Лекція

[**4.1. Завдання і цілі фізичного проектування**](https://rdb.dp.ua/uk/chapter_04#4.1)

[**4.2. Вибір СКБД**](https://rdb.dp.ua/uk/chapter_04#4.2)

[**4.3. Правила, згідно з якими СКБД може вважатися реляційною**](https://rdb.dp.ua/uk/chapter_04#4.3)

[**4.4. Мова SQL**](https://rdb.dp.ua/uk/chapter_04#4.4)

[**4.5. SQL-оператори, що реалізовують структуру реляційних даних**](https://rdb.dp.ua/uk/chapter_04#4.5)

[**4.6. Додавання, оновлення й вилучення даних**](https://rdb.dp.ua/uk/chapter_04#4.6)

[**4.7. КОНТРОЛЬНІ ПИТАННЯ**](https://rdb.dp.ua/uk/chapter_04#4.7)

**Мета**

*Мета розділу – визначення зв’язків між логічним та фізичним етапами проектуванням, а також мети та завдань етапу фізичного проектування БД, обґрунтування вибору цільової СКБД, ознайомлення з «еталонними» правилами, які обумовлюють приналежність СКБД до розряду дійсно реляційних систем. Розділ містить коротку характеристику мови SQL та опис основних прийомів забезпечення цілісності реляційних даних.*

**4.1. Завдання і цілі фізичного проектування**

Під час логічного проектування була визначена структура БД (відношення, їх атрибути та зв'язки). Хоча логічна модель не залежить від конкретної цільовий СКБД, вона створювалася з урахуванням вибраної моделі організації даних. Між логічним і фізичним проектуванням існує постійний зворотний зв'язок, оскільки рішення, які прийняті на етапі фізичного проектування з метою підвищення продуктивності системи, здатні вплинути на структуру логічної моделі даних [4, 6].

***Фізичне проектування БД****– це процес створення та опису реалізації БД на вторинних запам'ятовувальних пристроях з вказівкою структур зберігання та методів доступу, які використовуються для ефективної обробки даних.*

Основна мета фізичного проектування є – опис способу фізичної реалізації логічного проекту. Для реляційної моделі це:  
1) створення набору реляційних таблиць та встановлення обмежень для них на базі інформації, яка міститься в глобальній логічній моделі даних;  
2) визначення конкретних структур зберігання даних і методів доступу до них, які забезпечують оптимальну продуктивність БД;  
3) розробка засобів захисту системи, що створюється.

В ідеалі, фазу логічного проектування великих систем слід відокремлювати від фази їхнього фізичного проектування. На це є ряд причин:  
1) вони пов'язані із зовсім різними аспектами системи: що робити і як робити;  
2) вони виконуються в різний час, оскільки зрозуміти, що треба зробити, необхідно перед тим, ніж вирішити, як це зробити;  
3) вони вимагають зовсім різних навичок і умінь, якими звичайно володіють різні люди.

**4.2. Вибір СКБД**

Розпочинаючи фізичне проектування БД, насамперед, необхідно вибрати конкретну цільову СКБД. Розглянемо такі етапи її вибору:

1. Визначення ряду програмних продуктів, з яких буде вибиратися СКБД.
2. Скорочення списку претендентів до двох – трьох продуктів.
3. Оцінка продуктів.
4. Обґрунтування вибору єдиного продукту.

Розглянемо вибір продукту для вивчення організації реляційних БД.

Перший етап зводитися до того, що вибір буде виконуватися із СКБД, які підтримують реляційну модель представлення даних і архітектуру клієнт-сервер. Це пояснюється тим, що приблизно 80 % СКБД у світі використовують реляційну модель даних, а архітектура клієнт-сервер довела на практиці свою життєздатність та ефективність.

Сайт не ставить за своєї мети вивчення організації реляційних БД на основі певного програмного продукту або вивчення конкретної реалізації СКБД. Мета сайта – розглянути загальну методологію й підхід. На їхній базі читач має виробити навички зі створення реляційних БД. Отже, вибраний програмний продукт має бути простим і легким у вивченні. Для цього, на думку авторів, найбільше підходять Mysql і InterBase.

На третьому етапі спрацював суб'єктивний фактор. У зв'язку з тим, що автори сайта програмували у середовищах розробки від фірми Borland, то СКБД Interbase на час написання посібника була найбільш прийнятним варіантом із продуктів, які були визначені на другому етапі. Таким чином, завершений і четвертий етап, тому що на третьому етапі вибір СКБД обґрунтований.

На практиці, із-за впливу великої кількості факторів, вибір СКБД важко формалізувати. Наприклад, наявність на підприємстві ліцензій на використання певної СКБД, досвід розробників у реалізації проектів на основі певної СКБД, специфіка конкретного проекту, бюджет, який виділений на проект та ін.

