

# КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ БАЗИ ДАНИХ НА ОСНОВІ ER-МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ

Як правило, БД розробляється не окремо, а як частина інформаційної системи. Ні в ГОСТах, ні в Уніфікованому процесі, ні в інших стандартах не конкретизуються технології проектування і розробки БД. Сформулюємо послідовність дій по створенню БД відповідно до усталеної практики.

Зазвичай прийнято починати з вивчення *понять і описів* інформації, що підлягає моделюванню, а потім намагатися відображувати їх в рамках *ER-моделі*. Потім ER-проект перетвориться в *реляційну схему*, виражену засобами мови визначення даних для конкретної СУБД. В більшості випадків СУБД ґрунтуються на *реляційній моделі*. В цьому разі в ході прямолінійного процесу абстракція реалізується у конкретній *реляційній схемі БД* (relational database schema).

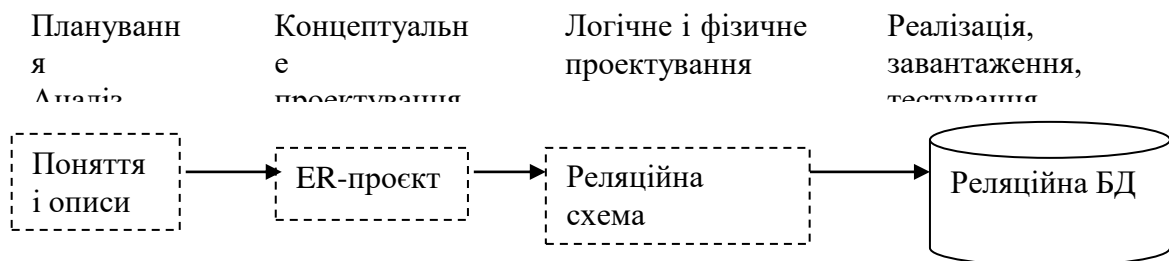


Рисунок 2.1 - Процес моделювання і реалізації БД

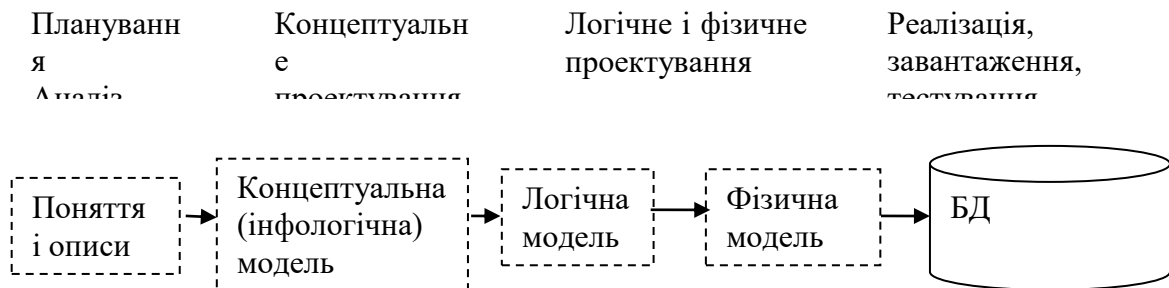


Рисунок 2.2 – Подання процесу моделювання і реалізації БД, від якого ми свідомо відмовились. ER-модель є інфологічною моделлю, а фізична модель створюється безпосередньо засобами СУБД.



## Модель «сутність-зв'язок»

Модель «сутність-зв'язок» була запропонована в 1976 році Пітером Пін-Шен Ченом (англ. Peter Pin-Shan Chen)<sup>1</sup> – американським професором комп'ютерних наук в університеті штату Луїзіана.

В ER-моделі структура даних відображається графічно, у вигляді діаграми сутностей і зв'язків (entity-relationship diagram), що складається з елементів трьох основних типів:

- a) множин сутностей;
- b) атрибутів;
- c) зв'язків.

<sup>1</sup> <http://cs-exhibitions.uni-klu.ac.at/index.php?id=185>

## Сутності

Сутність у філософії – первинне поняття, якому важко дати визначення. Згідно Аристотелю, сутність (субстанція) – це те, що існує, і вона має атрибути.

*Сутність* (entity) — це абстрактний об'єкт певного виду. Набір однорідних сутностей утворює множину сутностей (entity set).

*ER-проектування*

Сутність

Множина сутностей

*ОО-проектування*

Об'єкт

Клас об'єктів

*Приклад 2.1* стосується бази даних щодо кінофільмів, в яких беруть участь актори, студій, що здійснили зйомку, і т.п. Кожен з фільмів є сутністю, а колекція всіх фільмів є множиною сутностей. Актор, що знімається у фільмах, також є сутністю, але іншого виду, і їх множина — це множина сутностей. Кіностудія — це сутність ще одного виду, а перелік кіностудій формує третя множина сутностей.

## Атрибути

Множині сутностей відповідає набір *атрибутів* (attributes), які є властивостями сутностей цієї множини. Наприклад, множині сутностей *Movies* (кінофільми) можуть відповідати атрибути *title* (назва) і *length* (тривалість фільму у хвиликах). В нотації Чена ER-моделі виходимо з того, що атрибути мають атомарні значення (як-от рядки, цілі або дійсні числа і т.п.). В інших нотаціях ER-моделі атрибутами можуть бути:

- структура (як у С),
- кортеж, який містить фіксоване число атомарних компонентів,
- або множина значень одного типу.

## Зв'язки

...	...
Актор s	Фільм l
Актор s	Фільм m
Актор s	Фільм n
...	...
Актор t	Фільм n
...	...

*Зв'язки* - це з'єднання між двома або більше множинами сутностей. Якщо, напр., *Movies* (кінофільми) та *Stars* (актори) – дві множини сутностей, то можлива наявність зв'язку *Stars-in* („актори-учасники”, які знялись у фільмі), який з'єднує множини сутностей *Movies* і *Stars*: сутність *m* множини сутностей *Movies* з'єднана з сутністю *s* множини сутностей *Stars* зв'язком *Stars-in*, якщо актор *s* знявся в фільмі *m*.

Рисунок 2.2 - Зв'язок між сутностями *Актор* та *Фільм*

## Діаграма сутностей і зв'язків

*Діаграма сутностей і зв'язків* (entity-relationship diagram), або *ER-діаграма* - це графічне представлення множин сутностей, їхніх атрибутів та зв'язків. Множини сутностей і атрибути є вершинами графу, з'єднані зв'язками, і для вказання належності елемента до певного виду використовуються:

- *прямокутник* – для множин сутностей,
- *овал* – для атрибутів,
- *лінія*, або стрілка, та, можливо, ромб – для зв'язків.

Ребра графа з'єднують множини сутностей з атрибутами і служать для представлення зв'язків між множинами сутностей.

Приклад 2.2. На рис.2.3 наведено ER-діаграму зі структурою простої БД, яка містить інформацію про кінофільми.

Таблиця 2.1 - Сутності і атрибути БД фільмів

Назва сутності	Призначення сутності	Назва атрибуту
Movies	Кінофільми filmType – тип плівки, може бути <i>color</i> або <i>blackAndWhite</i>	title year length filmType
Stars	Кінозірки	name address
Studios	Студії, які випускають фільми	name address

Зв'язок Owns пов'язує фільм із кіностудією, яка випустила фільм і володіє правами на нього. Стрілка у сутності Studios свідчить про те, що кожний фільм має одну і лише одну студію-власника.

