ТЕМА 4. Реляційна модель даних та реляційна алгебра

Перетворення ER-моделі у реляційну модель даних

Легше за все проектувати БД за допомогою ER-моделі. Але цього недостатньо. Сучасні СУБД переважно грунтуються на інших принципах, які в сукупності мають назву *реляційна модель* (relational model). Реляційна модель проста і плідна, оскільки грунтується чи не на єдиному основному понятті *відношення* (relation).

> Відношення – це двомірна таблиця, призначена для впорядкованого збереження даних.

Основи реляційної моделі

Реляційна модель передбачає єдиний спосіб збереження даних – у вигляді двовимірних таблиць, які називаються відношеннями. Приклад – на рис.4.1.

title	*year*	*length*	*filmType*
Star Wars	1977	124	color
Mighty Ducks	1991	104	color
Wayne's World	1992	95	color

Рисунок 4.1 - Відношення *Movies*

Назва відношення – *Movies,* і призначене воно для зберігання сутностей *Movies* у контексті БД фільмів. Рядок таблиці відповідає сутності *Фільм*, а кожний стовпчик – одному з атрибутів множини сутностей *Movies.*

Атрибути

Як правило, атрибути відношення – це атрибути відповідної множини сутностей.

Схеми

Назву відношення та назви атрибутів у сукупності називають схемою (schema) відношення. Схема відношення – це назва відношення, за якою йдуть (невпорядковані) назви атрибутів у круглих дужках. Наприклад, схема відношення на рис.4.1 має вид:

```
Movies(title, year, length, filmType).
```

Проєкт БД, виконаний у рамках реляційної моделі, включає схеми відношень, набір яких є *реляційною схемою БД* (relational database schema) або просто *схемою БД.*

Кортежі

Рядки відношення, відмінні від рядка з назвами атрибутів, називають *кортежами* (tuples). Кортеж містить по одному компоненту для кожного атрибута відношення. Напр., перший кортеж відношення *Movies* (рис.4.1) виглядає так:

```
(Star Wars, 1977, 124, color).
```

Компоненти мають бути в тому стандартному порядку, який зафіксовано у схемі відношення.

Домени

В реляційній моделі кожний компонент кортежу має бути атомарним, тобто відноситись до певного базового типу, як-от рядок або число. Значеннями компонента не можуть бути записи, множини, списки, масиви чи інші складені об'єкти, які природньо припускають розділення на дрібніші елементи.

Крім цього, з кожним атрибутом відношення асоціюється певний *домен* (domain) - базовий тип. Значення компонентів усіх кортежів мають належати до певного домену, який визначається атрибутом відношення. Напр., у відношенні рис.4.1

title	рядки
year, lenth	цілі числа
filmType	„color" або "blackAndWhite"

Терміни ER, реляційні та бази даних

ER-модель	*Реляційна модель*	*База даних*
Множина сутностей	Відношення	Таблиця
Атрибут	Атрибут	Колонка, стовпчик
Сутність	Кортеж	Запис
	Компонент кортежу	Поле
Зв'язок	Відношення	Таблиця
Домен	Домен	Тип + обмеження
ER-діаграма	Реляційна схема	Схема БД

Форми представлення відношень

Відношення є множинами кортежів, але не впорядкованими списками. Порядок кортежів не є суттєвим. Для відношення рис.4.1 можливі 3*2=6 варіантів порядку рядків.
Крім порядку рядків, несуттєвим є також порядок стовпчиків. Таким чином, таблиця 3*4 (рис.4.1) має 6*(4*3*2)=6*24=144 представлення.

Традиційно системи баз даних підтримують лише одну версію будь-якого відношення: набір кортежів, який є в БД „в даний момент". Такий екземпляр відношення називають *поточним екземпляром* (current instance).

Від ER-діаграм до реляційних схем

У першому наближенні процес перетворення ER-діаграми у реляційну схему досить прямолінійний:

- Перетворити кожну множину сутностей у відношення з тим же набором атрибутів;
- Замінити кожний зв'язок відношенням, атрибути якого є ключами множин сутностей, які з'єднує зв'язок.

Хоча ці правила охоплюють значну частину питання, існує кілька особливих ситуацій, з якими треба рахуватись.

1. Слабкі множини сутностей (weak entity sets) не можуть бути перетворені у відношення безпосередньо.
2. Зв'язки isa і підкласи не можуть бути перетворені у відношення безпосередньо.
3. Інколи доцільно об'єднати в одному відношенні два інших – зокрема, тоді, коли одне з відношень містить зв'язок типу *багато до одного.*

Від множин сутностей до відношень

Спочатку розглянемо множини сутностей, які не є слабкими. Для кожної такої множини сутностей можна створити відношення з тими ж іменем і набором атрибутів. Таке відношення не містить інформації про зв'язки.

Приклад 4.1. Розглянемо наведену вище (рис.2.2) ER-діаграму БД фільмів:

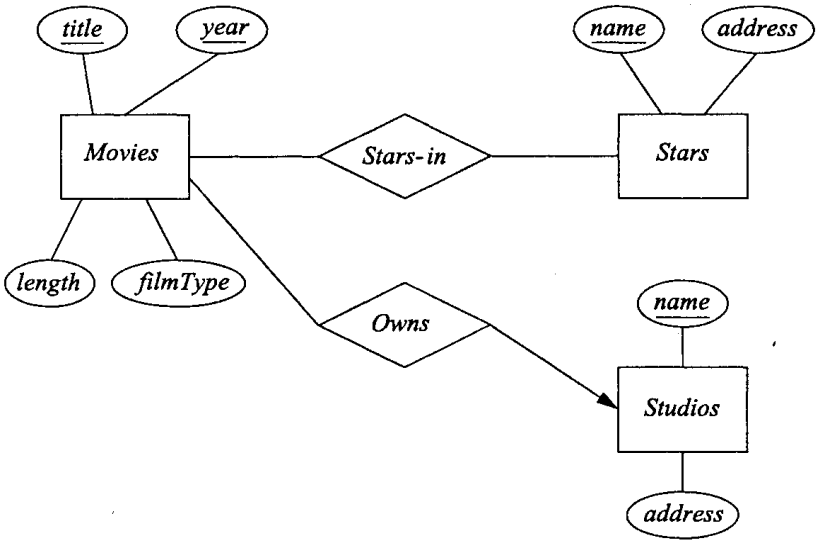


Рисунок 4.2 - ER-діаграма БД фільмів

Відношення *Movies* може виглядати як уже наведено вище

```
Movies(title, year, length, filmType).
```

Схема відношення Stars може бути представлена як Stars(name, address).

Від ER-зв'язків до відношень

Зв'язки ER-моделі також представляються відношеннями. Ці відношення повинні мати наступні атрибути:

1. Ключові атрибути кожної з з'єднаних зв'язком множин сутностей.
2. Власні атрибути зв'язку.

Якщо певна множина сутностей у контексті зв'язку згадується кілька разів, у різних „ролях", кожний з її ключових атрибутів уводиться у відношення стільки разів, скільки „ролей" він має.

Приклад 4.2. Розглянемо зв'язок *Qwns* ER-діаграми рис.4.2, яка з'єднує множини сутностей Movies і Studios. Його схема матиме вигляд:

```
Owns(title, year, studioName).
```

А це може бути типовий екземпляр відношення:

title	*year*	*studioName*
Star Wars	1977	Fox
Mighty Ducks	1991	Disney
Wayne's World	1992	Paramount

Приклад 4.3. Зв'язок *Stars-in* може бути перетворений на схему відношення *Stars-in(title, year, starName)*

Приклад 4.4. Багатосторонні зв'язки також можуть бути перетворені на відношення. Повернувшись до малюнку 2.6, визначимо реляційну схему з'єднуючого відношення.

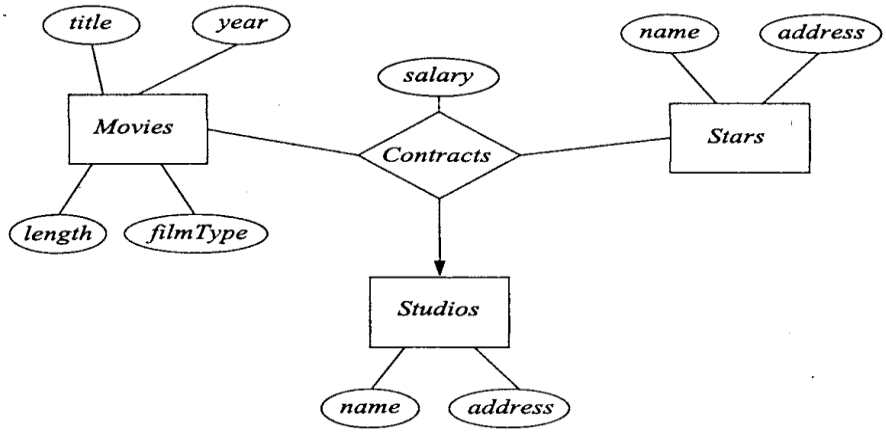


Рисунок 4.3 - З'єднуюче відношення з атрибутом.

Contracts(title, year, studioName, starName, salary) .

Додано власний атрибут зв'язку *salary*. Відмітимо, що ми вільні в виборі імен атрибутів.

