МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Г. Ю. Мартиненко

КОНЦЕПТУАЛЬНЕ ТА ЛОГІЧНЕ ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

Навчально-методичний посібник
для виконання завдань з дисципліни «Організація баз даних»
для студентів спеціальностей 122 «Комп'ютерні науки»,
113 «Прикладна математика»

Рекомендовано
редакційно-видавничою
радою університету,
протокол № 1 від 16.02.2023 р.

Харків
НТУ «ХПІ»
2023

УДК 004.65
М 29

Рецензенти:
Г. І. Львов, д-р. техн. наук, професор,
Національний технічний університет «Харківський політехнічний інститут»;
О. В. Каратанов, канд. техн. наук, доцент,
Національний аерокосмічний університет ім. М. С. Жуковського
«Харківський авіаційний інститут»

Мартиненко Г. Ю.
М 29 Концептуальне та логічне проектування реляційних баз даних [Електронний ресурс] : навч.-метод. посіб. / Мартиненко Г. Ю. – Харків: НТУ «ХПІ», 2023. – 91 с.

У посібнику викладено основні теоретичні положення та практичні підходи до проектування реляційних баз даних. Наведено докладні відомості щодо призначення і використання технології моделювання даних у процесі проектування реляційної бази даних та розглянуто три основні етапи проектування баз даних – концептуальне, логічне та фізичне проектування. Основну увагу приділено методології логічного проектування бази даних, тобто створенню логічної моделі, за допомогою методів нормалізації відношень та побудови діаграм «сутність-зв'язок». Розглянуто декілька практичних прикладів створення логічних моделей баз даних в різних предметних областях. Наведено завдання за варіантами для самостійного виконання студентами.

Призначено для студентів спеціальностей 122 «Комп'ютерні науки» та 113 «Прикладна математика».

Іл. 10. Табл. 57. Бібліогр. 15 назв.

УДК 004.65

ЗМІСТ

Список умовних позначень і скорочень

БД – база даних;
ПЕОМ – персональна електронна обчислювальна машина;
СУБД – система управління базами даних, теж саме що і система керування базами даних (СКБД);
МД – модель даних;
УК – уявлення користувача;
1НФ – перша нормальна форма;
2НФ – друга нормальна форма;
3НФ – третя нормальна форма;
НФБК – нормальна форма Бойса-Кодда;
4НФ – четверта нормальна форма;
5НФ – п’ята нормальна форма;
6НФ – шоста нормальна форма;
DB – Database;
DBMS – Database Management System;
DDL – Data Definition Language;
DML – Data Manipulation Language;
SQL – Structured Query Language;
OLTP – On-Line Transaction Processing;
OLAP – On-Line Analytical Processing;
ER – Entity-Relationship.

ВСТУП

Попередницями баз даних є файлові системи. Це набір прикладних програм, які виконують для користувачів деякі операції.

Такий підхід до зберігання даних мав певні недоліки, оскільки кожна створена програма зберігає свої власні дані та керує ними. При цьому можна відзначити основні обмеження файлових систем:

1. Розділення та ізоляція даних;
2. Дублювання даних;
3. Залежність від даних;
4. Несумісність файлів;
5. Фіксовані запити.

Виходячи з цього виник новий підхід, який полягає у застосуванні для накопичення і використання інформації *бази даних* (БД) або *Database* (DB), що формується за допомогою спеціальних узагальнених, математично обґрунтованих підходів та правил. Для фізичної реалізації такої бази використовуються спеціальні застосунки, які мають назву *системи управління базами даних* (СУБД), або, як варіант, системи керування базами даних (СКБД), або *Database Management System* (DBMS).

База даних – це спільно використовуваний набір логічно пов'язаних даних (і опис цих даних), призначений для задоволення інформаційних потреб. База даних є єдиним, великим сховищем даних, яке одноразово визначається, а потім використовується одночасно багатьма користувачами.

Базу даних також називають набором інтегрованих записів з самоописом. У сукупності опис даних називається *системним каталогом* (System Catalog), або *словником даних* (Data Dictionary), а самі елементи опису називають *метаданими* (Meta-Data), тобто «даними про дані». Наявність самоопису даних в БД забезпечує в ній незалежність програм від даних (Program-Data Independence).

Поняття «логічно пов'язаний» означає, що при аналізі інформаційних потреб виділяють сутності, атрибути та зв'язки.

Сутністю (Entity) називається окремий тип об'єкта (людина, місце або річ, поняття або подія), який потрібно представити в базі даних.

Атрибутом (Attribute) називається властивість, яка описує деяку характеристику даного об'єкту.

Зв'язок (Relationship) – це те, що об'єднує кілька сутностей.

Система управління базами даних – це програмне забезпечення, за допомогою якого користувачі можуть визначати, створювати, підтримувати БД і здійснювати контрольований доступ до неї.

В загальному випадку СУБД має наступні можливості:

1. Створення БД за допомогою мови визначення даних (DDL – Data Definition Language);

2. Вставка, оновлення, видалення та добування інформації з БД за допомогою мови маніпулювання даними (DML – Data Manipulation Language), з використанням, наприклад, мови структурованих запитів (SQL – Structured Query Language);

3. Надання контрольованого доступу до БД.

Середовище СУБД має такі основні компоненти:

1. Апаратне забезпечення (від одного персонального комп'ютера до мережі з багатьох комп'ютерів);

2. Програмне забезпечення (тобто програмне забезпечення самої СУБД і прикладних програм на різних мовах, наприклад, C, C ++, Java, SQL тощо, разом з операційною системою, включаючи і мережеве програмне забезпечення);

3. Дані (БД містить як робочі дані, так і метадані, тобто «дані про дані»);

4. Процедури (інструкції та правила, які повинні враховуватися при проектуванні та використанні бази даних);

5. Користувачі:

5.1. Адміністратори даних (управління даними, тобто планування БД, розробка і супровід стандартів, прикладних алгоритмів і процедур, концептуальне і логічне проектування БД) та адміністратори БД (фізична реалізація БД, включаючи фізичне проектування і втілення проекту);

5.2. Розробники логічної БД (управління даними, тобто планування БД, розробка і супровід стандартів, прикладних алгоритмів і процедур, концептуальне і логічне проектування БД) та розробники фізичної БД (фізична реалізація готової логічної моделі)

5.3. Прикладні програмісти (розробка застосунків, що надають необхідні функціональні можливості);

5.4. Кінцеві користувачі (клієнти БД).

В цьому посібнику увагу зосереджено саме на роботі розробників логічної бази даних, тобто на концептуальному та логічному проектуванні БД.

1. РІВНІ ОПИСУ ЕЛЕМЕНТІВ ДАНИХ ТА МОДЕЛІ ДАНИХ

1.1. Формалізація опису елементів даних

Процес створення робочої бази даних можна розділити на декілька рівнів. Вони являють собою рівні опису елементів даних. Наочно послідовність цих рівнів можна представити у вигляді схеми, яка зображена на рис. 1.1.

Рисунок 1.1 – Схематичне зображення рівнів опису елементів даних

Таким чином під час проектування бази даних використовуються декілька рівнів опису елементів даних [1].

Зовнішній рівень – це зображення бази даних з точки зору користувачів. Він описує ту частину бази даних, яка відноситься до кожного користувача і складається з декількох різних зовнішніх уявлень бази даних.

Концептуальний рівень – це узагальнююче подання бази даних. Він описує те, які дані зберігаються в базі даних, а також зв'язки між ними. Цей рівень містить наступну структуру:

- всі сутності, їх атрибути та зв'язки;
- обмеження, що накладаються на дані;
- семантична інформація про дані;
- інформація про безпеку і цілісність даних.

Результатом є логічна модель, що спроектована за допомогою певних правил, сукупність яких утворює деякий підхід до опису даних.

Внутрішній рівень являє собою фізичне подання бази даних як інформаційної системи в персональному комп'ютері з використанням СУБД.

Фізичний рівень, тобто робота створеної бази даних, контролюється операційною системою, але під керуванням СУБД. Фізичний рівень доступу до даних нижче системи управління базами даних. Він складається тільки з відомих операційній системі елементів (наприклад, покажчиків на те, як реалізовано послідовний розподіл і чи поля внутрішніх записів на диску зберігаються у вигляді безперервної послідовності байтів).

1.2. Моделі даних як правила опису даних

Для організації бази даних на всіх зазначених рівнях опису може використовуватись одна з існуючих моделей даних.

Модель даних – це інтегрований набір понять для опису та обробки даних, зв'язків між ними та обмежень, що накладаються на дані. Модель є зображенням «реального світу» об'єктів і подій, а також існуючих між ними зв'язків. Це деяка абстракція, у якій акцент робиться на найважливіших і невід'ємних аспектах предметної області, а всі другорядні властивості ігноруються.

Мета побудови моделі даних полягає в поданні даних в зрозумілому вигляді. Існують три пов'язані між собою моделі даних:

- *Зовнішня модель*, яка являє собою предметну область та її модель.
- *Концептуальна модель* – це логічне представлення моделі предметної області.
- *Внутрішня модель* – це фізичний опис даних з використанням СУБД.

Для абстрактного зображення реального світу можуть використовуватись різні моделі даних, які можна розділити на умовні категорії:

1. Об'єктні моделі даних (Object-Based Data Models):
   1.1. Модель типу «сутність-зв'язок» або ER-модель (Entity-Relationship Model),
   1.2. Семантична модель,
   1.3. Функціональна модель,
   1.4. Об'єктно-орієнтована модель;
2. Моделі даних на основі записів (Record-Based Data Models):
   2.1. Реляційна модель даних (Relational Data Model),
   2.2. Мережева модель даних (Network Data Model),
   2.3. Ієрархічна модель даних (Hierarchical Data Model);

3. Фізичні моделі даних (Physical Data Models):
   3.1. Узагальнююча модель (Unifying Model),
   3.2. Модель пам'яті кадрів (Frame Memory Model).

Фізичні моделі даних описують те, як дані зберігаються в комп'ютері, представляючи інформацію про структуру записів, їх впорядкованість та існуючі шляхи доступу. Ця категорія моделей даних використовується для опису даних на внутрішньому рівні, тоді як перші дві категорії використовуються для опису даних на концептуальному та зовнішньому рівнях. Серед них найбільш поширеними є реляційна модель та різні варіанти ER-діаграм.

Реляційна модель даних – це найбільш застосовна логічна модель даних, яка була запропонована у 1970 році Едгаром Франком Коддом (E.F. Codd) [2]. На сьогодні ця модель є фактичним стандартом, на який орієнтуються практично всі сучасні комерційні СУБД, що класифікуються як реляційні. У цій моделі є найбільш високий рівень абстракції даних, ніж в ієрархічній або мережевій, оскільки в ній подання даних не залежить від способу їх фізичної організації. Це є наслідком використання математичного поняття відношення (Relation).

Основні поняття реляційної моделі: 1. Відношення (аналог – таблиця); 2. Атрибут (аналог – стовпець таблиці); 3. Кортеж (аналог – рядок).

У реляційній моделі всі дані логічно структуровані всередині відношень (таблиць). Кожне відношення має ім'я і складається з іменованих атрибутів (стовпців) даних. Кожен кортеж (рядок) даних містить по одному значенню кожного з атрибутів. Реляційна модель заснована на математичному понятті відношення, елементах теорії множин і реляційній алгебрі.

Модель «сутність-зв'язок» (ER-модель) дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій-блоків. ER-модель – це мета-модель даних, тобто засіб опису моделей даних. З цієї моделі можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна) і тому вона є найзагальнішою, хоча з початку розвитку баз даних структурно її прийнято відносити до об'єктних моделей даних.

Виходячи з цього далі буде розглядатися методика використання саме реляційної моделі даних для створення баз даних в різних предметних областях, із застосуванням для побудови логічних моделей теорії нормалізації відношень, запропонованої Крістофером Дейтом (Christopher J. Date) [3, 4], та ER-моделі, у якій у разі реляційної бази даних кожен рядок кожної таблиці (як зображення відношення) являє собою один екземпляр сутності.

1.3. Життєвий цикл бази даних (БД)

Існує декілька традиційних етапів життєвого циклу бази даних:

1. Планування розробки БД;
2. Визначення вимог до системи;
3. Збір та аналіз вимог користувачів;
4. Проектування БД;
5. Вибір цільової СУБД;
6. Розробка застосунків;
7. Реалізація;
8. Перетворення і завантаження даних;
9. Тестування;
10. Експлуатація та супровід.

Перші чотири етапи виконуються без використання цільової СУБД. Для них важливим є тільки її тип, який пов'язаний з моделлю даних на якій базується ця СУБД. Результатом виконання цих етапів є логічна (або даталогічна) модель для обраної предметної області з визначеною моделлю цієї предметної області, яка базується на уявленнях майбутніх користувачів. Етапи з п'ятого по десятий виконуються вже з використанням обраної СУБД для реалізації розробленої логічної моделі. Результатом виконання цих етапів є працездатна наповнена інформацією база даних, яка прийнята в роботу користувачами.

До основних етапів проектування бази даних належить:

1. *Концептуальне проектування* – процес створення моделі інформації, незалежної від будь-яких фізичних аспектів її подання.
   Цей етап абсолютно не залежить від конкретної цільової СУБД, застосунків, мов програмування або будь-яких інших фізичних характеристик. Мета цього етапу – визначення сутностей, атрибутів, зв'язків та обмежень.
2. *Логічне проектування* – процес створення моделі інформації на основі обраної моделі організації даних, але без врахування типу цільової СУБД і фізичних аспектів реалізації.
   Мета цього етапу полягає у створенні логічної моделі даних, яка враховує особливості обраної моделі організації даних в цільовій СУБД (наприклад, реляційній моделі).
3. *Фізичне проектування* – фізична реалізація бази даних, яка виконується з використанням конкретної СУБД.

При цьому може використовуватись один з двох основних підходів до проектування баз даних [5-8]:

1. *Висхідний*, тобто від властивостей сутностей та зв'язків до відношень (прийнятний для проектування простих баз даних з невеликою кількістю атрибутів з використанням процедури нормалізації);
2. *Низхідний*, тобто від моделей з високорівневими сутностями та зв'язками до подальшого низхідного уточнення низькорівневих сутностей, зв'язків та атрибутів (підходить для проектування складних БД і реалізується з використанням концепції «сутність-зв'язок»).

Таким чином, підсумовуючи вищевикладене можна зазначити, що проектування баз даних складається з трьох стадій: концептуальної, логічної та фізичної [1, 4, 5].

Перша стадія передбачає створення концептуальної моделі даних, яка не залежить від будь-яких фізичних характеристик.

На другій стадії, призначення якої полягає у створенні логічної моделі даних, концептуальна модель піддається доопрацюванню шляхом видалення елементів, які не можуть бути реалізовані в реляційних системах (якщо використовується реляційна модель даних).

На третій стадії логічна модель даних перетворюється на фізичний проект, призначений для реалізації в середовищі конкретної цільової СУБД [9, 10]. При цьому аналізуються структури зберігання даних та методи доступу, необхідні для ефективної роботи з базою даних, розміщеною на зовнішніх пристроях запам'ятовувачах.

Предметом даного навчально-методичного посібника є виконання перших двох стадій проектування бази даних з використанням реляційної моделі даних за допомогою розгляду типових прикладів паралельно з теоретичними положеннями, що стосуються самої реляційної моделі даних, необхідних для її створення елементів реляційної алгебри та реляційного числення, а також двох методів побудови реляційної логічної моделі даних для визначеної предметної області, а саме – методу нормалізації форм відношень та одного з варіантів діаграм «сутність-зв'язок».

Третя стадія проектування БД із використанням систем управління базами даних докладно розглянута в [9, 10]. Вона може виконуватись з використанням будь-якої реляційної СУБД, наприклад, Microsoft Access чи Microsoft SQL Server, MySQL, PostgreSQL та багатьох інших [9-14].

2. РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ

2.1. Загальні положення

Реляційна модель даних описує, які дані можуть зберігатися в реляційних базах даних, а також способи маніпулювання такими даними. У спрощеному вигляді основна ідея реляційної моделі полягає в тому, що дані повинні зберігатися в таблицях, і тільки в таблицях. Класична реляційна модель даних вимагає, щоб дані зберігалися в так званих плоских таблицях. *Плоска таблиця* – це таблиця, кожна комірка якої може бути однозначно ідентифікована зазначенням рядка та стовпця таблиці, а в одному стовпці всі комірки повинні містити дані одного простого типу.

На найнижчому рівні опису (теорії множин) використовуються поняття «множина», «підмножина декартова добутку», «кортеж». На наступному рівні (реляційної моделі) їм відповідають «домен», «відношення», «кортеж». На самому верхньому рівні (стандарту мови структурованих запитів SQL – Structured Query Language та її конкретних реалізацій) використовуються відповідні їм терміни «тип даних», «таблиця», «рядок таблиці».

2.2. Елементи теорії множин

Множина – це сукупність елементів, що мають деяку загальну властивість [15]. Для того, щоб деяку сукупність елементів можна було назвати множиною, необхідно, щоб виконувалися дві умови:

1. Має існувати правило, що дозволяє визначити, чи належить вказаний елемент даній сукупності.
2. Має існувати правило, що дозволяє відрізняти елементи один від одного (це означає, що *множина не може містити двох однакових елементів*).

Друге правило є найголовнішим для формування реляційних баз даних, бо воно накладає певні обмеження на формат збережуваних даних.

Множини зазвичай позначаються великими літерами латинського алфавіту. Для них існує декілька основних позначень:

1. Елемент x *належить* множині A: $x \in A$.
2. Множина B є *підмножиною* множини A: $B \subset A$.
3. Підмножина B множини A називається *власною підмножиною*, якщо $B \neq A$.

Основними операціями над множинами є:

1. *Об'єднання* множин: $A \cup B = \{x \mid x \in A$ або $x \in B\}$;
2. *Перетин* множин: $A \cap B = \{x \mid x \in A$ та $x \in B\}$;
3. *Різниця* множин: $A \setminus B = \{x \mid x \in A$ та $x \notin B\}$;
4. *Доповнення* множини A: $\bar{A} = \Omega \setminus A$;
5. *Декартів (прямий) добуток* множин $A_1, A_2, ..., A_n$ – це множина упорядкованих n-ок $A_1 \times A_2 \times ... \times A_n = \{(a_1, a_2, ..., a_n) \mid a_i \in A_i\}$, при цьому:
   5.1. Упорядкована n-ка, набір або кортеж – це $(a_1, a_2, ..., a_n)$, де $a_1 \in A_1$, ..., $a_n \in A_n$,
   5.2. *Степінь* декартова добутку $A_1 \times A_2 \times ... \times A_n$ – це число множин n,
   5.3. Якщо всі множини A_i однакові, то використовують позначення $A^n = A \times A \times ... \times A$,
   5.4. Декартів добуток множин є одним із способів конструювання нових об'єктів з уже наявних множин, тому що створює всілякі перебори, наприклад, для множин $A=\{1,2,3\}$ та $B=\{4,5,6\}$ декартів добуток $A \times B$ містить всього 9 переборів (3×3), а саме (1, 4),(1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), а не три – (1, 4), (2, 5), (3, 6), як інколи помилково вважається;
6. *Відношення* ступеня n (n-арне відношення) – це підмножина R декартова добутку множин $A_1 \times A_2 \times ... \times A_n$, при цьому:
   6.1. *Потужність* відношення R – це потужність множин кортежів (їх кількість), що входять у відношення R,
   6.2. Всі елементи відношення є однотипні кортежі, наприклад, {(1, "Іванов", 1000), (2, "Петров", 2000), (3, "Сидоров", 3000)} – відношення, {(1), (1, 2), (1, 2, 3)} – множина з різнотипних числових кортежів,
   6.3. За винятком крайнього випадку, коли відношення є сам декартів добуток $A_1 \times A_2 \times ... \times A_n$, відношення включає в себе не всі можливі кортежі з декартова добутку, а це означає, що для кожного відношення є критерій, що дозволяє визначити, які кортежі входять у відношення, а які – ні,
   6.4. Цей критерій, по суті, визначає сенс (*семантику*) відношення, тому кожному відношенню можна поставити у відповідність деякий логічний вираз $P(x_1,x_2,...,x_n)$, що залежить від n параметрів і

визначає, чи буде кортеж (a_1, a_2, ...,a_n) належати відношенню R,

6.5. Цей логічний вираз $P(x_1, x_2,...,x_n)$ називають *предикатом* відношення R – кортеж (a_1, a_2,...,a_n) належить відношенню R тоді і тільки тоді, коли $P(a_1, a_2, ..., a_n)$ = «Істина»;

7. *Транзитивне замикання* відношення R, яке задано на декартовому квадраті A^2 деякої множини A – це нове відношення, що складається з кортежів (x, y), для яких або кортеж (x, y)$\in R$, або знайдеться скінченна послідовність елементів z_1, z_2, ... , $z_n \in A$, що всі кортежі (x, z_1), (z_1, z_2), ... ,(z_n, y) належать відношенню R.