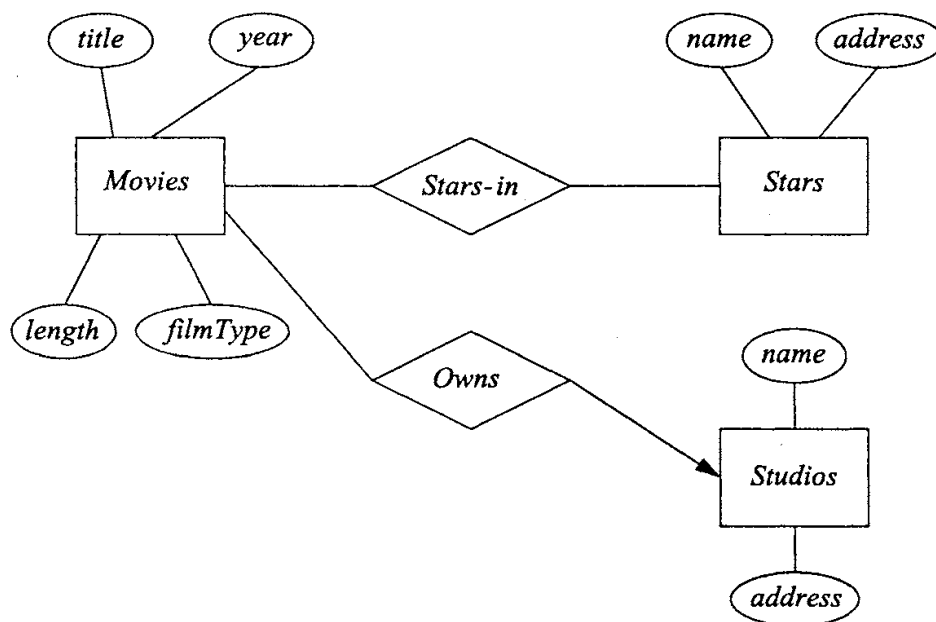


Рисунок 2.3 - Діаграма сутностей і зв'язків для бази кінофільмів

БД, яка відповідає конкретній ER-діаграмі та містить конкретний набір даних, є екземпляром БД (database instance). Множині сутностей у ній відповідає конкретний набір сутностей, кожна сутність має атрибути з певними значеннями. Зв'язок можна описати таблицею, як-от:

Таблиця 2.2 - Екземпляр зв'язку Stars-in для екземпляру бази кінофільмів

<i>Movies</i>	<i>Stars</i>
Basic Instinct	Sharon Stone
Total Recall	Arnold Schwarzenegger
Total Recall	Sharon Stone

### Множинність сутностей

Нехай  $R$  – зв'язок, що з'єднує множини сутностей  $E$  та  $F$ . Тоді можливе виконання однієї з трьох умов:

1. Якщо кожний член множини  $E$  зв'язком  $R$  може бути з'єднаний не більш як з одним членом  $F$ , кажуть, що  $R$  є зв'язок типу „багато до одного” (many-one relationship), спрямований від  $E$  до  $F$ . У цьому разі кожна сутність  $F$  припускає з'єднання з багатьма членами  $E$ . Якщо ж навпаки, кожний член множини  $F$  зв'язком  $R$  може бути з'єднаний не більш як з одним членом  $E$ , кажуть, що  $R$  є зв'язок типу „один до багатьох” (one-many relationship), якщо читати від  $E$  до  $F$ . У ER-діаграмі стрілка ставиться на стороні „1”. Приклад: зв'язок Owns на рис. 2.3.
2. Якщо зв'язок  $R$  в обох напрямках, від  $E$  до  $F$  та від  $F$  до  $E$ , належить до типу *багато до одного*, кажуть, що  $R$  є зв'язком „один до одного” (one-one relationship). У цьому разі кожний елемент однієї множини сутностей припускає з'єднання не більше ніж з одним елементом іншої множини сутностей.
3. Якщо зв'язок  $R$  в жодному напрямку, від  $E$  до  $F$  та від  $F$  до  $E$ , не належить до типу *багато до одного*, кажуть, що  $R$  є зв'язком „багато до багатьох” (many - many relationship).

Приклад зв'язку *багато до багатьох* на малюнку 2.3: Stars-in .

Приклад зв'язку *один до одного*: нехай студію очолює президент.



Рисунок 2.4 - Зв'язок типу *один до одного*.

Доречно уявити, що студію очолює лише один президент, і кожний президент очолює лише одну студію.

Такий зв'язок указує на *не більш ніж один* елемент кореспондуючої множини сутностей, проте наявність такого елемента не гарантована. Приклади:

- У „президента” має бути студія, інакше який же він президент.
- Проте студія тимчасово може обходитись без президента.

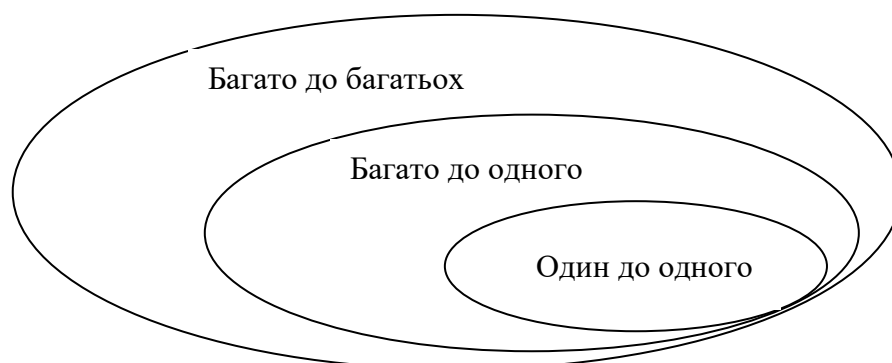


Рисунок 2.5 - Співвідношення типів зв'язків

Зв'язки *Один до одного* є підмножиною зв'язків *Багато до одного*.

Зв'язки *Багато до одного* є підмножиною зв'язків *Багато до багатьох*.

### Багатосторонні зв'язки

Крім бінарних зв'язків, у реальних ситуаціях *тернарні зв'язки* (ternary relationships) зрідка, але все ж зустрічаються. Приклад:

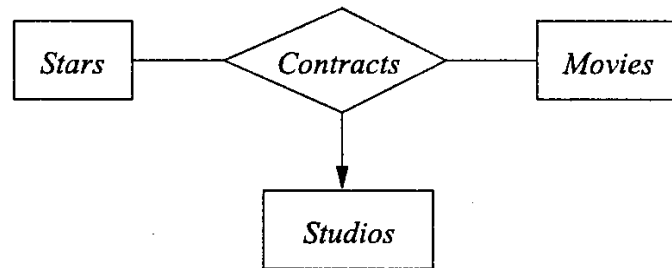


Рисунок 2.6 - Тернарний зв'язок

Зв'язок на рис. 2.6 відображає факт укладання контракту між студією і актором, який зобов'язується взяти участь у зйомках певного фільму. Таким чином, зв'язок *Contracts* може бути описаний набором кортежів виду (studio, star, movie).