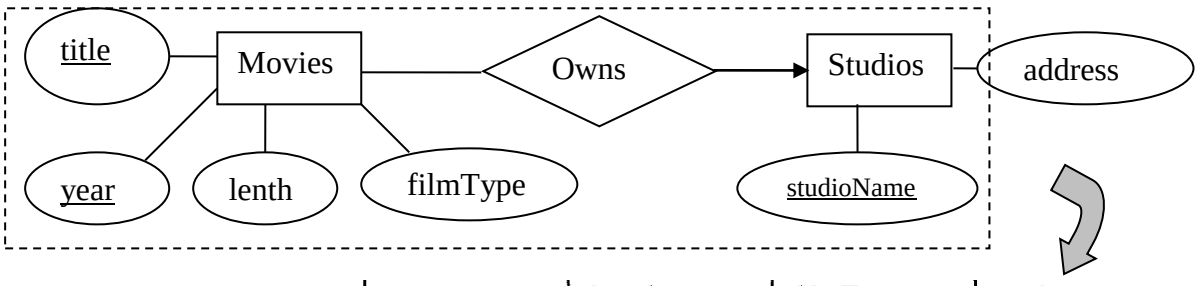
Об'єднання відношень

Нехай є множина сутностей *E*, з'єднана зв'язком *R* типу *Багато до одного* з множиною *F* від *E* до *F*. Оскільки зв'язок *R* відноситься до типу *Багато до одного,* усі атрибути *E* та *F* мають значення, які однозначно визначаються ключем множини *E,* та їх можна об'єднати в одній схемі відношення, яка складається з:

1. усіх атрибутів множини *E,*
2. ключових атрибутів множини *F,*
3. власних атрибутів зв'язку *R.*

У відношенні *E* для сутності *e* , яка не пов'язана з жодною сутністю з *F,* компоненти, що відповідають атрибутам пп. 2, 3, отримують значення *Null*. Це значення застосовується, якщо насправді значення є невизначеним.

Приклад 4.5. Об'єднання відношень *Movies* та *Owns* у БД фільмів.



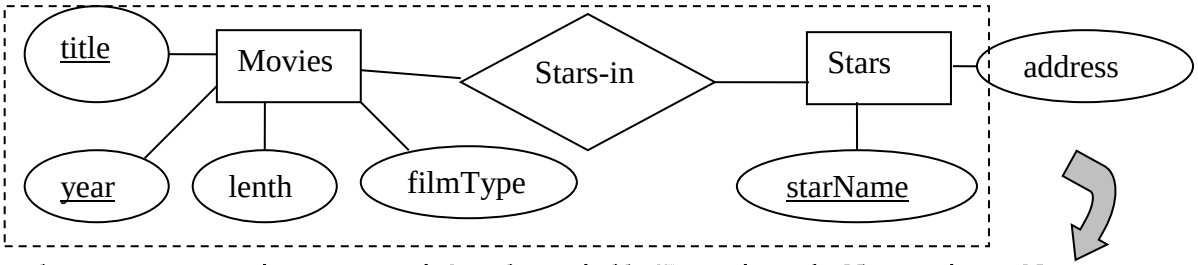
title	*year*	*length*	*filmType*	*studioName*
Star Wars	1977	124	color	Fox
Mighty Ducks	1991	104	color	Disney
Wayne's World	1992	95	color	Paramount

Рисунок 4.4 - Об'єднання відношень *Movies* та *Owns*

В підсумку екземпляр відношення – результат об'єднання відношень *Movies* та *Owns* має вигляд рис.4.4.

Відношення множини сутностей слід об'єднувати лише з відношенням зв'язку *багато до одного,* та, зокрема, *один до одного*; не слід об'єднувати з відношенням зв'язку *багато до багатьох,* і, зокрема, *один до багатьох.*

Приклад 4.6. Уявімо об'єднання відношення рис.4.4 з відношенням зв'язку *Stars-in* типу *багато до багатьох.*



title	*year*	*length*	*filmType*	*studioName*	*starName*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

Рисунок 4.5 - Не рекомендоване об'єднання відношення з відношенням зв'язку *багато до багатьох.*

Оскільки в кінофільмі знімається кілька акторів, інформацію про фільм, зокрема, і неключові атрибути *lenth, filmType,* доводиться повторювати кілька разів, для кожного актора. Подібна надмірність небажана, і методи *нормалізації* спрямовані на усунення таких надмірностей.

Перетворення слабких множин сутностей

Для перетворення слабкої множини сутностей у відношення треба взяти до уваги наступне.

1. Відношення для слабкої множини сутностей W має включати не тільки власні атрибути W, а також ключові атрибути множин сутностей, які беруть участь у формуванні ключа W. Ці атрибути на ER-діаграмі з'єднуються з W підтримуючими зв'язками з подвійним ромбом.
2. У відношенні для будь-якого зв'язку W, яким з'єднана ця множина, треба мати всі ключові атрибути W, в тому числі і „позичені".
3. Підтримуючий зв'язок R, що з'єднує W з „головною" множиною, яка позичає атрибути для ключа W, не потребує окремого відношення. Адже це зв'язок *багато до одного,* і ключові атрибути підтримуючої множини або уже присутні у W, або можуть бути приєднані.
4. Якщо два відношення створюються на основі одної слабкої множини сутностей, те з них, яке містить менше атрибутів , не має самостійного значення і може бути відкинуте. Адже кортежі підтримуючого зв'язку повністю містяться в кортежах власне слабкої множини. Хоча взагалі, для „неслабких" множин сутностей, це твердження невірне. Приклад: множини *люди* та *платники податків* для податкової служби: атрибути множини *люди* є підмножиною атрибутів *платники податків*, але *люди* не є слабкою сутністю до *платники податків.*

Приклад 4.7. Слабка множина сутностей *Contracts* у БД фільмів.

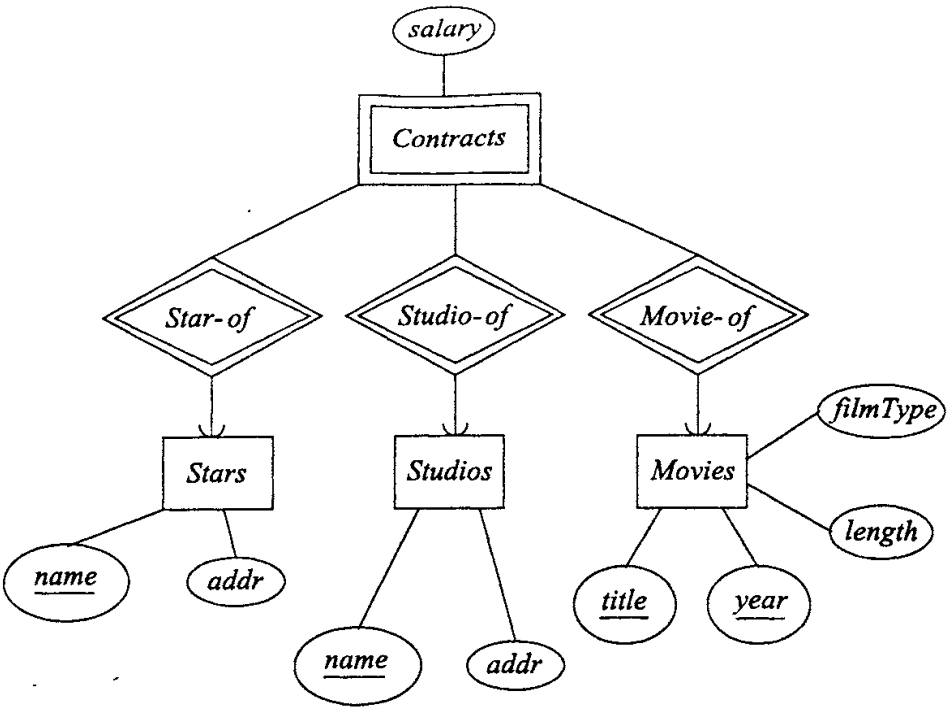


Рисунок 4.6 - Слабка множина *Contracts* у БД фільмів.

Схема відношення *Contracts:*
Contracts(starName, studioName, title, year, salary).

Атрибути starName, studioName – це перейменовані ключові атрибути множин сутностей Stars і Studios. Створення відношень для зв'язків Star-of, Studio-of, Movie-of не має сенсу, оскільки їхні схеми є підмножинами схеми Contracts.

Таку ж схему ми б отримали для тернарного з'єднуючого зв'язку Contracts (рис.2.7).

Остаточні правила побудови відношення для слабкої множини W:

1. Побудувати відношення, схема якого включає:
   - Усі атрибути множини W;
   - Усі атрибути підтримуючих зв'язків для множини W;
   - Усі ключові атрибути кожної з підтримуючих множин сутностей, з'єднаних з W зв'язками *багато до одного* у напрямку від W.
2. Не створювати відношення, що відповідають підтримуючим зв'язкам W.

Перетворення структур підкласів у відношення

Нагадаємо, що коренева множина сутностей містить ключові та неключові атрибути, які мають також усі її підкласи.

Основні стратегії перетворення:

1. Застосувати підхід *Сутність-зв'язок*. Для кожної множини сутностей ієрархії *E* створити відношення, яке містить усі ключові атрибути кореневої множини сутностей та всі атрибути, які належать *E*.
2. *Задіяти значення Null.* Створити одне відношення, що включає усі атрибути усіх множин сутностей, що входять в ієрархію. Кожна сутність представлена однаковими кортежами, і значення тих атрибутів, яких кортеж не має, дорівнюють *Null*.

Перетворення у стилі *Сутність-зв'язок*

Ми маємо створити відношення для кожної множини сутностей. Якщо множина сутностей *E* не є кореневою, відношення для *E* має містити ключові атрибути кореневої множини і всі власні атрибути *E*.

Хоча зв'язок *isa (є)* ми називаємо зв'язком, ми розуміємо його принципову відмінність від інших зв'язків, оскільки він з'єднує компоненти єдиної сутності, а не різні сутності. Тому окреме відношення для нього не створюється.

