Лабораторна робота 3: Побудова логічної моделі бази даних

Мета:

Навчити студентів перетворювати концептуальну модель, створену за допомогою UML, у логічну модель бази даних, представивши її у вигляді ER-діаграми (Entity-Relationship Diagram).

Завдання:

1. Аналіз концептуальної моделі: переглянути діаграму класів UML, створену в лабораторній роботі 2.
2. Перетворення атрибутів: визначити типи даних атрибутів та їх обмеження.
3. Визначення первинних ключів: визначити первинні ключі для кожної таблиці.
4. Визначення типів зв’язків та зовнішніх ключів: визначити типи зв'язків між сутностями (кардинальність), формалізувати зв’язки за допомогою зовнішніх ключів.
5. Нормалізація схеми даних: привезти схему до нормальної форми Бойса-Кодда.
6. Побудова ER-діаграми: створити ER-діаграму на основі концептуальної моделі, використовуючи відповідну нотацію (Chen, Crow's Foot тощо).
7. Документування переходу: описати процес переходу від UML діаграми до ER-діаграми, включаючи будь-які зміни або спрощення.

Результат:

Студенти повинні подати опис атрибутів з типами даних та ключами, ER-діаграму та опис процесу переходу від UML до ER, з поясненням прийнятих рішень.

Теоретичні відомості до виконання лабораторної роботи

Перехід від концептуальної до логічної моделі бази даних є ключовим етапом у процесі проектування бази даних. Це процес перетворення загальних концептуальних уявлень про дані та їх взаємозв'язки в більш детальну структуру, яка описує, як ці дані будуть зберігатися в СУБД.

Логічна модель бази даних – це деталізована модель, яка описує структуру даних з акцентом на логіку зберігання і взаємодії даних, без прив'язки до конкретної платформи або фізичного способу зберігання інформації. Вона є наступним кроком після концептуальної моделі і використовується для розуміння, як організовані дані на рівні бізнес-логіки. Логічна модель використовує поняття сутностей, атрибутів, зв'язків, ключів, але все ще не прив'язана до фізичної реалізації.

Основні елементи логічної моделі:

– Таблиці: кожна сутність у концептуальній моделі стає таблицею в логічній моделі.

– Колонки (поля): атрибути сутностей стають колонками в таблицях.

– Первинні ключі: унікальні ідентифікатори записів у таблицях (наприклад, ID студента).

– Зовнішні ключі: використовуються для встановлення зв'язків між таблицями.

Основні етапи переходу від концептуальної до логічної моделі

Етап 1. Визначення таблиць: сутності з концептуальної моделі перетворюються на таблиці логічної моделі.

Етап 2. Визначення атрибутів і їх типів даних: кожен атрибут сутності перетворюється на колонку (стовпець) таблиці. На цьому етапі визначаються типи даних для кожного атрибута (наприклад, текст, число, дата), а також обмеження на значення атрибутів.

Етап 3. Визначення первинних ключів: для кожної таблиці визначаються первинні ключі.

Етап 4. Визначення зв'язків між таблицям та зовнішніх ключів: здійснюється формалізація зв'язків концептуальної моделі та їх реалізація за допомогою зовнішніх ключів.

Етап 5. Нормалізація: логічна модель проходить процес нормалізації для усунення надлишкових даних і забезпечення узгодженості даних. На цьому етапі можуть бути створені додаткові таблиці для поділу складних атрибутів або зв'язків.

Етап 1 не потребує пояснень та автоматично реалізується.

Етап 2 аналогічно, тільки потребує зазначення типів даних без прив'язки до СУБД. Наприклад, чи буде поле числовим, текстовим, чи матиме фіксовану довжину. Також на цьому етапі аналізуються можливі значення атрибутів та виділяються обмеження на значення. Наприклад, дата народження не може бути більшою за поточну дату, ціна не може бути від'ємною тощо.

Для реалізації етапу 3 та частково 4 наведемо визначення ключів реляційної моделі даних.

1. Потенційний ключ (Candidate Key) – це будь-який атрибут або набір атрибутів, які унікально ідентифікують кожен запис у таблиці. Кожен потенційний ключ повинен бути унікальним і не містити значень NULL. В таблиці може бути декілька потенційних ключів.

2. Первинний ключ (Primary Key, PK) – це атрибут або набір атрибутів, які унікально ідентифікують кожен запис у таблиці. Жоден запис не може мати однакове значення первинного ключа, і це поле не може бути порожнім (NULL). В таблиці може бути тільки один первинний ключ. Визначення потенційного та первинного ключів схожі. Тобто, потенційних ключів в таблиці може бути багато, один з низ обирається для зв'язку з іншими таблицями і називається первинним, всі інші становляться альтернативними.