Стрілка до сутності *Studios* означає наступне: якщо ми оберемо по одній сутності зі всіх інших сутностей, охоплених зв'язком, ці сутності можуть бути пов'язані не більше ніж з одним елементом множини *Studios*. Це узагальнення правила стрілки для бінарних зв'язків. Таким чином,

Кожний актор для зйомки певного фільму може => Стрілка до *Studios*  
укласти контракт лише з одною студією

Будь-яка студія може запросити для знімання фільму => Лінія до *Stars*  
кілька акторів

Будь-який контракт може передбачати участь актора => Лінія до *Movies*  
у кількох фільмах

### Зв'язки та ролі

Можливо, що одна сутність згадується в контексті одного зв'язку багатократно. Це також відображається лініями на діаграмі. Кожна лінія, спрямована до множини сутностей, є окремою *ролю* (role), в якій множина виступає в цьому конкретному випадку. Роль позначається текстовою міткою, в ромбі або без нього.



Рисунок 2.7 – Різні ролі двох паралельних зв'язків

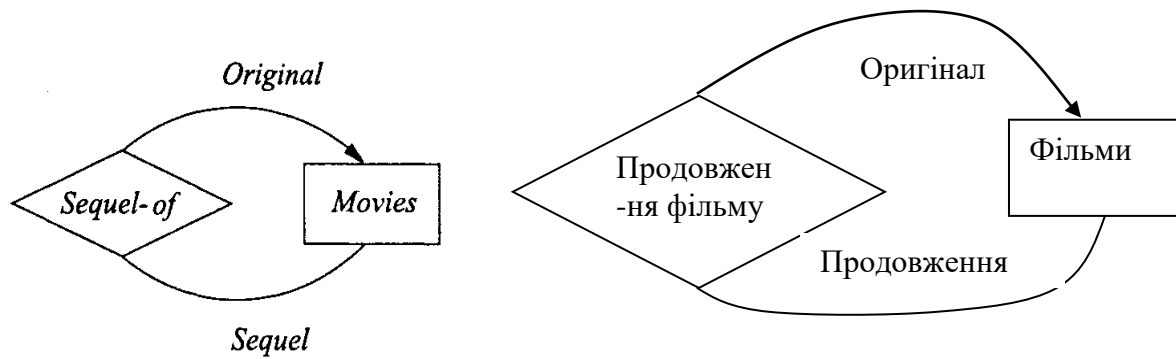


Рисунок 2.8 - Різні ролі зв'язку у його різних кінцях

Виходимо з того, що певний фільм може мати кілька продовжень, але для кожного продовження існує лише один „оригінальний” фільм. Таким чином, зв'язок *Sequel-of*, що з'єднує фільми-продовження з фільмами-оригіналами, належить до типу *Багато до одного*.

### Атрибути зв'язків

Буває зручно пов'язувати атрибут не з сутністю, а зі зв'язком. Для тернарного зв'язку (рис. 2.6) спробуємо зафіксувати розмір винагороди актора, встановлений згідно контракту. Правильно було б пов'язати винагороду не з фільмом, актором або студією, а з кортежем

(star, movie, studio)

з множини даних зв'язку *Contracts*.

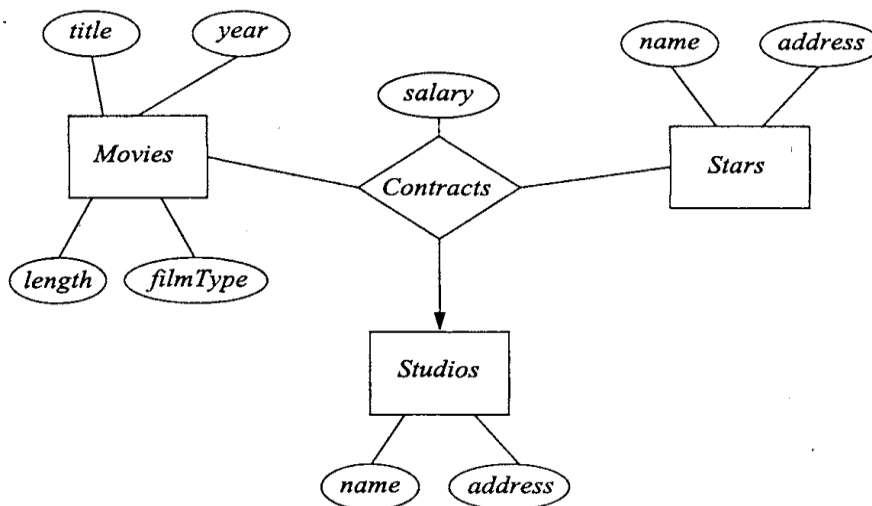


Рисунок 2.9 - Зв'язок з атрибутом

Якщо зв'язок має атрибути, він може бути замінений на з'єднуючу множину сутностей з тими ж атрибутами. В разі діаграми рис.2.10 ця сутність також може мати назву *Contracts*.

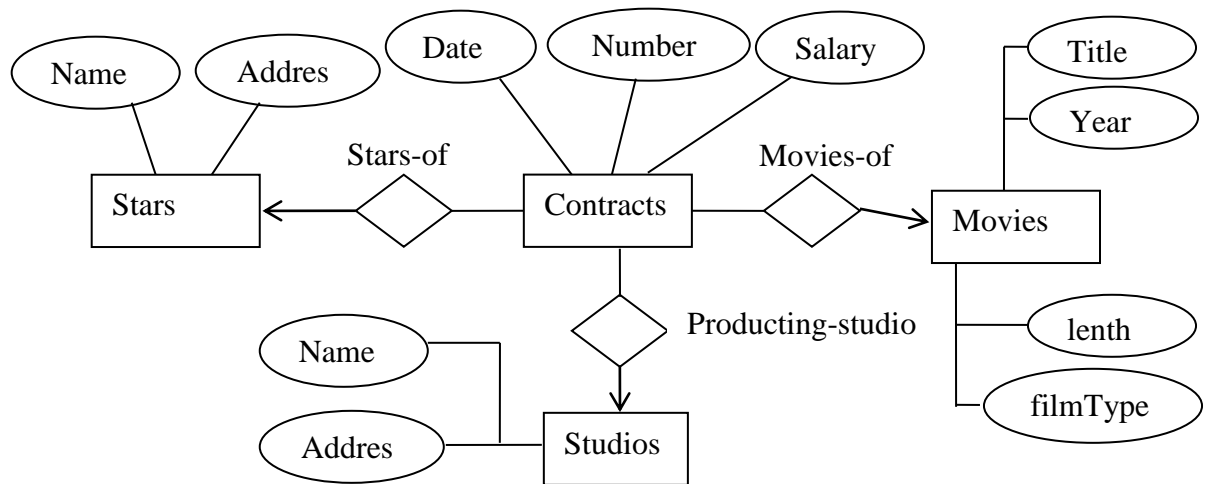


Рисунок 2.10 - ER-діаграма після передачі атрибутів від зв'язку до з'єднуючої множини сутностей. Тут контракт з одним актором на один фільм.