**4.3. Правила, згідно з якими СКБД може вважатися реляційною**

**4.3.1. Правила Кодда – як еталон реляційної СКБД**

На ринку існує декілька сотень реляційних СКБД. Ось лише деякі з них: Firebird [7]; Interbase [10, 14, 15]; Microsoft Access [16]; Microsoft SQL Server [17]; MySQL [18]; Oracle [19]; Visual FoxPro [20] і ін. На жаль, деякі з них не відповідають визначенню реляційної моделі. Ряд постачальників СКБД, які основані на мережній або ієрархічній моделі даних, реалізують у своїх продуктах тільки деякі риси реляційних систем. Це не заважає їм заявляти, що такі системи відносяться до реляційних. Стурбований тим, що потенційні можливості й зміст реляційного підходу спотворюються, Едгар Кодд у 1985 році запропонував 12 правил визначення реляційних систем (а точніше 13, якщо враховувати фундаментальне нульове правило). Ці правила утворюють свого роду еталон, за яким можна визначити приналежність СКБД до розряду дійсно реляційних систем.

Запропоновані Коддом правила протягом багатьох років викликали безліч нарікань з боку зацікавлених осіб і фахівців. Одні вважали їх не більш як чисто теоретичними вправами, інші заявляли, що їх продукти вже відповідають багатьом цим правилам. Ця дискусія сприяла зростанню розуміння найважливіших властивостей дійсно реляційних СКБД між користувачами й співтовариством розробників. Щоб підкреслити особливе значення цих правил, їх поділили на п'ять функціональних груп [4].

1. Фундаментальні правила.
2. Структурні правила.
3. Правила цілісності.
4. Правила керування даними.
5. Правила незалежності від даних.

**4.3.2. Фундаментальні правила (правила 0 і 12)**

За образним поданням правила 0 і 12 є «лакмусовим папірцем», який дозволяє визначати приналежність системи до реляційних СКБД. Якщо система не задовольняє цим правилам, то її не слід вважати реляційною.

***Правило 0****–****фундаментальне правило****. Будь-яка система, що рекламується або подається як реляційна СКБД, має бути здатною керувати БД виключно за допомогою її реляційних функцій.*

Це правило обумовлює, що СКБД не повинна прибігати до будь-яких не реляційних операцій для виконання таких видів робіт, як визначення даних і маніпулювання ними.

***Правило 12****–****заборона обхідних шляхів****. Якщо реляційна система має мову низького рівня (з послідовною порядковою обробкою), то вона не може бути використана для усунення або обходу правил і обмежень цілісності, складених реляційною мовою більш високого рівня (з обробкою відразу декількох рядків).*

Це правило гарантує, щоб всі спроби доступу до БД контролюються СКБД таким чином, що цілісність БД не була порушена без відома користувача або адміністратора. Це, однак, не виключає можливостей використання мови низького рівня з інтерфейсом послідовної обробки рядків.

**4.3.3. Структурні правила (правила 1 і 6)**

Фундаментальним структурним поняттям реляційної моделі є відношення. Кодд стверджує, що реляційна СКБД має підтримувати роботу з декількома структурними елементами: відношення, домени, первинні та зовнішні ключі. Для кожного відношення БД має бути визначений первинний ключ.

***Правило 1****–****подання інформації****. Уся інформація в реляційній БД подається в явному виді на логічному рівні й тільки одним способом – у вигляді значень у таблицях.*

Згідно з цим правилом уся інформація, навіть метадані в системному каталозі, має зберігатися у вигляді відношень, а керування здійснюється за допомогою тих самих функцій, що використовуються для роботи з даними. Згадування у цьому правилі «логічного рівня» означає, що такі фізичні конструкції, як індекси, нема сенсу подавати в моделі і користувач не повинен явно посилатися на них в операціях вилучення даних, навіть у тому разі, коли вони існують.

***Правило 6****–****оновлення представлення****. Усі представлення, які є теоретично обновлюваними, мають оновлюватися й у даній системі.*

Це правило має відношення тільки до представлень – віртуальних таблиць. Більш детально поняття представлення й умов його оновлення, які прийняті в мові *SQL*, будуть розглянуті пізніше. Це правило говорить, що якщо представлення є теоретично обновлюваним, то СКБД має вміти виконувати подібні оновлення. Насправді, жодна із існуючих систем не виконує цю вимогу, оскільки ще не визначені умови ідентифікації всіх теоретично обновлюваних представлень.