3. Альтернативний ключ (Alternate Key) – це потенційний ключ, який не був обраний як первинний ключ. Тобто це будь-який інший унікальний атрибут або набір атрибутів, який може ідентифікувати записи в таблиці, але не є первинним. Якщо є кілька потенційних ключів, один стає первинним, а інші називаються альтернативними.

4. Зовнішній ключ (Foreign Key, FK) – це атрибут або набір атрибутів у таблиці, який посилається на первинний ключ іншої таблиці. Зовнішній ключ використовується для створення зв'язків між таблицями. Він гарантує узгодженість даних між таблицями, забезпечуючи, що кожне значення зовнішнього ключа відповідає наявному значенню первинного ключа в іншій таблиці.

Приклад. У таблиці "Студент" є поля: "ID студента", "Номер студентського квитка", "Електронна адреса", "Група".

– Потенційні ключі: "ID студента", "Номер студентського квитка", "Електронна адреса". Усі ці поля можуть бути унікальними і ідентифікувати студента.

– Первинний ключ: "ID студента".

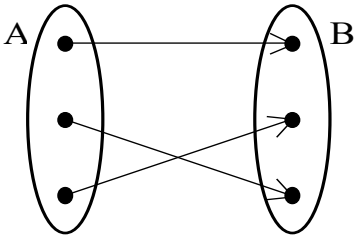
– Альтернативні ключі: "Номер студентського квитка", "Електронна адреса". Оскільки "ID студента" обрано первинним ключем, інші стають альтернативними.

– Зовнішній ключ: "Група" в таблиці "Студент", який посилається на "ID групи" в таблиці "Групи".

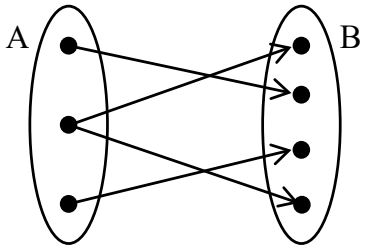
Етап 4 передбачає здійснення формалізації зв'язків. Для реалізації правил формалізації попередньо ознайомимось з визначенням понять кардинальності (кратності, потужності, ступеню) зв'язку, умовності/безумовності (обов'язковості/необов'язковості) зв'язку та арності зв'язку.

Визначення кардинальності зв’язків

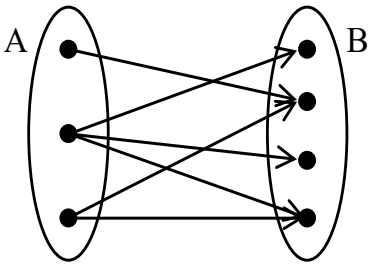
Це аспект, який визначає, скільки записів з однієї таблиці може бути пов'язано з записами в іншій таблиці.



Зв’язок 1:1 (або відображення 1 до 1, One-to-One) — кожному екземпляру сутності А відповідає в точності 1 екземпляр сутності В. Наприклад: "Кафедра" і "Завідувач кафедри": кожна кафедра має тільки одного завідувача і один співробітник може завідувати тільки однією кафедрою; "Студент" і "Паспорт": кожен студент має лише один паспорт і кожен паспорт пов'язаний тільки з одним студентом.



*Зв’язок 1:N (*One-to-Many) — кожному екземпляру сутності А відповідає 1÷N екземплярів сутності В. Наприклад: "Порт" і "Морське судно“: кожен порт має багато морських суден, які прописані в ньому, кожне морське судно має тільки один порт приписки; "Студент" і "Група“: кожен студент навчається в одній групі, а в кожній групі навчається багато студентів.



Зв’язок M:N (Many-to-Many) — кожному екземпляру сутності А відповідає 1÷N екземплярів сутності В і навпаки, кожен екземпляр сутності В відповідає 1÷М екземплярам сутності А. Наприклад: "Викладач" і "Дисципліна": кожен викладач може викладати одну або кілька дисциплін, тому один викладач (один запис у таблиці "Викладач") може бути пов'язаний із багатьма дисциплінами (кілька записів у таблиці "Дисципліна"), одна дисципліна викладається одним або декількома викладачами, тому одна дисципліна (один запис у таблиці "Дисципліна") може бути пов'язана із багатьма викладачами (кілька записів у таблиці "Викладач").

Особливий тип зв’язку: зв’язок супертип-підтип. Зв'язок супертип-підтип використовується для моделювання ієрархій між сутностями в базі даних, коли один клас сутностей (супертип) є узагальненим типом, а інші класи (підтипи) є спеціалізованими типами цього супертипу. Це дозволяє організувати дані більш ефективно, враховуючи спільні та специфічні властивості об'єктів.