На практиці з'єднуюча сутність завжди має атрибути, як у наступному прикладі. В галузі інвестування житла в основі лежить тристоронній договір між інвестором (фізична або юридична особа), забудовником (будівельна фірма) і банком:

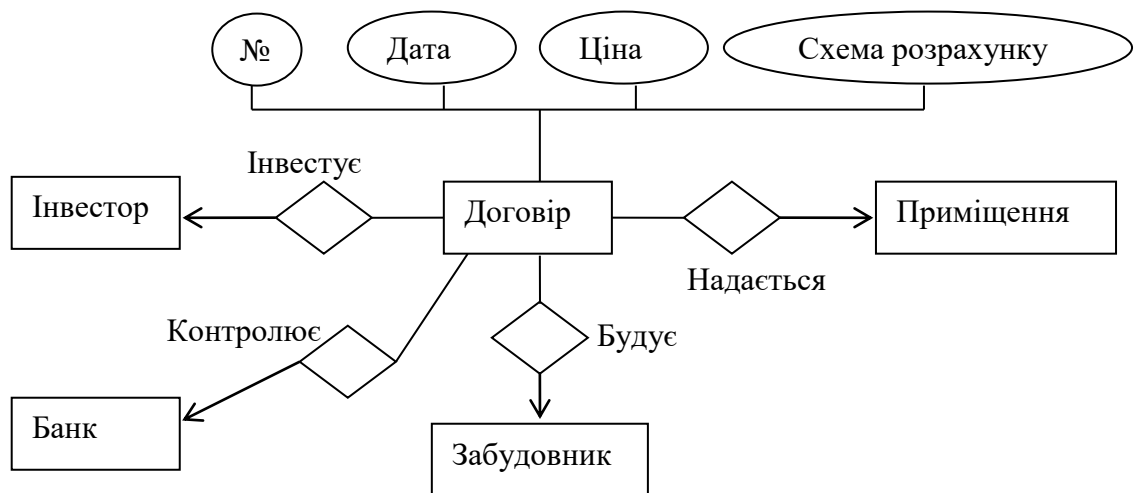


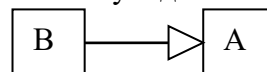
Рисунок 2.11 - Приклад ER-моделі тристороннього договору в галузі інвестування житла. Цей зв'язок чотирьохсторонній.

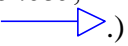
### Підкласи в ER-моделі

Нерідко буває, що деякі сутності множини сутностей, крім спільних властивостей, мають спеціальні властивості, не властиві іншим членам множини. В подібних випадках доцільно створювати спеціальні множини сутностей – *підкласи* (subclasses), кожна з яких має власний набір атрибутів та/або зв'язків. Для з'єднання „повної” множини сутностей – *базового класу* (superclass) з його підкласами використовуються зв'язки *isa* (від *is a*, можемо позначити *ε*.)

Напр., твердження  $B \in A$  виражає наявність зв'язку від множини сутностей (підкласу)  $B$  до

множини сутностей  $A$  – базового класу.



На ER-діаграмі зв'язок *isa* позначаємо стрілкою з трикутником («закрита стрілка»), спрямованим до базового класу. Кожний зв'язок *isa* належить до типу *один до одного*, але стрілки на лініях не проставляються (або проставляються стрілки з трикутником ).

Простий приклад зв'язків *isa*:

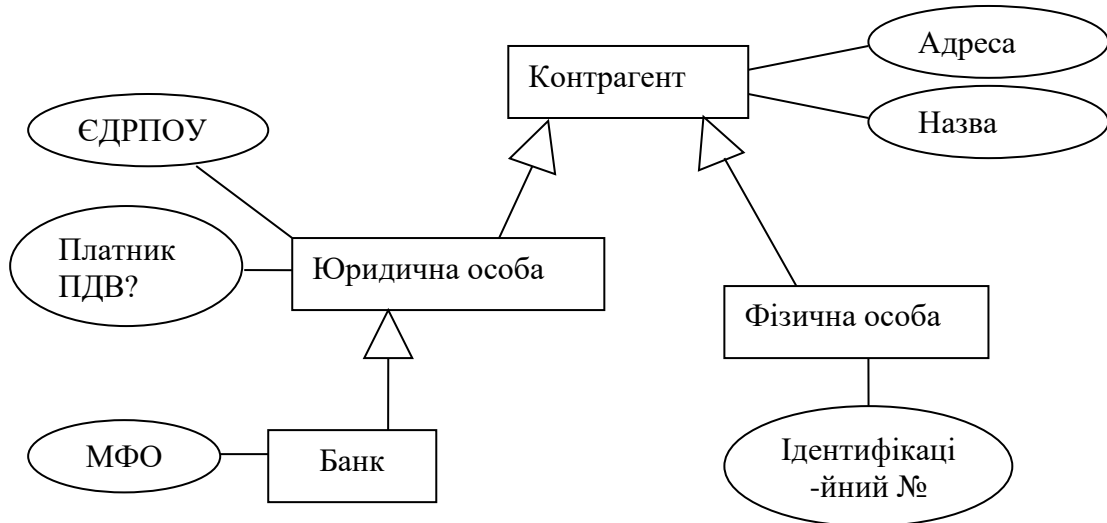


Рисунок 2.12 - Приклад зв'язків *isa* (є)

Сутність підкласу має всі атрибути базового класу + специфічні атрибути підкласу. Підклас бере участь у зв'язках базового класу + у специфічних зв'язках підкласу.

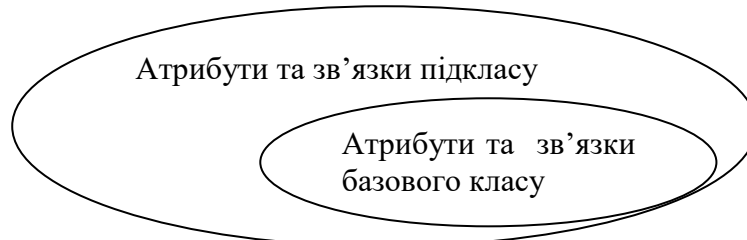


Рисунок 2.13 - Співвідношення між атрибутами та зв'язками базового класу та підкласу

### Підкласи в об'єктно-орієнтованих системах

Існує значна подібність між зв'язком *isa* та відношенням підкласу до базового класу в об'єктно-орієнтованій мові. Розбіжності: в ER-проектванні об'єкт може належати до різних підкласів, в ООП це, як правило, неможливо. Приклад:

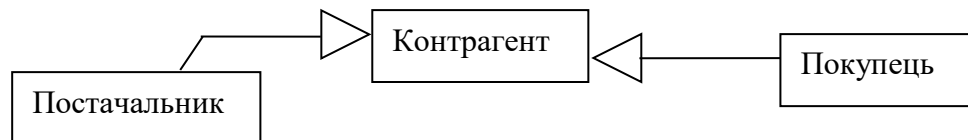


Рисунок 2.14 - Приклад можливої належності сутності до двох підкласів

У прикладі на рис.2.14 контрагент одночасно може бути і постачальником, і покупцем. В ООП для такого випадку доводиться створювати окремий підклас.

### Приклад ER-діаграми генеалогічної БД

Нижче наведений приклад ER-діаграми генеалогічної БД, яка містить зв'язки материнства, батьківства, шлюбу.



Не розглядаються:

- одностатеві шлюби;
- гареми (полігамія);
- проміскуїтет;
- шведська родина;
- розірвані шлюби.