2.3. Складові частини та базові поняття реляційної моделі даних

Реляційна модель складається з трьох частин:

1. *Структурна частина* описує, які об'єкти розглядаються реляційної моделлю. Постулюється, що єдиною структурою даних, використовуваних в реляційній моделі, є *нормалізовані n-арні відношення*.
2. *Цілісна частина* описує обмеження спеціального виду, які повинні виконуватися для будь-яких відношень в будь-яких реляційних базах даних. Це *цілісність сутностей* і *цілісність зовнішніх ключів.*
3. *Маніпуляційна частина* описує два еквівалентних способи маніпулювання реляційними даними – *реляційну алгебру* і *реляційне числення.*

2.3.1. Структурна частина реляційної моделі даних

Реляційна модель вимагає, щоб типи використовуваних даних були простими, наприклад, логічний (L), строковий (C), чисельний (N), цілий, речовинний, грошовий, перелічувальний, інтервальний, дата (D), час (T) тощо. Але можуть використовуватись й інші типи даних, наприклад, структуровані (масиви, записи) або посилальні (вказівники). Важливим для розуміння є те, що для реляційної моделі даних тип даних не важливий, а вимога, щоб він був простим, означає, що в реляційних операціях не повинна враховуватися внутрішня структура даних. В реляційній моделі даних з поняттям тип даних тісно пов'язане поняття домену, яке можна вважати уточненням типу даних.

Домен – це семантичне поняття, яке можна розглядати як підмножину значень деякого типу даних, що мають певний сенс. Якщо тип даних можна вважати великою кількістю всіх можливих значень даного типу, то домен нагадує підмножину в цій множині.

Домен характеризується його властивостями:

- Домен має *унікальне ім'я* (в межах БД).
- Домен визначений на деякому *простому* типі даних або на іншому домені.
- Домен може мати деяку *логічну умову,* що дозволяє описати підмножину даних, допустимих для даного домена.
- Домен несе певне *смислове навантаження.*

Наприклад, домен *D*, сенс – «Вік співробітника»: $D=\{n \in N: n \geq 18 \text{ and } n \leq 60\}$ (певні обмеження, що визначені законодавством).

Відмінність домену від поняття підмножини полягає саме в тому, що *домен відображає семантику,* визначену предметною областю. Може бути кілька доменів, які збігаються як підмножини, але які мають різний сенс.

Основне значення доменів полягає в тому, що *домени обмежують порівняння.* Некоректно порівнювати значення з різних доменів, навіть якщо вони мають однаковий тип (наприклад, ціна і вага). В цьому проявляється смислове обмеження доменів.

Атрибут відношення є пара виду <*Ім'я_атрибуту*: *Ім'я_домену*>. Імена атрибутів повинні бути унікальними в межах відношення. Часто імена атрибутів відношення збігаються з іменами відповідних доменів.

Відношення R, визначене на множині доменів D_1, D_2, ..., D_n (не обов'язково різних), містить дві частини: заголовок і тіло.

Заголовок відношення містить фіксовану кількість атрибутів: ($<A_1:D_1>$, $<A_2:D_2>$, ..., $<A_n:D_n>$). Він статичний і не змінюється під час роботи з БД.

Тіло відношення містить множину кортежів відношення. Воно може змінюватися під час роботи з БД – кортежі можуть змінюватися, додаватися, видалятися.

Кортеж відношення – це множина пар виду <*Ім'я_атрибуту*: *Значення_атрибуту*>: ($<A_1:Val_1>$, $<A_2:Val_2>$, ..., $<A_n:Val_n>$), таких що значення Val_i атрибуту $A_i \in$ домену D_i .

Степенем (або *-арністю) відношення* називають кількість атрибутів у відношенні. *Потужністю відношення* називають потужність множини кортежів відношення (тобто кількість кортежів у відношенні).

Позначення відношення:

$R(<A_1:D_1>, <A_2:D_2>, ..., <A_n:D_n>)$ або $R(A_1,A_2, ...,A_n)$ або *R*.

Наприклад, відношення «Співробітники» задано на доменах:

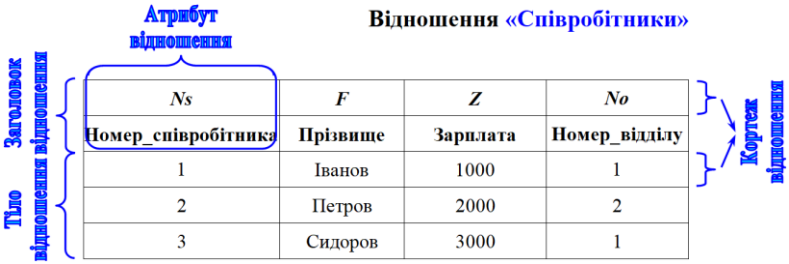
- «Номер_співробітника»,
- «Прізвище»,
- «Зарплата»,
- «Номер_відділу».

Заголовок відношення: Співробітники (Ns: Номер_співробітника, F: Прізвище, Z: Зарплата, No: Номер_відділу).

Поточний стан (3 кортежи):

- (1, Іванов, 1000,1),
- (2, Петров, 2000, 2),
- (3, Сидоров, 3000, 1).

У вигляді таблиці цей поточний стан відношення «Співробітники» зображено на рис. 2.1.



Ns	*F*	*Z*	*No*
Номер_співробітника	**Прізвище**	**Зарплата**	**Номер_відділу**
1	Іванов	1000	1
2	Петров	2000	2
3	Сидоров	3000	1

Рисунок 2.1 – Табличне зображення відношення

Реляційна БД – це набір відношень.

Схема реляційної БД – це набір заголовків відношень, що входять в базу даних.

Слід розуміти, що відношення не є таблицею, хоча таблична форма зображення відношення є зрозумілою та наочною.

Відмінності між відношеннями та таблицями визначаються *властивостями відношень*:

1. У відношеннях немає однакових кортежів (оскільки тіло відношення є множина кортежів);
2. Кортежі не впорядковані (зверху вниз);
3. Атрибути не впорядковані (зліва направо);
4. Всі значення атрибутів атомарні (тобто неподільні).

Відповідність реляційних і табличних понять дана в таблиці 2.1.

Таблиця 2.1 – Відповідність реляційних і табличних понять

Реляційне поняття	**Табличне поняття**
База даних	Набір таблиць
Схема бази даних	Набір заголовків таблиць
Відношення	Таблиця
Заголовок відношення	Заголовок таблиці
Тіло відношення	Тіло таблиці
Атрибут відношення	Найменування стовпця
Кортеж відношення	Рядок таблиці
Степінь відношення	Кількість стовпців
Потужність відношення	Кількість рядків
Домени і типи даних	Типи даних в комірках

При створенні реляційної моделі бази даних використовується покроковий процес розбиття одного відношення відповідно до алгоритму нормалізації, який докладно буде розглянуто далі на практичних прикладах з послідовним застосуванням різних нормальних форм. Цей процес має назву *нормалізації схеми бази даних* або *нормалізації відношень*.

Нормальна форма відношення – це властивість відношення в реляційній моделі даних. Вона характеризує відношення з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Звісно, що процес нормалізації розпочинається з першої нормальної форми, яка утворює підґрунтя для структурованої схеми бази даних.

Перша нормальна форма скорочено позначається як 1НФ або 1NF (Normal Form). Існує декілька рівнозначних визначень першої нормальної форми відношень. Так вважається, що відношення *R* знаходиться в 1НФ, якщо:

- воно задовольняє визначенню самого відношення;
- його атрибути містять тільки скалярні (атомарні) значення;
- воно є плоскою таблицею.

Більш детальне визначення першої нормальної форми буде дано далі при розгляді процедури нормалізації відношень після введення необхідних понять.

2.3.2. Цілісна частина реляційної моделі даних

Існує *два обмеження будь-якої реляційної БД*:

1. Цілісність сутностей;
2. Цілісність зовнішніх ключів (або посилань).

В реальному світі часто зустрічається ситуація, коли дані невідомі або не повні. Наприклад, місце проживання або дата народження людини можуть бути невідомі. Щоб обійти проблему неповних або невідомих даних, в БД можуть використовуватися типи даних, поповнені так званим *null-значеннями.*

Null-значення – це не значення, а деякий маркер, який показує, що значення невідомо.

За визначенням, тіло відношення є *множина* кортежів, тому відношення не можуть містити однакові кортежі. Це означає, що кожен кортеж повинен мати *властивість унікальності.* Насправді, властивість унікальності в межах відношення можуть мати окремі атрибути кортежів або групи атрибутів. Такі унікальні атрибути зручно використовувати для ідентифікації кортежів.

Потенційним ключем відношення *R* називається підмножина атрибутів *K*, якщо ця підмножина *K* має такі властивості:

1. *Властивість унікальності* – у відношенні не може бути двох різних кортежів, з однаковим значенням *K*;
2. *Властивість ненадлишковості* – жодна підмножина в *K* не має властивість унікальності.

Будь-яке відношення має, принаймні, один потенційний ключ, але може мати і декілька. тоді один з них оголошується *первинним,* а решта – *альтернативними.* Потенційний ключ, що складається з одного атрибута, називається *простим*, з декількох – *складеним* (або *складним*).

Потенційний ключ – це семантичне поняття, що відображає зміст понять предметної області. Наприклад, розглянемо два відношення «Співробітники» та «Деякі дані», поточний стан яких зображено в таблицях 2.2 та 2.3.

Таблиця 2.2 – Відношення «Співробітники»

Табельний номер	**Прізвище**	**Зарплата**
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000

Таблиця 2.3 – Відношення «Деякі дані»

A	***B***	***C***
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000

У відношенні «Співробітники» імена атрибутів співпадають з відповідними доменами, тому атрибут «Табельний номер» є унікальним, оскільки згідно з визначенням (а значить, і з моделлю предметної області) поняття «*табельний номер*» – це унікальний числовий код в межах будь-якого підприємства чи організації, який присвоюється кожному співробітнику). Виходячи з цього саме атрибут «Табельний номер» є потенційним ключем відношення «Співробітники».

На відміну від цього для атрибутів *A, B, C* відношення «Деякі дані» не можна зробити висновків про унікальність, не знаючи їх сенсу відповідно до предметної області та її моделі. Але все одно в цьому відношенні потенційний ключ призначити можна. При відсутності додаткової інформації це буде *повністю ключове відношення*, а його ключем буде сукупність атрибутів {*A, B, C*}. Це випливає з того, що всі кортежі у відношенні унікальні.

Правило цілісності сутностей: *атрибути, що входять до складу деякого потенційного ключа, не можуть приймати null-значення.*

Різні об’єкти предметної області завжди взаємопов’язані один з одним. Наприклад, співробітник *має* дітей. В цьому разі логічна модель бази даних може містити декілька пов’язаних між собою відношень.

Розглянемо приклад про поставки деталей деякій організації та її постачальників. Приймемо як модель предметної області той факт, що атрибути «Номер постачальника» та «Номер деталі» є унікальними. Інші атрибути та поточний стан відношення «Постачальники та деталі, що поставляються», яке відповідає моделі предметної області дано в таблиці 2.4.

Згідно з описом моделі предметної області потенційним ключем цього відношення буде сукупність двох атрибутів – {«Номер постачальника», «Номер деталі»}. Для того, щоб відрізняти ключові атрибути від неключових (тобто тих, що не входять до складу потенційного ключа) ключові атрибути виділяються підкреслюванням. При зазначеній моделі предметної області саме значень цих двох атрибутів у сукупності достатньо, для того щоб звернутись до

одного єдиного кортежу, бо значення інших атрибутів залежать або від значення атрибуту «Номер постачальника», або від значення атрибуту «Номер деталі», або від значень пари цих атрибутів.

Таблиця 2.4 – Відношення «Постачальники та деталі, що поставляються»

Номер постачальника	**Найменування постачальника**	**Номер деталі**	**Найменування деталі**	**Кількість, що поставляється**
1	Іванов	1	Болт	100
1	Іванов	2	Гайка	200
1	Іванов	3	Гвинт	300
2	Петров	1	Болт	150
2	Петров	2	Гайка	250
3	Сидоров	3	Гвинт	1000

В цьому відношенні є явні проблеми, які мають назву *аномалій* відношення. Вони полягають в наступному.

- Якщо змінилося найменування постачальника, то значення атрибуту «Найменування постачальника» потрібно одночасно змінити у всіх кортежах, де воно зустрічається, інакше дані стануть суперечливими.

Така ж ситуація з найменуваннями деталей.

Висновок – дані зберігаються в цьому відношенні з великою надмірністю.

- Якщо *Петров* тимчасово припинив поставки, то потрібно:

1. Або видалити всі кортежі, в яких «Найменування постачальника»=«Петров».

Висновок – це неможливо, оскільки якщо видалити всі кортежі з інформацією про постачання цього постачальника, то загубляться дані про самого *Петрова* як постачальника.

2. Або залишити у відношенні кортеж типу (2, Петров, NULL, NULL, NULL).

Висновок – згідно з правилом цілісності сутностей це неможливо, оскільки атрибут «Номер деталі» входить до складу потенційного ключа і не може містити null-значень.

Узагальнення – подібні проблеми виникають тому, що в одному відношенні змішані різні об’єкти предметної області – інформація про постачальників, про деталі та про постачання деталей.

Це означає, що *відношення погано нормалізовано*.

Для усунення проблем, тобто аномалій, спробуємо рознести дані по трьом відношенням – «Постачальники», «Деталі», «Поставки» та встановити, яким чином дані в цих відношеннях взаємопов'язані між собою. Табличне уявлення цих відношень представлено у таблицях 2.5, 2.6 та 2.7 відповідно.

Таблиця 2.5 – Відношення «Постачальники»

Номер постачальника	**Найменування постачальника**
1	Іванов
2	Петров
3	Сидоров

Таблиця 2.6 – Відношення «Поставки»

Номер постачальника	**Номер деталі**	**Кількість, що поставляється**
1	1	100
1	2	200
1	3	300
2	1	150
2	2	250
3	3	1000

Таблиця 2.7 – Відношення «Деталі»

Номер деталі	**Найменування деталі**
1	Болт
2	Гайка
3	Гвинт

Зв'язки між цими відношеннями визначається семантикою предметної області та можуть формально описуватися фразами, які відображають різні типи взаємозв'язків:

1. «Постачальники виконують Поставки»;
2. «Деталі поставляються через Поставки»;
3. «Деталі поставляються Постачальниками».

Існує три *типи взаємозв'язків* в реляційних БД, два з яких є основними:

1. «один-до-багатьох», наприклад, «*Один* Постачальник може виконувати *кілька* Поставок» або «*Одна* Деталь може поставлятися *кількома* Поставками»;
2. «багато-до-багатьох» (декілька «один-до-багатьох»), наприклад, «*Кілька* Деталей може поставлятися *кількома* Постачальниками».

Батьківське відношення – це те відношення, що входить у зв'язок з боку «один» (наприклад, «Постачальники»).

Дочірнє відношення – це те відношення, що входить у зв'язок з боку «багато» (наприклад, «Поставки»).

Механізм реалізації взаємозв'язку «один-до-багатьох» полягає у тому, що в дочірнє відношення додаються атрибути, які є посиланнями на ключові атрибути батьківського відношення.

Зовнішні ключі – це атрибути, які визначають, з якими кортежами батьківського відношення пов'язані кортежі дочірнього відношення (також називаються мігруючими з батьківського відношення).

Наприклад, у відношенні «Поставки» атрибути «Номер постачальника» і «Номер деталі» є посиланнями на ключові атрибути відношень «Постачальники» і «Деталі», а значить є зовнішніми ключами. Тепер у розглянутому прикладі після рознесення інформації по трьом відношенням (таблиці 2.5, 2.6 та 2.7) дані відношення вільні від недоліків, описаних вище, коли все зберігалося в одному відношенні (таблиця 2.4).

Зовнішнім ключем називається підмножина атрибутів *FK* відношення *R*, якщо виконуються дві умови:

1. Існує відношення *S* (*R* і *S* не обов'язково різні) з потенційним ключем *K*;
2. Кожне значення *FK* у відношенні *R* завжди збігається зі значенням *K* для деякого кортежу з *S*, або є null-значенням.

До визначення зовнішнього ключа можна зробити декілька зауважень:

1. Зовнішній ключ, так само як і потенційний, може бути простим і складеним;
2. Зовнішній ключ повинен бути визначений на тих же доменах, що і відповідний первинний ключ батьківського відношення;
3. Зовнішній ключ, як правило, *не має властивість унікальності*;
4. Якщо зовнішній ключ все-таки має властивість унікальності, то зв'язок між відношеннями має тип «*один до одного*» (третій тип

зв'язку, який є недоречним та незастосовним, оскільки означає, що відношення невірно розділено);

5. Хоч кожне значення зовнішнього ключа і зобов'язане збігатися зі значеннями потенційного ключа в деякому кортежі батьківського відношення, то зворотне невірно;
6. Для зовнішнього ключа не потрібно, щоб він був компонентом деякого потенційного ключа;
7. Null-значення для атрибутів зовнішнього ключа допустимі тільки в тому випадку, коли атрибути зовнішнього ключа не входять до складу жодного потенційного ключа;

Правило цілісності зовнішніх ключів: *зовнішні ключі не повинні бути неузгодженими, тобто для кожного значення зовнішнього ключа має існувати відповідне значення первинного ключа в батьківському відношенні*.

Іншими словами, цілісність посилань полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке веде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (ні на що не вказувати).

На практиці порушення правила цілісності зовнішніх ключів або посилальної цілісності може трапитись в результаті операцій, що змінюють стан бази даних. У визначенні посилальної цілісності беруть участь два відношення – батьківське і дочірнє. У кожному з них можливі три операції – вставка, оновлення, видалення, і тому повинні розглядатися шість різних варіантів. Але не всі ці варіанти порушують цілісність, наприклад, вставка кортежу в батьківське відношення або видалення кортежу з дочірнього відношення апріорі не може призвести до втрати цілісності БД. Тому залишається тільки чотири операції, що можуть порушити посилальну цілісність:

1. Оновлення кортежу в батьківському відношенні;
2. Видалення кортежу в батьківському відношенні;
3. Вставка кортежу в дочірнє відношення;
4. Оновлення кортежу в дочірньому відношенні.

Існують стратегії підтримки посилальної цілісності, які визначаються СУБД або користувачем, наприклад, RESTRICT (обмежити), CASCADE (каскадувати), SET NULL (встановити в NULL), SET DEFAULT (за замовчуванням), IGNORE (нехтувати) тощо. Але правильне моделювання ще на етапі створення логічної моделі реляційної БД може запобігти порушенню цілісності.

2.3.3. Маніпуляційна частина реляційної моделі даних

Третя частина реляційної моделі, маніпуляційна частина, стверджує, що доступ до реляційних даних здійснюється за допомогою *реляційної алгебри* або еквівалентного їй *реляційного числення*.

У реалізаціях конкретних реляційних СУБД зараз не використовується в чистому вигляді ні реляційна алгебра, ні реляційне числення, а фактичним стандартом доступу до реляційних даних є мова структурованих запитів SQL – Structured Query Language.

Мова SQL являє собою суміш операторів реляційної алгебри та виразів реляційного числення, яка використовує синтаксис, близький до фраз англійської мови та розширена додатковими можливостями, відсутніми в реляційній алгебрі та реляційному численні.

Реляційна алгебра є замкнутою, оскільки являє собою набір операторів, що використовують відношення в якості аргументів і повертають відношення як результат.

Реляційний оператор f має наступний вираз: $R=f(R_1, R_2, ..., R_n)$.

В реляційних виразах можна використовувати вкладені вирази будь-якої складної структури, оскільки в якості аргументів в реляційні оператори можна підставляти інші реляційні оператори, які підходять за типом, наприклад:

$$R= f(f_1 (R_{11}, R_{12}, ...), f_2 (R_{21}, R_{22}, ...), ...) . \quad (1)$$

Кожне відношення зобов'язане мати унікальне ім'я в межах бази даних, але можна не вимагати наявності імен від відношень, якщо ці відношення підставляються в якості аргументів в інші реляційні вирази. Такі відношення називаються *неіменованими відношеннями.*

Існує *дві групи реляційних операторів*.

Перша група – *теоретико-множинні оператори*:

1. Об'єднання;
2. Перетин;
3. Віднімання;
4. Декартів добуток.

Друга група – це *спеціальні реляційні оператори*:

1. Вибірка;
2. Проекція;
3. З'єднання;
4. Ділення.

Відношення є *сумісними за типом*, якщо вони мають ідентичні заголовки, тобто:

1. Відношення мають одну і ту ж множину імен атрибутів;
2. Атрибути з однаковими іменами визначені на одних і тих же доменах.

Деякі відношення не є сумісними за типом, але стають такими після перейменування атрибутів.

Оператор перейменування атрибутів має наступний синтаксис:

$$R \text{ RENAME } Atr_1, Atr_2, \ldots \text{ AS } NewAtr_1, NewAtr_2, \ldots , \quad (2)$$

де R – відношення, Atr_1, Atr_2, ... – його атрибути, $NewAtr_1$,$NewAtr_2$, ... – нові імена атрибутів, RENAME та AS ключові слова оператора.

Наприклад, *City* RENAME *City_Num* AS *Cityld* .

Розглянемо перші три оператори з *групи теоретико-множинних операторів* за допомогою прикладу, де вихідними є два відношення *A* та *B*, що надані в таблицях 2.8 та 2.9. Як і раніше атрибут «Табельний номер» є унікальним і тому його призначено потенційним ключем в цих двох відношеннях.