Приклад 4.8. Відтворимо малюнок 2.14:

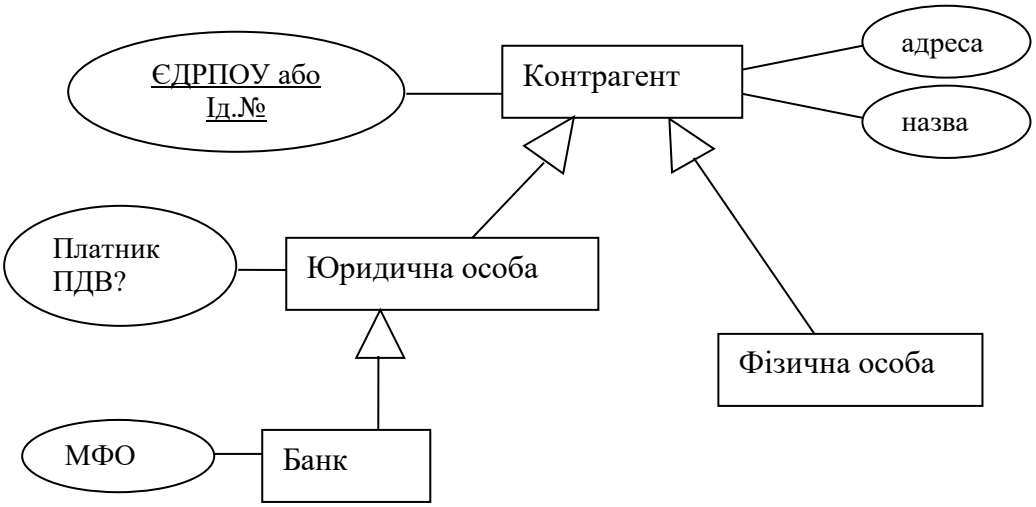


Рисунок 4.7 - Приклад діаграми базового класу *Контрагент* та його підкласів.

Схеми відношень, необхідних для представлення цієї ієрархії:
Контрагент(ЄДРПОУ або Ід.№, адреса, назва)
Юридична_особа(ЄДРПОУ або Ід.№, Платник ПДВ?)
Банк(ЄДРПОУ або Ід.№, Платник ПДВ?, МФО)
Фізична_особа(ЄДРПОУ або Ід.№)

Сутність *Юридична_особа* описується двома відношеннями: *Юридична_особа* та *Контрагент*, звідки ми візьмемо адресу і назву юридичної особи.

Сутність *Банк* описується трьома відношеннями: *Банк, Юридична_особа* (звідки візьмемо ознаку *Платник ПДВ?* банку) та *Контрагент*, звідки ми візьмемо адресу і назву банку. Наявність ключа *ЄДРПОУ або Ід.№* у всіх трьох відношеннях дозволяє знайти сутності *Юридична_особа* та *Контрагент*, які відповідають банку.

Приклад 4.9. Відтворимо та доповнимо малюнок 2.14:

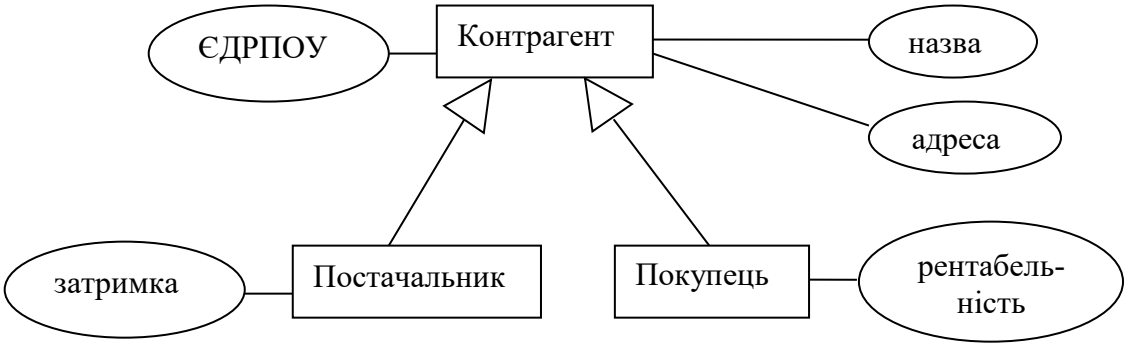


Рисунок 4.8 - Приклад можливої належності сутності до двох підкласів

Схеми відношень:
Контрагент(ЄДРПОУ, назва, адреса)
Постачальник(ЄДРПОУ, затримка)
Покупець(ЄДРПОУ, рентабельність)

Сутність *Постачальник* описується двома відношеннями: *Постачальник* та *Контрагент*, звідки ми візьмемо адресу і назву юридичної особи.

Сутність *Постачальник,* яка одночасно є сутністю *Покупець*, описується трьома відношеннями:

- у відношенні *Постачальник* через ЄДРПОУ знайдемо затримку,
- у відношенні *Покупець* знайдемо рентабельність,
- у відношенні *Контрагент* знайдемо назву і адресу.

Використання значень Null та об'єднання відношень

Якщо дозволити використання невизначених значень типу *Null*, можна вмістити усі множини сутностей в одному-єдиному відношенні, яке охоплює усі множини сутностей ієрархії. Будь-яка сутність в цій ієрархії представлена одним кортежем з єдиною схемою. Компоненти кортежу мають значення *Null*, якщо відповідний атрибут не визначений для відповідної множини сутностей.

Застосовуючи цей підхід до прикладу 4.8, отримуємо відношення зі схемою:

Контрагент(ЄДРПОУ або ід.№, назва, адреса, Платник ПДВ?, МФО).

Але ніщо не дається задарма: множину фізичних осіб доведеться визначати по довжині ідентифікаційного номеру – 10 знаків замість 8.

Застосовуючи цей підхід до прикладу 4.9, отримуємо відношення зі схемою:

Контрагент(ЄДРПОУ, назва, адреса, затримка, рентабельність).

Атрибути *затримка* сутностей *Контрагент*, які не є постачальниками, отримують значення *Null*. Компоненти *рентабельність* сутностей *Контрагент*, які не є покупцями, також отримують значення *Null*.

В цілому підхід з використанням *Null* виглядає найбільш привабливим, оскільки створюється лише одне відношення. Він стає неефективним при великій кількості полів зі значенням *Null*, тобто коли набори атрибутів підкласів сильно різняться.

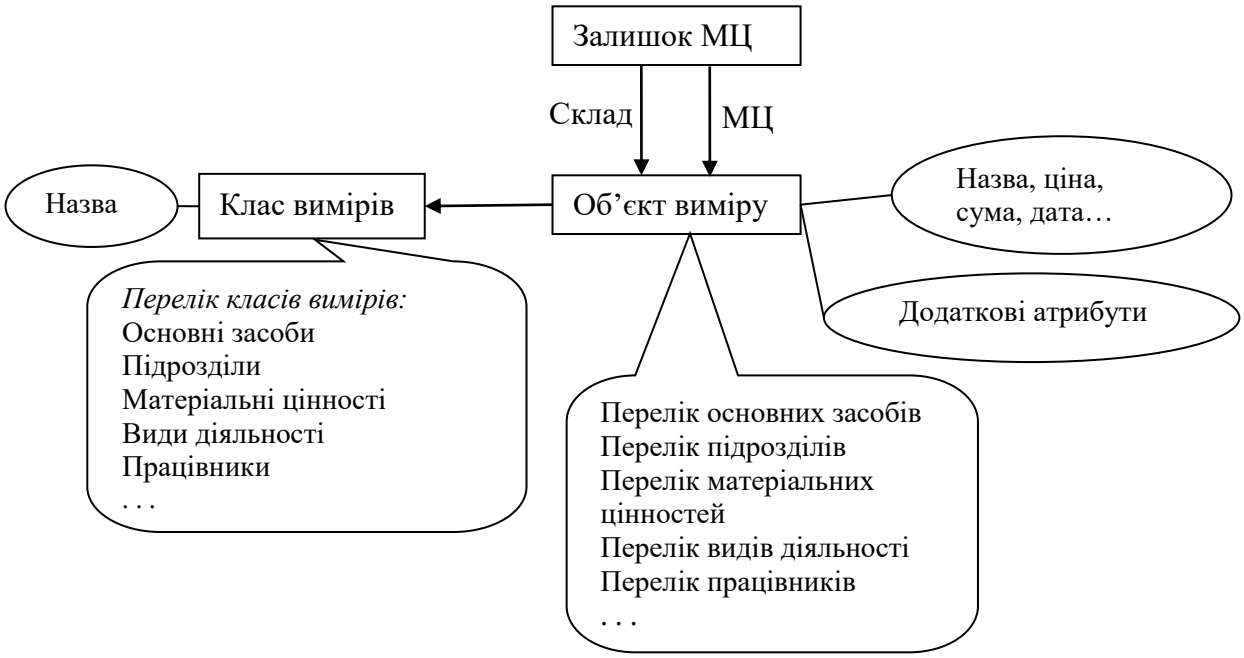


Рисунок 4.9 – Всі виміри в одній множині сутностей і відношенні Об’єкт виміру

Створення реляційної моделі на основі функціональних залежностей

Обмеження унікальності, яке називають *функціональна залежність* (functional dependency, FD), дуже важливе для уникнення інформаційної надмірності (дублювання).

Визначення

> *Функціональна залежність* від атрибутів $A_1, A_2, ... A_n$ інших атрибутів відношення R – це твердження наступного виду:
> Якщо два кортежі відношення R співпадають у атрибутах $A_1, A_2, ... A_n$ (тобто кортежі мають однакові значення компонентів для кожного з названих атрибутів), то вони мають співпасти і в іншому атрибуті B.

Формально така FD записується як $A_1, A_2, ... A_n \to B$ і свідчить, що $A_1, A_2, ... A_n$ обумовлюють B.

Якщо кілька атрибутів $A_1, A_2, ... A_n$ функціонально обумовлюють більше одного атрибута, тобто

$A_1, A_2, ... A_n \to B_1$

$A_1, A_2, ... A_n \to B_2$

...