**4.3.4. Правила цілісності (правила 3 і 10)**

У своїх роботах Кодд запропонував два правили підтримки цілісності даних. Підтримка цілісності даних є важливим критерієм оцінки придатності системи для практичних потреб. Чим більше обмежень цілісності підтримує сама СКБД, а не окремі її прикладні програми, тим краще якість даних.

***Правило 3****–****систематична обробка невизначених значень****(****NULL****). Невизначені значення (що задаються за допомогою визначника NULL) – це значення, які відмінні від порожнього рядка або рядка порожніх символів, а також від нуля або будь-якого іншого конкретного значення, підтримуються для систематичного подання відсутньої або неприйнятної інформації, причому незалежно від типу даних.*

***Правило 10****–****незалежність обмежень цілісності****. Специфічні для даної реляційної СКБД обмеження цілісності повинні визначатися на субмові реляційних даних і зберігатися в системному каталозі, а не в прикладних програмах.*

Як підкреслює Кодд, відомості про встановлені обмеження цілісності даних мають зберігатися в системному каталозі, а не інкапсулюватися в окремих прикладних програмах або інтерфейсах користувача. Зберігання обмежень цілісності в системному каталозі надає велику перевагу централізованому керуванню та приведенню їх у дію.

**4.3.5. Правила керування даними (правила 2, 4, 5 та 7)**

Ідеальна реляційна СКБД має підтримувати 18 функцій керування даними, які визначають повноту мови запитів (тут термін «запит» містить операції вставки, оновлення та вилучення). Правила маніпулювання даними визначають спосіб застосування 18 функцій керування даними. Обов’язкове їх дотримання дозволяє ізолювати користувача та прикладні програми від фізичного та логічного механізмів реалізації засобів маніпулювання даними.

***Правило 2****–****гарантований доступ****. Для всіх та кожного із елементів даних (тобто його атомарного значення) реляційної БД має бути гарантований логічний доступ на основі комбінації імені таблиці, значення первинного ключа та значення імені стовпця.*

***Правило 4****–****динамічний інтерактивний каталог, побудований за правилами реляційної моделі****. Опис БД має подаватися на логічному рівні за таким само способом, як і звичайні дані, що дозволить санкціонованим користувачам звертатися до цього опису тією ж реляційною мовою, що й до даних.*

Це правило вказує на те, що має існувати тільки одна мова, яка призначена для маніпулювання як метаданими, так і звичайними даними; причому в СКБД для зберігання системної інформації має використовуватися тільки одна логічна структура – відношення.

***Правило 5****–****вичерпна субмова даних****. Реляційна система може підтримувати кілька мов і різні режими роботи з терміналами (наприклад, режим заповнення форми – fill-in-the-blanks). Однак має існувати, принаймні, одна мова, оператори якої дозволяли б виконувати операції: 1) визначення даних; 2) визначення представлень; 3) маніпулювання даними (команди доступні як інтерактивно, так і з програм); 4) обмеження цілісності; 5) авторизація користувачів; 6) організація трансакцій (запуск, фіксація та скасування).*

Слід зазначити, що новий стандарт ISO для мови *SQL* забезпечує виконання всіх цих функцій таким чином, що будь-яка мова, яка підтримує цей стандарт, автоматично буде відповідати й цьому правилу.

***Правило 7****–****високорівневі операції вставки, оновлення й вилучення****. Здатність обробляти базові або похідні відношення (тобто представлення) як єдиний операнд має стосуватися не тільки процедур витягу даних, але й операцій вставки, оновлення й вилучення даних.*

**4.3.6. Правила незалежності від даних (правила 8, 9 та 11)**

У своїх роботах Кодд визначає три правила незалежності даних від прикладних програм, які ці дані використовують. Обов’язкове дотримання цих правил гарантує, що користувачі і розробники будуть захищені від необхідності вносити тотальні зміни в прикладні програми при кожній реорганізації БД на низькому рівні.

***Правило 8****–****физическая независимость от данных****. Прикладні програми та засоби роботи з терміналами мають залишатися логічно незайманими при внесенні будь-яких змін у способи зберігання даних або методи доступу до них.*

***Правило 9****–****логічна незалежність від даних****. Прикладні програми та засоби роботи з терміналами мають залишатися логічно незайманими при внесенні в базові таблиці будь-яких змін, що не міняють інформацію та теоретично не чіпають прикладне програмне забезпечення.*

***Правило 11****–****незалежність від розподілу даних****. Субмова маніпулювання даними в реляційній СКБД повинна дозволяти прикладним програмам та запитам залишатися логічно незмінними, незалежно від того, як зберігаються дані фізично: централізовано або в розподіленому вигляді.*

Незалежність від розподілу даних означає, що прикладна програма, здійснюючи доступ до СКБД на окремому комп'ютері, повинна без жодних модифікацій продовжувати працювати й тоді, коли дані в мережі будуть перенесені з одного комп'ютера на інший. Інакше кажучи, для кінцевого користувача мають бути створені умови, згідно з якими ілюзія того, що дані централізовано зберігаються на єдиному комп'ютері, а відповідальність за переміщення та пошук місць їх зберігання безпосередньо покладається на систему. Зверніть увагу, ще й на те, що тут не сказано про те, що реляційна СКБД має неодмінно підтримувати роботу з розподіленою БД. Тут мається на увазі, якщо можливість роботи з розподіленими даними реалізується в деякій СКБД, мова запитів повинна залишатися незмінною.