Таблиця 2.8 – Відношення *A*

Табельний номер	**Прізвище**	**Зарплата**
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000

Таблиця 2.9 – Відношення *B*

Табельний номер	**Прізвище**	**Зарплата**
1	Іванов	1000
2	Антонов	2500
4	Сидоров	3000

Об'єднання двох сумісних за типом відношень *A* і *B* – це відношення з тим же заголовком, що і у відношень *A* і *B*, та тілом, що складається з кортежів, які належать або *A*, або *B*, або обом відношенням. Синтаксис цього оператора:

$$A \text{ UNION } B , \quad (3)$$

де ключове слово UNION визначає операцію об'єднання.

Результат виконання операції об'єднання двох сумісних за типом відношень *A* і *B* (табл. 2.8 та 2.9) має вигляд, наведений в таблиці 2.10.

Таблиця 2.10 – Відношення *A* UNION *B*

Табельний номер	Прізвище	Зарплата
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000
2	Антонов	2500
4	Сидоров	3000

Потрібно особливо зазначити, що потенційні ключі відношень *A* і *B* *не успадковуються* їх об'єднанням. Слід звернути також увагу, що повторюваний кортеж (1, Іванов, 1000) в результуючому відношенні зустрічається тільки один раз, оскільки тіло відношення є множина кортежів, і тому кортежі не можуть повторюватись в жодному разі, навіть при виконанні будь-яких операцій.

Перетин двох сумісних за типом відношень *A* і *B* – це відношення з тим же заголовком, що і у відношень *A* і *B*, та тілом, що складається з кортежів, які належать одночасно обом відношенням *A* і *B*. Синтаксис цього оператора:

$$A \text{ INTERSECT } B, \tag{4}$$

де ключове слово INTERSECT визначає операцію перетину.

Результат виконання операції перетину двох сумісних за типом відношень *A* і *B* (табл. 2.8 та 2.9) має вигляд, наведений в таблиці 2.11.

Таблиця 2.11 – Відношення *A* INTERSECT *B*

Табельний номер	Прізвище	Зарплата
1	Іванов	1000

При виконанні цієї операції в результуючому відношенні атрибут «Табельний номер» не є потенційним ключем, оскільки *жодні реляційні оператори не передають результуючому відношенню жодних даних про потенційні ключі. Ключі явно задаються для кожного відношення*, виходячи з його змісту, ґрунтуючись на моделі предметної області.

Віднімання двох сумісних за типом відношень *A* і *B* – це відношення з тим же заголовком, що і у відношень *A* і *B*, та тілом, що складається з кортежів, які належать відношенню *A* і не належать відношенню *B*. Синтаксис оператора:

$$A \text{ MINUS } B, \tag{5}$$

де ключове слово MINUS визначає операцію віднімання.

Результат виконання операції віднімання двох сумісних за типом відношень *A* і *B* (табл. 2.8 та 2.9) має вигляд, наведений в таблиці 2.12.

Таблиця 2.12 – Відношення *A* MINUS *B*

Табельний номер	**Прізвище**	**Зарплата**
2	Петров	2000
3	Сидоров	3000

Декартів добуток двох відношень $A(A_1, A_2, ..., A_n)$ та $B(B_1, B_2, ..., B_m)$ – це відношення, заголовком якого є *зчеплення заголовків* відношень *A* і *B*: $(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$, а тіло складається з кортежів, які є *зчепленням кортежів* відношень *A*, *B*: $(a_1, a_2, ..., a_n, b_1, b_2, ..., b_m)$, таких що $(a_1, a_2, ..., a_n) \in A$, $(b_1, b_2, ..., b_m) \in B$. Синтаксис цього оператора:

$$A \text{ TIMES } B , \qquad (6)$$

де ключове слово TIMES визначає операцію декартова добутку.

Потужність добутку *A* TIMES *B* (кількість кортежів в ньому) дорівнює добутку потужностей відношень *A* і *B*, тому що кожен кортеж відношення *A* з'єднується з кожним кортежом відношення *B*.

Якщо у відношеннях *A* і *B* є атрибути з однаковими найменуваннями, то перед виконанням операції декартова добутку такі атрибути необхідно перейменувати. Перемножувати можна будь-які два відношення, сумісність за типом при цьому не потрібна.

Для реальних запитів *ця операція майже ніколи не використовується*. Однак вона важлива для виконання спеціальних реляційних операцій.

Для ілюстрації цього останнього оператора з групи теоретико-множинних розглянемо наступний приклад. Вихідні відношення *A* і *B* представлено в таблицях 2.13 та 2.14. Як і раніше атрибут «Номер постачальника» є унікальним і тому його призначено потенційним ключем в цих двох відношеннях, але поточний стан для спрощення трохи змінено.

Таблиця 2.13 – Відношення *A* (Постачальники)

Номер постачальника	**Найменування постачальника**
1	Іванов
2	Петров

Таблиця 2.14 – Відношення *B* (Деталі)

Номер деталі	**Найменування деталі**
1	Болт
2	Гайка

Результат виконання операції декартова добутку відношень *A* і *B* (табл. 2.13 та 2.14) має вигляд, наведений в таблиці 2.15.

Таблиця 2.15 – Відношення *A* TIMES *B*

Номер постачальника	**Найменування постачальника**	**Номер деталі**	**Найменування деталі**
1	Іванов	1	Болт
1	Іванов	2	Гайка
2	Петров	1	Болт
2	Петров	2	Гайка

Так само, як і у попередніх випадках, не можна переносити в результуюче відношення потенційні ключі з вихідних відношень, їх потрібно призначати безпосередньо для цього відношення, виходячи з його сенсу та моделі предметної області.

Наступна група реляційних операторів – це *спеціальні реляційні оператори*, серед яких найбільш застосовними на етапі моделювання є вибірка та проекція, а на етапі отримання інформації – з'єднання та ділення.

Вибірка (інші назви *обмеження*, *селекція*) на відношенні *A* з умовою *c* – це відношення з тим же заголовком, що і у відношення *A*, та тілом, що складається з кортежів, значення атрибутів яких при підстановці в *c* дають значення ІСТИНА, де *c* – це логічний вираз, до якого можуть входити атрибути відношення *A* та (або) скалярні вирази. Для *Θ-вибірки* (обмеження, селекції), як одного з видів вибірки, *c* має вид $X\Theta Y$, де Θ – це =, ≠, <, ≤, >, ≥ тощо, а *X* і *Y* – атрибути відношення *A* або скалярні значення. Синтаксис цього оператора:

$$A \text{ WHERE } c \text{ або} \tag{7}$$

$$A \text{ WHERE } X\Theta Y, \tag{8}$$

де ключове слово WHERE визначає операцію вибірки, а Θ (тета) – позначає один з математичних операторів порівняння.

Результат виконання операції вибірки з умовою «Зарплата<3000» для відношення *A* (табл. 2.8) має вигляд, наведений в таблиці 2.16. У цьому випадку атрибут «Табельний номер» залишається потенційним ключем, оскільки таке формулювання вибірки лише скорочує тіло відношення.

Таблиця 2.16 – Відношення *A* WHERE Зарплата<3000

Табельний номер	**Прізвище**	**Зарплата**
1	Іванов	1000
2	Петров	2000

Проекція відношення *A* по атрибутам *X*, *Y*, ..., *Z*, де кожен з атрибутів належить відношенню *A* – це відношення із заголовком (*X*, *Y*, ..., *Z*) і тілом, що містить множину кортежів виду (*x*, *y*, ..., *z*) таких, для яких у відношенні *A* знайдуться кортежі зі значенням атрибута *X*, що дорівнює *x*, значенням атрибута *Y*, що дорівнює *y*, ..., значенням атрибута *Z*, що дорівнює *z*. Синтаксис цього оператора:

$$A[X, Y, ..., Z] , \qquad (9)$$

де квадратні дужки […] визначають операцію проекції, а вказані всередині атрибути – це ті, по яким вона виконується.

Для ілюстрації цього спеціального реляційного оператора розглянемо наступний приклад. Вихідне відношення *A* представлено в таблиці 2.17.

Таблиця 2.17 – Відношення *A*

Номер постачальника	**Найменування постачальника**	**Місто постачальника**
1	Іванов	Київ
2	Петров	Харків
3	Сидоров	Харків
4	Сидоров	Дніпро

Результат виконання операції проекції по атрибуту «Місто постачальника» має вигляд, наведений в таблиці 2.18. Слід звернути увагу, що повторюваний кортеж (Харків) в результуючому відношенні зустрічається тільки один раз (як і у попередніх прикладах), оскільки тіло відношення є множина кортежів, і тому кортежі не можуть повторюватись.

Таблиця 2.18 – Відношення *A*[Місто постачальника]

Місто постачальника
Київ
Харків
Дніпро

Особливостями, цих двох операцій є те, що *операція вибірки* дає «*горизонтальний зріз*» відношення по деякій умові, *операція проекції* дає «*вертикальний зріз*» відношення з видаленням дублікатів кортежів.

Для реляційної операції з'єднання є чотири різновиди:

1. Загальна операція з'єднання;
2. Θ-з'єднання (тета-з'єднання);
3. Екві-з'єднання;
4. Природне з'єднання.

Розглянемо ці різновиди послідовно, звертаючи увагу на особливості їх виконання.

Загальна операція з'єднання відношень *A* і *B* за умовою *c* – це відношення:

$$(A \text{ TIMES } B) \text{ WHERE } c\,, \qquad (10)$$

де *c* являє собою логічний вираз, в який можуть входити атрибути відношень *A* і *B* та (або) скалярні вирази. Операція з'єднання є результат послідовного застосування операцій декартова добутку та вибірки.

Θ-з'єднання є звуженням загальної операції з'єднання. Воно полягає в наступному. Нехай відношення *A* містить атрибут *X*, відношення *B* містить атрибут *Y*, а Θ – один з математичних операторів порівняння (=, ≠, <, ≤, >, ≥ тощо). тоді *Θ-з'єднання* відношення *A* по атрибуту *X* з відношенням *B* по атрибуту *Y* – це відношення:

$$(A \text{ TIMES } B) \text{ WHERE } X\Theta Y\,, \qquad (11)$$

або в скороченій формі запису:

$$A[X\Theta Y]B\,. \qquad (12)$$

Для розуміння особливостей застосування цього оператора розглянемо приклад, в якому вихідні відношення *A* і *B* представлено в таблицях 2.19 та 2.20. На відміну від прикладу, де вихідні дані представлено в таблицях 2.5-2.7, модель предметної області розширено таким чином, що постачальники мають право постачати тільки ті деталі, статус яких не вище статусу постачальника,

тобто введено додаткові атрибути «Статус постачальника» та «Статус деталі». Потенційними ключами, як і раніше, згідно з моделлю предметної області залишаються атрибути «Номер постачальника» та «Номер деталі».

Таблиця 2.19 – Відношення *A* (Постачальники)

Номер постачальника	**Найменування постачальника**	**X (Статус постачальника)**
1	Іванов	4
2	Петров	1
3	Сидоров	2

Таблиця 2.20 – Відношення *B* (Деталі)

Номер деталі	**Найменування деталі**	**Y (Статус деталі)**
1	Болт	3
2	Гайка	2
3	Гвинт	1

Відповідь на запитання – «Які постачальники мають право поставляти які деталі?» дає наступна операція Θ-з'єднання: $A[X \geq Y]B$. Результат її виконання наведено в таблиці 2.21.

Таблиця 2.21 – Відношення «Що мають право поставляти постачальники»

Номер постачальника	**Найменування постачальника**	**X (Статус постачальника)**	**Номер деталі**	**Найменування деталі**	**Y (Статус деталі)**
1	Іванов	4	1	Болт	3
1	Іванов	4	2	Гайка	2
1	Іванов	4	3	Гвинт	1
2	Петров	1	3	Гвинт	1
3	Сидоров	2	2	Гайка	2
3	Сидоров	2	3	Гвинт	1

Отримане відношення дає розуміння, що мають право поставляти постачальники згідно зі статусами, які є характеристиками постачальників і деталей.

Для розуміння наступних операцій розглянемо умови прикладу, який було розглянуто раніше (див. табл. 2.4-2.7) та буде застосовано і далі. Оскільки при реалізації фізичної моделі із застосуванням реляційної СУБД краще застосовувати літери латиниці (а не кирилиці) для запобігання неточного відображення на різних системах символів внаслідок некоректної нумерації, тому далі будемо застосовувати *словник для імен відношень та атрибутів*. З урахуванням такого словника розглянемо базу даних для відомостей про деяку організацію з тією самою моделлю предметної області, як і раніше, тобто приймемо, що атрибути «Номер постачальника» та «Номер деталі» є унікальними. Тоді така база даних з дещо зміненим поточним станом складається з трьох пов'язаних відношень, що представлені в таблицях 2.22-2.24.

Таблиця 2.22 – Відношення *PD* (поставки)

PNUM (номер постачальника)	**DNUM** (номер деталі)	**VOL** (кількість, що поставляється)
1	1	100
1	2	200
1	3	300
2	1	150
2	2	250
3	1	1000

Таблиця 2.23 – Відношення *P* (постачальники)

PNUM (номер постачальника)	**PNAME** (найменування постачальника)
1	Іванов
2	Петров
3	Сидоров

Таблиця 2.24 – Відношення *D* (деталі)

DNUM (номер деталі)	**DNAME** (найменування деталі)
1	Болт
2	Гайка
3	Гвинт

Екві-з'єднання – це окремий випадок *Θ-з'єднання*, коли Θ є просто рівність:

$$A[X=Y]B\ . \qquad (13)$$

Тоді відповідь на запитання – «Які деталі поставляються якими постачальниками?» дає наступна операція екві-з'єднання: *P*[*PNUM=PNUM*]*PD* або (*P* RENAME *PNUM* AS *PNUM1*)[*PNUM1=PNUM2*](*PD* RENAME *PNUM* AS *PNUM2*). Результат її виконання наведено в таблиці 2.25.

Таблиця 2.25 – Відношення «Які деталі поставляються якими постачальниками?»

PNUM1	PNAME	PNUM2	DNUM	VOL
1	Іванов	1	1	100
1	Іванов	1	2	200
1	Іванов	1	3	300
2	Петров	2	1	150
2	Петров	2	2	250
3	Сидоров	3	1	1000

Недоліком виконання цієї операції для отримання відповіді є те, що в результуючому відношенні виникають два атрибута з однаковими значеннями. Уникнути цього дозволяє операція природного з'єднання.

Нехай дано відношення $A(A_1, A_2, ..., A_n, X_1, X_2, ..., X_p)$ і $B(X_1, X_2, ..., X_p, B_1, B_2, ..., B_m)$, що мають однакові атрибути $X_1, X_2, ..., X_p$ (тобто атрибути з однаковими іменами і визначені на однакових доменах).

Тоді *природне з'єднання* відношень *A* і *B* – це відношення із заголовком $(A_1, A_2, ..., A_n, X_1, X_2, ..., X_p, B_1, B_2, ..., B_m)$ і тілом, що містить множину кортежів $(a_1, a_2, ..., a_n, x_1, x_2, ..., x_p, b_1, b_2, ..., b_m)$, таких, що $(a_1, a_2, ..., a_n, x_1, x_2, ..., x_p) \in A$ і $(x_1, x_2, ..., x_p, b_1, b_2, ..., b_m) \in B$. Синтаксис цього оператора:

$$A \text{ JOIN } B\ , \qquad (14)$$

де ключове слово JOIN визначає операцію природного з'єднання.

Природне з'єднання виконується *по всім* однаковим атрибутам. Ця операція еквівалентна наступній послідовності реляційних операцій:

- перейменувати однакові атрибути у відношеннях;
- виконати декартів добуток відношень;
- виконати вибірку за співпадаючими значеннями атрибутів, які мали однакові імена;
- виконати проекцію, видаливши повторювані атрибути;

- перейменувати атрибути, повернувши їм початкові імена.

Природне з'єднання має властивість *асоціативності*, тобто:

$$(A \text{ JOIN } B) \text{ JOIN } C = A \text{ JOIN } (B \text{ JOIN } C) = A \text{ JOIN } B \text{ JOIN } C . \quad (15)$$

У цьому випадку відповідь на запитання – «Які деталі поставляються якими постачальниками?» дає наступна операція природного з'єднання, застосована для всіх трьох відношень (табл. 2.22-2.24):

$$P \text{ JOIN } PD \text{ JOIN } D .$$

Результат її виконання наведено в таблиці 2.26.

Таблиця 2.26 – Відношення *P JOIN PD JOIN D*

PNUM1	PNAME	DNUM	DNAME	VOL
1	Іванов	1	Болт	100
1	Іванов	2	Гайка	200
1	Іванов	3	Гвинт	300
2	Петров	1	Болт	150
2	Петров	2	Гайка	250
3	Сидоров	1	Болт	1000

Розглянемо останню операцію з групи спеціальних реляційних операторів. Нехай дано відношення $A(X_1, X_2, ..., X_p, Y_1, Y_2, ..., Y_m)$ і $B(Y_1, Y_2, ..., Y_m)$, причому атрибути $Y_1, Y_2, ..., Y_m$ – загальні для двох відношень.

Ділення відношень A на B – це відношення з заголовком $(X_1, X_2, ..., X_p)$ і тілом, що містить множину кортежів $(x_1, x_2, ..., x_p)$ таких, що для *всіх* кортежів $(y_1, y_2, ..., y_m) \in B$ у відношенні A знайдеться кортеж $(x_1, x_2, ..., x_p, y_1, y_2, ..., y_m)$. Відношення A – ділене, відношення B – дільник. Синтаксис цього оператора:

$$A \text{ DEVIDEBY } B , \quad (16)$$

де ключове слово DEVIDEBY визначає операцію ділення.

Для останнього прикладу (табл. 2.22-2.24) відповідь на запитання – «Які постачальники поставляють *всі* деталі?» дає наступна операція ділення відношень: X DEVIDEBY Y , де $X=PD[PNUM, DNUM]$ (таблиця 2.27) і $Y=D[DNUM]$ (таблиця 2.28). Результат її виконання наведено в таблиці 2.29.

Типові запити, які реалізуються за допомогою операції ділення в своєму формулюванні мають слово «всі».

Розглянуті вісім реляційних операцій дають можливість отримання відповідей на будь-які запитання при роботі з реляційними базами даних.

Таблиця 2.27 – Проекція *X=PD*[*PNUM, DNUM*]

PNUM	DNUM
1	1
1	2
1	3
2	1
2	2
3	1

Таблиця 2.28 – Проекція *Y=D*[*DNUM*]

DNUM
1
2
3

Таблиця 2.29 – Відношення *X* DEVIDEBY *Y*

PNUM
1

Для розуміння яким чином можна отримувати відповіді за допомогою реляційних операцій розглянемо декілька таких прикладів з використанням бази даних, яка містить три пов'язані між собою відношення (табл. 2.22-2.24).

Приклади:

1. Отримати імена постачальників, що поставляють деталь номер 2.

Розв'язок: ((*PD* JOIN *P*) WHERE *DNUM* = 2)[*PNAME*] .

2. Отримати імена постачальників, що поставляють принаймні одну гайку.

Розв'язок: (((*D* WHERE *DNAME* = *Гайка*) JOIN *PD*) JOIN *P*)[*PNAME*] або

(((*D* JOIN *PD*) JOIN *P*) WHERE *DNAME = Гайка*) [*PNAME*] .

3. Отримати імена постачальників, що поставляють всі деталі.

Розв'язок: ((*PD* [*PNUM, DNUM*] DEVIDEBY *D* [*DNUM*]) JOIN *P*)[*PNAME*] .

4. Отримати імена постачальників, що не постачають деталь номер 2.

Розв'язання цього прикладу залишається для самостійного виконання.

Звісно, що в багатьох випадках досягнути результату є можливим різними способами із застосуванням різної послідовності реляційних операцій.

3. ПОБУДОВА РЕЛЯЦІЙНОЇ ЛОГІЧНОЇ МОДЕЛІ ДАНИХ МЕТОДОМ НОРМАЛІЗАЦІЇ ФОРМ ВІДНОШЕНЬ

3.1. Етапи розробки бази даних

Метою розробки будь-якої бази даних є зберігання і використання інформації про деяку предметну область.

Для реалізації цієї мети використовуються такі інструменти:

1. Реляційна модель даних як зручний спосіб представлення даних предметної області;
2. Мова SQL як універсальний спосіб маніпулювання такими даними.

Для однієї і тієї ж предметної області реляційні відношення можна спроектувати безліччю різних способів. Наприклад, кілька відношень з великою кількістю атрибутів, або навпаки, рознести всі атрибути по великому числу дрібних відношень. Тому завжди виникає питання як правильно створити логічну модель реляційної бази даних, по-перше, щоб раціонально розподілити атрибути по відношенням, і, по-друге, щоб уникнути можливих аномалій та порушень цілісності ще на етапі проектування.

Під час розробки бази даних зазвичай виділяється кілька рівнів моделювання (етапів розробки БД), за допомогою яких відбувається перехід від предметної області до конкретної реалізації бази даних засобами конкретної СУБД.