$A_1, A_2, ... A_n \to B_m$,

дозволяється використовувати наступне скорочене представлення набору таких FD:

$A_1, A_2, ... A_n \to B_1, B_2, ... B_m$.

Наприклад, нехай є 2 кортежа t і u.

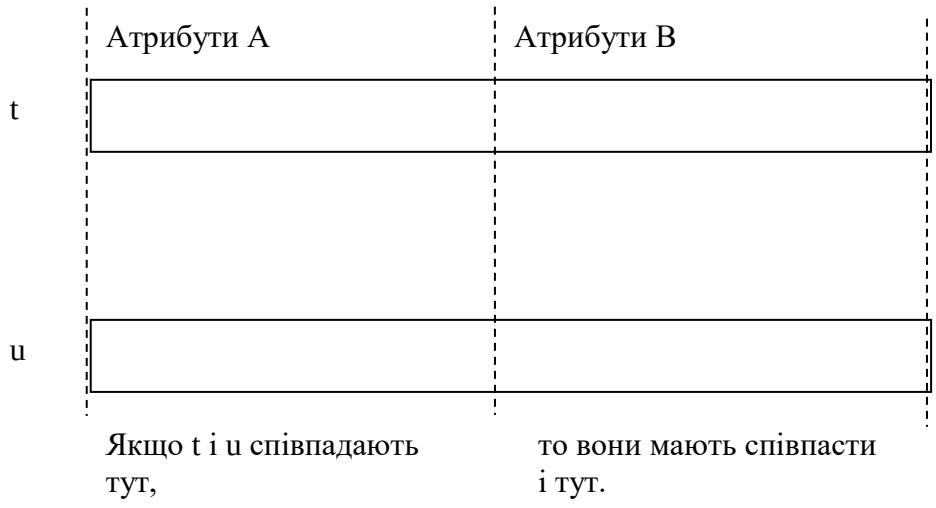


Рисунок 4.10 - Ефект функціональної залежності атрибутів B від A на прикладі двох кортежів

Визначення.

> Відношення $A_1, A_2, ... A_n \rightarrow B_1, B_2, ... B_m$ називають *тривіальним,* якщо $\{B_1, B_2, ... B_m\} \subseteq \{A_1, A_2, ... A_n\}$, тобто атрибути B є підмножиною атрибутів A.

Приклад 4.10. Розглянемо відношення
`Movies(title, year, length, filmType, studioName, starName)`

Вкажемо кілька функціональних залежностей:
`title year → length`
`title year → filmType`
`title year → studioName`

Екземпляр цього відношення:

title	*year*	*length*	*filmType*	*studioName*	*starName*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

Рисунок 4.11 - Екземпляр відношення Movies

Оскільки ліва частина усіх трьох FD однакова (title year), їх можна коротко подати так:

`title year → length filmType studioName.`

Конкретні значення title і year однозначно визначають сутність *фільм*, якому відповідають конкретні значення lenth і filmType. Крім того, згідно зв'язку *багато до одного* фільму відповідає лише одна сутність Studio.

З іншої сторони, вираз
title year -> starName
невірний, оскільки одному фільму взагалі може відповідати кілька сутностей-акторів.

Ключі відношень

Множина виду $\{A_1, A_2, ... A_n\}$, що складається з одного чи кількох атрибутів, є ключом відношення R, якщо виконуються наступні умови:

- Атрибути $A_1, A_2, ... A_n$ функціонально обумовлюють усі інші атрибути відношення. Нема двох кортежів, де би значення всіх атрибутів $A_1, A_2, ... A_n$ співпали.
- Жодна з припустимих підмножин $\{A_1, A_2, ... A_n\}$ атрибутів не є функціональним обгрунтуванням решти атрибутів відношення R, тобто ключ $\{A_1, A_2, ... A_n\}$ є мінімальним.

Приклад 4.11: Відношення *Залишки* містить залишки матеріалів на складах.
Залишки(IdСкладу, IdМатеріалу, кількість)

Склад і матеріал обумовлюють певну кількість матеріалу на складі, якщо він там узагалі існує.
Але лише матеріал не визначає залишок, оскільки залишки одного матеріалу можуть бути на кількох складах. Так само склад не визначає залишок, оскільки на складі зберігається багато різних матеріалів.

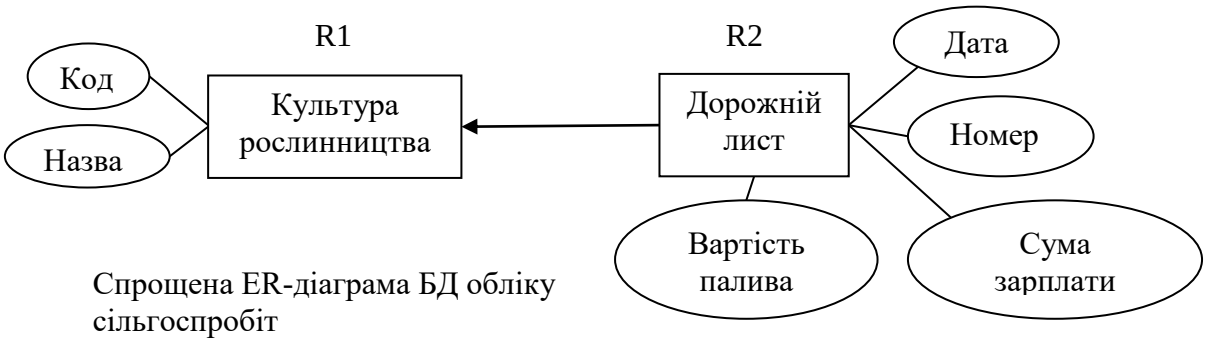
Інколи для відношення може бути створено кілька ключів. Як правило, в цій ситуації одному з них надається роль *первинного ключа* (primary key).
Співробітник(Id, ІПН, №вДемогафРеєстрі, ТабельнийN, ПІБ, ДатаНародження, …)

В тексті схеми відношення атрибути первинного ключа потрібно підкреслювати.

Власний і зовнішній ключ

Неформальне визначення: *зовнішнім ключем* є множина атрибутів відношення R2, значення яких повинні збігатися із значеннями деякого потенційного ключа відношення R1. *Потенційний ключ* – реальний первинний ключ або інша множина атрибутів, що утворюють унікальний набір значень.

Розглянемо облік сільськогосподарських робіт в агрофірмі. Витрати на технологічну операцію, передбачену для культури рослинництва, фіксуються первинним документом *Дорожній лист*, який серед іншого містить посилання на культуру рослинництва (*Crop* – рослина).



Спрощена ER-діаграма БД обліку сільгоспробіт

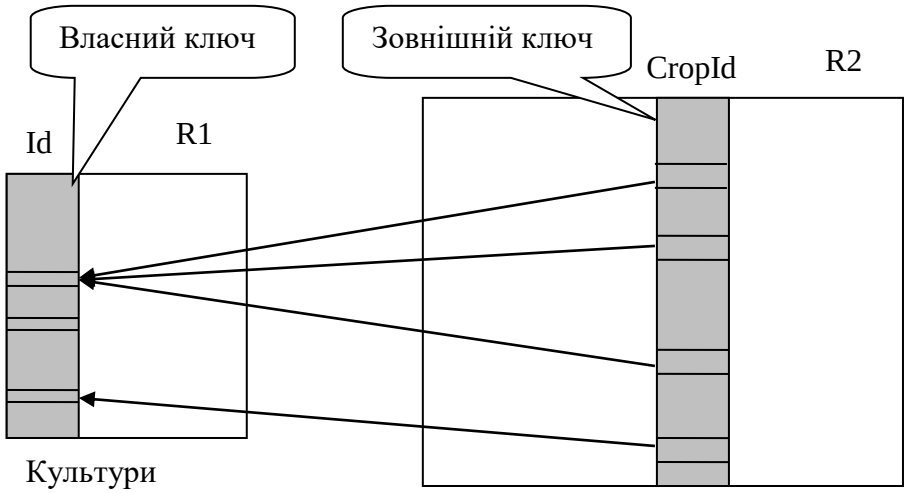


Рисунок 4.12 - Приклад власного і зовнішнього ключів у базі агрофірми

Вибір ключів для відношення

Якщо реляційна схема створюється з ER-діаграми, можна сформулювати правила:

- Якщо відношення отримане на основі множини сутностей, ключ відношення формується з ключових атрибутів множин сутностей. Приклад:

```
Movies(title, year, length, filmType)
Stars(name, address)
```

- Якщо відношення R створюється на основі зв'язку, то:
  - Якщо зв'язок належить до типу *Багато до багатьох,* ключами відношення R стають ключові атрибути обох множин сутностей, що з'єднуються зв'язком.
  - Якщо зв'язок належить до типу *Багато до одного* та з'єднує множини сутностей E_1 та E_2 у напрямку від E_1 до E_2, ключами відношення R мають бути ключові атрибути множини E_1 (але не E_2), тобто ключові атрибути сторони «*Багато*».
  - Якщо зв'язок належить до типу *Один до одного* , ключами відношення R можуть бути ключі будь-якої множини, що з'єднуються.

Приклади з БД фільмів:

```
Owns(title, year, studioName)
```

Це зв'язок типу *Багато до одного.* Тому ключом є ключ сутності на стороні *Багато*.

```
Stars-in(title, year, starName)
```

Всі атрибути цього відношення мають бути ключовими.

Бази даних і структурна лінгвістика

Поняття лінгвістики:
Номенклатура – „матеріал" мови; вокальні чи графічні знаки, що позначають речі.
Структура – „форма" мови: ансамбль правил, що регулюють поєднання цих вокальних чи графічних знаків.

Рисунок 4.13 - Фердинанд де Сосюр (1857-1913) – засновник структурної лінгвістики.

