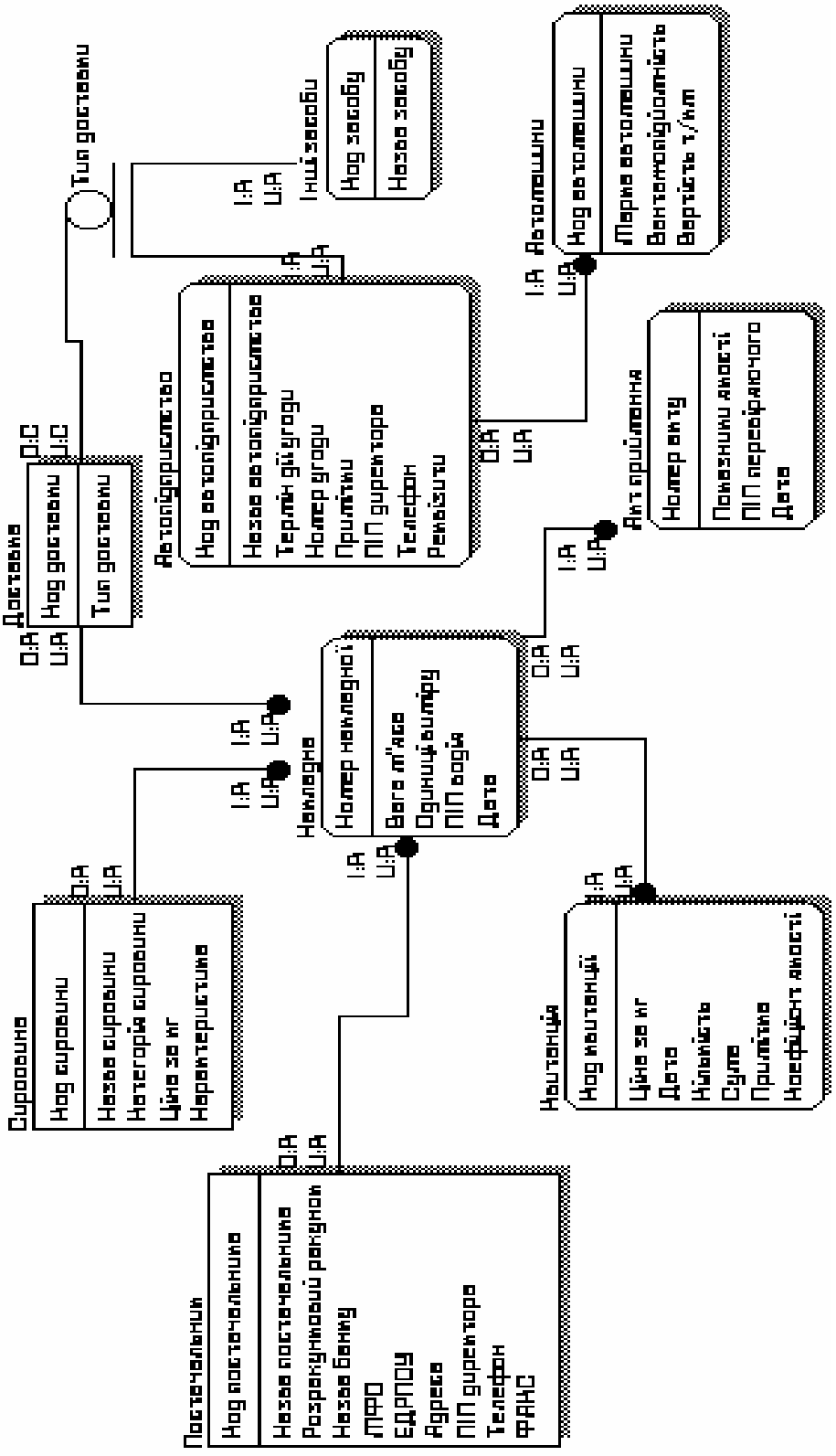



Рис.5. Атрибутивна модель бази даних відділу постачання м’ясокомбінату з правилами збереження цілісності посилань