При створенні будь-якої бази даних можна виділити наступні збільшені *рівні моделювання*:

1. Предметна область;
2. Модель предметної області;
3. Логічна модель даних;
4. Фізична модель даних;
5. База даних і додаткові застосунки.

Дамо визначення кожному з цих рівнів, зосередившись саме на етапах концептуального та логічного проектування.

Предметна область – це частина реального світу, дані про яку ми хочемо відобразити в базі даних.

Модель предметної області – це наші знання про предметну область. Знання можуть бути як у вигляді неформальних знань в мозку експерта, так і виражені формально за допомогою будь-яких засобів.

Логічна модель описує поняття предметної області, їх взаємозв'язок, а також обмеження на дані, що накладаються предметною областю. Вона є початковим прототипом майбутньої БД і будується в термінах інформаційних одиниць, але без прив'язки до конкретної СУБД. Наприклад, поняття – «співробітник», «відділ», «проект», взаємозв'язки між поняттями – «співробітник працює рівно в одному відділі», «над одним проектом може працювати кілька співробітників», обмеження – «вік співробітника не менше 16 і не більше 60 років». Логічна модель даних є початковим прототипом майбутньої БД і будується у термінах інформаційних одиниць, але без прив'язки до конкретної СУБД. У даному випадку передбачається використання реляційних СУБД, і тому логічна модель даних формулюватиметься в термінах реляційної моделі даних [2-5]. При виконанні даного етапу повинні також враховуватись критерії оцінки якості логічної моделі даних: адекватність БД предметної області, легкість розробки та супроводу БД, швидкість виконання операцій оновлення даних (вставка, оновлення, видалення кортежів), швидкість виконання операцій вибірки даних.

Фізична модель даних описує дані засобами конкретної СУБД.

База даних реалізується на конкретній програмно-апаратній основі.

Виконання перелічених перших трьох рівнів моделювання разом з теорією, що застосовується при нормалізація схеми реляційної бази даних, буде далі розглядатися разом з прикладами. Для перших трьох нормальних форм відношень буде розглянуто один основний приклад, для інших нормальних форм більш високих порядків будуть розглянуті окремі приклади з огляду на ті особливості моделі предметної області, які розв'язують ці форми.

3.2. Предметна область та її модель

Предметна область нескінченна і містить як важливі поняття та дані, так і малозначущі або взагалі не значущі дані. Визначення важливості даних, які будуть використовуватися в БД, що розробляється, залежить від вибору предметної області та виконується на етапі побудови моделі.

При побудові моделі предметної області є велика кількість методик опису предметної області. Найбільш відомі з них: методика структурного аналізу SADT (Structured Analysis and Design Technique), діаграми потоків даних Гейна-Сарсона, методика об'єктно-орієнтованого аналізу UML (Unified Modeling Language), що містить багатий набір діаграм та нотацій [1].

На початковому етапі побудови БД виконуються попередні дії [1], а саме:

1. Планування розробки БД, тобто планування ефективного способу реалізації етапів життєвого циклу БД;
2. Визначення вимог до системи, тобто визначення діапазону дій та меж застосунку БД, складу його користувачів та областей застосування;
3. Збір та аналіз вимог користувачів, тобто збір та аналіз користувальницьких уявлень про предметну область, формування вимог по кожному користувальницькому уявленню та сукупного уявлення всіх користувачів.

Уявлення користувача (УК) – це визначення вимог до застосування бази даних конкретної категорії користувачів (наприклад, менеджера чи продавця, якщо предметна область – магазин) чи інших потреб (наприклад, потреби відділу постачання).

Збір та аналіз вимог користувачів – це процес збору та аналізу інформації про предметну область (наприклад, бухгалтерія підприємства), яка описуватиметься створюваним застосунком бази даних, робота якої буде підтримуватися з його допомогою. Ця інформація використовується для визначення вимог користувачів до створюваної системи. Збір та аналіз вимог є попереднім етапом концептуального проектування бази даних, під час якого специфікації вимог користувачів аналізуються з метою з'ясування всіх необхідних відомостей.

3.3. Основний приклад проектування БД – модель предметної області

Розглянемо умови основного прикладу, за допомогою якого буде проілюстровано процес побудови логічної моделі реляційної бази даних із застосуванням методу нормальних форм або нормалізації форм відношень.

Нехай *предметна область* створюваної бази даних – «Проектна організація», а для неї *модель предметної області* формується на основі зібраних уявлень майбутніх користувачів та містить наступні факти:

1. *Співробітники* організації виконують *проекти*;
2. *Проекти* складаються з декількох *завдань*;
3. Кожен *співробітник* може брати участь в одному або декількох *проектах*, або тимчасово не брати участь в жодному проекті;
4. Над кожним *проектом* може працювати кілька *співробітників*, або тимчасово *проект* може бути припинений, тоді над ним не працює жоден *співробітник*;

5. Над кожним *завданням* в *проекті* працює рівно один *співробітник*;
6. Кожен *співробітник* числиться в одному *відділі*;
7. Кожен *співробітник* має *телефон*, що знаходиться у *відділі співробітника* (у кожному *відділі* є тільки один *телефон*).

У даному випадку модель предметної області описується неформальним текстом, який містить відомості про предметну область. Це один з варіантів опису інформації в обраній предметній області, який відповідає організації виробничих процесів в розглянутій проектній організації в даний час. Саме ці факти буде покладено в основу логічної моделі БД. Кожен з них має своє значення і призводить до певного вигляду логічної моделі взагалі, та відношень, потенційних ключів, зовнішніх ключів тощо зокрема. Зміна будь-якої умови в цій моделі предметної області призведе до іншої логічної моделі реляційної бази даних, що їй відповідає. Наприклад, якщо змінити факт 5 на наступну редакцію: «Над кожним *завданням* в *проекті* може працювати декілька *співробітників*» або «Кожен *співробітник* може працювати над декількома завданнями в одному *проекті*», то це буде мати свої наслідки ще на етапі призначення потенційних ключів, що призведе до іншої моделі БД.

Зібрана таким чином інформація дає уявлення, що саме повинна містити база даних (виділені нахилом іменники є кандидатами в атрибути відношення), але не дає розуміння як інформація має бути пов'язана між собою, що від чого може залежати та що може бути унікальним в межах предметної області. Останнє є особливо важливим, оскільки інформація про унікальність надає змогу правильно обирати потенційні ключи на основі поточної моделі предметної області. Тому необхідно зібрати з майбутніх користувачів БД додаткову інформацію з фокусом саме на ці моменти.

Нехай зібрані додаткові уточнення полягають в наступному:

1. Про кожного співробітника необхідно зберігати *табельний номер* і *прізвище*, *табельний номер* є унікальним для кожного співробітника;
2. Кожен відділ має унікальний *номер*;
3. Кожен проект має *номер* і *найменування*, *номер* проекту є унікальним;
4. Кожне завдання в проекті має *номер*, унікальний в межах цього проекту, роботи в різних проектах можуть мати однакові *номери*.

Ці додаткові уточнення визначають, які дані необхідно зберігати та які факти необхідно враховувати при створенні логічної моделі, призначенні потенційних ключів та визначення різних залежностей між атрибутами в БД.

Для завершення збору інформації до цього потрібно ще додати *поточний стан предметної області*:

1. *Іванов* працює у *відділі 1*, виконує в першому проекті «*Турбіна*» *завдання 1* та у *другому проекті* «*Трактор*» *завдання 1*;
2. *Петров* працює у *відділі 2*, виконує в *проекті 1 завдання 3* та в *проекті 2 завдання 2*;
3. *Сидоров* працює у *відділі 1*, виконує *в проекті 1 завдання 2*.

Звісно, що відомості про поточний стан предметної області з часом можуть змінюватися, але така зміна буде справою користувачів після вводу бази даних в експлуатацію.

Тепер перелічених відомостей цілком достатньо для створення логічної моделі реляційної бази даних для визначеної моделі предметної області «Проектна організація». Для побудови такої логічної моделі застосуємо метод нормалізації відношень, який, звісно, розпочинається з *першої нормальної форми*.

3.4. Перша нормальна форма (1НФ) відношення

Пригадаємо, що згідно з викладеним в п. 2.3.1 *перша нормальна форма* (1НФ або 1NF) – це звичайне відношення, тобто це форма уявлення інформації, яка задовольняє визначенню відношення, з атрибутами, що містять тільки скалярні (атомарні) значення, та яка є плоскою таблицею.

Засіб, що дозволяє однозначно ідентифікувати кортеж відношення, – це потенційні ключі відношення, тобто один атрибут (простий ключ) або набір атрибутів (складений ключ) відношення, що має властивості унікальності та ненадлишковості. Доступ до кожного кортежу здійснюється за значенням потенційного ключа для цього кортежу.

3.5. Перша нормальна форма в основному прикладі

На першому кроці при побудові логічної моделі для предметної області «Проектна організація» будемо зберігати дані в одному відношенні:

СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ (<u>НСПІВР</u>, ПРІЗВ, НВІД, ТЕЛ, <u>НПРО</u>, ПРОЕКТ, НЗАВД) ,

що має такі атрибути:

<u>НСПІВР</u> – табельний номер *співробітника*,

ПРІЗВ – прізвище *співробітника*,

НВІД – номер відділу, в якому числиться *співробітник*,

ТЕЛ – телефон у відділі *співробітника*,

НПРО – номер проекту, над яким працює *співробітник*,

ПРОЕКТ – найменування проекту, над яким працює співробітник,

НЗАВД – номер завдання, над яким працює *співробітник*.

Такий опис атрибутів є *словником бази даних*, який дозволяє розуміти сенс кожного атрибута.

Зверніть увагу, що всі атрибути прив'язані до співробітника, тобто модель бази даних будується з точки зору співробітника, як головного об'єкта бази даних.

Оскільки кожен співробітник в кожному проекті виконує рівно одне завдання, то потенційний ключ –**{НСПІВР, НПРО}** (тут і далі всі атрибути, що входять до складу потенційного ключа прийнято позначати підкреслюванням для простоти їхньої ідентифікації при перевірці певних умов в алгоритмі нормалізації).

Примітка: якщо б згідно з моделлю предметної області було б визначено, що «Кожен *співробітник* може працювати над декількома завданнями в одному *проекті*», потенційний ключ мав би вигляд –**{НСПІВР, НПРО, НЗАВД}**.

Зауваження: незважаючи на те, що в даному випадку атрибути позначені за допомогою букв кирилиці, при реальному проектуванні логічної моделі бази даних рекомендується використовувати назви атрибутів, що складаються з букв латиниці. Це пов'язано з тим, що СУБД може не підтримувати кирилицю, і тоді під час створення фізичної моделі виникне необхідність зміни назв атрибутів.

Поточний стан предметної області «Проектна організація» можна відобразити у вигляді таблиці 3.1.

Таблиця 3.1 – Відношення СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ

НСПІВР	ПРІЗВ	НВІД	ТЕЛ	НПРО	ПРОЕКТ	НЗАВД
1	Іванов	1	707-11-11	1	Турбіна	1
1	Іванов	1	707-11-11	2	Трактор	1
2	Петров	2	707-22-22	1	Турбіна	3
2	Петров	2	707-22-22	2	Трактор	2
3	Сидоров	1	707-11-11	1	Турбіна	2

Таблиця 3.1, що зображує стан відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_**

ПРОЕКТИ, характеризується тим, що дані зберігаються в ній з великою надмірністю. В багатьох рядках повторюються прізвища співробітників, номери телефонів, найменування проектів. Крім того, у цьому відношенні зберігаються разом незалежні дані – і дані про співробітників, і дані про відділи, і дані про проекти, і про завдання в проектах. Поки ніяких дій із відношенням не проводиться, це ні на що не впливає. Але як тільки стан предметної області змінюється, то при спробах відповідним чином змінити стан бази даних виникає велика кількість проблем. Ці проблеми називаються *аномаліями оновлення*.

3.6. Аномалії оновлення у відношенні

Таким чином, таблиця 3.1 зображує поточний стан бази даних «Проектна організація» і може змінюватися, тому будь-які висновки потрібно робити, розглядаючи саме модель предметної області, а не цей поточний стан предметної області. Зважаючи на це можна зробити певні висновки щодо існуючих аномалій в цьому відношенні, які можуть призводити до некоректності та невідповідності бази даних предметній області при внесенні змін, тобто до порушення цілісності бази даних. Аналізуючи модель предметної області можна прийти до висновку, що в даному відношенні спостерігаються всі можливі аномалії. Розглянемо ці *три види аномалій* послідовно.

1. Аномалії вставки (INSERT).

У відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ** не можна вставити дані про співробітника, який поки не бере участь в жодному проекті, наприклад, не можна вставити кортеж (4, Антонов, 2, 707-22-22, null, null, null), оскільки це буде порушенням цілісності сутностей (п. 2.3.2), бо ключовий атрибут **НПРО** буде приймати null-значення. Така ж ситуація відбувається з проектами – не можна вставити дані про проект, над яким поки не працює жоден співробітник.

2. Аномалії оновлення (UPDATE).

Якщо співробітник змінює *прізвище*, або проект змінює *найменування*, або змінюється *номер* телефону, то такі зміни необхідно одночасно виконати у всіх місцях, інакше відношення стане некоректним, тому оновлення бази даних однією дією неможливе.

3. Аномалії видалення (DELETE).

При видаленні деяких даних може трапитись втрата іншої інформації, на-

приклад, якщо закрити проект «Турбіна» і видалити всі рядки, в яких він зустрічається, то будуть втрачені всі дані про співробітника Сидорова.

Причина аномалій – зберігання в одному відношенні різнорідної інформації (про співробітників, про проекти, про завдання в проектах).

Висновок – логічна модель даних неадекватна моделі предметної області, тому база даних, заснована на такій моделі, буде працювати неправильно.

3.7. Функціональні залежності атрибутів у відношенні

Для усунення зазначених аномалій, тобто для правильного проектування моделі даних, застосовується *метод нормалізації відношень*. Нормалізація заснована на понятті *функціональної залежності атрибутів* відношення.

Нехай R – відношення. Множина атрибутів Y *функціонально залежна* від множини атрибутів X, коли для будь-яких кортежів $r_1, r_2 \in R$ – з того що $r_1X=r_2X$ випливає, що $r_1Y=r_2Y$. Позначення функціональної залежності: $X \rightarrow Y$.

Функціональна залежність означає, що у всіх кортежах, що мають однакові значення атрибутів X, значення атрибутів Y також збігаються *в будь-якому стані* відношення R.

В функціональній залежності множина атрибутів X (те, від чого залежить) називається *детермінантом* функціональної залежності, а множина атрибутів Y (те, що залежить) називається *залежною частиною*.

Зауваження: якщо атрибути X складають потенційний ключ відношення R, то *будь-який атрибут* відношення R функціонально залежить від X.

3.8. Функціональні залежності атрибутів в основному прикладі

На основі аналізу моделі предметної області (не зважаючи на її поточний стан) випишемо всі наявні функціональні залежності між атрибутами у відношенні **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ:**

1. Залежність атрибутів від *ключа* відношення:

{НСПІВР, НПРО}→ПРІЗВ;
{НСПІВР, НПРО}→НВІД;
{НСПІВР, НПРО}→ТЕЛ;
{НСПІВР, НПРО}→ПРОЕКТ;
{НСПІВР, НПРО}→НЗАВД;

2. Залежність атрибутів, що характеризують співробітника від *табельного номера* співробітника:

НСПІВР→ПРІЗВ;

НСПІВР→НВІД;

НСПІВР→ТЕЛ;

3. Залежність найменування проекту від *номера проекту*:

НПРО→ПРОЕКТ;

4. Залежність номера телефону від *номера відділу*:

НВІД→ТЕЛ.

Зауваження: 1. Наведені функціональні залежності не виведені з зовнішнього вигляду відношення в таблиці 3.1, ці залежності відображають взаємозв'язки між об'єктами предметної області і є додатковими обмеженнями, які визначаються предметною областю; 2. Функціональна залежність атрибутів стверджує лише те, що для кожного конкретного стану бази даних за значенням одного атрибута (детермінанта) можна однозначно визначити значення іншого атрибута (залежної частини), але конкретні значення залежної частини можуть бути різні в різних станах бази даних, наприклад, знаючи *табельний номер* співробітника, можна визначити його *прізвище*, а за *номером відділу* можна визначити *номер телефону*.

3.9. Друга нормальна форма відношення (2НФ)

Відношення *R* знаходиться в *другій нормальній формі* (2НФ або 2NF) тоді і тільки тоді, коли відношення знаходиться в першій нормальній формі (1НФ) і немає неключових атрибутів, залежних від частини складеного ключа, при цьому *неключовий атрибут* – це атрибут, який не входить до складу жодного потенційного ключа, ані первинного, ані жодного альтернативного (п. 2.3.2).

Зауваження: якщо потенційний ключ відношення є простим, то відношення автоматично перебуває в 2НФ.

Приведення до 2НФ відбувається наступним чином. Для того, щоб усунути залежність атрибутів від частини складеного (складного) ключа, потрібно зробити *декомпозицію* (тобто розбиття) відношення на кілька відношень. При цьому *ті атрибути, які залежать від частини складеного ключа*, виносяться в окреме відношення разом з цією частиною потенційного ключа, і ця ж частина ключа залишається у вихідному відношенні для зв'язку відношень.

3.10. Приведення відношень до 2НФ в основному прикладі

Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ** не знаходиться в 2НФ, тому що є атрибути, які залежать від частини складеного ключа:

НСПІВР→ПРІЗВ;

НСПІВР→НВІД;

НСПІВР→ТЕЛ;

НПРО→ПРОЕКТ.

Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ** декомпозуємо на три відношення:

1. **СПІВРОБІТНИКИ_ВІДДІЛИ**;
2. **ПРОЕКТИ**;
3. **ЗАВДАННЯ**.

Проаналізуємо ці відношення докладно.

1. Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ (НСПІВР, ПРІЗВ, НВІД, ТЕЛ)**.

Це відношення має чотири раніше визначені функціональні залежності, які були в основі декомпозиції:

НСПІВР→ПРІЗВ; **НСПІВР→Н_ВІД**;

НСПІВР→ТЕЛ; **НВІД→ТЕЛ**;

2. Відношення **ПРОЕКТИ (НПРО, ПРОЕКТ)**

Це відношення має одну функціональну залежність, яка була в основі декомпозиції:

НПРО→ПРОЕКТ;

3. Відношення **ЗАВДАННЯ (НСПІВР, НПРО, НЗАВД)**.

Це відношення є залишком початкового відношення (див. табл. 3.1) і також, як і попереднє, має тільки одну функціональну залежність:

{НСПІВР, НПРО}→НЗАВД.

Поточний стан цих трьох відношень показаний в таблицях 3.2-3.4.

Таблиця 3.2 – Відношення СПІВРОБІТНИКИ_ВІДДІЛИ

НСПІВР	ПРІЗВ	НВІД	ТЕЛ
1	Іванов	1	707-11-11
2	Петров	2	707-22-22
3	Сидоров	1	707-11-11

Таблиця 3.3 – Відношення ПРОЕКТИ

НПРО	**ПРОЕКТ**
1	Турбіна
2	Трактор
1	Турбіна
2	Трактор
1	Турбіна

Таблиця 3.4 – Відношення ЗАВДАННЯ

НСПІВР	**НПРО**	**НЗАВД**
1	1	1
1	2	1
2	1	3
2	2	2
3	1	2

Проаналізуємо отримані відношення з точки зору аномалій оновлення, які були виявлені у відношенні **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ**.

Відношення, отримані в результаті декомпозиції, знаходяться в 2НФ.

Дійсно, відношення **СПІВРОБІТНИКИ_ВІДДІЛИ** і **ПРОЕКТИ** мають прості ключі, отже автоматично знаходяться в 2НФ. Відношення **ЗАВДАННЯ** має складений ключ, але єдиний неключових атрибут **НЗАВД** функціонально залежить від всього ключа **{НСПІВР, НПРО}**.

Таким чином, частина аномалій оновлення усунена. Тепер дані про співробітників і про проекти зберігаються в різних відношеннях. Так, наприклад, *при появі співробітників, які не беруть участь ні в одному з проектів,* просто додаються кортежі у відношення **СПІВРОБІТНИКИ_ВІДДІЛИ**. Так само, *при появі проекту, над яким не працює жоден співробітник*, просто вставляється кортеж у відношення **ПРОЕКТИ**. Прізвища співробітників і найменування проектів тепер зберігаються без надмірності. *Якщо співробітник змінить прізвище або проект змінить назву*, то таке оновлення буде зроблено в одному місці.

Але при цьому частину аномалій усунути не вдалося. Розглянемо цю решту аномалій послідовно за трьома видами.

1. Аномалії вставки (INSERT).

У відношення **СПІВРОБІТНИКИ_ВІДДІЛИ** не можна вставити кортеж

(4, Антонов, 1, 707-22-22), оскільки при цьому вийде, що два співробітника з першого відділу (Іванов і Антонов) мають різні номери телефонів, а це суперечить моделі предметної області, оскільки в одному відділі встановлено тільки один телефон.

2. Аномалії оновлення (UPDATE).

Одні й ті ж номери телефонів повторюються в багатьох кортежах відношення, тому оновлення БД однією дією неможливе.

3. Аномалії видалення (DELETE).