**4.4. Мова SQL**

**4.4.1. Стисла характеристика мови SQL**

Мова *SQL* є першою й поки єдиною стандартною мовою роботи з БД, яка набула найбільшого поширення [4, 6 – 8]. Є ще одна стандартна мова роботи з БД – Network Data-base Language (*NDL*), яка побудована на основі мережної моделі *CODASYL*, але застосовується вона не в усіх розробках. Практично усі розробники СКБД нині створюють свої продукти з використанням мови *SQL* або *SQL*-інтерфейсом.

В ідеалі будь-яка мова роботи з БД має дозволяти користувачу:  
1) створювати БД і таблиці з повним описом їх структури;  
2) виконувати основні операції маніпулювання даними, такі як вставка, модифікація й вилучення даних з таблиць;  
3) виконувати прості та складні запити, здійснювати перетворення неопрацьованих даних у необхідну інформацію;  
4) застосовувати стандартний синтаксис і структуру команд при переході від однієї СКБД до іншої.

Крім того, мова роботи з БД повинна вирішувати всі розглянуті раніше завдання при мінімальних зусиллях з боку користувача, а структура й синтаксис його команд мають бути досить простими й доступними при вивченні. Мова *SQL* відповідає практично всім цим вимогам.

*SQL* є прикладом мови з орієнтацією на трансформацію, або ж мови, яка призначена для роботи з таблицями з метою перетворення вхідних даних до необхідного вихідного вигляду. Мова *SQL*, складається з:  
1) мови *DDL* (*Data Definition Language*), призначена для визначення структур БД і керування доступом до даних;  
2) мови *DML* (*Data Manipulation Language*), призначена для вибірки та оновлення даних.

Мова *SQL* включає тільки команди визначення й маніпулювання даними. У неї відсутні будь-які команди керування ходом обчислень. Інакше кажучи, у цій мові немає команд *IF ... THEN ... ELSE, GO TO, DO ... WHILE* і т.п., що призначені для керування ходом обчислювального процесу. Подібні завдання мають вирішуватися за допомогою мов програмування або інтерактивно, як результат дій користувача. Через подібну незавершеність стосовно обчислювального процесу мова *SQL* може використовуватися двома способами. Перший – передбачає інтерактивну роботу, що полягає в уведенні користувачем з термінала окремих *SQL*-операторів. Другий – полягає у впровадженні *SQL*-операторів до програм, що написані на процедурних мовах.

Мова *SQL* відносно проста у вивченні.

1. Це не процедурна мова, а тому в ній необхідно вказувати, яка інформація повинна бути отримана, а не спосіб її отримання. Інакше кажучи, *SQL* не вимагає опису методів доступу до даних.
2. Як і більшість сучасних мов, *SQL* має вільний формат запису операторів. Це означає, що при введенні окремі елементи операторів не пов'язані з фіксованими позиціями екрана.
3. Структура команд задається набором ключових слів, що являють собою звичайні слова та фрази англійською мовою, наприклад, *CREATE TABLE* (створити таблицю), *INSERT* (вставити), *SELECT* (вибрати) та ін.
4. Мова *SQL* може використовуватися великою групою фахівців, включаючи адміністраторів БД, керівників, програмістів та безліч інших типів кінцевих користувачів.

На час створення сайту мова *SQL* визначаэться міжнародним стандартом ISO/IEC 9075 (1 – 4, 9 – 11, 13, 14):2008, що формально перетворює її у стандартну мову визначення й маніпулювання реляційними БД. Однак слід зазначити, що розробники СКБД відходять від стандартів, або використовують стандарти, які були розроблені значно раніше. Тому для фізичної реалізації БД потрібно використовувати відповідну технічну документацію розробника СКБД. Усі приклади та фізична реалізація БД «БІБЛІОТЕКА» відпрацьовані в СКБД Interbase від Embarcadero.