Сосюрова революційна для епохи думка про те, що мовні знаки є довільними, означала, що певний знак має значення не тому, що подібний до якоїсь речі чи явища, а тому що відмінний від інших знаків, що означають інші речі і явища.
„Мова – це система, у якій усі терміни солідарні, в якій цінність одного випливає тільки з одночасної присутності інших" – писав Сосюр. Значення знаку визначається не його подібністю до речі, яку він позначає, а його місцем у системі знаків. Відома фраза Сосюра: „У мові є тільки відмінності".

Кльод Леві-Строс (1908-2009) – засновник структурної антропології.
Спроба Леві-Строса подивитися на культуру як на мову тягла висновки: соціальні феномени є *повідомленнями*; уся множина повідомлень є *мовленням*, що передбачає безліч варіацій – однак варіацій, які рухаються в межах чіткої *структури* правил та класифікацій, яка зветься *мовою*. Єдність мови не скасовує, а радше уможливлює безмежну множину висловлювань, які в ній можуть існувати[1].

Подивимся на БД з позиції структурної лінгвістики.

Мова	*БД*
Номенклатура мови - словники	Умовно-постійна нормативно-довідкова інформація (як-от таблиця культур рослинництва ClCrop у БД агрофірми). Довідники, на які часто посилаються первинні документи як на таблиці власних ключів (виміри).
Структура мови - правила	Схема або структура БД (структура таблиць та зв'язки)
Повідомлення, речення	Змінна, оперативна інформація, часто це первинні документи. Ця інформація відображає факти, що відбулись або відбудуться, як-от твердження: „01.04.10 трактор Т-150 з держ.№ ЧН0007 працював на підживлення пшениці озимої, виконав роботу в обсязі 10ет.га, витратив 30л дизпалива, нарахована пряма зарплата тракториста 80грн."
Мовлення	Документообіг на основі функціонуючої БД (Workflow)

З цієї позиції розробка та експлуатація БД є процесом визначення правил певної мови, заповненням її словників, і мовленням на ній.

[1] Критика, число 11–12 (145–146), https://shron1.chtyvo.org.ua/Yermolenko_Volodymyr/Tropik_Lievi-Strosa.pdf?PHPSESSID=lbdvjqgd065kq7pkth4l0tcrs1

Аксіоми Армстронга

Ці 3 тривіальні аксіоми складають повну множину правил, які шляхом логічного виводу дозволяють отримати будь-яку похідну функціональну залежність (FD), яка витікає з множини заданих FD.

1. *Рефлективність* (reflectivity). Якщо $\{B_1, B_2, ..., B_m\} \subseteq \{A_1, A_2, ..., A_n\}$, то $A_1, A_2, ..., A_n \to B_1, B_2, ..., B_m$. Тобто множина атрибутів залежить від охоплюючої її надмножини.
2. *Розширення* (augmentation). Якщо $A_1, A_2, ..., A_n \to B_1, B_2, ..., B_m$, то $A_1, A_2, ..., A_n, C_1, C_2, ..., C_k \to B_1, B_2, ..., B_m, C_1, C_2, ..., C_k$ для будь-якої множини $C_1, C_2, ..., C_k$.
3. *Транзитивність* (transitivity). Якщо $A_1, A_2, ..., A_n \to B_1, B_2, ..., B_m$ та $B_1, B_2, ..., B_m \to C_1, C_2, ..., C_k$, то $A_1, A_2, ..., A_n \to C_1, C_2, ..., C_k$.

Проекція функціональних залежностей

Нехай є відношення R з множиною FD F та ми виконуємо „проекцію” R, вилучаючи деякі атрибути з його схеми. Нехай S – відношення, отримане з R шляхом відкидання компонентів, які відповідають усім видаленим атрибутам в усіх кортежах R. Оскільки S є множиною, а не мультімножиною, співпадаючі кортежі в ньому заміняються одною копією.

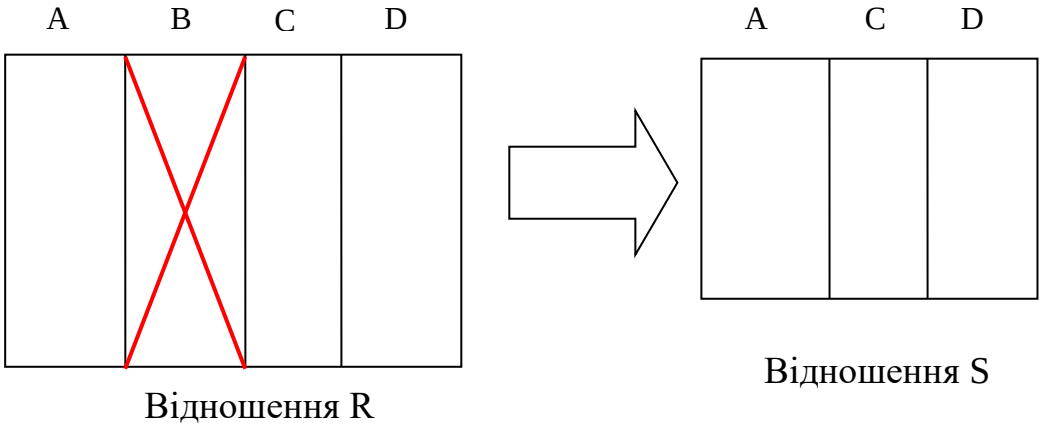


Рисунок 4.14 - Можливий наслідок вилучення атрибута при проекції відношення – зменшення кількості кортежів

Ті FD, які залишаються справедливими для S,

- випливають з F,
- включають атрибути, які належать лише S.

Можна їх обчислити, застосувавши аксіоми Армстронга. Складність пошуку в гіршому випадку експоненційно залежить від числа атрибутів S.

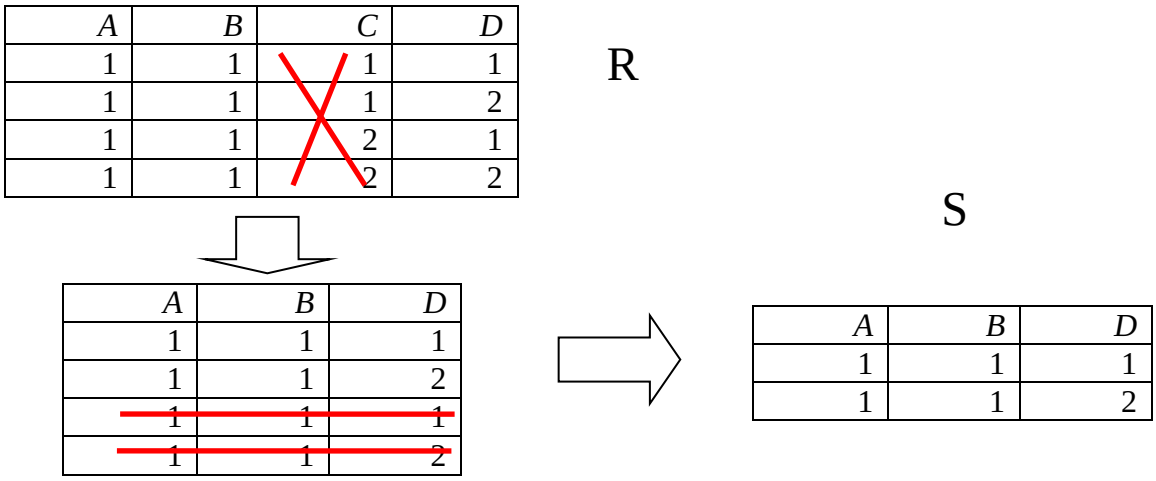


Рисунок 4.15 - Приклад проекції відношення

Приклад 4.13. Нехай відношення R(A, B, C, D) має FD $A \to B$, $B \to C$ та $C \to D$. Нехай нам треба відкинути атрибут B та перейти до схеми S(A,C,D). Шляхом математичного доведення отримуємо, що найпростіша множина FD, яка відповідає S, є:
$A \to C$, $C \to D$.
Першу з цих FD можна отримати з аксіоми транзитивності.

Наприклад, хай є відношення R, що містить FD $A \to B$, $B \to C$ та $C \to D$.

A	*B*	*C*	*D*
1	2	3	3
2	3	4	0
3	4	5	1
4	5	6	2

Виконаємо проекцію відношення R на S, вилучивши з нього атрибут B.

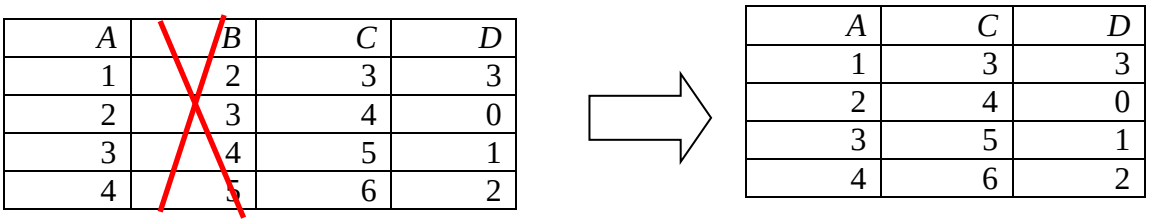


Рисунок 4.16 - Приклад проекції відношення зі збереженням FD

Проектування реляційних схем на основі форм нормалізації

Аномалії

Проблеми, подібні надмірності (див. Об'єднання відношень, приклад 4.6), які виникають при спробі „запхання" в одне відношення надмірної кількості атрибутів, називають *аномаліями* (anomalies).

title	*year*	*length*	*filmType*	*studioName*	*starName*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

Рисунок 4.17 - Відношення, яке містить аномалії (відтворений рис.4.10)

Різновиди аномалій:

1. *Аномалії надмірності* (redudancy anomalies). Однакові елементи інформації повторюються в кількох кортежах. (на рис.4.17 – у перших трьох і двох останніх).
2. *Аномалії зміни* (update anomalies). Один і той же фрагмент даних може змінитись в одному кортежі та не змінитись в іншому (наприклад, при зміні тривалості фільму Star Wars).
3. *Аномалії видалення* (deletion anomalies). Якщо певна підмножина кортежів стає пустою, це непрямо може потягнути втрати деякої іншої інформації. Наприклад, видаливши з відношення Movies (рис.4.17) дані про участь актора Еміліо Естевезе у фільмі Mighty ducks, ми повністю втрачаємо дані про цей фільм.

