

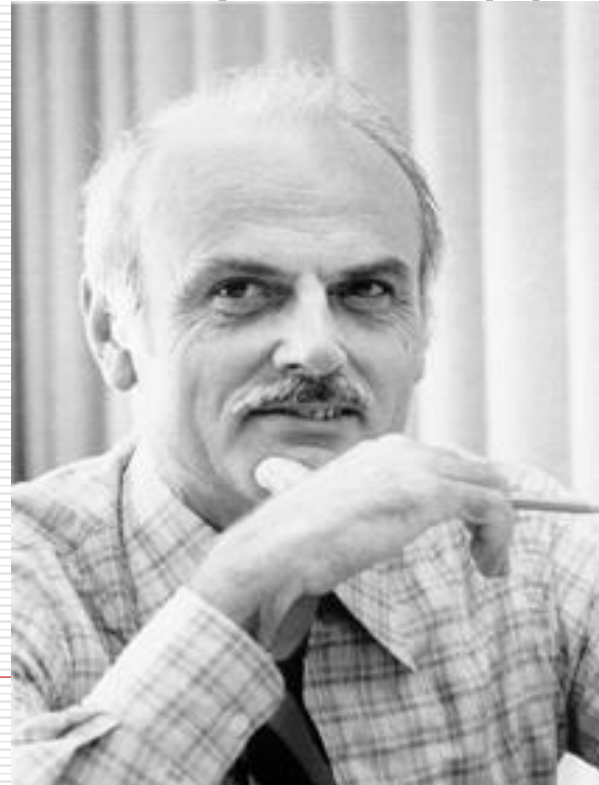
Тема лекції 2:

Реляційна модель даних

- ❑ Історія реляційної моделі даних
 - ❑ Реляційна структура даних
 - ❑ Структурна частина реляційної бази даних
 - ❑ Частина цілісності реляційної бази даних
 - ❑ Маніпулятивна частина реляційної бази даних
 - ❑ Стандарти SQL
 - ❑ Типи даних в SQL. Перетворення типів
 - ❑ SQL-операції
-

Історія реляційної моделі даних

- Теоретичні основи закладені американським вченим Едгаром Коддом на початку 70-х рр. XX ст.



Історія реляційної моделі даних

- ❑ перші прототипи реляційних СУБД - 70-х рр. XX ст.
 - ❑ реляційні системи витіснили зі світового ринку попередні СУБД ієрархічного та мережевого типів - 80-х рр. XX ст.
-

12 (13) правил Кодда

- 13 правил (пронумерованих від нуля до дванадцяти)
запропоновані Е.Коддом для
визначення того чи є СУБД реляційною
 - Кодд створив ці правила як частину
своєї кампанії запобігання розмиванню
його бачення реляційності
 - оскільки продавці систем керування
базами даних на початку 1980х просто
видавали свої старі продукти за реляційні
розробки
-

Переваги та недоліки реляційної моделі даних

Переваги відносно мережевої та ієрархічної моделі:

- ❑ простота,
- ❑ гнучкість структури,
- ❑ зручність реалізації на комп'ютері,
- ❑ наявність теоретичного опису

Недоліки:

- ❑ обмеженість під час використання складних структур даних (у системах автоматизованого проектування),
 - ❑ відсутність засобів, які адекватно відображають семантику предметної області
-

Теоретична основа реляційних баз даних

- теорія множин
 - реляційна алгебра (алгебра відношень)
-

Академічна основа реляційних баз даних

- **Крістофер Дейт** (Christopher J. Date) — один з найвидатніших спеціалістів в галузі баз даних, особливо в реляційної моделі даних, незалежний автор, лектор та консультант



Академічна основа реляційних баз даних

- ❑ Крістофер Дейт — автор класичного підручника «Вступ до баз даних», що використовується як стандарт в безлічі університетах світу.
 - ❑ Монографія видана тиражем 700 тис. екземплярів тільки мовою оригіналу, не враховуючи перекладів
 - ❑ *Вважається, що книга Дейта входить в «золотий фонд» комп'ютерної літератури, за своєю значимістю відповідає таким відомим працям як «Мистецтво програмування» Кнута*
-

Реляційна структура даних (за К. Дейтом)

- **Структурна частина** описує, які об'єкти розглядаються реляційною моделлю. За Дейтом, реляційна база даних – це база даних, яка складається з відношень. Схемою реляційної бази даних називається набір заголовків відношень, які входять у базу даних.
 - **Частина цілісності** описує обмеження спеціального виду, які повинні виконуватися для будь-яких відношень у будь-яких базах даних. Це цілісність сутностей і цілісність посилань.
 - **Маніпулятивна частина** описує два еквівалентних способи маніпулювання реляційними даними – реляційна алгебра і реляційне числення. З практичної точки зору, важливим є висновок про реляційну повноту структурованої мови запитів SQL у тому чи іншому стандарті, яка і реалізує маніпулятивну частину реляційної моделі у реальних СУБД.
-

Структурна частина реляційної бази даних. Відношення, таблиця.