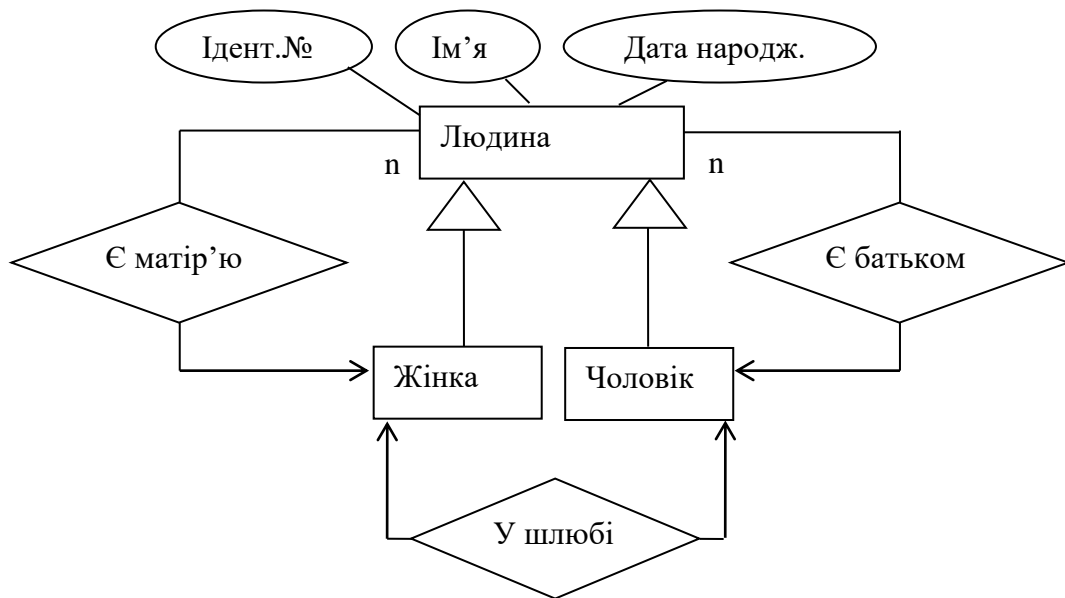


Рисунок 2.15 - ER-діаграма генеалогічної БД

Термін *проміскуїтет* був введений в XIX столітті для позначення стадії нерегульованих статевих стосунків у первісному людському суспільстві, що передували виникненню шлюбу і сім'ї. [<http://ru.wikipedia.org/wiki/Промискуитет> ]

*Шведська сім'я* — так в просторіччі називається форма *поліаморії*, при якій троє людей різної статі (один чоловік і дві жінки або двоє чоловіків і одна жінка) живуть разом, під одним дахом.

## Принципи проєктування ER-моделі

Найбільші вади у проєкт закладаються на перших етапах, зокрема, етапі проєктування ER-моделі. Ці вади існують у проєкті довгі роки, оскільки їх важко виправити. Тому до цього етапу треба підходити дуже обачно. Виділяємо наступні принципи ER-проєктування:

### Достовірність

Множини сутностей та їхні атрибути мають відповідати реальним вимогам. Перед тим як вводити в модель нову множину сутностей, атрибут або зв'язок, треба спитати себе, чи все достеменно відомо про ту частину реального світу, яка має бути змодельована.

### Відсутність надмірності

Зберігання одної інформації в кількох місцях крім зайвого розходу пам'яті, тягне більш серйозні наслідки: потенційні протиріччя (якщо зміни вносяться не у всі місця).

Приклад надмірності: в базі фільмів (рис.2.3) крім зв'язку Owns зберігати назву студії як властивість фільму.

## Простота

Включаєте в проєкт лише ті структурні елементи, без яких ніяк не можна обійтись.

## Вибір підходящих зв'язків

Коли ми вирішуємо питання, чи потрібний той чи інший зв'язок, загальна відповідь така: „Невідомо; рішення залежить від того, який сенс ми вкладаємо в кожний зв'язок.”

Приклад: У базі фільмів чи потрібний зв'язок *Stars-in*, чи не можна цю інформацію отримати з множини сутностей Contracts? Відповідь залежить від сенсу, який ми вкладаємо в Contracts. Зокрема, під цим ми можемо розуміти довгострокові контракти, які прямо не пов'язані зі зйомками фільмів; артист може бути наданий іншій студії для зйомок згідно окремого контракту між двома студіями.

## Використання елементів адекватних типів

Важлива функція ER-моделі – визначити її межі, тобто що ми включаємо і що не включаємо в модель. Зв'язок *Багато до одного* може бути замінений ідентифікуючим атрибутом, якщо окрема сутність із власними атрибутами не потрібна, і навпаки.

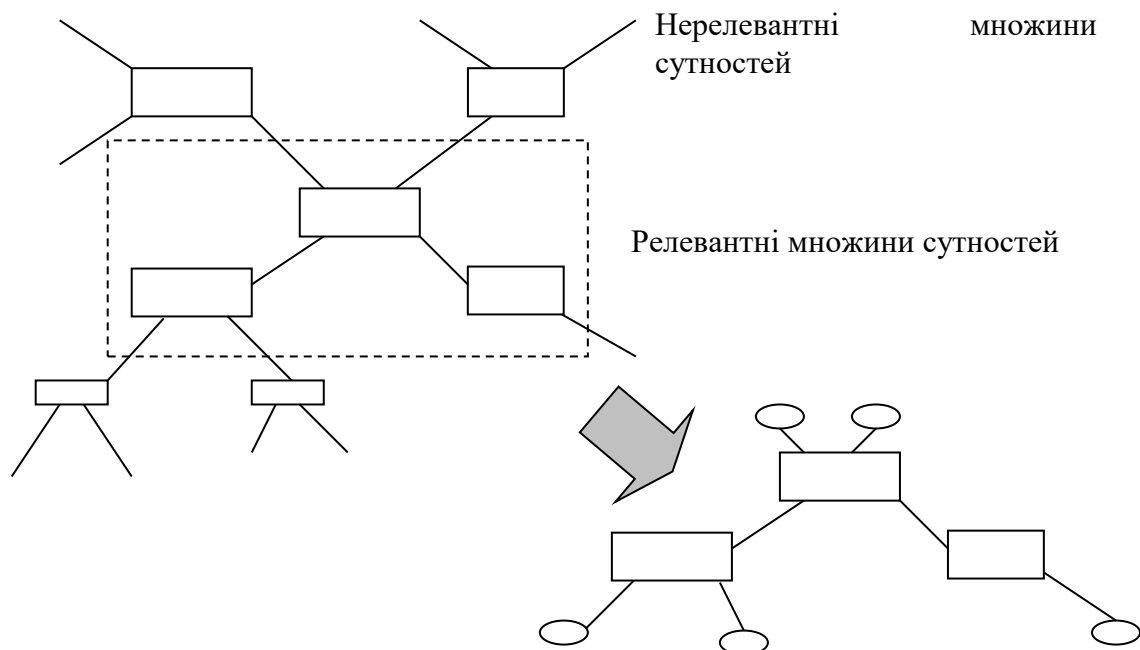


Рисунок 2.16 - Ілюстрація використання елементів адекватних типів у ER-проєктуванні.