Супертип — це узагальнена сутність, що включає атрибути, спільні для кількох підтипів. Супертип описує загальні характеристики, які притаманні всім підтипам.

Підтип — це спеціалізована сутність, що містить унікальні атрибути та відображає певний тип сутностей, який є підмножиною супертипу. Кожен підтип має додаткові атрибути або поведінку, які відрізняють його від інших підтипів. Наприклад: в університеті є різні співробітники: професорсько-викладацький персонал, інженери, працівники відділу кадрів, бухгалтерії, деканату і так далі. Вони відрізняються деякими властивостями: професорсько-викладацький персонал має науковий ступінь, наукове звання; інженери – категорію, тощо. Отже, сутність супертипу буде "Співробітник", яка містить всі загальні властивості для всіх співробітників (ПІБ, дата народження, дата прийняття на роботу, посада тощо), підтипами будуть "Викладача", "Інженер", "Адмін_Працівник", кожний з яких буде містити атрибути, притаманні саме цим об’єктам.

Визначення умовності зв’язків

Безумовність (обов’язковість) (Mandatory Relationship) виникає коли кожен запис в одній таблиці має бути пов'язаний з одним або кількома записами в іншій таблиці. В обов'язкових зв'язках зовнішній ключ не може бути порожнім (NULL). Це означає, що кожен запис в одній таблиці повинен мати відповідний запис в іншій таблиці.

Наприклад: кожна дисципліна в навчальному процесі обов’язково має викладача, тому таблиця "Дисципліна" матиме зовнішній ключ на таблицю "Викладач".

Умовність (необов'язковість) (Optional Relationship) у зв'язках виникає тоді, коли деякі екземпляри однієї або обох сутностей *не беруть участь* у зв'язку. У необов'язкових зв'язках зовнішній ключ може бути порожнім (NULL), що означає, що певний запис може не мати відповідного запису в іншій таблиці.

Наприклад: Не всі студенти будуть мати тему кваліфікаційної роботи (студенти 1-3 курсів апріорі ще не мають теми згідно плану навчального процесу), тому зв'язок між сутностями "Студент" та "Кваліфікаційна робота" може бути необов'язковим зі сторони сутності "Студент".

Визначення арності зв'язків

Арність зв'язку — це характеристика, яка визначає кількість сутностей або таблиць, що беруть участь у зв'язку. Іншими словами, арність вказує на кількість учасників у відношенні між даними.

Основні типи арності:

1. Унарний зв'язок (Unary Relationship) – це зв'язок, де одна й та сама таблиця або сутність пов'язана сама з собою. Іншими словами, сутність бере участь у зв'язку з іншою сутністю того ж самого типу. Унарний зв'язок пов'язує записи однієї й тієї ж таблиці між собою. Це зазвичай виражається через рефлексивний зовнішній ключ, де один атрибут вказує на інший запис у тій же таблиці.

Приклад: У таблиці "Співробітник" кожен співробітник може мати менеджера, який теж є співробітником цієї ж компанії. Це зв'язок "співробітник-менеджер", де обидва учасники відносяться до тієї ж самої таблиці "Співробітник".

2. Бінарний зв'язок (Binary Relationship) – це найпоширеніший тип зв'язку, який включає дві сутності або дві таблиці. Бінарний зв'язок визначає відношення між двома різними таблицями. Бінарний зв'язок між двома таблицями виражається через зовнішній ключ, де один або більше атрибутів однієї таблиці посилаються на первинний ключ іншої таблиці.

Приклад: Таблиці "Студент" та "Група" можуть мати зв'язок "один до багатьох", де один студент може бути зарахований тільки в одну групу, а в групі може навчатися багато студентів.

3. Тернарний зв'язок (Ternary Relationship) – це зв'язок, у якому беруть участь три сутності або три таблиці. Він складніший за бінарний, оскільки в ньому необхідно враховувати взаємодії між трьома різними таблицями. Він формалізується через зв'язкову таблицю, яка містить зовнішні ключі до кожної з трьох сутностей. Ця зв'язкова таблиця може містити додаткові атрибути.

Приклад: В університеті може існувати зв'язок між таблицями "Група", "Дисципліна" та "Викладач", де викладач для певної групи або груп може викладати певну дисципліну або дисципліни.