**4.4.2. Запис SQL-операторів**

*SQL*-оператор складається із зарезервованих слів, а також зі слів, які визначаються користувачем [4, 6]. Зарезервовані слова є незмінною частиною мови *SQL* і мають фіксоване значення. Їх слід записувати відповідно до встановлених стандартів, тобто не розбивати на частини для переносу з одного рядка в інший. Слова, які визначаються користувачем, задаються ним самим (відповідно до певних синтаксичних правил) і являють собою імена різних об'єктів БД: таблиць, стовпців, представлень, індексів і т.д. В *SQL*-операторі слова розміщуються відповідно до встановлених синтаксичних правил. Хоча в стандарті це не зазначено, багато діалектів мови *SQL* вимагають використання наприкінці оператора деякого символу, що позначає закінчення його тексту. Як правило, це «крапка з комою».

Більшість компонентів *SQL*-операторів не чутливі до регістру. Це означає, що можуть використовуватися будь-які літери – як малі, так і великі. Одним важливим виключенням із цього правила є символьні літерали – дані, які не мають відрізнятися від тих, що зберігаються в БД. Наприклад, якщо в БД зберігається символьний літерал прізвища ‘*ІВАНОВ*’, а в умові пошуку воно подано малими літерами, то цей запис не буде знайдено.

Оскільки мова *SQL* має вільний формат, то окремі *SQL*-оператори та їх послідовності з використанням відступів та вирівнювання будуть мати більшу читабельність. Рекомендується дотримуватися таких правил.

1. Кожна фраза оператора має починатися з нового рядка.
2. Початок кожної фрази має бути вирівняний з початком інших фраз оператора.
3. Якщо фраза складається з декількох частин, то кожна з них має починатися з нового рядка з деяким відступом відносно початку фрази, що буде вказувати на їхню підпорядкованість.

Для визначення формату *SQL*-операторів будемо дотримуватися такої форми їх запису.

1. Великі літери будуть використовуватися для запису зарезервованих слів і повинні вказуватися в операторах так само, як це буде показано.
2. Малі літери будуть використовуватися для запису слів, які визначені користувачем.
3. Вертикальна риска ( | ) вказує на вибір одного значення з декількох наведених. Наприклад, *a | b | c*.
4. Фігурні дужки визначають обов'язковий елемент, наприклад, {*а*}.
5. Квадратні дужки визначають необов'язковий елемент, наприклад, [*а*].
6. Три крапки (...) використовуються для відображення необов'язкової можливості повторення конструкції, від нуля до декількох разів, наприклад, {*а | b*} [, *c*…]. Цей запис означає, що після *a* або *b* може слідувати від нуля до декількох повторень *c*, які поділені комами.

На практиці для визначення структури БД використовується *DDL*-оператори, а для заповнення відношень даними й вибору з них інформації за допомогою запитів – *DML*-оператори.

**4.5. SQL-оператори, що реалізовують структуру реляційних даних**

На етапі вибору системи керування БД ми зупинилися на СКБД Interbase [10, 14, 15]. Відзначили, що в наші плани не входить детальне вивчення всіх нюансів роботи із цим програмним продуктом. Тих, кого зацікавлять окремі деталі створення й керування БД саме за допомогою цього продукту, традиційно запропонуємо ознайомитися з документацією, що поставляється розробником.

СКБД Interbase дозволяє створювати БД як в інтерактивному режимі, по черзі виконуючи *SQL*-оператори, так і в режимі інтерпретації SCRIPT-файлу. Практична частина курсу побудована на створенні SCRIPT-файлу та його послідовному наповненні командами. Рекомендовану послідовність команд в SCRIPT-файлі розглянемо на практичних заняттях. Тут обмежимося загальним розглядом *SQL*-операторів, що створюють на жорсткому диску файл БД і його структуру, що відповідає логічній моделі.

Для створення файлу (або файлів) БД на жорсткому диску використовують оператор *CREATE DATABASE*. Він дозволяє:

1. Зазначити ім'я файлу (або файлів) БД та місце його розташування на локальному або мережному жорсткому диску.
2. Указати ім'я й пароль користувача, який використовується при приєднанні до БД.
3. Визначити набір символів, застосовуваних у БД за умовчуванням. Ми будемо використовувати набір WIN1251.
4. Для БД указати розмір файлу в сторінках, розмір кожної сторінки в байтах, якщо БД розташовується в декількох файлах – указати, з якої сторінки починається кожний із цих файлів.

Домени створюються оператором *CREATE DOMAIN*, в якому необхідно кожному домену надати унікальне ім'я. Він дозволяє:

1. Зазначити тип даних, на базі яких створюється домен.
2. Указати значення, прийняте за замовчуванням.
3. Встановити конструкції умов перевірки значень домену.
4. Визначити таблицю, за якою буде здійснюватися сортування значень домену.