## Моделювання обмежень у ER- моделі

### Класифікація обмежень

Приблизна класифікація обмежень:

1. *Ключ (key)* – атрибут або підмножина атрибутів, що унікальним чином визначає певну сутність у складі підмножини сутностей. Ніякі дві сутності не можуть мати однакові комбінації значень атрибутів, що складають ключ. Втім, припустимим є співпадання окремих, але не всіх, значень ключових атрибутів будь-яких двох сутностей.
2. *Обмеження унікальності (single-value constraint)*: певне значення в певному контексті має бути унікальним. Ключ – приклад обмеження унікальності. Взагалі, це може бути значення на стороні *один* у зв'язку *багато до одного*.
3. *Обмеження посилальної цілісності (referential integrity constraint)*: певне значення, на яке посилається інший об'єкт, має існувати в БД. Це є заборона на „висячі” посилання.
4. *Обмеження домену (domain constraint)*: довільна вимога, яка також має бути зафіксована в БД, як-от: кожній сутності *кінофільм* може бути співставлено не більше 10 сутностей *актор*.

### Ключі в ER-моделі

Мають місце такі вимоги до ключів:

- Кожна множина сутностей повинна мати ключ.
- Ключ може складатися з більш ніж одного атрибута (як-от для множини сутностей *Залишки на складі*, ключем якої можуть бути два атрибути: ідентифікатор (ключ) деталі + ідентифікатор складу).
- Для множини сутностей припускається наявність кількох ключів (навіть у найпростішому прикладі - для множини сутностей *Марки машин*, де можливі ключі: *Id*, *Код*, можливо, також *Назва*. Одначе, доцільно обрати один ключ як *первинний ключ (primary key)* і надалі переважно працювати через нього.
- Якщо певна множина сутностей залучена в ієрархію *isa*, коренева множина сутностей повинна мати всі атрибути, необхідні для формування ключа. Напр., Рисунок 2.17 задовольняє цій вимозі. Для її задоволення треба було перемістити ідентифікаційні атрибути (ЄДРПОУ для контрагента та ідентифікаційний № для фізособи) з підкласів до базового класу.

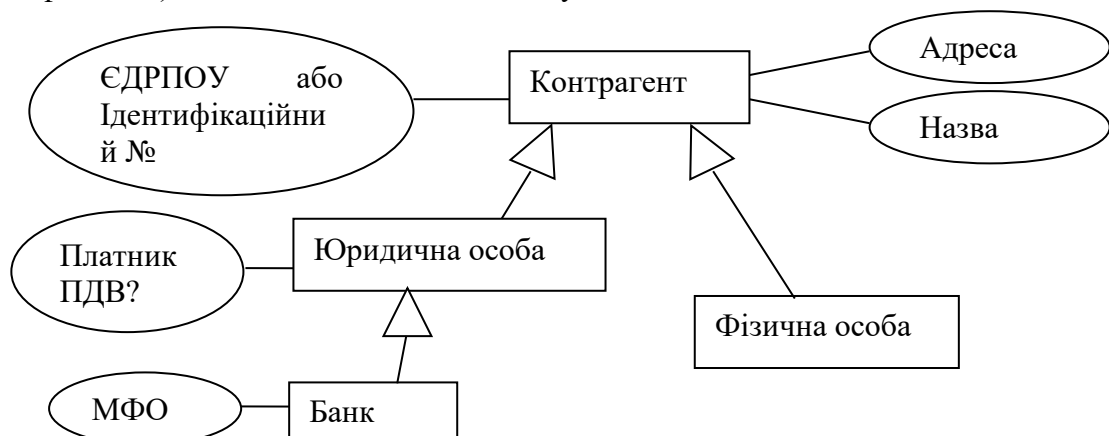


Рисунок 2.17. Приклад діаграми базового класу *Контрагент* з атрибутом – первинним ключем.

Наступний приклад: визначимо ключ для множини сутностей *кінофільми*. Оскільки існують фільми з однаковими назвами (як-от *King Kong*, *Три мушкетери*), доцільно створити ключ на основі атрибутів *title* і *year*, ігноруючи імовірність виходу в одному році фільмів з однаковою назвою.

В реальних ситуаціях, що підлягають моделюванню, часто створюються спеціальні атрибути, єдиним призначенням яких є підтримка ключових ознак сутності у множині однорідних сутностей. Приклади:

- Вже згадані ЄДРПОУ для юридичних осіб, ідентифікаційний № для фізичних осіб.
- № авіарейсу, № потяга.
- Табельний № співробітника, № залікової книжки студента.
- Поштовий код.
- № телефону.

У ER-діаграмі первинні ключі виділяються підкресленням, для інших ключів позначень нема.

### Обмеження унікальності

1. Це обмеження на атрибут, якому дозволено бути NULL. Маючи кілька значень NULL, цей атрибут все одно вважається унікальним. Вимога про те, що певний атрибут не може мати невизначених значень типу NULL, у ER-моделі не має спеціальних засобів представлення. При необхідності можна додати до атрибута текстове пояснення.
2. Зв'язок типу *багато до одного* між множинами сутностей *E* та *R*, від *E* до *R*, також передбачає унікальність у множині *R*.

### Обмеження посилальної цілісності

Обмеження посилальної цілісності (referential integrity constraints) передбачають наявність у точності одного значення на стороні «Один».

Розглянемо зв'язок *Owns* („володіє”), що з'єднує сутності *Movies* і *Studios* на малюнку 2.18. Цей зв'язок типу *багато до одного* передбачає лише те, що жодний фільм не може належати кільком студіям одночасно.

Обмеження посилальної цілісності в контексті зв'язку *Owns* передбачає, що для кожної сутності *Фільм* у БД має бути сутність *Студія*, на яку посилається *Фільм*. Способи підтримки посилальної цілісності на прикладі даного зв'язку:

1. Якщо БД поповнюється новим фільмом, має існувати (або має бути додана) сутність *Студія*, на яку посилається *Фільм*. Якщо це посилання змінюється, нове посилання також має вказувати на сутність множини студій.
2. Сутність *Студія* не може бути видалена, поки існують сутності *Фільм*, які на неї посилаються.
3. Якщо сутність *Студія* підлягає примусовому видаленню, мають бути видалені також усі сутності *Фільм*, які на неї посилаються.

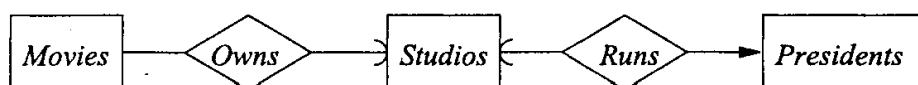


Рисунок 2.18 - ER-діаграма з обмеженнями посилальної цілісності

Напівкругла стрілка, що з'єднує зв'язок *Owens* з сутністю *Studios*, виражає обмеження посилальної цілісності.

Аналогічно інша напівкругла стрілка, що з'єднує зв'язок *Runs* з сутністю *Studios*, виражає наступне обмеження посилальної цілісності: кожний президент очолює одну студію і ця студія має існувати у множині *Studios*. В разі видалення студії з БД відповідна сутність з множини *Presidents* також має бути видалена. В той же час, зв'язок *Runs* з'єднується з множиною сутностей *Presidents* звичайною стрілкою, оскільки при видаленні президента студія не зникає, і у студії певний час може не бути президента.

Для обмежень інших типів ER-модель не забороняє писати текстові примітки.



Рисунок 2.19 - Позначення обмеження на кількість сутностей-акторів, які відповідають кожному фільму.

Таким чином, звичайна стрілка рівноцінна обмеженню виду „<=1”, а напівкругла стрілка (рис.2.17) - виду „=1”.