При видаленні деяких даних як і раніше може статися втрата іншої інформації, наприклад, якщо видалити дані про співробітника Петрова, то буде втрачена інформація про те, що у *відділі номер 2* є *телефон* 707-22-22.

Причина аномалій – зберігання в одному відношенні різнорідної інформації (про співробітників і про відділи).

Висновок – логічна модель даних неадекватна моделі предметної області, тому база даних, заснована на такій моделі, буде працювати неправильно.

3.11. Третя нормальна форма відношення (3НФ)

Відношення *R* знаходиться в *третій нормальній формі* (3НФ або 3NF) тоді і тільки тоді, коли відношення знаходиться в 2НФ і всі неключові атрибути взаємно незалежні, при цьому *взаємно незалежними атрибутами* називаються такі атрибути відношення, жоден з яких не є функціонально залежним від іншого.

Приведення до 3НФ відбувається наступним чином. Для того, щоб усунути залежність неключових атрибутів, потрібно зробити *декомпозицію* відношення на кілька відношень. При цьому *ті неключові атрибути, які є залежними*, виносяться в окреме відношення разом з детермінантами функціональних залежностей, і ці ж детермінанти залишаються в вихідному відношенні для зв'язку відношень.

3.12. Приведення відношень до 3НФ в основному прикладі

Відношення **ПРОЕКТИ** та **ЗАВДАННЯ** знаходяться в 3НФ, оскільки мають тільки по одному неключовому атрибуту і тому в них не може бути взаємно залежних неключових атрибутів.

Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ** не знаходиться в 3НФ, тому що є функціональна залежність неключових атрибутів (тобто залежність *номера* телефону від *номера* відділу): **НВІД→ТЕЛ**.

Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ** декомпозуємо на два відношення:

1. **СПІВРОБІТНИКИ**;
2. **ВІДДІЛИ**.

Проаналізуємо ці відношення докладно.

1. Відношення **СПІВРОБІТНИКИ (НСПІВР, ПРІЗВ, НВІД)**.

Це відношення має дві раніше визначені функціональні залежності, які були в основі декомпозиції:

НСПІВР→ПРІЗВ;

НСПІВР→НВІД.

2. Відношення **ВІДДІЛИ (НВІД, ТЕЛ)**.

Це відношення має одну функціональну залежність, яка була в основі декомпозиції:

НВІД→ТЕЛ

Поточний стан цих двох відношень показаний в таблицях 3.5 та 3.6.

Таблиця 3.5 – Відношення СПІВРОБІТНИКИ

НСПІВР	ПРІЗВ	НВІД
1	Іванов	1
2	Петров	2
3	Сидоров	1

Таблиця 3.6 – Відношення ВІДДІЛИ

НВІД	ТЕЛ
1	707-11-11
2	707-22-22

Зауваження: атрибут **НВІД**, *що не був ключовим* у відношенні **СПІВРОБІТНИКИ_ВІДДІЛИ**, *стає потенційним ключем* в відношенні **ВІДДІЛИ** і за рахунок цього усувається надмірність, пов’язана з багаторазовим зберіганням одних і тих же номерів телефонів.

Повна логічна модель бази даних в основному прикладі зображена на рис. 3.1. Це реляційна модель, що складається з чотирьох відношень **ПРОЕКТИ** (табл. 3.3), **ЗАВДАННЯ** (табл. 3.4), **СПІВРОБІТНИКИ** (табл. 3.5), **ВІДДІЛИ** (табл. 3.6).

Слід зазначити, що відношення **ЗАВДАННЯ** є залишком від первісного відношення, а інші три відношення (**СПІВРОБІТНИКИ**, **ПРОЕКТИ**, **ВІДДІЛИ**) були утворені в результаті декомпозиції. Всі отримані в результаті приведення до 3НФ відношення тепер вільні від зазначених вище аномалій, а це означає, що в базі даних побудованій на такій логічній моделі автоматично унеможливлюється порушення цілісності при виконанні операцій вставки, оновлення та видалення кортежів.

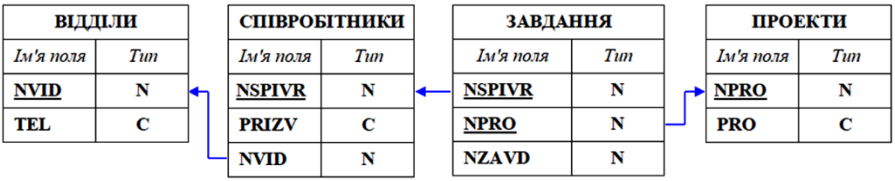


Рисунок 3.1 – Логічна модель (схема) бази даних «Проектна організація»

Зв’язок між відношеннями отриманими в результаті декомпозиції, здійснюється за допомогою зовнішніх ключів. Якщо якийсь атрибут є у кількох відношеннях, його наявність відображає певний зв’язок між кортежами цих відношень. Так, наприклад, у відношенні **СПІВРОБІТНИКИ** (табл. 3.5) атрибут **НСПІВР** є первинним потенційним ключем, а у відношенні **ЗАВДАННЯ** (табл. 3.4) є зовнішнім ключем і служить для встановлення відповідності між відомостями про співробітників та відомостями про проекти та завдання, над якими вони працюють. У цьому разі кажуть, що атрибут **НСПІВР** у дочірньому відношенні **ЗАВДАННЯ** посилається на первинний ключ, тобто. на атрибут **НСПІВР** у батьківському відношенні **СПІВРОБІТНИКИ**.

3.13. Схематизація алгоритму нормалізації при приведенні до 3НФ

Крок 1 – приведення до 1НФ.

Задається одне або кілька відношень, що відображають поняття предметної області. За моделлю предметної області (а не за зовнішнім виглядом отриманих відношень!) виписуються виявлені функціональні залежності. Всі відношення автоматично знаходяться в 1НФ.

Крок 2 – приведення до 2НФ.

Якщо в деяких відношеннях виявлена залежність атрибутів від частини складеного (складного) ключа, то проводимо декомпозицію цих відношень на

кілька відношень наступним чином: ті атрибути, які залежать від частини складеного ключа виносяться в окреме відношення разом з цією частиною ключа. Усі ключові атрибути залишаються і у вихідному відношенні для забезпечення зв'язку між кортежами цих відношень через первинний та зовнішній ключі.

Алгоритм приведення до 2НФ схематизовано можна записати наступним чином. Нехай є такі вихідні дані:

- початкове відношення – $R(K_1, K_2, A_1, ..., A_n, B_1, ..., B_m)$;
- складений ключ – (K_1, K_2);
- функціональні залежності –
  $(K_1, K_2) \rightarrow A_1, .., A_n, B_1, ..., B_m$ – залежність всіх атрибутів від ключа;
  $(K_1) \rightarrow A_1, .., A_n$ – залежність деяких атрибутів від частини складеного ключа.

Отримані декомпозовані відношення:

- $R_1(K_1, K_2, B_1, ..., B_m)$ – залишок від вихідного відношення, в якому залишається ключ $\{K_1, K_2\}$;
- $R_2(K_1, A_1, ..., A_n)$ – відношення з атрибутами, винесеними з початкового відношення разом з частиною складеного ключа, де ключем є K_1.

Крок 3 – приведення до 3НФ.

Якщо в деяких відношеннях виявлена залежність деяких неключових атрибутів від інших неключових атрибутів, то проводимо декомпозицію цих відношень наступним чином: ті неключові атрибути, які залежать від інших неключових атрибутів виносяться в окреме відношення разом з детермінантом функціональної залежності. У відношенні, з якого виносяться атрибути, залишається детермінант функціональної залежності для забезпечення зв'язку між кортежами цих відношень через первинний та зовнішній ключі. У новому відношенні ключем стає детермінант функціональної залежності.

Алгоритм приведення до 3НФ схематизовано можна записати наступним чином. Нехай є такі вихідні дані:

- початкове відношення – $R(K, A_1, .., A_n, B_1, ..., B_m)$;
- ключ – K;
- функціональні залежності –
  $K \rightarrow A_1, .., A_n, B_1, ..., B_m$ – залежність всіх атрибутів від ключа відношення;
  $(A_1, .., A_n) \rightarrow B_1, ..., B_m$ – залежність деяких неключових атрибутів від інших неключових атрибутів.

Отримані декомпозовані відношення:

- $R_1(K, A_1, ..., A_n)$ – залишок від вихідного відношення, в якому залишається ключ K;
- $R_2(A_1, ..., A_n, B_1, ..., B_m)$ – атрибути, винесені з початкового відношення разом з детермінантом функціональної залежності, де ключем є детермінант функціональної залежності $\{A_1, ..,A_n\}$.

3.14. Коректність процедури нормалізації

Декомпозиція відношень – це взяття однієї або декількох проекцій вихідного відношення так, щоб ці проекції в сукупності містили (можливо з повтореннями) всі атрибути вихідного відношення. При декомпозиції не повинні губитися атрибути відношень і самі дані, тобто повинна бути можлива зворотна операція – по декомпозованим відношенням можна відновити вихідне відношення в точності в попередньому вигляді. Оскільки при відновленні вихідного відношення з'єднанням проекцій не повинні з'явитися нові атрибути, то як операцію, що зворотна проекції, необхідно використовувати природне з'єднання (JOIN).

Проекція $R[X]$ відношення R на множині атрибутів X називається *власною проекцією*, якщо множина атрибутів X є власною підмножиною множини атрибутів відношення R (тобто множина атрибутів X не збігається з множиною всіх атрибутів відношення R).

Власні проекції R_1 і R_2 відношення R називаються *декомпозицією без втрат*, якщо відношення R точно відновлюється з них за допомогою природного з'єднання для будь-якого стану відношення, тобто:

$$R_1 \text{ JOIN } R_2 = R \,. \qquad (17)$$

Для ілюстрації цього розглянемо приклад. Нехай дано відношення R, яке зображене в табличному вигляді в таблиці 3.7.

Таблиця 3.7 – Відношення R

Номер	Прізвище	Зарплата
1	Іванов	1000
2	Петров	1000

Зробимо будь-як перший варіант декомпозиції, розбивши вихідне відношення, не покладаючись на наявні функціональні залежності (таблиці 3.8, 3.9).

Таблиця 3.8 – Відношення R_1

Номер	**Зарплата**
1	1000
2	1000

Таблиця 3.9 – Відношення R_2

Прізвище	**Зарплата**
Іванов	1000
Петров	1000

Така декомпозиція не є декомпозицією без втрат, оскільки вихідне відношення не відновлюється в точному вигляді по проекціям. Табличне зображення результату виконання операції природного з'єднання R_1 JOIN R_2 дано в таблиці 3.10. Нахилом позначено кортежі, яких не було у вихідному відношенні R, але які з'являються у результуючому відношенні при спробі відновлення.

Таблиця 3.10 – Відношення R_1 JOIN $R_2 \neq R$

Номер	**Прізвище**	**Зарплата**
1	Іванов	1000
1	*Петров*	*1000*
2	*Іванов*	*1000*
2	Петров	1000

Зауваження: поява нових кортежів, яких не було у вихідному відношенні, також є втратою інформації, оскільки це невідповідність поточному стану предметної області, яку внесено в базу даних неправильною декомпозицією!

Розглянемо інший варіант декомпозиції відношення R (таблиці 3.11, 3.12).

Таблиця 3.11 – Відношення R_1

Номер	**Прізвище**
1	Іванов
2	Петров

Таблиця 3.12 – Відношення R_2

Номер	**Зарплата**
1	1000
2	1000

Тепер здається, що початкове відношення *R* відновлюється в точному первісному вигляді. Це помилково! Не можна сказати, що дана декомпозиція є декомпозицією без втрат, оскільки розглянутий тільки один конкретний стан відношення *R*, і неможливо сказати, чи буде і в інших станах відношення *R* відновлюватися точно.

Нехай відношення *R* перейшло в стан, що зображено в таблиці 3.13. Важливим є те, що значення в атрибуті **НОМЕР** повторюються, оскільки нічого не було сказано про ключі цього відношення і взагалі про додаткові обмеження типу унікальності в моделі предметної області!

Таблиця 3.13 – Відношення *R*

Номер	**Прізвище**	**Зарплата**
1	Іванов	1000
2	Петров	1000
2	Сидоров	2000

Зробимо ту ж саму декомпозицію (таблиці 3.14, 3.15).

Таблиця 3.14 – Відношення R_1

Номер	**Прізвище**
1	Іванов
2	Петров
2	Сидоров

Таблиця 3.15 – Відношення R_2

Номер	**Зарплата**
1	1000
2	1000
2	2000

Спробуємо відновити вихідне відношення R, виконавши операцію природного з'єднання R_1 JOIN R_2. Результат виконання цієї операції дано в таблиці 3.16. Так само як і у попередньому випадку нахилом позначено кортежі, яких не було у вихідному відношенні *R*, але які з'являються в результуючому відношенні при спробі відновлення.

Тому така декомпозиція також не є декомпозицією без втрат!

Таблиця 3.16 – Відношення R_1 JOIN $R_2 \neq R$

Номер	**Прізвище**	**Зарплата**
1	Іванов	1000
1	Петров	1000
2	*Петров*	*2000*
2	*Сидоров*	*1000*
2	Сидоров	2000

Зауваження: знову поява нових кортежів, яких не було у вихідному відношенні, є втратою інформації, оскільки це невідповідність поточному стану предметної області, яку внесено в базу даних неправильною декомпозицією!

Висновок: без додаткових обмежень на відношення *R* не можна говорити про декомпозиції без втрат.

Такими додатковими обмеженнями є функціональні залежності.

Теорема Хеза. Нехай $R(A,B,C)$ є відношенням, а A,B,C – атрибути або множини атрибутів цього відношення. Якщо є функціональна залежність $A \rightarrow B$, то проекції $R_1=R[A,B]$ і $R_2=R[A,C]$ утворюють декомпозицію без втрат.

Основний сенс теореми Хеза – при декомпозиції і наступному відновленні відношення не з'являться нові кортежі, відсутні у вихідному відношенні.

Ця теорема доводить, що саме декомпозиція відношень на основі функціональних залежностей (а не будь-як!), які випливають з моделі предметної області (а не поточного стану предметної області!), гарантує відповідність інформації в базі даних предметній області.

Саме тому правильним варіантом розподілу атрибутів по відношенням, тобто правильним варіантом побудови логічної моделі реляційної бази даних, є саме виявлення потенційних ключів відношень і функціональних залежностей у відношеннях з подальшим застосуванням процедури нормалізації відношень. Розподіл атрибутів по відношенням будь-як може призвести до внесення ще на етапі побудови логічної моделі бази даних передумов для втрати цілісності бази даних та втрати інформації, з якими дуже часто неможливо впоратись при побудові фізичної моделі БД за допомогою СУБД.

Але декомпозиція відношень на основі функціональних залежностей з цієї точки зору не завжди дозволяє досягти мети. Іноді модель предметної області вимагає перехід до нормальних форм більш високих порядків, як окремих, а не загальних ситуацій, які виникають тільки в окремих випадках.

3.15. Нормальні форми високих порядків

При побудові логічної моделі реляційної бази даних за допомогою нормалізації відношень можуть виникати особливі ситуації, які не дають змоги усунути аномалії оновлення лише на підставі виявлення функціональних залежностей та застосування теореми Хеза, тобто доведення відношень до 3НФ. Для моделювання БД у разі виникнення таких особливих ситуацій застосовуються нормальні форми високих порядків, відповідність яким перевіряється при різних умовах. До таких нормальних форм високих порядків належать:

- нормальна форма Бойса-Кодда (НФБК) – виникає необхідність перевірки та доведення до цієї форми за допомогою декомпозиції у разі наявності у відношенні декількох незалежних потенційних ключів;
- четверта нормальна форма (4НФ) – виникає необхідність перевірки та доведення до цієї форми за допомогою декомпозиції у разі наявності у відношенні нетривіальних багатозначних залежностей;
- п'ята нормальна форма (5НФ) – виникає необхідність перевірки та доведення до цієї форми за допомогою декомпозиції у разі наявності у відношенні нетривіальних залежностей з'єднання.

Всі зазначені поняття та алгоритми доведення то тієї чи іншої з перелічених нормальних форм розглядаються далі з окремими прикладами.

3.16. Нормальна форма Бойса-Кодда (НФБК)

При приведенні відношень за допомогою алгоритму нормалізації до відношень в 3НФ неявно передбачалося, що всі відношення містять один потенційний ключ. Це не завжди вірно. Якщо у відношенні, виходячи з моделі предметної області, виявлено декілька потенційних ключів, то виникає необхідність перевірки на відповідність нормальній формі Бойса-Кодда, а при невідповідності, доведення до цієї нормальної форми.

Відношення *R* знаходиться в *нормальній формі Бойса-Кодда* (НФБК або BCNF) тоді і тільки тоді, коли детермінанти *всіх* функціональних залежностей є потенційними ключами.

Зауваження: якщо відношення знаходиться в НФБК, то воно автоматично знаходиться і в 3НФ.

Приведення до НФБК відбувається наступним чином. Щоб усунути залежність від детермінантів, які не є потенційними ключами, необхідно провести декомпозицію, виносячи ці детермінанти і залежні від них атрибути в окреме

відношення, при цьому залишаючи детермінанти і у вихідному відношенні для зв'язку відношень.

Визначення НФБК не потребує жодних попередніх умов щодо знаходження відношення в нижчих нормальних формах (2НФ та 3НФ). Якщо проводити нормалізацію послідовно, то в переважній більшості випадків при досягненні 3НФ автоматично будуть задовольнятися вимоги НФБК. 3НФ не збігається з НФБК лише тоді, коли одночасно виконуються такі 3 умови:

1. Відношення має більше одного потенційного ключа;
2. Ці потенційні ключі складені з більше ніж одного атрибута;
3. Ці потенційні ключі перекриваються, тобто мають щонайменше один спільний атрибут.

Розглянемо приклад відношення, що містить два таких ключа. Нехай необхідно зберігати інформацію про номер співробітника згідно зі штатним розкладом (**НСПІВР**), його прізвище (**ПРІЗВ**), номер проекту, над яким він працює (**НПРО**) та номер завдання (**НЗАВД**), причому, як і раніше, поставимо умову, що кожен співробітник може брати участь в одному чи кількох проектах, або тимчасово не брати участь в жодних проектах. Проте, особистість співробітника може повністю визначатися як його номером, так і прізвищем, тобто. не може бути двох співробітників з однаковим прізвищем (дана ситуація не завжди відповідає дійсності, а вибирається тільки як приклад).

Будемо зберігати дані у відношенні: **СПІВРОБІТНИКИ_ПРОЕКТИ (НСПІВР, ПРІЗВ, НПРО, НЗАВД)**. Поточний стан предметної області можна відобразити у вигляді таблиці 3.17.

Таблиця 3.17 – Відношення СПІВРОБІТНИКИ_ПРОЕКТИ

НСПІВР	**ПРІЗВ**	**НПРО**	**НЗАВД**
1	Іванов	2	1
2	Петров	1	1
3	Сидоров	2	2

Як первинний потенційний ключ даного відношення можна вибрати одну з пар атрибутів: **{НСПІВР, НПРО}** або **{ПРІЗВ, НПРО}**.

Розглянемо всі функціональні залежності:

1. Залежності від ключів відношень:

{НСПІВР, НПРО}→ПРІЗВ;

{НСПІВР, НПРО}→НЗАВД;

{ПРІЗВ, НПРО}→НСПІВР;

{ПРІЗВ, НПРО}→НЗАВД;

2. Інші залежності:

НСПІВР→ПРІЗВ;

НСПІВР→НПРО;

ПРІЗВ→НСПІВР;

ПРІЗВ→НПРО.

Це відношення не містить неключових атрибутів, що залежать від частини складеного ключа (див. визначення 2НФ, п. 3.9). Таким чином, відношення знаходиться у 2НФ. З іншого боку, відношення не містить залежних один від одного неключових атрибутів, оскільки неключовий атрибут лише один – **НЗАВД** (див. визначення 3НФ, п. 3.11). Таким чином, показано, що відношення, що розглядається (табл. 3.17), знаходиться в 3НФ.

Однак, той факт, що є функціональні залежності атрибутів відношення від атрибуту, що є частиною первинного ключа, призводить до аномалій. Наприклад, зміна прізвища співробітника з цим номером, узгодженим чином, вимагатиме модифікувати всі кортежі, що включають його номер.

Висновок – логічна модель даних неадекватна моделі предметної області, тому база даних, заснована на такій моделі, буде працювати неправильно.

Для усунення цих аномалій відношення слід привести до НФБК, тобто. усунути залежність від детермінантів, які не є потенційними ключами і провести декомпозицію, виносячи ці детермінанти та залежні від них частини в окреме відношення. Відношення **СПІВРОБІТНИКИ** (табл. 3.18) та **ПРОЕКТИ** (табл. 3.19), отримані внаслідок декомпозиції, перебувають у НФБК.

У відношенні **СПІВРОБІТНИКИ** можливими ключами є **НСПІВР** та **ПРІЗВ**, а наявні функціональні залежності:

НСПІВР→ПРІЗВ та **ПРІЗВ→НСПІВР**.