Декомпозиція відношень

Одним з прийнятних способів уникнення аномалій є *декомпозиція* (decomposing) відношень. Декомпозиція відношення R передбачає розбиття множини атрибутів R з метою побудови схем двох нових відношень з наступним занесенням у ці відношення певних кортежів R.

Відношення R зі схемою $\{A_1, A_2, ..., A_n\}$ може бути піддано декомпозиції у 2 відношення S і T, яким відповідають схеми $\{B_1, B_2, ..., B_n\}$ та $\{C_1, C_2, ..., C_n\}$, які задовольняють наступним умовам:

1. $\{A_1, A_2, ..., A_n\} = \{B_1, B_2, ..., B_m\} \cup \{C_1, C_2, ..., C_k\}$.

2. Кортежі відношення S є проекціями всіх кортежів R на множину атрибутів $\{B_1, B_2, ..., B_n\}$. Тобто, з кожного кортежу R відбираються лише ті компоненти (поля), які відповідають лише атрибутам $\{B_1, B_2, ..., B_n\}$. У разі утворення ідентичних кортежів у S заноситься лише один з них.

3. Аналогічно, кортежі відношення T є проекціями всіх кортежів R на множину атрибутів $\{C_1, C_2, ..., C_n\}$.

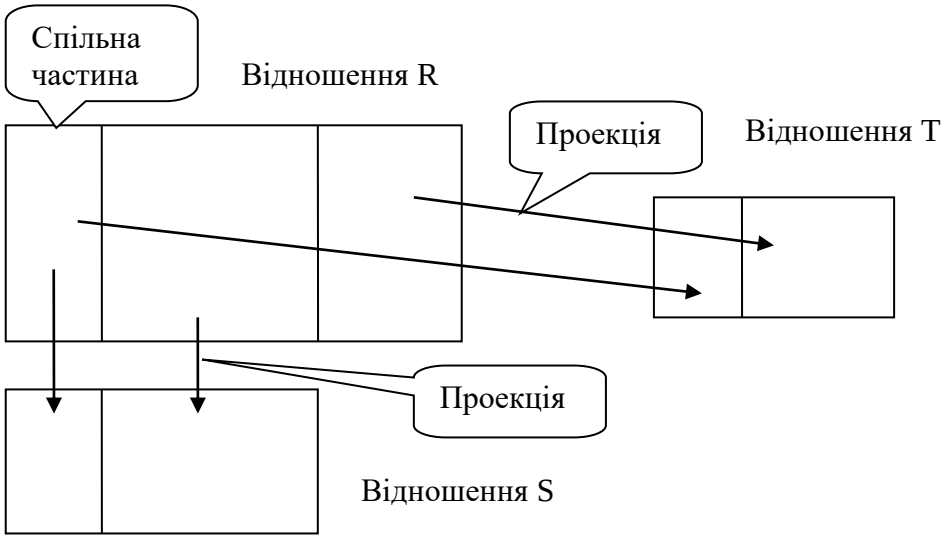


Рисунок 4.18 - Декомпозиція відношень

Приклад декомпозиції відношення

Приклад 4.13. Виконаємо декомпозицію відношення Movies (рис.4.17) наступним чином:

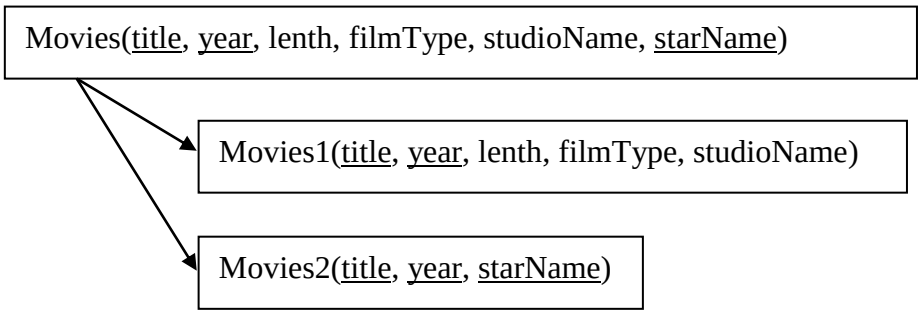
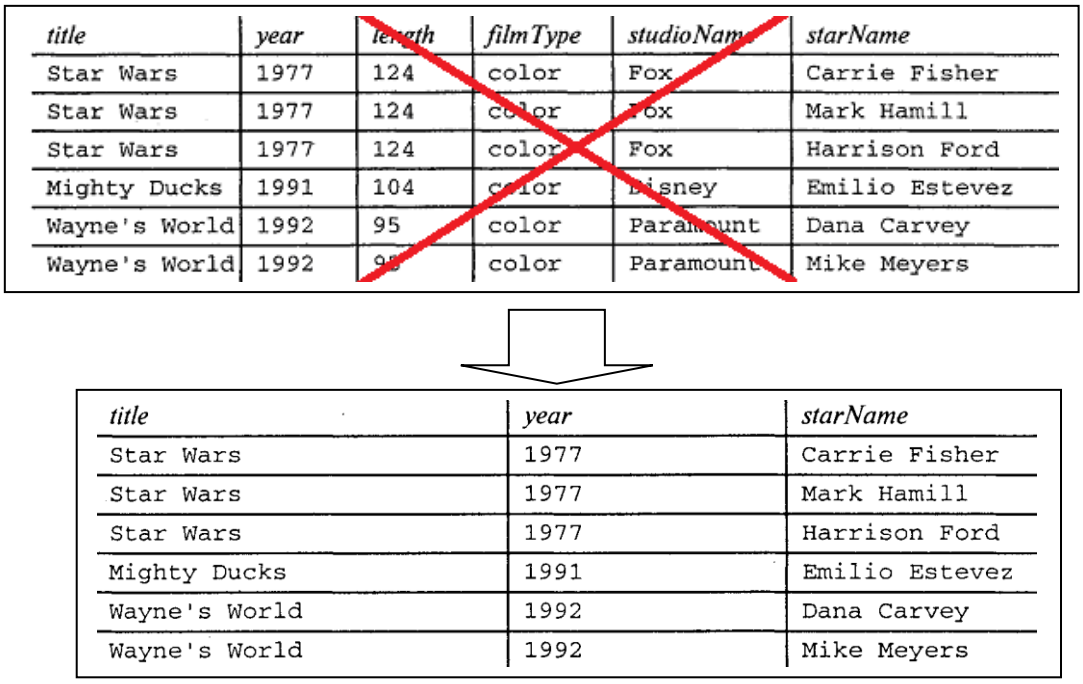


Рисунок 4.19. Декомпозиція відношення Movies

title	*year*	*length*	*filmType*	*studioName*	*starName*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

title	*year*	*length*	*filmType*	*studioName*
Star Wars	1977	124	color	Fox
Mighty Ducks	1991	104	color	Disney
Wayne's World	1992	95	color	Paramount

Рисунок 4.20 - Результат проекції екземпляра відношення Movies на схему Movies1



title	year	length	filmType	studioName	starName
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Mighty Ducks	1991	Emilio Estevez
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

Рисунок 4.21 - Результат проекції екземпляра відношення Movies на схему Movies2

Чи така декомпозиція ліквідує аномалії?

- *Надмірність* ліквідована, адже, напр., тривалість фільму включена у відношення Movies1 лише 1 раз.
- Небезпека аномалії *зміни* також ліквідована по тій же причині, адже, напр., ніяк не вдасться задати для тривалості фільму 2 різних значення.
- Імовірна аномалія *видалення* також ліквідована, адже при видаленні з Movies2 інформації про участь актора Еміліо Естевезе у фільмі Mighty ducks інформація про сам фільм залишається в Movies1.

Може здатися, що у відношеннях Movies1 та Movies2 існує надмірність, оскільки пари (title, year) повторюються кілька разів. Але це є ключові атрибути, і більш лаконічного способу ідентифікації фільму в межах даної схеми не існує. В разі зміни року, напр., пара (title, year) може вказувати на інший фільм з тою ж назвою та іншим роком випуску. (Більш лаконічний спосіб ідентифікації фільму в іншій схемі – сурогатний ключ Id.)

Нормальна форма Бойса-Кодда

Мета декомпозиції полягає в заміні одного відношення кількома іншими, які не містять аномалій. Існує проста умова, яка гарантує відсутність аномалій. Це *нормальна форма Бойса-Кодда* (Boyce-Codd normal form – BCNF).

> Відношення R задовольняє умові BCNF, якщо та тільки якщо ліва частина (аргумент) кожної його нетривіальної функціональної залежності містить ключ R.

Відношення Movies з рис. 4.17 не задовольняє умовам BCNF. Вище обгрунтовано, що ключом цього відношення є множина атрибутів {title, year, starName}. В той же час одна з двох функціональних залежностей, які містяться в ньому:

`title year → length filmType studioName`

не містить ключ у лівій частині.

Навпаки, відношення Movies1 (рис.4.19) задовольняє вимогам BCNF, оскільки його єдиним ключем є {title, year} та єдина FD, яка в ньому міститься,
`title year → length filmType studioName`,
містить цей ключ у лівій частині.