4. N-арний зв'язок (N-ary Relationship) – це загальний випадок, коли в зв'язку беруть участь більше ніж три сутності (таблиці). Хоча на практиці рідко зустрічаються зв'язки з більш ніж трьома сутностями, вони можливі в складних моделях даних. В більш складних моделях, коли є N-арний зв'язок, він теж формалізується через зв'язкову таблицю, яка містить зовнішні ключі на всі N сутностей.

Формалізація зв'язків

Формалізація зв'язків в реляційній моделі даних включає кілька важливих аспектів: визначення арності (кількості учасників), кардинальності (кількості пов'язаних записів) та обов'язковості чи необов'язковості зв'язків. Чітке визначення цих правил допомагає розробити структуровану, ефективну і узгоджену базу даних, що правильно відображає реальні бізнес-процеси і запобігає аномаліям даних.

Зв'язок 1:1 (тобто безумовний). Це єдиний тип зв'язку, що допускає два варіанти

представлення. Якщо між двома сутностями існує безумовний зв'язок 1:1, то для їхнього представлення досить *одного відношення*. Причому первинним ключем цього відношення може бути ідентифікатор кожної із сутностей. Якщо ж за умовами задачі або системи необхідно кожну сутність представити у вигляді окремого відношення, то для формалізації зв'язків необхідно атрибути первинного ключа одного з відношень додати в схему іншого відношення як зовнішній ключ. Причому в будь-яке з відношень.

Зв'язок 1:1у. Якщо між сутностями встановлений умовний (необов'язковий) зв'язок 1:1у, то

а) кожна сутність представляється окремим відношенням. Причому ID сутності стає первинним ключем відповідного відношення.

б) ID сутності, що відповідає умовності зв'язку, додається як *допоміжний атрибут* в іншу сутність і, відповідно, як зовнішній ключ у схему її відношення.

Зв'язок 1у:1у (тобто біумовний або необов'язковий з двох сторін). Для формалізації зв'язку цього типу необхідно використовувати 3 відношення: по одному для кожної сутності й одне для зв'язку, яке називається асоціативна сутність. Атрибутами цього відношення обов'язково будуть первинні ключі двох інших. При цьому типі зв'язку обидві множини атрибутів асоціативного об'єкта є потенційними ключами. Причому, якщо один з них вибирається як первинний ключ цього відношення, то другий автоматично вважається зовнішнім. Крім того, асоціативний об'єкт може мати і множину власних атрибутів.

Зв'язки 1:N і 1у:N. Для формалізації зв'язків типу 1:N *не має значення*, чи є зв'язок з боку *однозв'язної* сутності умовним чи безумовним. Кожна сутність, яка бере участь у зв'язку типу 1:N чи 1у:N, представляється своїм відношенням. При цьому первинний ключ відношення, що відповідає однозв'язній сутності, додається як зовнішній ключ у схему відношення *N*-зв'язної сутності.

Зв'язки 1:Ny і 1у:Ny. Для визначення зв'язку 1:N, *умовного з боку N-зв'язної сутності,* необхідно сформувати по одному відношенню для кожної сутності й одне відношення для зв'язку, у схему відношення якого повинні бути включені первинні ключі відношень сутностей. Потенційними ключами відношення, що відповідає асоціативній сутності, може бути ключ відношення N-зв'язної сутності або множина, що складається з ключів обох відношень.

Зв'язок *M:N* будь-якої умовності (безумовний, умовний, биумовний) також вимагає трьох відношень для формалізації: по одному для кожної сутності й одне для зв'язку. Причому, останнє повинно мати в схемі відношення первинні ключі двох інших. Більш того, якщо асоціативний об'єкт не має чи не повинен мати власного атрибута-ідентифікатора, то первинним ключем може бути тільки множина, що містить обидва зовнішніх ключі. Якщо при цьому відсутні і будь-які інші власні атрибути цього відношення, то кажуть, що воно цілком є ключем.

При формалізації зв'язків „супертип-підтип" відношення підтипів повинні включати як первинні ключі первинний ключ відношення супертипу.

Для формалізації *n*-арного зв'язку необхідно *n+1* відношення: по одному для кожної сутності, що беруть участь у зв'язку, і одне для самого зв'язку. Причому відношення для зв'язку, точніше, його схема повинна включати первинні ключі всіх інших *n* відношень. Більш того, множина усіх цих ключів буде первинним ключем відношення зв'язку.

Для реалізації етапу 5 розглянемо поняття нормалізації даних.

Нормалізація - це процес перевірки і реорганізації сутностей і атрибутів з метою задоволення вимог до реляційної моделі даних. Процес нормалізації зводиться до послідовного приведення структур даних до нормальних форм – формалізованим вимогам до організації даних – шляхом розбивки сутності на дві та більше. Нормалізація застосовується для усунення надлишковості, підвищення ефективності зберігання даних і запобігання аномаліям при додаванні, видаленні або оновленні даних.