Таблиця 3.18 – Відношення СПІВРОБІТНИКИ

НСПІВР	**ПРІЗВ**
1	Іванов
2	Петров
3	Сидоров

У відношенні **ПРОЕКТИ** ключем є пара атрибутів – **{НСПІВР, НПРО}**, а єдина функціональна залежність:

{НСПІВР, НПРО}→НЗАВД.

Таблиця 3.19 – Відношення ПРОЕКТИ

НСПІВР	НПРО	НЗАВД
1	2	1
2	1	1
3	2	2

Тепер після декомпозиції обидва відношення знаходиться в НФБК, оскільки детермінанти всіх залежностей є потенційними ключами.

3.17. Четверта нормальна форма (4НФ)

Деякі відношення можуть не мати функціональних залежностей, але в них можуть бути присутні аномалії оновлення, і тому вони потребують декомпозиції. В таких окремих випадках декомпозиція може базуватися на понятті нетривіальної багатозначної залежності, якщо така є у цьому відношенні.

Нехай R – відношення, а X,Y,Z – деякі з його атрибутів (або непересічні множини атрибутів). Тоді атрибути (множини атрибутів) Y і Z *багатозначно залежать* від X (позначається $X\rightarrow\rightarrow Y|Z$) тоді і тільки тоді, коли з того, що у відношенні R містяться кортежі $r_1=(x,y,z_1)$ і $r_2=(x,y_1,z)$ випливає, що у відношенні R міститься також і кортеж $r_3=(x,y,z)$.

Зауваження: 1. У відношенні R міститься також і кортеж $r_4=(x,y_1,z_1)$, тобто атрибути Y і Z, багатозначно залежать від X, поводяться «симетрично» відносно атрибуту X; 2. Якщо у відношенні R є не менше трьох атрибутів X,Y,Z і є функціональна залежність $X\rightarrow Y$, то є і багатозначна залежність $X\rightarrow\rightarrow Y|Z$.

Розглянемо останнє твердження з урахуванням визначення багатозначної залежності атрибутів. Якщо у відношенні є функціональна залежність $X\rightarrow Y$, тоді у згаданих у визначенні кортежах буде рівність значень атрибутів $y=y_1$, що відповідають x. Це означає, що кортеж $r_3=(x,y,z)$ співпадає з кортежем $r_2=(x,y_1,z)$ і міститься у відношенні R. Значить у відношенні є і багатозначна залежність $X\rightarrow\rightarrow Y|Z$.

З огляду на це *поняття багатозначної залежності є узагальненням поняття функціональної залежності*.

Багатозначна залежність $X \rightarrow\rightarrow Y|Z$ називається *нетривіальною багатозначною залежністю*, якщо у відношенні не існує функціональних залежностей $X \rightarrow Y$ і $X \rightarrow Z$.

Оскільки у таких відношеннях немає функціональних залежностей, то не можна скористатися теоремою Хеза для декомпозиції.

Теорема (Фейджина). Нехай X, Y, Z – неперетинні множини атрибутів відношення $R(X,Y,Z)$. Декомпозиція відношення R на проекції $R_1=R[X,Y]$ і $R_2=R[X,Z]$ буде декомпозицією без втрат тоді і тільки тоді, коли є багатозначна залежність $X \rightarrow\rightarrow Y|Z$.

Зауваження: якщо залежність $X \rightarrow\rightarrow Y|Z$ є тривіальною, тобто існує одна з функціональних залежностей $X \rightarrow Y$ або $X \rightarrow Z$, то отримуємо теорему Хеза.

Відношення R знаходиться в *четвертій нормальній формі* (4НФ або 4NF) тоді і тільки тоді, коли відношення знаходиться в НФБК і не містить нетривіальних багатозначних залежностей.

Розглянемо приклад: нехай необхідно зберігати інформацію про співробітників з унікальними номерами (**НСПІВР**), які працюють над проектами з номерами (**НПРО**) і виконують в цих проектах завдання з номерами (**НЗАВД**). Причому модель предметної області складають наступні факти:

1. Кожен співробітник має право працювати у кількох проектах;
2. Кожен проект має свій список завдань, що в ньому виконуються;
3. Номери завдань унікальні тільки в межах кожного проекту і тому можуть повторюватися в різних проектах;
4. Якщо співробітник долучається до проекту, то він виконує всі завдання, що передбачаються цим проектом.

Будемо зберігати дані у відношенні:

ПРОЕКТИ (<u>НПРО</u>, <u>НСПІВР</u>, <u>НЗАВД</u>).

Оскільки згідно з моделлю предметної області не можна скоротити кількість атрибутів, що мають властивість унікальності, то це є повністю ключове відношення з потенційним ключем **{<u>НПРО</u>, <u>НСПІВР</u>, <u>НЗАВД</u>}**.

Кожен кортеж цього відношення пов’язує деякий проект із співробітником, який бере участь у цьому проекті, та кожним завданням в рамках цього проекту (оскільки передбачається, що будь-який співробітник, який бере участь у проекті, виконує всі завдання, передбачені цим проектом). Нехай поточний стан предметної області є таким, що відображений у вигляді таблиці 3.20.

Тобто є два проекти – в першому три завдання, у другому два завдання, і перший співробітник працює тільки в першому проекті, а другий – в обидвох проектах.

Таблиця 3.20 – Відношення ПРОЕКТИ

НПРО	**НСПІВР**	**НЗАВД**
1	1	1
1	1	2
1	1	3
1	2	1
1	2	2
1	2	3
2	2	1
2	2	2

Через сформульовані вище умови визначено, що єдиним можливим ключем відношення є складений ключ **{НПРО, НСПІВР, НЗАВД}** і немає жодних інших детермінантів. Отже, відношення **ПРОЕКТИ** перебуває у НФБК. Але у відношенні є аномалії.

1. Аномалії вставки (INSERT).

При додаванні у відношення **ПРОЕКТИ** кортежу (2, 3, 1) необхідно додати кортеж (2, 3, 2), інакше БД не буде відповідати моделі предметної області (див. п. 4 моделі предметної області – якщо деякий співробітник приєднується до цього проекту, необхідно вставити у відношення **ПРОЕКТИ** стільки кортежів, скільки завдань у ньому передбачено).

2. Аномалії оновлення (UPDATE).

Дублюються номери проектів, співробітників та завдань.

3. Аномалії видалення (DELETE).

При видаленні у відношенні **ПРОЕКТИ** кортежу (2, 2, 1) необхідно видалити кортеж (2, 2, 2), а при узгодженому видаленні у відношенні **ПРОЕКТИ** кортежів (2, 2, 1) та (2, 2, 2) відбудеться втрата інформації про завдання, що мають виконуватися в межах другого проекту, хоча проект не закритий (просто на нього не розподілено співробітників).

Зауваження: 1. Декомпозиція відношення **ПРОЕКТИ** не може бути виконана на основі функціональних залежностей, оскільки їх у відношенні немає; 2. Це відношення повністю ключове, а взаємозв’язок описується поняттям багатозначною залежності.

У відношенні **ПРОЕКТИ** існує така багатозначна залежність:

НПРО→→НСПІВР|НЗАВД.

Для усунення перелічених вище недоліків (аномалій) повинна бути проведена нормалізація відношення **ПРОЕКТИ**, яка ґрунтується на цій багатозначній залежності та теоремі Фейджина, тобто це декомпозиція на два відношення – **ПРОЕКТИ_СПІВРОБІТНИКИ (НПРО, НСПІВР)** (табл. 3.21) та **ПРОЕКТИ_ЗАВДАННЯ (НПРО, НЗАВД)** (табл. 3.22).

Таблиця 3.21 – Відношення ПРОЕКТИ_СПІВРОБІТНИКИ

НПРО	НСПІВР
1	1
1	2
2	2

Таблиця 3.22 – Відношення ПРОЕКТИ_ЗАВДАННЯ

НПРО	НЗАВД
1	1
1	2
1	3
2	1
2	2

Таким чином, згідно з теоремою Фейджина відношення **ПРОЕКТИ** можна без втрат декомпозувати на два відношення з усуненням знайдених аномалій оновлення. Отримані відношення перебувають у 4НФ.

Зауваження: отримані відношення є повністю ключовими, і в них, як і раніше, немає функціональних залежностей.

3.18. П’ята нормальна форма (5НФ)

Функціональні та багатозначні залежності дозволяють зробити декомпозицію вихідного відношення без втрат на *дві* проекції. Можна, однак, навести приклади відношень, які не можна декомпозувати без втрат на жодні дві проекції, але в них можуть бути аномалії оновлення і тому вони потребують декомпозиції. В таких окремих випадках декомпозиція може базуватися на понятті нетривіальної залежності з’єднання, якщо така є у цьому відношенні.

Нехай R є відношенням, а $A,B,...,Z$ – довільними (можливо пересічними) підмножинами множини атрибутів відношення R. Тоді відношення R задовольняє *залежності з'єднання* $*(A,B,...,Z)$ тоді і тільки тоді, коли воно рівносильне з'єднанню всіх своїх проекцій з підмножинами атрибутів $A,B,...,Z$, тобто $R=R[A]$ JOIN $R[B]$ JOIN... JOIN $R[Z]$.

Зауваження: залежність з'єднання є узагальненням поняття багатозначної залежності.

Згідно з теоремою Фейджина, відношення $R(X,Y,Z)$ може бути декомпозовано без втрат на проекції $R_1=R[X,Y]$ та $R_2=R[X,Z]$ тоді і тільки тоді, коли є багатозначна залежність $X\rightarrow\rightarrow Y|Z$.

Теорема Фейджина (інше формулювання). Відношення $R(X,Y,Z)$ задовольняє залежності з'єднання $*(XY,XZ)$ тоді і тільки тоді, коли є багатозначна залежність $X\rightarrow\rightarrow Y|Z$.

Зауваження: багатозначна залежність є окремим випадком залежності з'єднання, тобто, якщо у відношенні є багатозначна залежність, то є і залежність з'єднання.

Залежність з'єднання $*(A,B,...,Z)$ називається *нетривіальною залежністю з'єднання*, якщо виконується дві умови:

1. Одна з множин атрибутів $A,B, ...,Z$ не містить потенційного ключа відношення R;
2. Жодна з множин атрибутів не збігається з усією множиною атрибутів відношення R.

Залежність з'єднання $*(A,B, ...,Z)$ називається *тривіальною залежністю з'єднання*, якщо виконується одна з умов:

1. Або всі множини атрибутів $A,B, ...,Z$ містять потенційний ключ відношення R;
2. Або одна з множин атрибутів збігається з усією множиною атрибутів відношення R.

Відношення R знаходиться в *п'ятій нормальній формі* (5НФ або 5NF) тоді і тільки тоді, коли будь-яка наявна залежність з'єднання є тривіальної, або навпаки, відношення R не знаходиться в 5НФ, якщо у відношенні знайдеться нетривіальна залежність з'єднання.

Зауваження: не знаючи нічого про те, які потенційні ключі є у відношенні і як взаємопов'язані атрибути, не можна робити висновки про те, чи знаходиться дане відношення в 5НФ (як, втім, і в інших нормальних формах).

Розглянемо як приклад відношення, яке не можна декомпозувати без втрат на жодні дві проекції. Сформулюємо модель предметної області таким чином. Нехай один і той самий співробітник може працювати в декількох відділах і працювати в кожному відділі над декількома (не всіма) проектами.

Відношення, що містить інформацію про унікальні номери співробітників (**НСПІВР**), відділів (**НВІД**), в яких вони працюють, і проектів (**НПРО**), які вони виконують:

СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ (НСПІВР, НВІД, НПРО).

Поточний стан предметної області показано у вигляді таблиці 3.23.

Таблиця 3.23 – Відношення СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ

НСПІВР	**НВІД**	**НПРО**
1	1	2
1	2	1
2	1	1
1	1	1

Первинним ключем цього відношення є повна сукупність його атрибутів **{НСПІВР, НВІД, НПРО}**, відсутні функціональні та багатозначні залежності. Тому відношення перебуває у 4НФ, але при додаванні або видаленні кортежів у відношенні можуть виникнути проблеми, тобто існують аномалії, які можна усунути шляхом декомпозиції вихідного відношення на три нових відношення.

Розглянемо всілякі проекції ([**НСПІВР**, **НВІД**], [**НСПІВР**, **НПРО**] та [**НВІД**, **НПРО**]) (див. п. 2.3.3) відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ**, що включають по два атрибута.

Отримані відношення мають відповідний зазначеним проекціям вигляд, що зображено в таблицях 3.24, 3.25 та 3.26.

Таблиця 3.24 – Відношення СПІВРОБІТНИКИ_ВІДДІЛИ

НСПІВР	**НВІД**
1	1
1	2
2	1

Таблиця 3.25 – Відношення СПІВРОБІТНИКИ_ПРОЕКТИ

НСПІВР	**НПРО**
1	2
1	1
2	1

Таблиця 3.26 – Відношення ВІДДІЛИ_ПРОЕКТИ

НВІД	**НПРО**
1	2
2	1
1	1

Відношення **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ** не відновлюється по жодному з попарних з'єднань:

СПІВРОБІТНИКИ_ВІДДІЛИ JOIN СПІВРОБІТНИКИ_ПРОЕКТИ,
СПІВРОБІТНИКИ_ВІДДІЛИ JOIN ВІДДІЛИ_ПРОЕКТИ,
СПІВРОБІТНИКИ _ПРОЕКТИ JOIN ВІДДІЛИ_ПРОЕКТИ.

Це легко перевірити, пригадавши принцип виконання операції природного з'єднання (див. п. 2.3.3). Наприклад, природне з'єднання проекцій **СПІВРОБІТНИКИ_ВІДДІЛИ JOIN СПІВРОБІТНИКИ_ПРОЕКТИ** дає результат, зображений в таблиці 3.27, який не співпадає з вихідним відношенням **СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ** (нахилом позначено кортежі, яких не було у вихідному відношенні, але які з'являються у результуючому відношенні при спробі відновлення).

Таблиця 3.27 – Відношення СПІВРОБІТНИКИ_ВІДДІЛИ JOIN СПІВРОБІТНИКИ_ПРОЕКТИ

НСПІВР	**НВІД**	**НПРО**
1	1	2
1	1	1
1	*2*	*2*
1	2	1
2	1	1

Те ж саме виникає при спробі відновити вихідне відношення по іншим попарним з'єднанням, але вихідне відношення відновлюється при з'єднанні

всіх трьох попарних проекцій, тобто таким чином:

СПІВРОБІТНИКИ_ВІДДІЛИ JOIN СПІВРОБІТНИКИ_ПРОЕКТИ JOIN ВІДДІЛИ_ПРОЕКТИ.

Це відповідає визначенню залежності з'єднання. Логічна модель, що складається з декомпозованих саме таким чином відношень, вільна від можливих аномалій, а самі відношення знаходяться в 5НФ.

3.19. Продовження алгоритму нормалізації (приведення до 5НФ)

Крок 4 – приведення до НФБК.

Якщо є відношення, що містять кілька потенційних ключів, то необхідно перевірити, чи є функціональні залежності, детермінанти яких не є потенційними ключами. Якщо такі функціональні залежності є, то необхідно провести подальшу декомпозицію відношення.

Ті атрибути, що залежать від детермінантів, які не є потенційними ключами, виносяться в окреме відношення разом з детермінантами, але детермінанти залишаються і у вихідному відношенні для забезпечення зв'язку між кортежами цих відношень через первинний та зовнішній ключі.

Крок 5 – приведення до 4НФ.

Якщо у відношеннях виявлені нетривіальні багатозначні залежності, то необхідно провести декомпозицію для виключення таких залежностей, яка ґрунтується на теоремі Фейджина. Тобто ті атрибути, що багатозначно залежать від іншого атрибута (детермінанта) виносяться кожен в окреме відношення разом з детермінантом. У цьому випадку детермінант є в кожному з цих відношень і забезпечує зв'язок кортежів цих відношень.

Алгоритм приведення до 4НФ схематизовано можна записати наступним чином. Нехай є такі вихідні дані:

- початкове відношення – $R(X,Y,Z)$, де X, Y, Z – атрибути або неперетинні множини атрибутів;
- багатозначна залежність – $X \rightarrow\rightarrow Y|Z$.

Отримані декомпозовані відношення:

- $R_1=R[X,Y]$ і $R_2=R[X,Z]$.

Крок 6 – приведення до 5НФ.

Якщо у відношеннях виявлені нетривіальні залежності з'єднання, то необхідно провести декомпозицію для виключення і таких залежностей.

3.20. Додаткові визначення нормальних форм високих порядків

Додаткові нормальні форми – це по суті розширення розглянутих понять.

Доменно-ключова нормальна форма (ДКНФ) вимагає, аби у схемі не було інших обмежень окрім ключів та доменів.

Шоста нормальна форма визначає, що відношення знаходиться у цій формі, якщо воно знаходиться у 5НФ та задовольняє вимозі відсутності будь-яких нетривіальних залежностей (зазвичай 6НФ ототожнюють з ДКНФ).

3.21. Аналіз критеріїв для різного ступеня нормалізованих моделей

Порівняння за чотирма основними критеріями для нормалізованих і ненормалізованих моделей зведено в таблицю 3.28.

Таблиця 3.28 – Порівняння нормалізованих і ненормалізованих моделей

КРИТЕРІЙ	Відношення слабо нормалізовані (1НФ, 2НФ)	Відношення сильно нормалізовані (3НФ та вище)
1. Адекватність БД предметній області	ГІРШЕ (-)	КРАЩЕ (+)
2. Легкість розробки і супроводу БД	СКЛАДНІШЕ (-)	ЛЕГШЕ (+)
3. Швидкість операцій вставки, оновлення, видалення	ПОВІЛЬНІШЕ (-)	ШВИДШЕ (+)
4. Швидкість операцій вибірки даних	ШВИДШЕ (+)	ПОВІЛЬНІШЕ (-)

Використання слабо та сильно нормалізованих відношень:

- сильно нормалізовані моделі даних добре підходять для так званих OLTP-додатків (On-Line Transaction Processing – оперативна обробка транзакцій), наприклад, системи складського обліку, замовлень квитків, банківські системи, що виконують операції з переказу грошей, тобто основна функція подібних систем полягає у виконанні великої кількості коротких транзакцій.

- слабо нормалізовані моделі даних підходять для OLAP-додатків (On-Line Analytical Processing – оперативна аналітична обробка даних), що оперують заздалегідь обчисленими основними підсумковими даними, тобто великими масивами даних, вже накопиченими в OLTP-додатках, наприклад, для проведення аналізу «що якщо... », при цьому додавання у систему нових даних відбувається відносно рідко, і ці дані зазвичай ніколи не видаляються, але цілісність даних відслідковується за допомогою додаткових дій.

4. МОДЕЛЬ «СУТНІСТЬ-ЗВ'ЯЗОК» (ER-ДІАГРАМИ)

4.1. Семантичне моделювання

Моделювання за допомогою алгоритму нормалізації має недоліки:

1. Первісне розміщення всіх атрибутів в одному відношенні є неприродною операцією, особливо якщо цих атрибутів дуже багато;
2. Неможливо відразу визначити повний список атрибутів, а якщо потрібно щось буде додати, то необхідно буде повертатися до 1НФ;
3. Для проведення процедури нормалізації необхідно виділити залежності атрибутів на основі моделі предметної області, що не завжди є очевидним на початку розробки БД.

Іншим методом логічного моделювання є семантичне моделювання, тобто моделювання структури даних, спираючись на зміст цих даних. Це може виконуватись за допомогою різних варіантів ER-діаграм або ER-моделей (Entity-Relationship, тобто діаграм або моделей «сутність-зв'язок»).

Взагалі існує дуже багато різних моделей для концептуального представлення даних з метою збереження. *Модель «сутність-зв'язок» (ER-модель)* – це така мета-модель даних або засіб опису даних, що дозволяє описувати концептуальні або логічні схеми даних за допомогою узагальнених конструкцій блоків. Вона є зручним інструментом уніфікованого наочного представлення та уявлення даних незалежно від далі використовуваних програмних засобів або СУБД. Головною перевагою моделі «сутність-зв'язок» є той факт, що з неї можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна тощо). Саме тому вона є найзагальнішою моделлю.

Модель «сутність-зв'язок» є результатом систематичного процесу, який описує та визначає певну предметну область. Вона не визначає сам процес, а лише дає можливість візуалізувати його.

Суттю цього підходу до моделювання даних є те, що дані представляються у вигляді компонентів (сутностей), які пов'язані між собою певними зв'язками. Ці зв'язки виражають залежності та вимоги між ними. Сутності можуть мати різні властивості (атрибути), які характеризують їх. Діаграми, що створюються для графічного зображення цих сутностей, їх атрибутів та зв'язків, називають *діаграмами «сутність-зв'язок»*.

Отже, модель «сутність-зв'язок» або ER-діаграма є зручним способом створення логічних моделей і для реляційних баз даних.

4.2. Основні поняття ER-діаграм

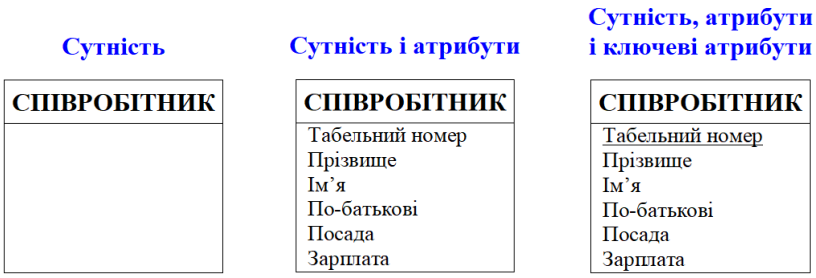
Розглянемо методику побудови, основні поняття та позначення одного із способів створення ER-діаграм, тобто моделей «сутність-зв'язок».