### Слабкі множини сутностей

Інколи трапляється, що ключ для певної множини сутностей доводиться формувати на основі атрибутів, які повністю або частково належать до іншої множини сутностей. Така сутність є *слабкою*, а зв'язок до сутності, з якої додаються ключові атрибути, є *підтримуючим* (supporting relationship). Вони ще називаються *ідентифікаційно-залежні сутності* (ID-dependent entities). Слабка сутність зображається подвійним прямокутником, а підтримуючий зв'язок – подвійним ромбом.

Перше джерело слабких сутностей. Якщо сутності множини *E* є одиницями більш „крупних” сутностей множини *F*, імовірна ситуація, коли атрибути множини *E*, які претендують на унікальність, не будуть унікальними, поки до них не будуть додані атрибути множини *F*, яким „підлягають” сутності *E*.

Приклади є в базі Технічної служби: якщо не використовувати сурогатний ключ, то група, підгрупа, деталь не ідентифікуються власними кодами, для цього до ключа треба додати коди груп верхніх рівнів довідника включно з кодом марки машини.

*E* – група (підсистема) в каталозі,

*F* – марка машини.

Інший приклад: ієрархія приміщень у БД інвестування житла:

**ПРИМІЩЕННЯ**

Тип	№ кв.	пш№кв	Кімн	Пов.	Ціна	м2 проє	Стан
Тип 1	1		4	2	0,00	161,18	Обов'язкове відр
Тип 1	2		1	2	0,00	58,11	Обов'язкове відр
Тип-1	3	35	1	2	10 281,60	58,11	Довідка про повн
Тип-1	4		4	2	0,00	161,18	Обов'язкове відр
Тип 2	5	37	4	3	9 865,80	161,28	Довідка про повн
Тип 2	6	38	1	3	10 092,60	55,64	Довідка про повн
Тип-2	7	39	1	3	10 659,60	55,67	Довідка про повн
Тип 3	8	40	4	3	9 865,80	163,50	Довідка про повн
Тип 2	9	41	4	4	9 875,25	161,28	Довідка про повн
Тип 2	10	42	1	4	10 054,80	55,67	Довідка про повн
Тип-2	11	43	1	4	10 054,80	55,67	Довідка про повн
Тип 3	12	44	4	4	9 875,25	163,50	Довідка про повн
Тип 2	13	45	4	5	10 017,00	161,28	Довідка про повн
Тип 2	14	46	1	5	11 277,00	55,67	Довідка про повн
Тип-2	15	47	1	5	11 277,00	55,67	Довідка про повн
Тип 3	16	48	4	5	9 954,00	163,50	Довідка про повн

Запис: 1 из 32

Рисунок 2.20. Ієрархія приміщень у БД інвестування житла

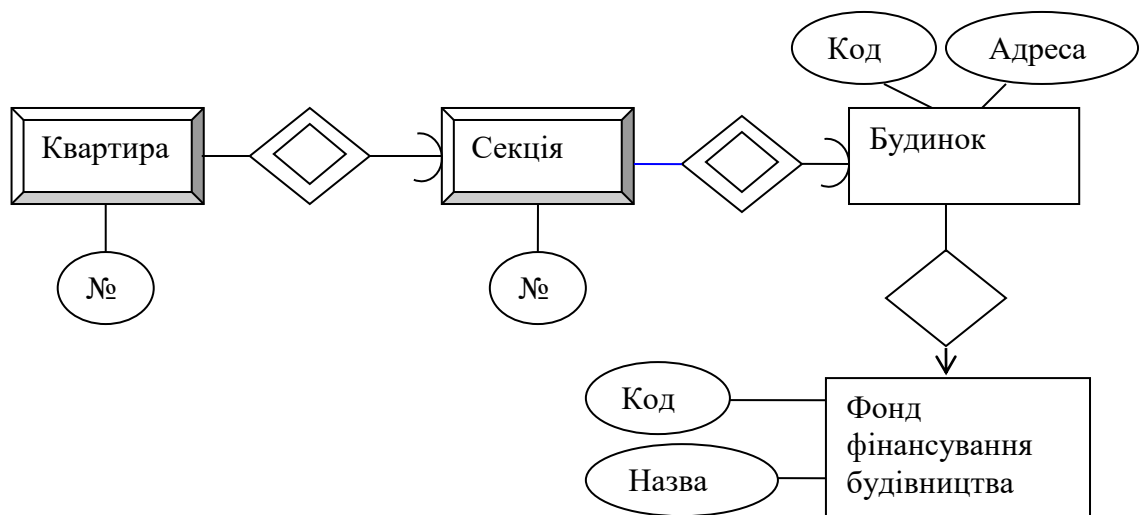


Рисунок 2.21. Слабкі сутності у БД інвестування житла

Друге джерело слабких сутностей. З'єднуючі множини сутностей, які створюються при спробі уникнути багатосторонніх зв'язків, часто не містять власних атрибутів, тому ключі таких множин складаються з атрибутів інших пов'язаних з ними множин.

Приклад – сутність *Contracts* є у БД фільмів. Унікальна сутність *контракт* визначається сукупністю значень атрибутів:

- *Name* множини сутностей *Stars* і *Studios*,
- *Title* і *year* множини сутностей *Movies*.

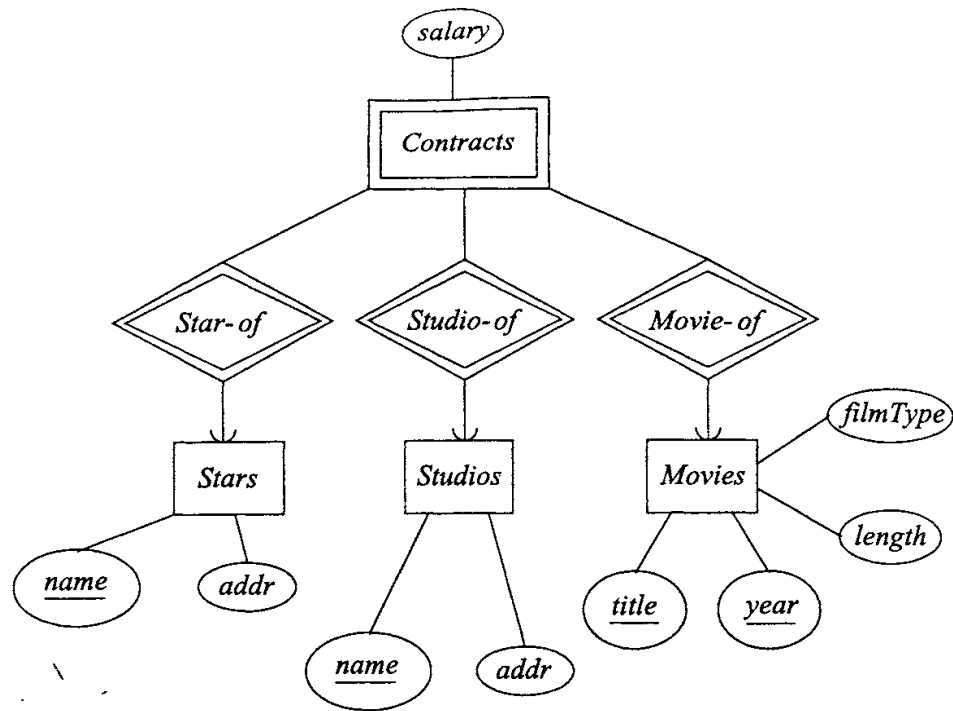


Рисунок 2.22 - Слабка з'єднуюча сутність

Підтримуючий зв'язок  $R$  від множини сутностей  $E$  до множини сутностей  $F$  має відповідати наступним вимогам:

1.  $R$  – зв'язок типу багато до одного (зокрема, і один до одного).
2. Для  $R$  має місце обмеження посилальної цілісності.
3. Атрибути  $F$ , надані для утворення ключа  $E$ , є ключовими і для  $F$ .
4. Якщо і  $F$  є слабкою сутністю, необхідні для утворення ключа атрибути по ланцюжку „позичаються” у „головніших” множин сутностей (як-от у прикладі БД інвестування житла, Рисунок 2.21).