Для створення таблиць та ключів застосовується оператор *CREATE TABLE*. Таблиця повинна мати унікальне ім'я, причому імена стовпців у її межах не повинні повторюватися. За допомогою цього оператора можна визначити:

1. для кожного стовпця таблиці домен або тип даних і конструкції, що дозволяють визначити припустимі значення, які можна буде вводити в цей стовпець;
2. список стовпців, що входять у первинний ключ;
3. список стовпців потенційних ключів;
4. список стовпців, що входять у зовнішні ключі таблиці, і вказати імена таблиць, на первинні ключі яких посилаються значення зовнішніх ключів;
5. дії при каскадному оновленні або вилученні рядків;
6. умови перевірки значень стовпців при додаванні нового рядка в таблицю.

Послідовність створення таблиць у БД відіграє велике значення. Спочатку створюються ті таблиці, у яких відсутні зовнішні ключі, дали таблиці, де зовнішні ключі мають зв'язок з первинними ключами вже створених таблиць. Як приклад розглянемо процес створення таблиць «ПОСАДА», «ЧИТАЧІ», «ТИПИ ТЕЛЕФОНІВ», «ТЕЛЕФОНИ» (рис. 3.4). Тут можна поміняти місцями послідовність створення таблиць «ЧИТАЧІ» і «ТИПИ ТЕЛЕФОНІВ». Однак, таблиця «ЧИТАЧІ» завжди має створюватися після таблиці «ПОСАДА», а таблиця «ТЕЛЕФОНИ» – після таблиць «ТИПИ ТЕЛЕФОНІВ» і «ЧИТАЧІ». А якщо ні, то СКБД укаже на помилку.

Структура БД «БІБЛІОТЕКА», яка отримана за допомогою відповідних операторів *CREATE TABLE*, відображає зв'язок між логічною й фізичною моделями, (дод. Б, де літерою *Р* позначені первинні ключі, а літерою *F* – зовнішні). Позначення первинних і зовнішніх ключів відповідають позначенням зв'язків між відношеннями, що наведені в дод. А. Стрілки вказують, що значення зовнішніх ключів у дочірніх відношеннях обов'язково мають посилатися на відповідні значення первинних ключів батьківських відношень.

**4.6. Додавання, оновлення й вилучення даних**

**4.6.1. Невідоме або неприйнятне значення**

***Визначник NULL****вказує на те, що значення атрибута в даний момент невідоме або неприйнятне для цього кортежу.*

Визначник *NULL* слід сприймати як логічну величину «невідомо». Інакше кажучи, або це значення не входить в область визначення деякого кортежу, або жодне з них ще не задане. Ключове слово *NULL* являє собою спосіб обробки неповних або незвичайних даних. Однак визначник *NULL* не слід розуміти як нульове чисельне значення або заповнений пробілами текстовий рядок. Нулі й пробіли є значеннями, тоді як ключове слово *NULL* має вказувати на відсутність будь-якого значення. Деякі автори використовують термін «значення *NULL*», але насправді визначник *NULL* не є значенням, а лише позначає його відсутність, а тому термін «значення *NULL*» використовувати не рекомендується.

Застосування визначника *NULL* може викликати проблеми на етапі реалізації. Складності виникають через те, що реляційна модель побудована на обчисленні предикатів першого порядку, яке має двозначну, або булеву логіку, тобто припустимими є тільки два значення: істина й хибність. Застосування визначника *NULL* означає, що потрібно працювати з логікою більш високого рівня, наприклад, тризначною або навіть чотиризначною (Codd, 1986, 1987, 1990).

Використання поняття *NULL* у реляційній моделі є дискусійним питанням. У своїх працях Кодд (1990) розглядає поняття *NULL* як складову частину цієї моделі, коли інші фахівці вважають цей підхід не зовсім точним. Вони припускають, що включення визначника *NULL* у реляційну модель є передчасним (Date, 1995). Слід зазначити, що не усі реляційні системи підтримують роботу з визначником *NULL*, але в СКБД Interbase цей визначник присутній.

**4.6.2. Забезпечення цілісності реляційних даних**

Насамперед, цілісність забезпечується первинними (потенційними) ключами базових або батьківських відношень [2, 4, 6]. Тут батьківське відношення визначається як таке, що відповідає деякому суб'єкту (сутності) технологічного процесу в логічній моделі. Наприклад, відношення «ПОСАДИ» є батьківським для відношення «ЧИТАЧІ» (рис. 3.4). Відношення «ЧИТАЧІ» є батьківським для того відношення, де значення зовнішнього ключа посилаються на значення одного з потенційних ключів відношення «ЧИТАЧІ».