Сутність – це клас однотипних об'єктів, інформація про які повинна бути врахована в моделі.

Зауваження: кожна сутність повинна мати найменування, виражене іменником в однині.

Приклад сутності: класи об'єктів «Співробітник», «Постачальник».

Сутність в моделі зображується у вигляді прямокутника з найменуванням, як наведено на рис. 4.1а.



а) б) в)

Рисунок 4.1 – Схематичне зображення опису елементів даних у вигляді ER-діаграми: а) сутність; б) сутність з атрибутами; в) сутність з атрибутами та ключем

Екземпляр сутності – це конкретний представник даної сутності.

Зауваження: екземпляри сутностей повинні бути відмінні один від одного, тобто сутності повинні мати деякі властивості, унікальні для кожного екземпляра цієї сутності.

Приклад: сутність «Співробітник», екземпляр сутності «Співробітник Іванов».

Атрибут сутності – це іменована характеристика, що є деякою властивістю сутності (зображуються в межах прямокутника, що визначає сутність).

Зауваження: найменування атрибута повинно бути виражено іменником в однині (можливо, з прикметниками, що його характеризують).

Приклад атрибутів сутності: сутність «Співробітник», атрибути «Табельний номер», «Прізвище», «Ім'я», «По батькові», «Посада», «Зарплата», як наведено на рис. 4.1б.

Ключ сутності – це ненадлишковий набір атрибутів, значення яких в сукупності є унікальними для кожного екземпляра сутності (зображуються на діаграмі підкресленням).

Зауваження: 1. Ненадлишковість полягає у тому, що видалення будь-якого атрибута з ключа порушує його унікальність; 2. Сутність може мати кілька різних ключів.

Приклад позначення ключа сутності надано на рис. 4.1в.

Зв'язок – це деяка асоціація між двома сутностями. Одна сутність може бути пов'язана з іншого сутністю або сама з собою (зображується лінією, що з'єднує дві сутності).

Зауваження: 1. Зв'язки дозволяють по одній сутності знаходити інші сутності, пов'язані з нею; 2. Кожен зв'язок має два кінця і одне або два найменування. Найменування зазвичай виражається в невизначеній дієслівній формі: «мати», «належати» тощо. Кожне з найменувань ставиться до свого кінця зв'язку.

Приклади зв'язків сутностей: «СПІВРОБІТНИК може мати кілька ДІТЕЙ», «кожен СПІВРОБІТНИК зобов'язаний числитися рівно в одному ВІДДІЛІ».

Приклад зображення зв'язку між сутностями надано на рис. 4.2.

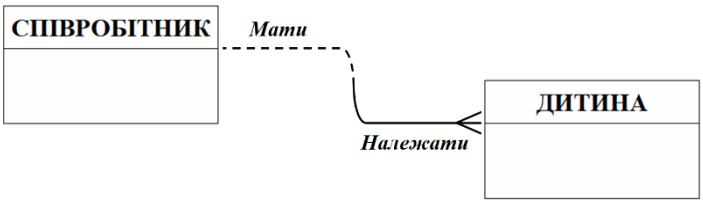


Рисунок 4.2 – Схематичне зображення зв'язку між двома сутностями на ER-діаграмі

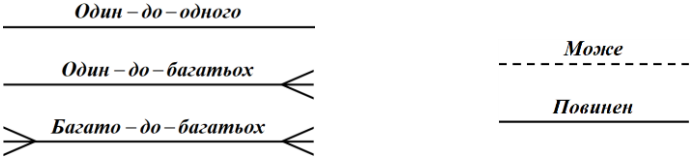
Існує декілька *типів зв'язків* між сутностями:

1. Зв'язок типу *один-до-одного* означає, що один екземпляр першої сутності (лівої) пов'язаний з одним екземпляром другої сутності (правої). Зв'язок один-до-одного найчастіше свідчить про те, що є всього одна сутність, неправильно поділена на дві;
2. Зв'язок типу *один-до-багатьох* означає, що один екземпляр першої суті (лівої) пов'язаний з декількома екземплярами другої суті (правої).

Це найбільш часто використовуваний тип зв'язку. Ліва сутність (зі сторони «один») називається *батьківською*, права (зі сторони «багато») – *дочірньою*;

3. Зв'язок типу *багато-до-багатьох* означає, що кожен екземпляр першої сутності може бути пов'язаний з декількома екземплярами другої сутності, і навпаки (є тимчасовим і повинен бути замінений двома зв'язками типу один-до-багатьох шляхом створення проміжної сутності).

Приклад зображення типів зв'язків між сутностями надано на рис. 4.3а.



а) б)

Рисунок 4.3 – Схематичне зображення елементів зв'язків на ER-діаграмі: а) типів; б) модальності

Існує також декілька *модальностей зв'язків* між сутностями:

1. Модальність «може» означає, що екземпляр однієї сутності може бути пов'язаний з одним або декількома екземплярами іншої сутності, а може бути і не пов'язаний з жодним екземпляром;
2. Модальність «повинен» означає, що екземпляр однієї сутності зобов'язаний бути пов'язаний не менше ніж з одним екземпляром іншої сутності.

Зв'язок може мати різну модальність з різних кінців, як показано на рис. 4.2.

Графічний синтаксис дозволяє однозначно читати діаграми, користуючись наступною схемою побудови фраз:

<Кожен *екземпляр* СУТНОСТІ 1> <МОДАЛЬНІСТЬ ЗВ'ЯЗКУ> <НАЗВА ЗВ'ЯЗКУ> <ТИП ЗВ'ЯЗКУ> <*екземпляр* СУТНОСТИ 2>.

Зв'язок може бути прочитаний як зліва направо, так і справа наліво.

Наприклад: зліва направо – «Кожен співробітник може мати кілька дітей»; справа наліво – «Кожна дитина має належати рівно одному співробітнику».

4.3. Приклад розробки ER-діаграми для реляційної бази даних

Для розуміння процедури застосування моделі «сутність-зв'язок» з побудовою ER-діаграми розглянемо послідовність дій при проектуванні логічної моделі даних для типового прикладу з предметною областю «Оптова фірма» з метою реалізації реляційної бази даних.

При розробці ER-моделей треба отримати наступну інформацію про предметну область:

1. *Список сутностей* предметної області.
2. *Список атрибутів* сутностей.
3. *Опис взаємозв'язків* між сутностями.

ER-діаграми зручні тим, що процес виділення сутностей, атрибутів і зв'язків є ітераційним. Розробивши перший наближений варіант діаграм, можна уточнювати їх, опитуючи експертів предметної області. При цьому документацією, в якій фіксуються результати бесід, є самі ER-діаграми.

Розглянемо основні етапи розробки ER-діаграми на зазначеному прикладі з метою зберігання інформації, необхідної при функціонуванні оптової торгівельної організації.

Етап 1. Предметна область.

Необхідно розробити інформаційну систему на замовлення деякої оптової торгової фірми.

Етап 2. Модель предметної області.

При створенні моделі предметної області потрібно зібрати уявлення майбутніх користувачів, тому необхідно:

- опитати співробітників фірми;
- вивчити документацію, форми замовлень і накладних;
- тощо.

Наприклад, проектована система повинна виконувати наступні дії:

- зберігати інформацію про покупців;
- друкувати накладні на відпущені товари;
- стежити за наявністю товарів на складі.

Зауваження: всі іменники в цих реченнях – це будуть потенційні кандидати на сутності та атрибути.

Проаналізуємо ці *факти* (при цьому незрозумілі поки терміни будуть виділені знаком питання):

- *покупець* – явний кандидат на сутність;

- *накладна* – явний кандидат на сутність;
- *товар* – явний кандидат на сутність;
- (?) *склад* – а взагалі, скільки складів має фірма? Якщо кілька, то це буде кандидатом на нову сутність;
- (?) *наявність товару* – це, швидше за все, атрибут, але атрибут якої сутності?

Очевидний зв'язок між сутностями:

- «*покупці* можуть купувати багато *товарів*»;
- «*товари* можуть продаватися багатьом *покупцям*».

Перший варіант діаграми виглядає таким чином, як показано на рис. 4.4.

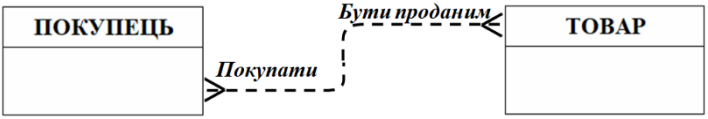


Рисунок 4.4 – Перший варіант ER-діаграми бази даних «Оптова фірма»

Отримані додаткові відомості:

- фірма має кілька *складів*;
- кожен *товар* може зберігатися на декількох *складах* і бути проданим з будь-якого *складу*.

Після отримання додаткових відомостей виникають питання:

- Куди помістити сутності «*Накладна*» та «*Склад*» і з чим їх зв'язати?
- Як зв'язані ці сутності між собою і з сутностями «*Покупець*» та «*Товар*»?

Для розуміння спробуємо описати ситуацію, що має бути відтворена в БД:

- *покупці* купують *товари*, отримуючи при цьому *накладні*, в які внесені дані про *кількість* і *ціну* купленого товару;
- кожен *покупець* може отримати кілька *накладних*;
- кожна *накладна* зобов'язана виписуватися на одного *покупця*;
- кожна *накладна* повинна містити кілька *товарів* (не буває порожніх накладних);
- кожен *товар*, в свою чергу, може бути проданий декільком *покупцям* через кілька *накладних*;
- кожна *накладна* повинна бути виписана з певного *складу*, і з будь-якого складу може бути виписано багато накладних.

Діаграма після уточнення виглядає таким чином, як показано на рис. 4.5.

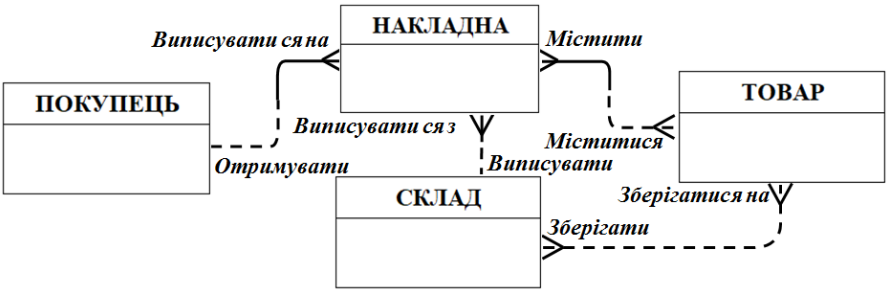


Рисунок 4.5 – Уточнений варіант ER-діаграми бази даних «Оптова фірма»

Додамо до існуючих додаткові відомості для визначення атрибутів сутностей:

- кожен *покупець* є *юридичною особою* і має *найменування*, *адресу*, *банківські реквізити*;
- кожен *товар* має *найменування*, *ціну*, а також характеризується *одиницями виміру*;
- кожна *накладна* має *унікальний номер*, *дату виписки*, *список товарів* з *кількостями* і *цінами*, а також загальну *суму накладної*. *Накладна* виписується з певного *складу* і на певного *покупця*;
- кожен *склад* має своє *найменування*.

Випишемо всі іменники, які будуть потенційними атрибутами, і проаналізуємо їх:

- *юридична особа* – термін не є характеристикою *покупця*, оскільки модель предметної області передбачає, що покупцем може бути тільки юридична особа;
- *найменування покупця* – явна характеристика *покупця*;
- *адреса* – явна характеристика *покупця*;
- *банківські реквізити* – явна характеристика *покупця*;
- *найменування товару* – явна характеристика *товару*;
- (?) *ціна товару* – схоже, що це характеристика *товару* (чи відрізняється ця характеристика від *ціни в накладній*?);
- *одиниця виміру* – явна характеристика *товару*;
- *номер накладної* – явна унікальна характеристика *накладної*;

- *дата накладної* – явна характеристика *накладної*;
- (?) *список товарів в накладній* – список не може бути атрибутом (ймовірно, потрібно виділити цей список в окрему сутність);
- (?) *кількість товару в накладній* – це явна характеристика, але характеристика чого? – це характеристика не просто «*товару*», а «*товару в накладній*»;
- (?) *ціна товару в накладній* – знову ж таки це має бути не просто характеристика товару, а характеристика *товару в накладній* (але ціна товару вже зустрічалася вище – це одне й те саме?);
- *сума накладної* – явна характеристика *накладної*, яка не є незалежною бо сума накладної дорівнює сумі вартостей усіх товарів, що входять в накладну;
- *найменування складу* – явна характеристика *складу*.

Деякі з описаних фактів потребують додаткових пояснення та уточнень.

Нехай такі додаткові відомості формулюються наступним чином:

- про поняття цін додатково встановлено: кожен *товар* має деяку *поточну ціну*, за якою товар продається в даний момент.
- ця *ціна* може змінюватися з часом.
- *ціна* одного і того ж товару в різних *накладних*, виписаних в різний час, може бути різною.
- таким чином, є *дві ціни – ціна товару в накладній* та *поточна ціна товару*.

Етап 3. Побудова концептуальної (логічної) моделі.

Сутності «*Накладна*» та «*Товар*» зв'язані одна з одною зв'язком типу багато-до-багатьох. Такий зв'язок має бути розщеплений на два зв'язка типу один-до-багатьох. Для цього потрібна додаткова сутність «*Список товарів в накладній*». Її зв'язок з сутностями «*Накладна*» та «*Товар*» характеризується наступними фразами:

- «кожна *накладна* повинна мати кілька записів зі *списку товарів в накладній*»;
- «кожен *запис зі списку товарів в накладній* зобов'язаний включатися рівно в одну *накладну*»;
- «кожен *товар* може включатися в кілька записів зі *списку товарів в накладній*»;
- «кожен *запис зі списку товарів в накладній* повинен бути зв'язаний рівно з одним *товаром*».

Атрибути «*Кількість товару в накладній*» та «*Ціна товару в накладній*» є атрибутами сутності «*Список товарів в накладній*».

Для зв'язку, що з'єднує сутності «*Склад*» та «*Товар*» аналогічно введемо додаткову сутність «*Товар на складі*». Атрибутом цієї сутності буде «*Кількість товару на складі*». Таким чином, товар буде значитися на будь-якому складі, а кількість його на кожному складі буде своєю.

Після цього ER-діаграма з атрибутами та ключами для БД «Оптова база» набуває свого остаточного вигляду, який наведено на рис. 4.6.

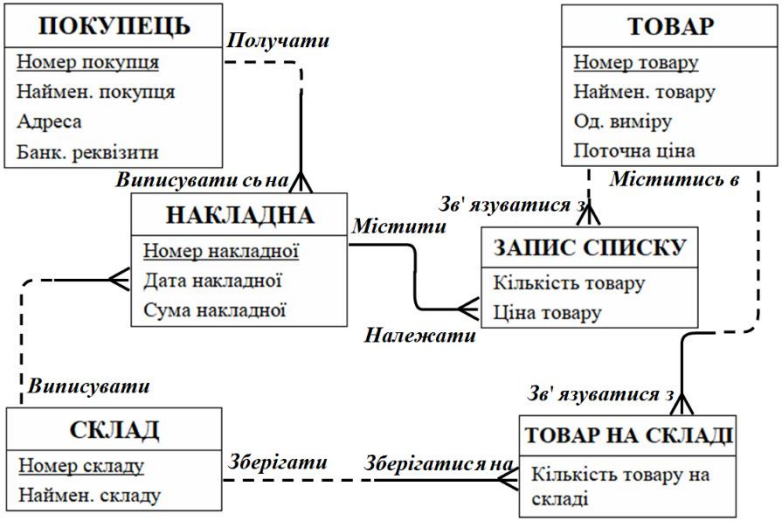


Рисунок 4.6 – Остаточний варіант ER-діаграмі бази даних «Оптова фірма»

Етап 4. Фізична ER-модель.

Розглянутий вище приклад ER-діаграми є прикладом *концептуальної діаграми*. Це означає, що діаграма *не враховує* особливості конкретної СУБД.

З цієї концептуальної діаграми можна побудувати *фізичну діаграму*, яка вже буде враховуватися такі особливості СУБД, як допустимі типи і найменування полів та таблиць, обмеження цілісності тощо.

Розглянемо, як може будуватись *фізична модель*.

На даній діаграмі кожна сутність являє собою таблицю бази даних, кожен атрибут стає колонкою відповідної таблиці.

Фізична модель реляційної бази даних «Оптова фірма» наведена на рис. 4.7.

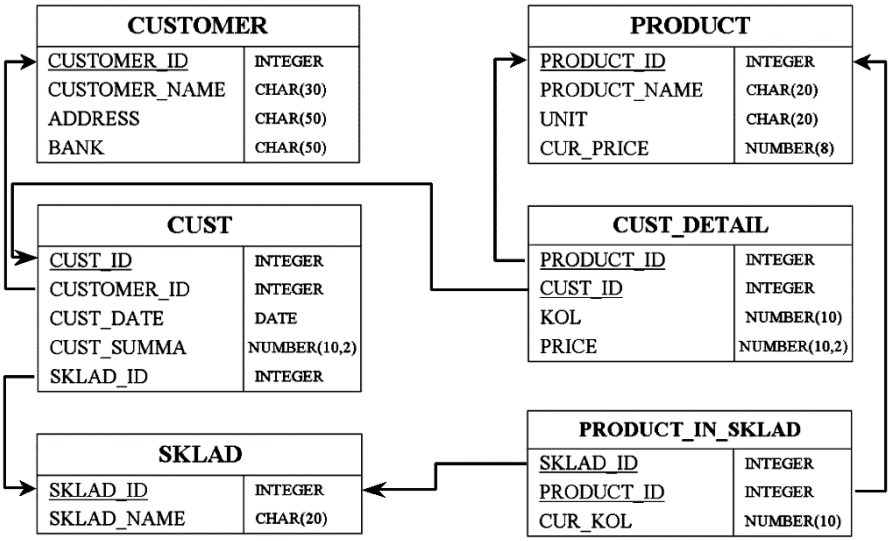


Рисунок 4.7 – Фізична ER-модель бази даних «Оптова фірма»

Необхідно звернути увагу на те, що в багатьох таблицях, наприклад, “CUST_DETAIL” та “PROD_IN_SKLAD”, відповідних сутностей «*Запис списку накладної*» та «*Товар на складі*», з’явилися нові атрибути, яких не було в концептуальній моделі – це ключові атрибути батьківських таблиць, які *мігрували* в дочірні таблиці для того, щоб забезпечити зв’язок між таблицями за допомогою зовнішніх ключів.

Отже, основними етапами створення моделі бази даних при застосуванні семантичного моделювання з побудовою ER-діаграм є:

1. Предметна область.
2. Модель предметної області.
3. Концептуальна (логічна) модель.
4. Фізична ER-модель.

При цьому, однією з переваг такого підходу є те, що отримані в результаті такої послідовності дій таблиці відразу знаходяться в 3НФ. Далі кожна таблиця потребує перевірки на відповідність нормальним формам високих порядків і в разі необхідності відповідної декомпозиції (див. Розділ 3).

5. ВИСНОВОК

Таким чином, розглянуто два найбільш застосовних підходи до створення концептуальних та логічних моделей реляційних баз даних, а саме метод нормалізації форм відношень та метод семантичного моделювання із застосуванням одного з варіантів побудови ER-діаграм. Обидва способи мають свої переваги та недоліки, але суттєво доповнюють один одного. Найбільш прийнятним підходом при створенні логічної моделі реляційної бази даних є використання ER-моделі з подальшою перевіркою на відповідність тій чи іншій нормальній формі високого порядку при наявності особливих ситуацій з подальшою декомпозицією у разі необхідності.

Алгоритм нормалізації, наведений у Розділі 3 дозволяє побудувати логічну модель даних, адекватну предметній області та уникнути некоректної роботи бази даних при виконанні операцій по її модифікації. Ця некоректна робота пов'язана з можливими аномаліями оновлення та порушеннями цілісності БД.

У той самий час, в цьому алгоритмі при приведенні до 3НФ є недоліки, а саме початкове розміщення всіх атрибутів в одному відношенні є дуже неприродною операцією. Інтуїтивно розробник одразу проектує кілька відношень відповідно до виявлених сутностей. Тому для отримання 3НФ доцільним є використання ER-діаграм, що описано у Розділі 4. В цьому випадку отримується логічна модель, в якій одночасно є кілька пов'язаних між собою відношень (сутностей). Однак, при цьому всі відношення, що входять в логічну модель, повинні піддаватися подальшій перевірці нормалізованості та приведення до відповідного стану за допомогою алгоритму нормалізації.