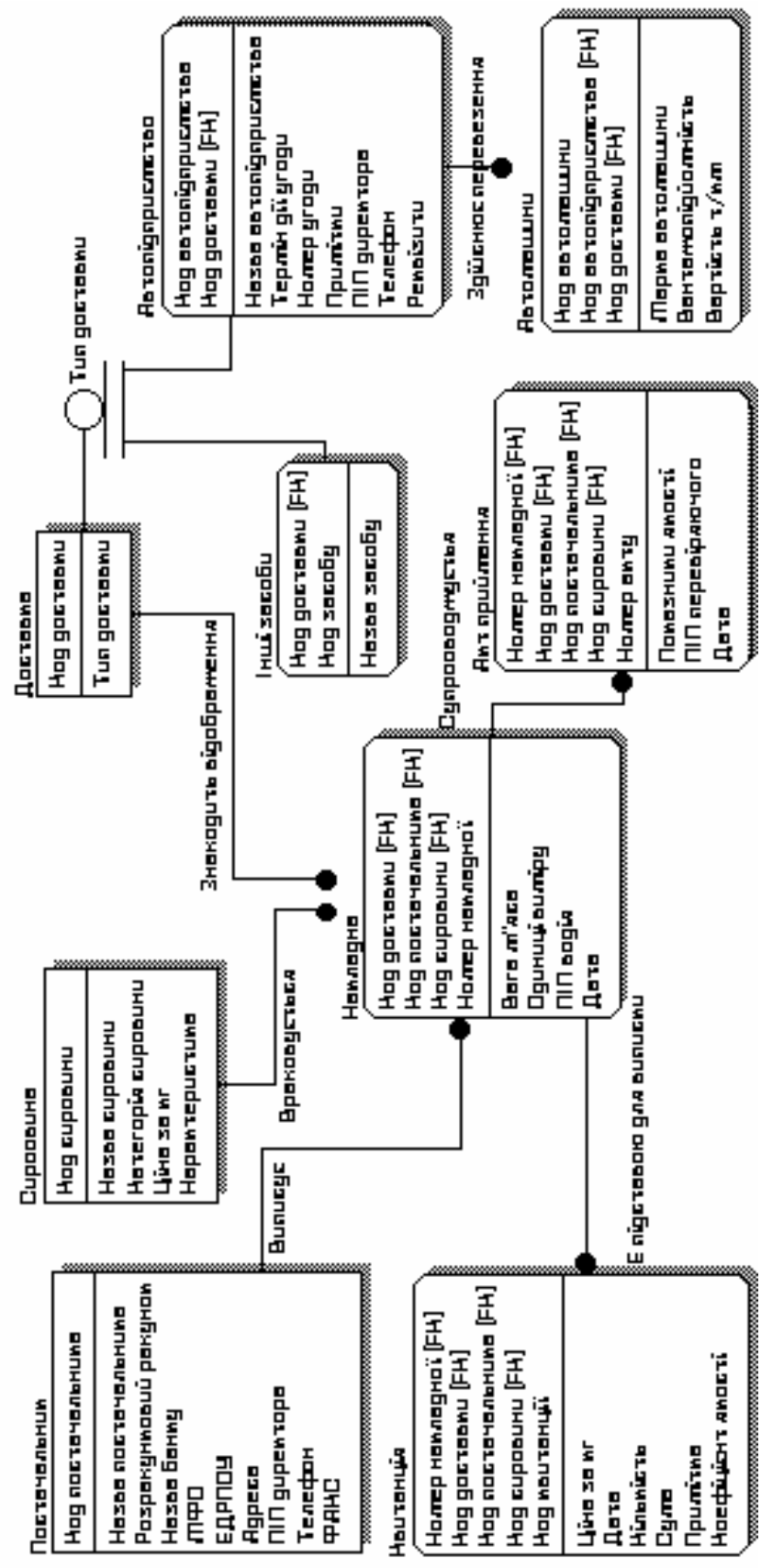



Рис.6 Нормалізована атрибутивна модель бази даних відділу постачання м’ясокомбінату

Навчальне видання

**О.М. М'ЯКШИЛО**

# МОДЕЛЮВАННЯ БАЗ ДАНИХ ЗАСОБАМИ CASE - ТЕХНОЛОГІЇ ERWIN

## КОНСПЕКТ ЛЕКЦІЙ

**з дисципліни "Структурне моделювання систем"**
для студентів спеціальності 6.080400 «Інформаційні управляючі системи та технології»
напряму 0804 "Комп'ютерні науки"
всіх форм навчання

**Видання подається в авторській редакції**

Підп. до друку 04.12.08. Ум. друк. арк. 3,81. Наклад 150 пр.
Зам. № 047-08А

РВЦ НУХТ. 01033 Київ-33, вул. Володимирська, 68
www.book.nuht.edu.ua
Свідоцтво про реєстрацію серія ДК № 1786 від 18.05.04 р.