***Цілісність сутностей****у базовому відношенні забезпечується тим, що жоден із атрибутів первинного ключа не може містити невідомих або неприйнятних значень (див. визначник NULL).*

Припущення присутності визначника *NULL* у будь-якій частині потенційного ключа рівноцінно твердженню, що не всі його атрибути необхідні для унікальної ідентифікації кортежів. Наведене твердження суперечить визначенню потенційного ключа.

Цілісність реляційних даних забезпечується зовнішніми ключами в дочірніх відношеннях. Дочірніми називаються відношення, де присутні зовнішні ключі, значення яких повинні посилатися на значення потенційних ключів у батьківських відношеннях.

***Посилальна цілісність****забезпечується, якщо значення зовнішнього ключа дочірнього відношення посилається на існуюче значення потенційного ключа батьківського відношення, або задається за допомогою визначника NULL.*

Допустимість наявності визначника *NULL* серед значень атрибутів, що використовуються як зовнішній ключ, встановлюється вимогами користувачів. Наприклад, якщо користувач вважає, що інформація про посаду є необов'язковою, то в стовпці «Код посади» відношення «ЧИТАЧІ» може знаходитися або визначник *NULL*, або значення первинного ключа, яке має місце в атрибуті «Код» відношення «ПОСАДА». Якщо користувач вважає, що інформація про посаду є обов'язковою, тоді в стовпці «Код посади» наявність визначника *NULL* неприпустима.

**4.6.3. Оператори для додавання, зміни й вилучення даних**

Оператор *INSERT* дозволяє додавати новий рядок у таблицю з іменем, яке Ви вказуєте. Він дозволяє перелічувати стовпці таблиці, у яких значення в новому рядку будуть відрізнятися від тих, які прийняті за замовчуванням. Якщо на рівні домену або в оператору *CREATE TABLE* значення за замовчуванням не визначені, то в чарунках нового рядка, які розташовані в стовпцях, що не попали в список, буде записаний визначник *NULL*. Кількість стовпців і кількість доданих значень, які перелічені в операторі, має збігатися. Якщо хоча б одне значення виходить за область його визначення, то дія оператора *INSERT* буде відхилена й СКБД видасть відповідне повідомлення.

Послідовність додання нових рядків у таблиці не є довільною. Це пов'язане з тим, що поки в батьківську таблицю не буде доданий рядок з деяким значенням потенційного ключа, в дочірню таблицю не можна буде додати рядок із значенням зовнішнього ключа, яке відповідає значенню потенційного ключа в батьківській таблиці. Для виключення помилок додавайте нові рядки в таблиці, дотримуючись послідовності їх створення в БД операторами *CREATE TABLE*.

Для визначення унікального значення первинного ключа в СКБД Interbase застосовується механізм генераторів. Для кожного атрибута, який передбачається використовувати як первинний ключ, визначається генератор. Для цього служить оператор *CREATE GENERATOR*. За його допомогою можна задати крок, на який нове значення потенційного ключа буде відрізнятися від попереднього. Оператор

**SET GENERATOR** {Ім'я} **TO** {Початкове значення}

дозволяє вказати значення, з якого почне свою роботу генератор. В інших СКБД механізм генераторів вбудований в оператор *CREATE TABLE*, де можна вказати інкрементний тип даних для стовпця таблиці.

Оператор *UPDATE* змінює значення в рядках таблиці, ім'я якої обов'язково потрібно вказати. Він дозволяє змінювати значення водночас у будь-якій кількості стовпців таблиці. Як і для оператора *INSERT*, при виході нового значення за межі області визначення, СКБД згенерує повідомлення про помилку, і дію оператора *UPDATE* буде скасовано

Перелік рядків таблиці, у яких буде працювати оператор *UPDATE*, визначається умовами пошуку. У більшості випадків в умові пошуку наводиться область припустимих значень первинного або потенційного ключа таблиці. Дані оновляються у тих рядках, де логічний вираз для пошуку повертає значення «істинно». Якщо Ви не вкажете умови пошуку рядків для оновлення, то в усіх рядках таблиці дані будуть змінені.

Вилучення рядків з таблиці здійснюється оператором *DELETE*. Умова пошуку працює так само, як і в операторі *UPDATE*. Якщо умови пошуку не вказувати, то з таблиці будуть вилучені усі рядки.

**4.6.4. Каскадне оновлення й вилучення**

Цей механізм, що є одним із засобів забезпечення посилальної цілісності даних, реалізується в операторові *CREATE TABLE*. Він дозволяє розробнику вказати дію для СКБД, яку необхідно здійснити системі при виконанні операторів *UPDATE* або *DELETE*, що спробують вилучити або оновити таке значення потенційного ключа в батьківській таблиці, на яке посилається один або кілька рядків дочірньої таблиці.