Перша нормальна форма (1НФ). Сутність знаходиться в першій нормальній формі тоді і тільки тоді, коли всі атрибути містять атомарні значення. Серед атрибутів не повинно зустрічатися повторюваних груп, тобто кілька значень для кожного екземпляра.

Друга нормальна форма (2НФ). Сутність знаходиться в другій нормальній формі, якщо вона знаходиться в першій нормальній формі, і кожен неключовий атрибут повністю залежить від первинного ключа (не може бути залежності від частини ключа).

Третя нормальна форма (3НФ). Сутність знаходиться в третій нормальній формі, якщо вона знаходиться в другій нормальній формі і ніякий не ключовий атрибут не залежить від іншого неключового атрибута (не повинно бути залежності між не ключовими атрибутами).

Відношення знаходяться в *нормальній формі Бойса-Кодда* (*НФБК)* тоді і тільки тоді, коли кожен детермінант (атрибут, який визначає інші атрибути) є потенційним ключем.

Нормалізація — це важливий процес в проектуванні баз даних, що дозволяє зменшити надмірність і уникнути аномалій в роботі з даними. Однак надмірна нормалізація може призвести до зниження продуктивності через збільшення кількості таблиць та складність запитів. Тому завжди варто шукати баланс між нормалізацією і ефективністю виконання запитів.

Перехід від концептуальної до логічної моделі

Перехід від концептуальної моделі у вигляді UML-діаграми класів до логічної ER (Entity-Relationship) моделі є важливим кроком у процесі проектування бази даних. Цей процес допомагає детальніше структуризувати дані та підготувати їх для фізичної реалізації в СУБД. Давайте розглянемо, як перетворювати концептуальну модель у логічну модель і що слід враховувати при трансформації зв'язків.

Основні етапи переходу від UML-діаграми класів до ER-моделі:

1) Кожен простий клас (без зв’язку-узагальнення) перетворюється в таблицю.

2) Кожен атрибут стає стовпцем таблиці з таким же ім’ям.

3) Стовпці, які відповідають необов’язковим атрибутам, можуть мати невизначені значення, а стовпці, які відповідають обов’язковим атрибутам, – не можуть.

4) Унікальні атрибути класу перетворюються в потенційні ключі таблиці.

5) Зв’язки «один до багатьох», «один до одного» формалізуються за допомогою зовнішнього ключа: створюється копія унікального атрибуту класу на кінці зв’язку «один», а в таблиці, яка відповідає класу на кінці зв’язку «багато», стовпці становля зовнішній ключ.

6) Якщо між двома класами А та В є зв’язок «один до одного», то зовнішній ключ може бути оголошено в таблиці В як потенційний.

7) Щоб відобразити в визначені таблиці обмеження, яке полягає в тому, що ступінь кінця зв’язку повинен дорівнювати одиниці, відповідний стовпець повинен бути додатково специфіковано як потенційний ключ таблиці.

8) Зв’язки «багато до багатьох» між класами А та В створюються через додаткову таблицю АВ з двома стовпцями: один містить стовпець-атрибут класу А, а другий стовпець-атрибут класу В. В результаті отримаємо два окремих зв’язки типу «один-до-багатьох». В таблиці АВ формується складений первинний ключ зі всіх стовпців.

9) Якщо в діаграмі присутні зв’язки-узагальнення (успадкування), то можливі три способи представлення їх класів в реляційній БД:

а) зібрати всі підкласи в одній таблиці, зазвичай додається дискримінатор (поле, що вказує на тип).

б) для кожного підкласу створити окрему таблицю з колонками, які відповідають атрибутам підкласу та атрибутами батьківського класу;

в) для кожного підкласу створити окрему таблицю з колонками, які відповідають атрибутам цього підкласу при забезпеченні зв’язку з батьківським класом.

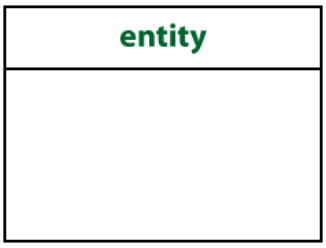
Представлення логічної моделі даних

Для представлення логічної моделі даних використовують різні **нотації** і **інструменти**, які допомагають візуалізувати структуру бази даних, зв’язки між сутностями та їхні атрибути. Вибір нотації може залежати від методології проєктування, вимог системи або особистих уподобань аналітиків і розробників.