Нормальні форми високих порядків перевіряються тільки в особливих ситуаціях. Наприклад, для нормальної форми Бойса-Кодда (НФБК) виникає необхідність перевірки та декомпозиції у разі наявності у відношенні декількох незалежних потенційних ключів, а для четвертої нормальної форми (4НФ) така необхідність виникає у разі наявності у відношенні нетривіальних багатозначних залежностей. Слід також зазначити, що п'ята нормальна форма (5НФ), що перевіряє наявність нетривіальних залежностей з'єднання, – це майже остання нормальна форма, яку можна отримати шляхом декомпозиції. Її умови досить нетривіальні і невідповідність їй практично виникає мало коли.

Вихідні дані за варіантами для виконання розрахунково-графічних завдань або курсових робіт, а також зміст завдань та звітів наведено у Додатку.

СПИСОК ЛІТЕРАТУРИ

1. Connolly T.M., Begg C.E. Database Systems. A Practical Approach to Design, Implementation, and Management. Third Edition. Addison-Wesley Longman, 2002. 1443 p.
2. Codd E.F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*. New York, 1970. Vol. 13, Number 6. Pp. 377-387.
3. Date C.J., Darwen H., Lorentzos N. Temporal Data and the Relational Model: 1st Edition. Elsevier, Morgan Kaufmann, 2002. 422 p.
4. Date C.J. Introduction to Database Systems. Addison-Wesley, 2004. 1024 p.
5. Riordan R.M. Designing Relational Database Systems. Microsoft Press, 1999. 320 p.
6. Голуб Б.Л., Ящук Д.Ю. Основи організації баз даних. Київ: НУБіП України, 2017. 139 с.
7. Мулеса О.Ю. Інформаційні системи та реляційні бази даних: навч. посібник. Електронне видання, 2018. 118 с.
8. Трофименко О.Г., Прокоп Ю.В., Логінова Н.І., Копитчук І.М. Організація баз даних: навч. посіб. 2-ге вид. виправ. і доп. Одеса: Фенікс, 2019. 246 с.
9. Каратанов О.В. Організація даних: навч. посіб. до лаб. практикуму. Харків: Нац. аерокосм. ун-т ім. М.Є. Жуковського «Харків. авіац. ін-т», 2020. 60 с.
10. Методичні вказівки до виконання лабораторних робіт засобами СУБД MySQL для студентів спеціальності 122 «Комп'ютерні науки» / Уклад. О.Г. Сімонова, О.В. Охотська, І.Б. Шеліхова. Харків: «НТМТ», 2022. 40 с.
11. Покришень Д.А., Крєпкий Ю.О., Атрошенко І.Т., Дрозд О.П., Сподаренко І.Й. Основи баз даних. СКБД Access 2010 (2013): практ. посіб. Чернігів: ТОВ НВП «Інтерсервіс», 2013. 225 с.
12. Балик Н.Р., Мандзюк В.І. Бази даних MySQL: навч. посіб. Тернопіль: Навчальна книга – Богдан, 2010. 160 с.
13. Карпуша В.Д., Панченко Б.Є. Моделювання та проектування реляційних баз даних: навч. посіб. Суми: СДУ, 2010.
14. Бази даних та засоби управління. Практикум. [Електронний ресурс]: навч. посіб. для студ. спеціальності 123 – Комп'ютерна інженерія. / В.І. Павловський, А.В. Петрашенко, Д.В. Победа; КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2021. 112 с.
15. Коцовський В.М. Основи дискретної математики: навч. посіб. Ужгород: Рік-У, 2020. 123 с.

ДОДАТОК. РОЗРАХУНКОВО-ГРАФІЧНЕ ЗАВДАННЯ

Д.1. Зміст завдання

На основі реляційної моделі створити логічну модель бази даних, призначеної для зберігання та обробки інформації про визначену предметну область:

1. Провести вивчення предметної області та збір користувацьких уявлень, за результатами вивчення предметної області за необхідності доповнити заданий перелік атрибутів та/або додаткових обмежень з обов'язковим обґрунтуванням внесених змін;
2. Розробити модель предметної області з зазначенням унікальності атрибутів та їх залежностей один від одного.
3. Спроектувати логічну модель бази даних з використанням методу нормалізації форм відношень та методу семантичного моделювання з побудовою ER-діаграм або їх комбінації;
4. Застосувати розроблену логічну модель для реалізації фізичної моделі бази даних.

Д.2. Зміст звіту з розрахунково-графічного завдання (або проектування реляційної бази даних у курсовій роботі)

1. Титульна сторінка;
2. Опис предметної області – об'єкти та їх атрибути (імена полів), додаткові обмеження та зв'язки;
3. Етапи проектування бази даних (побудова нормалізованих відношень, створення ER-діаграми з перевіркою нормалізованості тощо);
4. Список нормалізованих відношень із зазначенням зв'язків;
5. Логічна модель (схема) бази даних (зобразити згідно з наведеним на рис. 3.1 або 4.6 зразком);
6. Фізична модель бази даних (див. рис. 4.7) зі список ідентифікаторів даних із зазначенням типу, довжини та числа значущих цифр після коми, наприклад, ГУСТИНА – RHO («N», 6, 2);
7. Приклад заповнення інформацією кожного відношення (не менше 5 кортежів, тобто рядків) або кожної сутності (не менше 5 екземплярів сутності), бажано таким чином, щоб частина інформації була зв'язана через зв'язані відношення (3-4 рядки), а частина інформації (1-2 рядки) була незалежна.

Д.3. Індивідуальні завдання за варіантами

Д.3.1. Варіант 1

Предметна область (назва БД): «ПОВЗУЧІСТЬ МЕТАЛІВ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Номер зразка;
2. Табельний номер та ПІБ співробітника, який проводив експеримент;
3. Номер підрозділу, у якому працює співробітник;
4. Номер робочого телефону співробітника;
5. Дата проведення експерименту;
6. Час (у [годинах]);
7. Розмір відносного подовження зразка;
8. Ідентифікатор форми зразка;
9. Назва матеріалу зразка;
10. Вид навантаження зразка;
11. Величина навантаження на зразок (у [Н]);
12. Діаметр поперечного перерізу зразка (у [м]);
13. Довжина стрижня зразка (у [м]);
14. Вид закріплення («затиснений» або «вільно опертий»);
15. Модуль пружності матеріалу (у [МПа]);
16. Коефіцієнт Пуассона матеріалу ([-]);
17. Початкова температура (у [К]);
18. Кінцева температура (у [К]);
19. Закон зміни температури (наприклад «лінійний» або «параболічний»).

Додаткові умови та обмеження:
1. Експеримент проводиться над зразком з унікальним номером, співробітником (ПІБ), у період, що визначається датою проведення та часом;
2. Результатом кожного експерименту над зразком є величина відносного подовження зразка;
3. За номером зразка визначається його форма (ідентифікатор), матеріал, вид навантаження, величина навантаження;
4. За ідентифікатором форми можна визначити діаметр перерізу, довжину стрижня, вид закріплення;
5. За назвою для кожного матеріалу завжди можна визначити його властивості (модуль пружності та коефіцієнт Пуассона);
6. Вид навантаження визначає початкову і кінцеву температуру та закон зміни температури;
7. Табельний номер у межах підприємства унікальний;
8. У кожному підрозділі розташовано один телефон.

Д.3.2. Варіант 2

Предметна область (назва БД): «ЕКСПЕРИМЕНТ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:

1. Номер випробування;
2. Шифр геометрії;
3. Форма зразка («круглий», «плаский»);
4. Орієнтація зразка (у [°], наприклад, «0», «45», «90»);
5. Пропорції зразка (наприклад, «1/5», «7/5», «10/1»);
6. Марка матеріалу (наприклад, «сталь», «мідь», «латунь», «дюраль»);
7. Модуль пружності матеріалу (у МПа);
8. Границя плинності матеріалу (у МПа);
9. Границя міцності матеріалу (у МПа);
10. Температура плавлення матеріалу (у К);
11. Дата початку випробування;
12. Температура випробування;
13. Навантаження на зразок (у Н);
14. Час руйнування (у годинах);
15. Характер руйнування («в’язке», «крихке», «змішане»).

Додаткові умови та обмеження:

1. Номер випробування є унікальним;
2. Одночасно може проводитись кілька випробувань;
3. Геометричні характеристики зразка (пп. 3-5) однозначно визначаються шифром геометрії;
4. З кожного матеріалу може бути зроблений зразок будь-якої геометрії;
5. Випробування однакових зразків за тих самих умов можуть проводитися багаторазово.

Д.3.3. Варіант 3

Предметна область (назва БД): «НАУКОВА БІБЛІОТЕКА».

Базовий перелік об’єктів (атрибутів) вихідного відношення:

1. Індекс статті у бібліотеці;
2. Назва статті;
3. Назва журналу;
4. Рік видання статті;
5. Номер журналу, у якому видано статтю;
6. Сторінки у журналі (наприклад, «10-15»);
7. Область знань, якій присвячено роботу (наприклад, «техніка», «медицина», «соціологія»);
8. Тема дослідження (наприклад, «Міцність», «Динаміка»);
9. Об’єкт досліджень (наприклад, «лопатка», «турбіна»);
10. Характеристика вкладу у тему (наприклад, «розвиток», «новий напрям»);
11. Мова журналу;
12. Адреса видавництва журналу;
13. Номер телефону видавництва;
14. Кількість співавторів за цією статтею;
15. ПІБ автора;
16. Хронологічний номер роботи автора в бібліотеці;
17. Місце роботи автора;
18. Вчений ступінь автора (наприклад, «к.т.н.», «д.т.н.», «PhD», «Dr. Eng.»);
19. Номер робочого телефону автора;
20. E-mail автора.

Додаткові умови та обмеження:

1. Індекс статті у бібліотеці унікальний;
2. Назва журналу унікальна;
3. В одному номері журналу не може бути статей з однаковими назвами.
4. Сторінки в журналі – перша та остання сторінки, на яких надруковано статтю в даному номері журналу;
5. Кожна стаття може мати кілька авторів (співавтори);
6. Кожен автор має один номер робочого телефону та E-mail або їх немає взагалі.

Д.3.4. Варіант 4

Предметна область (назва БД): «ПРОЕКТ КОНСТРУКЦІЇ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Шифр проекту конструкції;
2. Шифр вузла конструкції;
3. Шифр деталі конструкції;
4. Варіант деталі;
5. Назва проекту (наприклад, «Трактор»);
6. ПІБ головного конструктора проекту;
7. Дата закінчення проекту;
8. Назва графічного файлу проекту (наприклад, «Trk-45.grf»);
9. Назва вузла (наприклад, «Шасі»);
10. ПІБ відповідального конструктора вузла;
11. Дата закінчення проекту вузла;
12. Назва графічного файлу вузла;
13. Назва деталі (наприклад, «Стійка»);
14. ПІБ виконавця проекту деталі;
15. Місце роботи, відділ виконавця проекту;
16. Номер робочого телефону виконавця проекту деталі;
17. Дата останньої зміни варіанта деталі;
18. Назва графічного файлу варіанта деталі.

Додаткові умови та обмеження:
1. Шифр проекту конструкції є унікальним;
2. Кожен проект конструкції містить кілька вузлів;
3. Кожен вузол містить кілька деталей;
4. Один і той самий вузол може входити в різні проекти конструкції;
5. Одна й та сама деталь може входити в різні вузли;
6. Один і той самий співробітник може бути відповідальним конструктором кількох вузлів;
7. Один і той самий співробітник може бути виконавцем проектів кількох деталей;
8. У кожному відділі встановлено один телефон.

Д.3.5. Варіант 5

Предметна область (назва БД): «ДЕКАНАТ (іспити)».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Табельний номер викладача;
2. ПІБ викладача;
3. Домашня адреса викладача;
4. Номер домашнього телефону викладача;
5. Номер робочого телефону викладача;
6. Кафедра;
7. Шифр дисципліни;
8. Назва дисципліни;
9. Кількість академічних годин з дисципліни у семестрі;
10. Екзаменатор;
11. Факультет;
12. Курс;
13. Код групи;
14. Навчальний корпус;
15. Аудиторія;
16. Дата;
17. Час.

Додаткові умови та обмеження:
1. База даних містить розклад іспитів академічних груп під час сесії;
2. Табельний номер викладача є унікальним;
3. На кожній кафедрі встановлено один телефон;
4. Екзаменатор вибирається із числа викладачів;
5. Одночасно в одній аудиторії може проходити лише один іспит;
6. У екзаменатора не може бути двох і більше іспитів одночасно;
7. У групи не може бути двох і більше іспитів одночасно;
8. Врахувати, що в особливих умовах екзамени можуть відбуватись онлайн, тоді замість навчальний корпусу та аудиторії потрібно зберігати назву корпоративної платформи (онлайн-сервісу) та посилання на подію.

Д.3.6. Варіант 6

Предметна область (назва БД): «УНІВЕРСИТЕТ (співробітники)».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Табельний номер співробітника;
2. ПІБ співробітника;
3. Домашня адреса співробітника;
4. Дата народження співробітника;
5. ПІБ та дати народження дітей співробітника;
6. Оклад співробітника;
7. Номер домашнього телефону співробітника;
8. Номер мобільного телефону співробітника;
9. Номер робочого телефону співробітника;
10. Посада співробітника (наприклад, «молодший науковий співробітник», «науковий співробітник», «старший науковий співробітник», «асистент», «викладач», «старший викладач», «доцент», «професор»);
11. Вчений ступінь та вчене звання співробітника (наприклад, «к.т.н.», «д.т.н.», «PhD», «Dr. Eng.»; «старший науковий співробітник», «старший дослідник», «доцент», «професор»);
12. Шифр спеціальності за дипломом;
13. Наукова спеціалізація, напрям (наприклад, «компресорна техніка»);
14. Тема дослідження (наприклад, «міцність», «динаміка»);
15. Об’єкт досліджень (наприклад, «лопатка», «турбіна»);
16. Код найменування та місцезнаходження підрозділу (наприклад, «група міцності», «група динаміки», «обчислювальний центр»);
17. Назва кафедри;
18. ПІБ завідувача кафедри;
19. Номер телефону завідувача кафедри;
20. Розташування кафедри (корпус).

Додаткові умови та обмеження:
1. Табельний номер співробітника є унікальним;
2. На кожній кафедрі встановлено лише один телефон;
3. У співробітника або один домашній телефон, або його немає взагалі;
4. У співробітника може бути один або декілька мобільних телефонів;
5. Співробітник може мати кілька дітей або їх може не бути взагалі;
6. Розташування кафедри та її підрозділу може не співпадати;
7. У кожному науковому напрямі може працювати кілька співробітників.

Д.3.7. Варіант 7

Предметна область (назва БД): «ВІДДІЛ КАДРІВ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:

1. Табельний номер співробітника;
2. ПІБ співробітника;
3. Стать співробітника;
4. Дата народження співробітника;
5. Освіта співробітника;
6. Всі займані дотепер посади (наприклад, «програміст», «інженер», «майстер», «науковий співробітник», «начальник відділу» тощо);
7. Відповідні до посад зарплати;
8. Дати вступу на кожну посаду та дати звільнення з цих посад;
9. Типи, назви стягнень (наприклад, «депреміювання», «догана», «сувора догана»);
10. Дати кожного стягнення;
11. Номер відділу, де зараз працює співробітник;
12. Назва відділу, де зараз працює співробітник;
13. ПІБ начальника відділу, де зараз працює співробітник;
14. Номер робочого телефону співробітника;
15. Номер мобільного телефону співробітника;
16. Ім’я дитини співробітника;
17. Дата народження дитини співробітника;
18. Стать дитини співробітника.

Додаткові умови та обмеження:

1. Табельний номер співробітника є унікальним;
2. Кожній посаді відповідає оклад згідно зі штатним розписом;
3. Одні й ті самі посади співробітник міг займати у різний час у різних відділах;
4. У кожному відділі встановлено лише один телефон;
5. У співробітника може бути один або декілька мобільних телефонів;
6. Співробітник може мати кілька дітей або не мати їх взагалі;
7. Співробітник може мати кілька різних стягнень, накладених у різний час, або не мати їх взагалі.

Д.3.8. Варіант 8

Предметна область (назва БД): «ЛІКАРНЯ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Номер палати;
2. Кількість ліжок у палаті;
3. Відділення, у якому розташована палата;
4. Табельний номер лікаря;
5. ПІБ лікаря;
6. Відділення, у якому працює лікар;
7. Посада лікаря;
8. Спеціалізація лікаря;
9. Реєстраційний номер пацієнта;
10. ПІБ пацієнта;
11. Домашня адреса пацієнта;
12. Стать пацієнта;
13. Номер страхового поліса пацієнта;
14. Дата надходження пацієнта;
15. Номер палати, в якій перебуває (перебував) пацієнт;
16. Лікуючий лікар пацієнта;
17. Діагноз пацієнта;
18. Дата виписки пацієнта;
19. Завідувач відділення;
20. Номер телефону завідувача відділення.

Додаткові умови та обмеження:
1. Табельний номер лікаря унікальний;
2. В одній палаті не можуть лежати чоловіки та жінки;
3. У кожній палаті кількість пацієнтів не може перевищувати кількості ліжок;
4. Номер палати унікальний у межах відділення;
5. Кожне відділення має один номер телефону;
6. Дата виписки пацієнта не може мати значення більше, ніж дата надходження, або її може не бути взагалі, якщо пацієнт ще перебуває в лікарні;
7. Відділення, у якому розташована палата пацієнта, та відділення, де працює лікар, можуть не співпадати.

Д.3.9. Варіант 9

Предметна область (назва БД): «АДВОКАТСЬКА КОНТОРА».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Шифр (номер) статті Кримінального кодексу (КК);
2. Характер правопорушення за статтею КК;
3. Можливі види покарань за статтею КК;
4. Мінімальний строк позбавлення волі відповідно до статті КК;
5. Максимальний строк позбавлення волі відповідно до статті КК;
6. Пом’якшувальні підстави;
7. Номер справи (відповідно до реєстру адвокатської контори);
8. ПІБ клієнта;
9. Домашня адреса клієнта;
10. Дата народження клієнта;
11. Дата початку справи клієнта;
12. Номер камери;
13. Розмір гонорару, отриманого від клієнта;
14. Вид покарання за вироком суду;
15. Строк позбавлення волі за вироком суду;
16. Дата закінчення справи;
17. ПІБ адвоката;
18. Освіта адвоката;
19. Номер домашнього або мобільного телефону адвоката;
20. Номер робочого телефону адвоката;
21. Номер кабінету адвоката в офісі.

Додаткові умови та обмеження:
1. Шифр (номер) статті КК є унікальним;
2. У кожному кабінеті адвокатської контори є один телефон;
3. В одного адвоката у провадженні може бути кілька справ або не бути взагалі;
4. В одному кабінеті може бути робоче місце кількох адвокатів;
5. Строк за вироком суду або є, або може бути інший вид покарання, або його може не бути взагалі, якщо вирок виправдальний (у випадку коли є строк, то він має бути ≤ максимального та ≥ мінімального строку);
6. Обвинувачені в одній справі не можуть сидіти в одній камері;
7. Дата початку справи не може мати значення більше за дату закінчення (дати закінчення може не бути взагалі, якщо справа не закрита).

Д.3.10. Варіант 10

Предметна область (назва БД): «ГОТЕЛЬ».

Базовий перелік об’єктів (атрибутів) вихідного відношення:
1. Серія та номер паспорта постояльця;
2. Займаний постояльцем номер;
3. ПІБ постояльця;
4. ПІБ чоловіка чи дружини постояльця;
5. Стать постояльця;
6. Місце роботи постояльця;
7. Домашня адреса постояльця;
8. Дата поселення постояльця;
9. Дата виселення постояльця;
10. Тип пансіону (наприклад, «сніданок», «сніданок+обід», «сніданок+обід+вечеря»);
11. Вартість типу пансіону (на добу);
12. Оплата за проживання та пансіону;
13. Цифровий ідентифікатор номера (наприклад, «312», «401»);
14. Клас номера (наприклад, «люкс», «напівлюкс», «окремий номер на одного», «окремий номер на двох», «загальний номер»);
15. Кількість місць у номері;
16. Кількість заброньованих місць у номері;
17. Вартість місця у номері;
18. Номер телефону в номері.

Додаткові умови та обмеження:
1. Номер паспорта унікальний у межах серії;
2. Кількість гостей в одному номері не може перевищувати кількість місць у номері;
3. Вартість місця залежить від класу, до якого належить номер;
4. Оплата за проживання визначається як добуток кількості днів на вартість місця плюс оплата пансіону;
5. При виселенні оплата за проживання має бути погашена;
6. У одному загальному номері не можуть одночасно проживати постояльці різної статі (в інших номерах, якщо то не сімейна пара);
7. У кожному номері встановлено один телефон.

Навчальне видання

МАРТИНЕНКО Геннадій Юрійович

КОНЦЕПТУАЛЬНЕ ТА ЛОГІЧНЕ ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

Навчально-методичний посібник
для виконання завдань з дисципліни «Організація баз даних»
для студентів спеціальностей 122 «Комп'ютерні науки»,
113 «Прикладна математика»

Відповідальний за випуск доц. О. О. Водка

Роботу до видання рекомендував доц. В. О. Федоров

В авторській редакції

План 2023 р., п. 31

Підп. до друку 25.09.2023 р.
Гарнітура Times New Roman. Обсяг – 3,5 друк. арк. Електронний ресурс

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, м. Харків, вул. Кирпичова, 2