### Діаграми «Сутність-зв'язок» в стилі UML

Database Modeling in UML. SPARX SYSTEMS,  
[https://sparxsystems.com/resources/uml\\_datamodel.html](https://sparxsystems.com/resources/uml_datamodel.html)

Статичне подання системи здійснюється за допомогою діаграми класів. Його можна використати для генерації більшості оголошень структури даних у програмі.

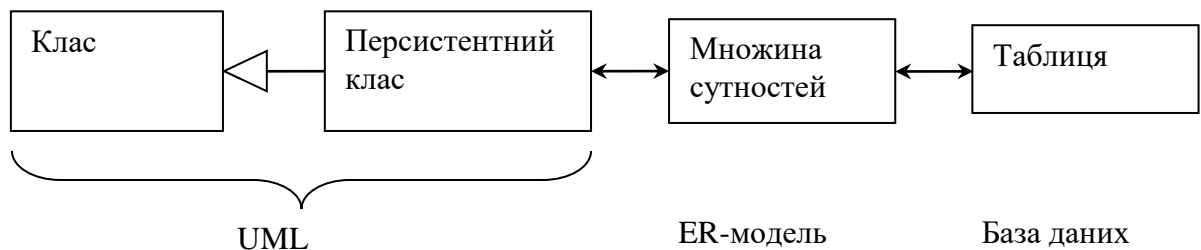


Рисунок 2.23 – Персистентні класи, для яких передбачене збереження їх об'єктів, є множинами сутностей в ER-моделі

Робота з цими діаграмами відбувається точно так само, як і з традиційними ER-діаграмами. Відтворимо рисунок 2.3 в стилі UML за допомогою діаграми класів:

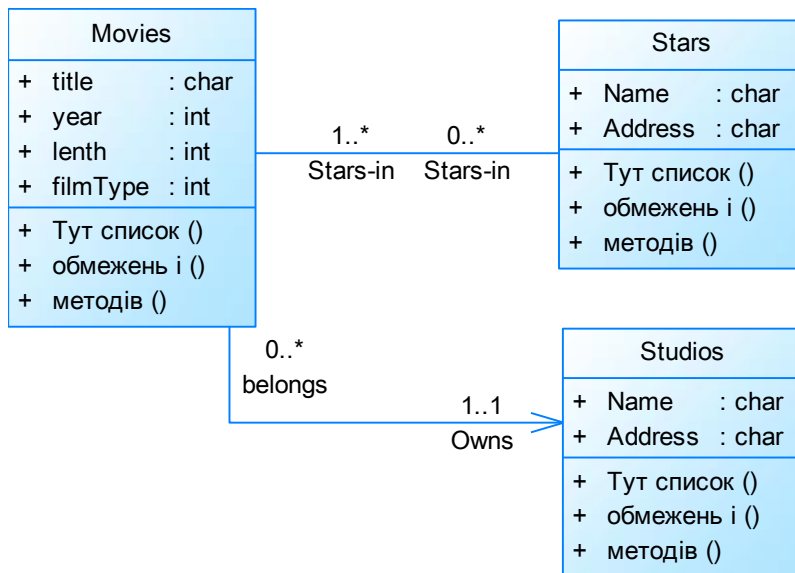


Рисунок 2.23. Діаграма сутностей і зв'язків для бази кінофільмів у стилі UML

Всі класи персистентні (це не відображається). Всі атрибути публічні, тому що в БД прихованих атрибутів не передбачено. Роль визначається не для зв'язку, а для сторони зв'язку, тому для зв'язку їх дві.

Успадкування множин сутностей моделюється успадкуванням класів.

Оскільки слабка множина сутностей не є самостійною і не може існувати без підтримуючої множини сутностей, моделюємо підтримуючий зв'язок як зв'язок композиції в діаграмі класів.

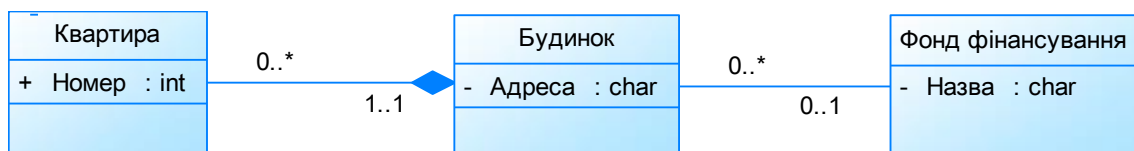


Рисунок 2.24. Діаграма сутностей і зв'язків у стилі UML для слабкої множини сутностей

На Рисунок 2.24 слабкою множиною сутностей є квартира, яка не існує окремо від будинку. На відміну від цього будинок не є слабкою множиною сутностей по відношенню до фонду, оскільки може існувати незалежно від фонду.

### Сурогатний ключ

Це додаткове службове поле, додане до вже існуючих інформаційних полів таблиці, єдине призначення якого — слугувати первинним ключем і тим самим ідентифікувати сутність. Значення цього поля не утворюється на основі якихось інших даних з БД, а генерується штучно.

Як правило, сурогатний ключ — це просто числове поле, в яке заносяться значення з числової послідовності, що збільшується. У ряді СУБД (напр., PostgreSQL, Sybase, MySQL або Microsoft SQL Server) є спеціальний тип даних для таких полів — числове поле, в яке при додаванні запису в таблицю автоматично записується унікальне для цієї таблиці числове значення — т.зв. «автоінкремент».



Головна перевага сурогатного ключа полягає в тому, що він ніколи не змінюється, оскільки не є інформативним полем таблиці (не несе ніякої змістовної інформації щодо об'єкту, який описується записом.). ... Саме на нього вказують усі посилання в пов'язаних таблицях. Сурогатний ключ „надає сили” слабким сутностям.

Для великих БД краще обрати властивість *нові значення* – випадкові. Для 4-байтного ключа (тип числа – довге ціле) діапазон приблизно  $\pm 2 \cdot 10^9$ , для 16-байтного ключа (тип GUID) набагато більше.

Алгоритм визначення випадкового сурогатного ключа для нової сутності:

1. Генеруємо випадкове число в певному діапазоні.
2. Перевіряємо, чи вже є нове значення серед ідентифікаторів існуючих сутностей. Якщо є, повертаємося на п.1.
3. Вставляємо нову сутність зі згенерованим ідентифікатором.

Перевагою випадкового сурогатного ключа перед лічильником є те, що в процесі додавання і видалення сутностей з їх множини нові випадкові значення ніколи не закінчуються, в той час як значення лічильника завжди зростають, і в якийсь неочікуваний момент стає потрібною реорганізація БД.