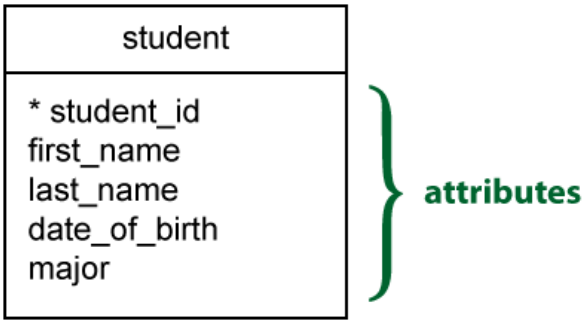
Нотація Crow's Foot

Розглянемо як приклад нотацію Crow's Foot (або "вороняча лапка") це одна з найпоширеніших нотацій для реляційних баз даних.

Сутності зображуються у вигляді прямокутників з назвою, як і в нотації Chen. І’мя сутності зазначається іменником в однині:



Атрибути включаються всередину прямокутників сутностей. Ключевий атрибут містить символ «зірочка» або розміщується в окремому розділі.



Зв'язки зображуються у вигляді ліній, що з'єднують прямокутники сутностей. Зазвичай, кожне відношення має ім'я, виражене дієсловом, написаним на лінії відношення. Воно визначає, який тип відношення пов'язує об'єкти.

Кардинальність. Відношення мають два індикатори. Вони показані по обидва боки лінії. На кінцях ліній є символи, що представляють кардинальність: обов'язкове відношення представлено прямою лінією, перпендикулярною до лінії відношення; необов’язковий зв’язок позначається порожнім колом; кардинальність один представлено прямою лінією, перпендикулярною до лінії відношення; кардинальність багато представлена трикутним символом «гусяча лапка».

Приклади відношень

Назва	Графічне представлення
Нуль або багато	
Один або багато	
Один і тільки один	
Нуль або один	

Інструменти для моделювання логічних моделей даних:

1. MySQL Workbench

Інструмент для проєктування баз даних, який підтримує ER-моделювання, дозволяючи будувати діаграми "сутність-зв'язок" за допомогою Crow’s Foot Notation. Він інтегрується з MySQL, але підтримує й інші СУБД.

2. Microsoft Visio

Потужний інструмент для візуалізації, який підтримує різні нотації для ERD, включаючи Crow’s Foot, UML і інші. Visio дозволяє легко будувати діаграми і є популярним серед фахівців.

3. Lucidchart

Веб-інструмент для створення діаграм, який підтримує моделювання баз даних і UML. Lucidchart підтримує спільну роботу, що робить його популярним у командних проєктах.

4. Erwin Data Modeler

Спеціалізований інструмент для моделювання баз даних, що підтримує розробку концептуальних, логічних і фізичних моделей даних. Erwin широко використовується для реляційного моделювання і нормалізації даних.

5. DBDesigner

Безкоштовний інструмент для моделювання баз даних з підтримкою ERD, дозволяє створювати логічні моделі даних, а також генерувати SQL-скрипти для фізичної реалізації.

6. Enterprise Architect

Інструмент для проєктування систем і моделювання баз даних з використанням UML та інших нотацій. Підтримує як UML-класи для моделювання, так і більш традиційні ERD.

7. Toad Data Modeler

Інструмент, що дозволяє створювати моделі даних, підтримує широкий спектр СУБД та пропонує багаті функції для візуалізації та моделювання баз даних.

8. PowerDesigner (SAP)

Потужний інструмент для проектування баз даних, що підтримує концептуальне, логічне і фізичне моделювання. Він пропонує багатий набір можливостей для створення ERD, UML, IDEF1X та інших діаграм.

Продовжимо приклад з попередньої лабораторної роботи та наведемо перехід *від UML-діаграми класів до ER-моделі*. Для зручності наведено UML-діаграму класів з попередньої лабораторної роботи.

Етап 1. Визначення таблиць: перенесли сутності з концептуальної моделі на логічну модель у вигляді таблиць.

Етап 2. Визначення атрибутів і їх типів даних: кожен атрибут кожної сутності перенесли на логічну модель у вигляді стовпців таблиці. Визначили типи даних для кожного атрибута та обмеження на їх значення.

Етап 3. Визначення первинних ключів: для кожної таблиці визначили первинні ключі (PK).

Етап 4. Визначення зв’язків між таблицям та зовнішніх ключів: провели формалізація зв’язків концептуальної моделі згідно правил. Більшість зв’язків реалізували за допомогою зовнішніх ключів (FK), а зв’язок між таблицями «Авіакомпанія» і «Літак» реалізовано за допомогою асоціативної сутності згідно правила формалізації зв’язку один до багатьох умовний.