Декомпозиція в BCNF

Стратегія декомпозиції полягає в наступному: на кожному кроці треба знайти і виділити в окреме відношення нетривіальну FD, яка порушує умову BCNF.

Нехай відношення R містить нетривіальну FD A→B, яка порушує умову BCNF. Множина атрибутів розбивається на дві схеми, що перекриваються.

- Схема праворуч містить всі атрибути «порушуючої» FD.
- Схема ліворуч містить атрибути лівої частини «порушуючої» FD в сукупності з тими атрибутами, які не входять у «порушуючу» FD.

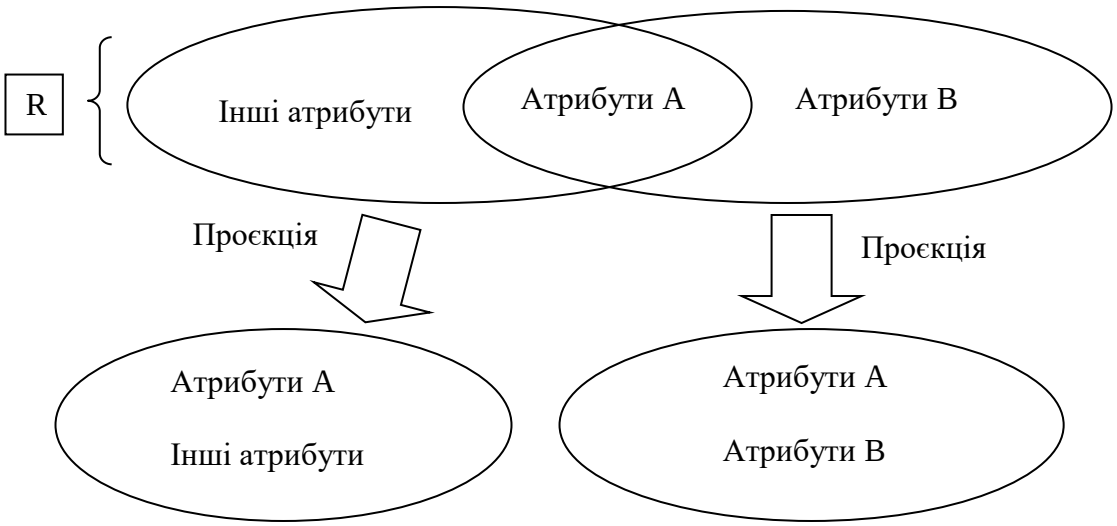


Рисунок 4.22 - Декомпозиція в BCNF

Приклад 4.14. Ще раз звернімося до аномального відношення рис.4.17. Розіб'ємо його схему на 2 схеми. Оскільки FD
`title year → length filmType studioName`
порушує BCNF,

- в першу схему винесемо всі атрибути цієї FD:
  `{title, year, length, filmType, studioName};`

- у другу схему винесемо всі атрибути відношення Movies, за виключенням тих, які входять у праву частину цієї FD, тобто:
  `{title, year, starName}.`

Приклад 4.15. Розглянемо відношення MovieStudio, яке містить інформацію про фільми, студії-власників фільмів та адреси студій.

title	*year*	*length*	*filmType*	*studioName*	*studioAddr*
Star Wars	1977	124	color	Fox	Hollywood
Mighty Ducks	1991	104	color	Disney	Buena Vista
Wayne's World	1992	95	color	Paramount	Hollywood
Addams Family	1991	102	color	Paramount	Hollywood

Рисунок 4.23 - Екземпляр відношення MovieStudio

Друга функціональна залежність **studioName → studioAddr,** яка захована в останніх двох атрибутах, продукує аномалію надмірності. Її ключ StudioName не містить ключа всього відношення MovieStudio (це {title, year}), і тим порушує умову BCNF. Для декомпозиції застосуємо ту ж попередню стратегію:

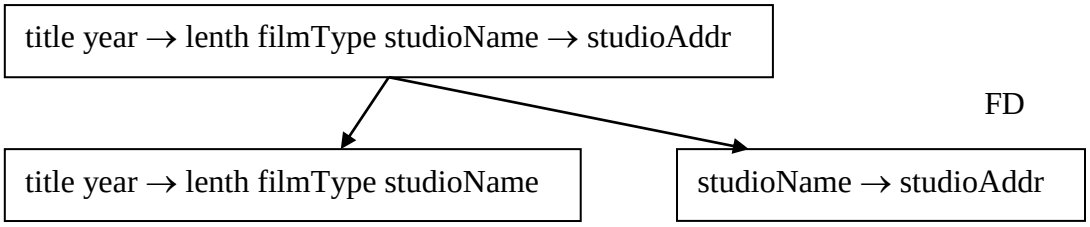


Рисунок 4.24 - Декомпозиція відношення MovieStudio

У друге відношення попадає виявлена функціональна залежність FD, а в перше – всі атрибути MovieStudio крім залежного атрибута FD (studioAddr).

title	*year*	*length*	*filmType*	*studioName*	*studioAddr*
Star Wars	1977	124	color	Fox	Hollywood
Mighty Ducks	1991	104	color	Disney	Buena Vista
Wayne's World	1992	95	color	Paramount	Hollywood
Addams Family	1991	102	color	Paramount	Hollywood

title	*year*	*length*	*filmType*	*studioName*
Star Wars	1977	124	color	Fox
Mighty Ducks	1991	104	color	Disney
Wayne's World	1992	95	color	Paramount
Addams Family	1991	102	color	Paramount

studioName	*studioAddr*
Fox	Hollywood
Disney	Buena Vista
Paramount	Hollywood

Рисунок 4.25 - Декомпозиція екземпляру відношення MovieStudio

Третя нормальна форма

Третя нормальна форма (third normal form – 3NF) є варіантом BCNF, незначно ослабленим для випадків, коли декомпозиція недоцільна або навіть неможлива.

Визначення (Едгар Кодд, 1971)

> Відношення R задовольняє умовам 3NF, якщо для її кожної нетривіальної FD або атрибути її лівої частини містять ключ відношення R, або атрибути її правої частини входять у певний ключ.

Різниця між BCNF та 3NF полягає в додатковій умові *або атрибути її правої частини входять у певний ключ.* Ця різниця охоплює досить рідкісні випадки (див.Ульман 2003, ст.134-135). На практиці різниці між цими формами нема. В проектуванні БД стандартом є саме 3NF, але її розуміють спрощено. Наочне визначення 3NF Кодда належить Біллу Кенту :

> Кожний неключовий атрибут повинен надавати інформацію про ключ, повний ключ і ні про що окрім ключа.

Це визначення часто доповнюється словами: so help me Codd (і хай поможе мені Кодд).

Ми завжди можемо, не ризикуючи втратити інформацію, виконати декомпозицію відношення у схеми, які задовольняють 3NF і дозволяють перевірити справедливість усіх FD шляхом зворотньої композиції в одне відношення. Якщо при цьому відношення не задовольняють вимогам BCNF, залишається імовірність певної надмірності.

Приклад 4.16 декомпозиції відношення (Ульман, ст.136).

Нехай є відношення:	*Його декомпозиція:*
R(A,B,C,D) з FD $B \rightarrow C$ *і* $B \rightarrow D$.	*R1(B,C); R2(B,D); R3(A,B)*
R(A,B,C,D,E) з FD $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow B$ і $D \rightarrow E$	*R1(A,B,C); R2(C,D); R3(D,B,E)*

Багатозначна залежність

Багатозначна залежність (multivalued dependency – MVD) – це твердження про те, що два атрибути або множини атрибутів незалежні один від одного. Точніше, багатозначна залежність

$A_1, A_2, ..., A_n \rightarrow\rightarrow B_1, B_2, ..., B_m$

для відношення R має місце, якщо для кожної пари кортежів *t* і *u* відношення *R*, які співпадають у всіх атрибутах $A_1, A_2, ..., A_n$, можна знайти в *R* певний кортеж *v*, співпадаючий:

- з кортежами *t* і *u* в атрибутах $A_1, A_2, ..., A_n$;
- з кортежем *t* в атрибутах $B_1, B_2, ..., B_m$;
- з кортежем *u* у всіх атрибутах відношення *R*, які не належать ні множині $\{A_1, A_2, ..., A_n\}$, ні множині $\{B_1, B_2, ..., B_m\}$.

Як наслідок, для будь-яких фіксованих значень атрибутів $A_1, A_2, ..., A_n$ пов'язані з ними значення $B_1, B_2, ..., B_m$ та інших атрибутів присутні в будь-яких можливих поєднаннях.

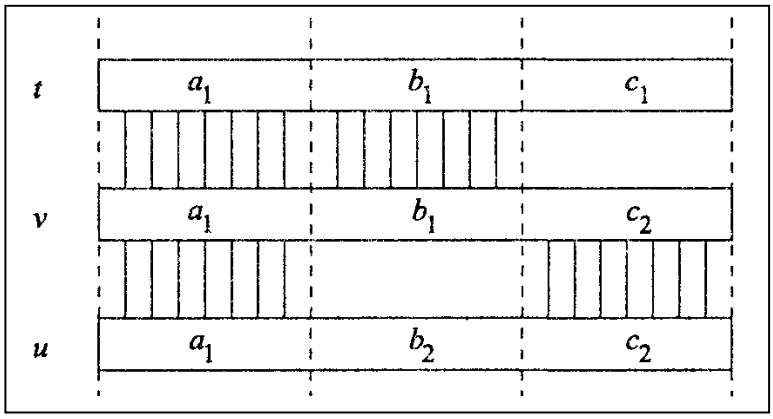


Рисунок 4.26 - Багатозначна залежність гарантує існування кортежу v.

c не визначається a і b. b також не визначається a і c.