Мова *SQL* передбачає чотири варіанти реакції, як для оновлення, так і для вилучення значень потенційного ключа з батьківської таблиці, на яке є посилання значень зовнішнього ключа з рядків дочірньої таблиці. Варіант дії наведеться окремо для оператора *UPDATE* або *DELETE* при визначенні зовнішнього ключа в дочірній таблиці. Дії вказуються після ключових фраз *ON UPDATE* або *ON DELETE*. Вони встановлюються відповідно до вимог користувачів, які були отримані на етапі логічного проектування БД.

Якщо після *ON UPDATE* указати слово *CASCADE*, то в цьому випадку буде виконуватися каскадне оновлення значень зовнішнього ключа в рядках дочірньої таблиці при зміні оператором *UPDATE* відповідного значення потенційного ключа в батьківській таблиці. Якщо *CASCADE* написати після *ON DELETE*, то виконання оператора *DELETE*, що вилучає рядок зі значенням потенційного ключа в батьківській таблиці, спричиняє до вилучення рядків у дочірній таблиці, де значення зовнішнього ключа збігається зі значенням потенційного ключа, яке вилучається.

Фраза *SET NULL*, яка написана після *ON UPDATE* або *ON DELETE*, говорить системі, що після вилучення або зміни значення потенційного ключа в батьківській таблиці необхідно прописати визначник *NULL* у всіх рядках дочірньої таблиці, у яких значення зовнішніх ключів відповідали вилученим або зміненим значенням потенційного ключа.

Наступним варіантом реакції СКБД на вилучення значення потенційного ключа з батьківської таблиці є прописування у всіх атрибутах зовнішнього ключа значень, визначених за замовчуванням. Для цього після фрази *ON DELETE* необхідно написати *SET DEFAULT*. Нагадаємо, що для кожного атрибута можна вказати значення, які в ньому прописуються за замовчуванням або при ініціалізації доменів, або безпосередньо в операторі *CREATE TABLE*.

Пропозиція *NO ACTION*, що написана після *ON UPDATE* або *ON DELETE*, визначає четвертий варіант реакції СУБД на зміну значення потенційного ключа у рядку батьківської таблиці або вилучення такого рядка. У цьому випадку система видає повідомлення про помилку, оскільки така дія порушує посилальну цілісність даних.

Наприклад, спроба вилучити рядок з таблиці «ПОСАДА» (рис. 3.4) може призвести до того, що у деяких читачів значення зовнішнього ключа буде посилатися на неіснуюче значення первинного ключа. Така ситуація неприпустима. Однак вилучення рядка з таблиці «ПОСАДА» зі значенням первинного ключа, на яке посилання з таблиці «ЧИТАЧІ» відсутнє, відбудеться без проблем. У цьому випадку посилальна цілісність даних порушена не буде.

Якщо в операторі *CREATE TABLE*, який описує структуру дочірньої таблиці, не вказані фрази *ON UPDATE* або *ON DELETE* у визначенні зовнішнього ключа, то механізм забезпечення посилальної цілісності при вилученні або зміні значення потенційного ключа в батьківській таблиці працює так само, як і при фразі *NO ACTION*.

Вміст БД «БІБЛІОТЕКА», яка буде використовуватися для вивчення прийомів вибірки інформації, наведений в дод. В.

**4.7. КОНТРОЛЬНІ ПИТАННЯ**

1. У чому полягає сутність фізичного проектування БД?
2. Що впливає на вибір СКБД для створення фізичної моделі БД?
3. За якою групою правил СКБД вважають реляційною?
4. Для чого використовують відомі Вам компоненти мови *SQL*?
5. Яких правил треба дотримуватися при записі *SQL*-операторів?
6. Які характеристики має *SQL*-оператор, що створює БД?
7. Якими можливостями володіє *SQL*-оператор, що визначає домени?
8. Які структурні елементи таблиць визначаються *SQL*-оператором?
9. На що вказує і для чого використовується визначник *NULL*?
10. Як забезпечуються цілісність сутностей та посилальна цілісність?
11. За допомогою яких *SQL*-операторів маніпулюють даними у таблицях?
12. Що визначають умови пошуку для *SQL*-операторів, які оновлюють або вилучають дані з таблиці?
13. Які засоби для забезпечення унікальності значень первинних ключів Вам відомі?
14. Як СКБД реалізує каскадне оновлення и вилучення даних?

**Висновок**

*Успіх фізичної реалізації БД залежить від результатів логічного моделювання. Вибираючи цільову реляційну СКБД, пам'ятайте про правила, які визначають її приналежність до дійсно реляційних систем. Останнє допоможе використовувати першу й поки єдину стандартну мову, яка надає широкі можливості для визначення й маніпулювання реляційними БД, – мову SQL.*