Етап 5. Нормалізація: проведено нормалізацію логічної моделі до нормальної форми Бойса-Кодда (НФБК).

Перевіримо схему даних на відповідність НФБК, згідно якої кожен атрибут відношення повинен залежати від потенційного ключа.

Розглянемо на прикладі відношення *Рейс*. Покажемо послідовний перехід від одної нормальної форми до іншої.

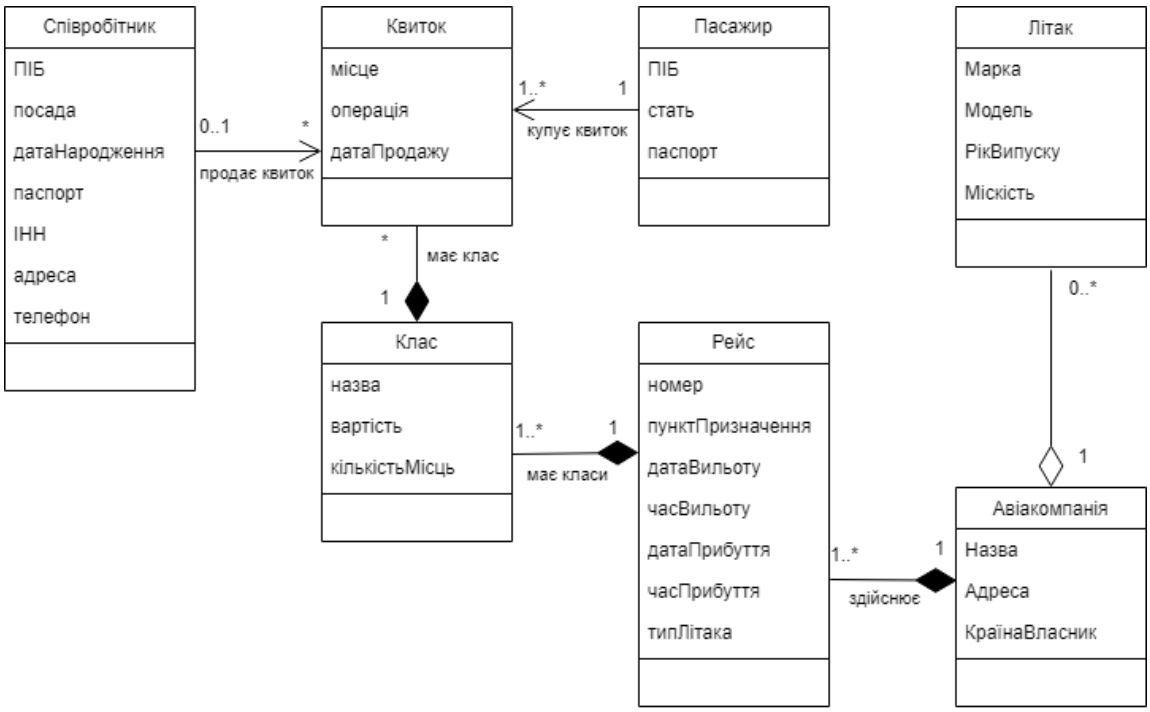


Рисунок 1 – UML-діаграма класів предметної області «Аеропорт»