Приклад 4.17. Нехай є відношення, в яке входить список навчальних дисциплін, рекомендована література і імена лекторів, що читають відповідні курси:

Таблиця 4.1.Навчальні дисципліни

	Атрибути A	*Атрибути B*	*Інші атрибути*
	Дисципліна	*Книга*	*Лектор*
t	Історія середніх віків	Грушевський	Іваненко
v	Історія середніх віків	Грушевський	Петренко
	Історія середніх віків	Яковенко	Іваненко
u	Історія середніх віків	Яковенко	Петренко
	Історія середніх віків	Грушевський	Коваленко
	Історія середніх віків	Яковенко	Коваленко
	Історія двадцятого сторіччя	Гунчак	Іваненко
	Історія двадцятого сторіччя	Попович	Петренко
	Історія двадцятого сторіччя	Попович	Іваненко
	. . .		

Як бачимо, має місце незалежність книг від лекторів, оскільки лектори рекомендують ті самі книги. Якщо з'явиться нова вартісна книга, не виключено, що її порекомендує кожний лектор. Існує кортеж v, який демонструє багатозначність залежності від A:

- Дисципліна і рекомендована книга ті самі, а лектор інший.
- Або (що те саме): Дисципліна і лектор ті самі, а рекомендована книга інша. В цьому разі *t* і *u* ті самі, а роль „руйнівника" *v* грає третій атрибут.

Приклад 4.18. Будемо виходити з того, що актори, як заможні люди, можуть мати кілька адрес. Спробуємо надати в одному відношенні акторів, фільми і (вигадані)адреси.

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return to the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Return to the Jedi	1983

Рисунок 4.27 - Множина адрес не залежить від множини фільмів.

Попри очевидну надмірність, це відношення не порушує умови BCNF, оскільки взагалі не містить нетривіальних FD. Напр, атрибут *city* не обумовлений жодним з інших атрибутів. Певний актор може володіти двома будинками в різних містах, але з однаковою адресою. Тому можливе існування двох кортежів, які співпадають в усіх атрибутах, крім *city*.

Нетривіальна багатозначна залежність

MVD $A_1, A_2, ..., A_n \rightarrow\rightarrow B_1, B_2, ..., B_m$ для відношення R є *нетривіальною* (nontrivial), якщо

1. Жоден з атрибутів $B_1, B_2, ..., B_m$ не співпадає з атрибутом з $A_1, A_2, ..., A_n$;
2. Не всі атрибути R належать $\{A_1, A_2, ..., A_n\}$ та $\{B_1, B_2, ..., B_m\}$.

Четверта нормальна форма

Умова *четвертої нормальної форми* (fourth normal form – 4NF) є умовою BCNF, але застосовується до MVD, а не FD.

> Відношення знаходиться в 4NF, якщо воно знаходиться в BCNF і не містить нетривіальних багатозначних залежностей.

Тобто всі залежності є, по суті, функціональними (однозначними) залежностями від ключів відношення.

На рис.4.27 наведено приклад відношення, яке задовольняє умови BCNF, але порушує умови 4NF. В ньому жодний атрибут не залежить від інших атрибутів.

Декомпозиція в 4NF

Алгоритм 4NF-декомпозиції аналогічний алгоритму BCNF-декомпозиції. Згідно факту порушення умови 4NF існує певна нетривіальна MVD $A_1, A_2, ..., A_n \rightarrow\rightarrow B_1, B_2, ..., B_m$, така, що множина $\{A_1, A_2, ..., A_n\}$ не містить ключ. Ця MVD може бути визначена явно або виведена з відповідної FD. Затим схема відношення R, у якій помічено порушення умови 4NF, розбивається на 2 схеми, які містять:

1. Множини атрибутів $A_1, A_2, ..., A_n$ та $B_1, B_2, ..., B_m$;
2. Множину атрибутів $A_1, A_2, ..., A_n$ та всі атрибути R, які не належать ні $A_1, A_2, ..., A_n$, ні $B_1, B_2, ..., B_m$.
3. Якщо одна або кілька схем, отриманих у результаті одного кроку декомпозиції, все ще порушує умову 4NF, процес декомпозиції треба продовжити.

Тобто треба послідовно вилучати з відношення MVD, поки вони є.

Приклад 4.19. Продовжимо розгляд прикладу 4.18. Ми встановили, що MVD

`name ↠ street city`

порушує умови BCNF. Правило декомпозиції диктує: треба замінити схему з п’ятьма атрибутами двома схемами:

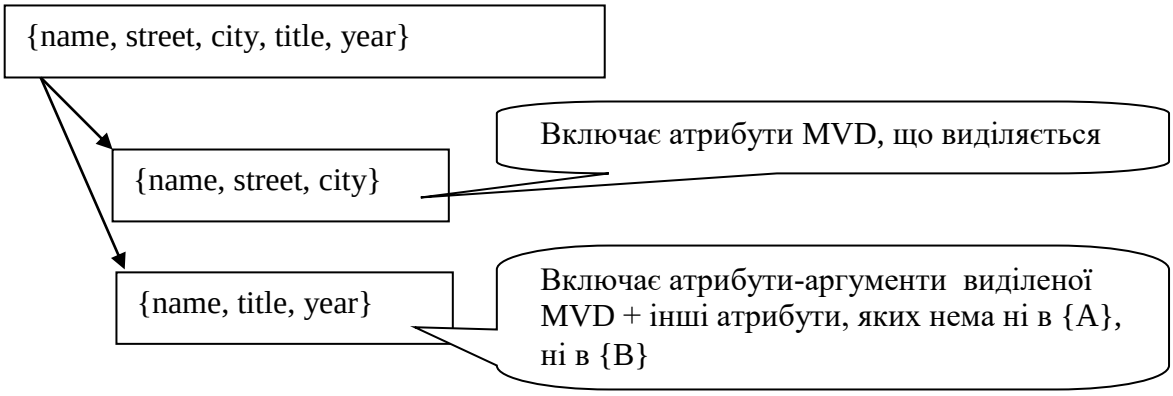


Рисунок 4.28 - Декомпозиція MVD в 4NF

Співвідношення нормальних форм

Інші нормальні форми:

- *Перша нормальна форма* (first normal form) встановлює просту вимогу: кожний компонент кожного кортежу має містити атомарне значення.
- *Друга нормальна форма* (second normal form) – менш жорсткий варіант третьої нормальної форми, який припускає наявність у відношенні транзитивних FD (як у останньому прикладі 4.15, де адреса студії є атрибутом фільму), але забороняє існування нетривіальних FD з лівою частиною, яка є підмножиною ключа (як у прикладі 4.13 - Movies(title, year, lenth, filmType, studioName, starName)).
- Існують також *п'ята*, *шоста* та інші нормальні форми (див.Вікіпедію).

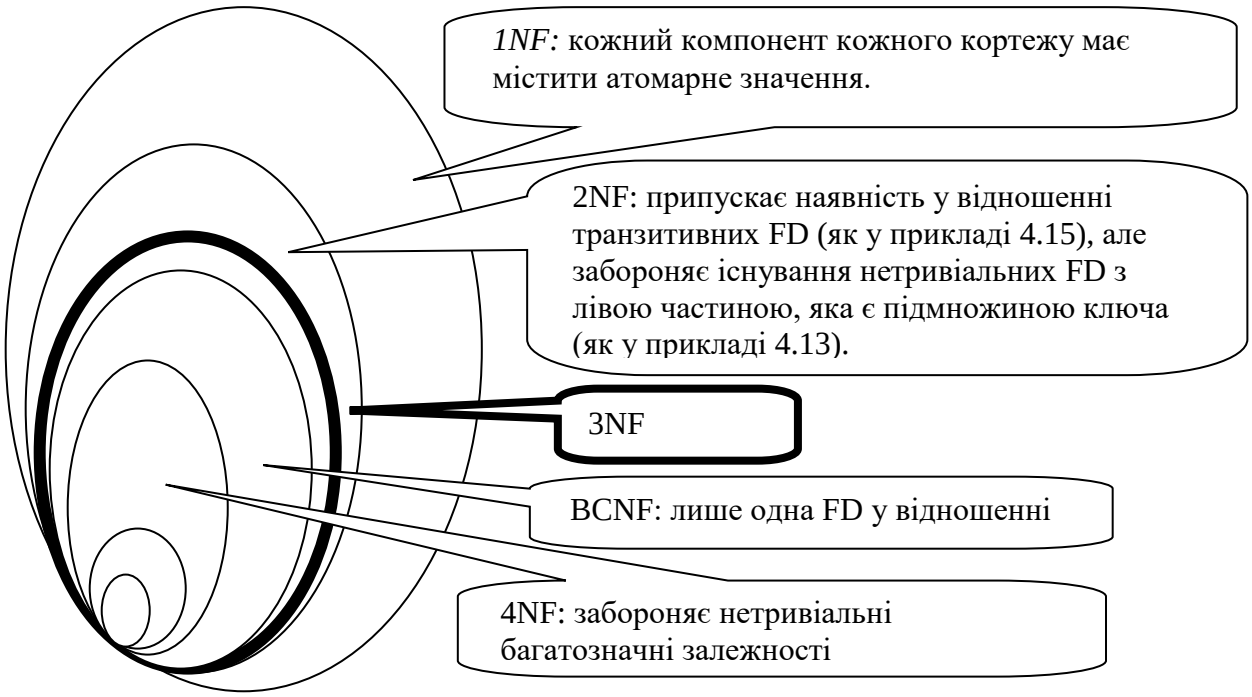


Рисунок 4.29 - Співвідношення нормальних форм

Переваги, які надають найбільш популярні нормальні форми:

Таблиця 4.2. Переваги нормальних форм

Властивість	*3NF*	*BCNF*	*4NF*
Відсутність надмірності з-за FD	У більшості випадків	Так	Так
Відсутність надмірності з-за MVD	Ні	Ні	Так