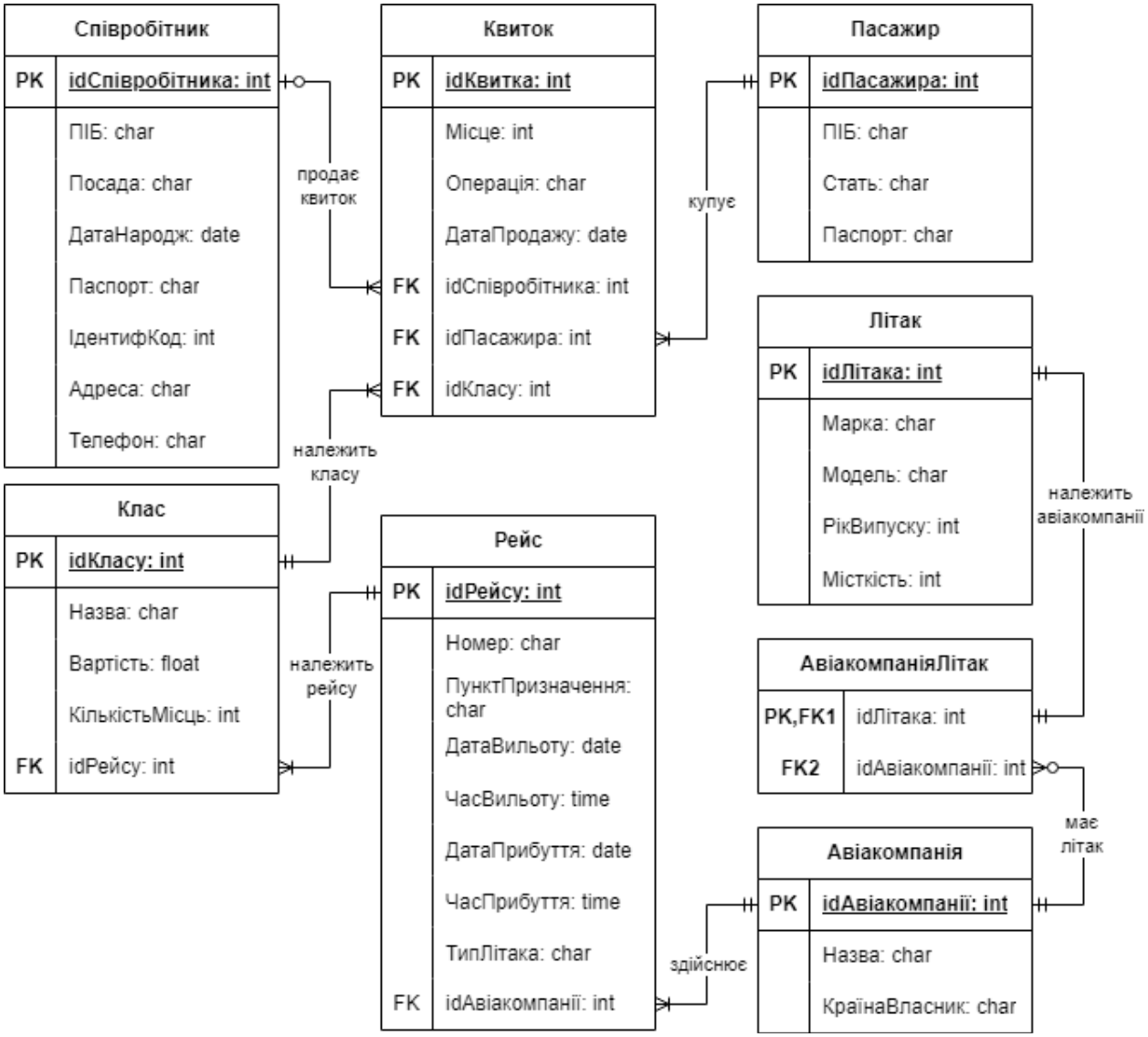


Рисунок 2 – ER-діаграма в нотації Crow’s Foot предметної області «Аеропорт»

Дане відношення знаходиться в першій нормальній формі, так як значення кожного атрибуту не розділяється на декілька значень.

Дане відношення знаходиться у другій нормальній формі, так як кожен неключовий атрибут функціонально повно залежить від первинного ключа — *id_рейсу*.

Дане відношення знаходиться у третій нормальній формі, так як кожен неключовий атрибут залежить тільки від первинного ключа *id_рейсу* та не виникає інформаційної надмірності та аномалій.

Дане відношення знаходиться в нормальній формі Бойса-Кодда, так як в ньому відсутні функціональні залежності атрибутів складеного ключа від неключових атрибутів. Ця умова виконується за замовчуванням, так як в даному відношенні ключ не являється складеним.

Аналогічним чином перевіряються інші відношення.

Структура звіту до лабораторної роботи

1. Завдання, тобто предметна область згідно варіанту.
2. Копія концептуальної моделі, яку створену в лабораторній роботі 2.
3. Перелік атрибутів з зазначенням типів даних та обмежень. Представити у вигляді таблиці:

Сутність	Властивість	Тип даних	Ключ	Обмеження на значення
Студенти	id	цілочисельний	первинний	
	ПІБ	символьний		
	група	цілочисельний	зовнішній	
…	…	…	…	
Спеціалізація	id	цілочисельний	первинний	
	…	…	альтернативний	
	Вартість навчання	дійсний		
…	…	…	потенційний	

4. Розроблена ER-діаграма на основі концептуальної моделі у відповідній нотації (Chen, Crow's Foot тощо).
5. Опис перевірки схеми на відповідність нормальній формі Бойса-Кодда.
6. Опис процесу переходу від UML-діаграми до ER-діаграми, включаючи будь-які зміни або спрощення.

Звіт до лабораторної роботи 3 можна здати онлайн на сайті ДО edu.op.edu.ua до початку вашого заняття.