- ❑ Реляційна модель базується на математичному понятті **відношення** (лат. relatio) , а фізичне представлення його – це **таблиця**.
 - ❑ **Таблиця** має жорстко обумовлену кількість поіменованих та впорядкованих стовпців (структуру), і може необмежено рости за кількістю рядків. В таблиці рядки відповідають певним записам, а стовпці – атрибутам.
-

Структурна частина реляційної бази даних. Відношення і таблиця

1. Відношення – це проста таблиця, в якій усі рядки містять однакову кількість комірок і у відповідних комірках містяться однакові типи даних
 2. Відношення – це таблиця з критерієм який дозволяє визначити, які рядки входять у таблицю, а які ні. Цей критерій визначає сенс, або семантику, відношення
 3. Оскільки відношення не має однакових кортежів, і кортежі є невпорядковані зверху донизу, то одне і те ж відношення можна представити різними таблицями, в яких рядки мають різний порядок.
-

Структурна частина реляційної бази даних. Атрибут.

- ❑ **Атрибут** – це поіменований стовпець відношення.
 - ❑ Атрибути можуть бути розміщеними в будь-якому порядку. Незалежно від їх розміщення відношення буде залишатись одним і тим же, а тому мати той же зміст.
-

Структурна частина реляційної бази даних. Домен.

- ❑ Кожен атрибут реляційної бази даних визначається на деякому домені.
 - ❑ **Домен** – це набір допустимих значень для одного або декількох атрибутів.
 - ❑ Через домени користувач може визначати зміст та джерело значень, які можуть отримувати атрибути.
 - ❑ У багатьох реляційних СУБД домени підтримуються лише частково.
-

Структурна частина реляційної бази даних. Кортеж.

- ❑ Елементами відношення є **кортежі**, тобто рядки таблиці.
 - ❑ Кортежі можуть бути розміщеними в будь-якому порядку, при цьому відношення залишається одним і тим же.
-

Структурна частина реляційної бази даних. Заголовок відношення.

- ❑ Опис структури відношення разом зі специфікацією доменів та інших обмежень щодо можливих значень атрибутів називають його **заголовком** (або змістом (intension)).
 - ❑ Заголовок відношення містить фіксовану кількість назв атрибутів. Імена атрибутів повинні бути унікальними у межах відношення.
 - ❑ Заголовок є фіксованим до тих пір, поки зміст відношення не зміниться за рахунок додавання в нього додаткових атрибутів.
-

Структурна частина реляційної бази даних. Тіло відношення.

- ❑ Кортежі називаються розширенням (extension), станом (state), а набір кортежів – **тілом** відношення, яке постійно змінюється.
 - ❑ Тіло відношення є підмножиною декартового добутку доменів, що і є відношенням з математичної точки зору.
-

Структурна частина реляційної бази даних. Відношення.



Термінологія в реляційній моделі

Офіційні терміни	Альтернативний варіант 1	Альтернативний варіант 2
Відношення	Таблиця	Таблиця
Кортеж	Рядок	Запис
Атрибут	Стовпець	Поле

Частина цілісності реляційної бази даних. Обмеження домену

- ❑ Домен розглядається як підмножина значень деякого типу даних, які мають певний зміст.
 - ❑ Домен – це семантичне поняття, яке несе певне змістовне навантаження.
 - ❑ Домен має унікальне ім'я у межах бази даних, він визначений на простому типі даних або на іншому домені.
 - ❑ Наявність логічної умови, яка дозволяє описати підмножину даних, допустимих для цього домену. Наприклад, домен D , який має зміст «вік співробітника», можна описати як наступну підмножину множини натуральних чисел: $(D = n \in \mathbb{N} : n \geq 18 \text{ and } n \leq 65)$.
 - ❑ Основне призначення доменів: вони обмежують порівняння. Некоректно, з логічної точки зору, порівнювати значення з різних доменів, навіть якщо вони мають однаковий тип.
-

Ключове слово NULL в реляційній моделі

- ❑ NULL вказує, що значення атрибута в даний момент невідоме, або неприйнятне для цього кортежу
 - ❑ NULL є способом опрацювання невизначених, неповних або незвичних даних
 - ❑ NULL не слід розуміти як нульове значення або заповнений пробілами текстовий рядок.
-

Частина цілісності реляційної бази даних. Реляційний ключ

- ❑ **Ключ** відношення – це атрибут чи множина атрибутів, який однозначно ідентифікує кортеж даного відношення.
 - ❑ Простий ключ складається з одного атрибута, а складений – з декількох атрибутів.
 - ❑ Поля, за якими побудовано ключ, називаються ключовими.
-

Частина цілісності реляційної бази даних. Реляційний ключ

STUDENT

STUDENT_ID

STUDENT_NAME

ADDRESS

DOB

COURSE

PERSON

EMPLOYEE_NAME

EMPLOYEE_DOB

EMP_ADDRESS

EMP_PASSPORT_NUM

LICENCE_NUMBER

SSN

Складений реляційний ключ

STUDENT_COURSE		
STUDENT_ID	COURSE_ID	No_of_Hours
100	401	20
100	402	15
101	401	25
103	403	10

Типи реляційних ключів.

Потенційний ключ

- ❑ **Потенційний ключ** – це ключ, який є унікальним і ненадлишковим.
 - ❑ Будь-яке відношення має хоча б один потенційний ключ. Дійсно, якщо ніякий атрибут або група атрибутів не є потенційним ключем, то завдяки унікальності кортежів усі атрибути разом утворюють потенційний ключ, т.зв. комбінований ключ (Compound key).
 - ❑ Відношення може мати декілька потенційних ключів. Традиційно один з них оголошується первинним ключем, а інші альтернативними.
-

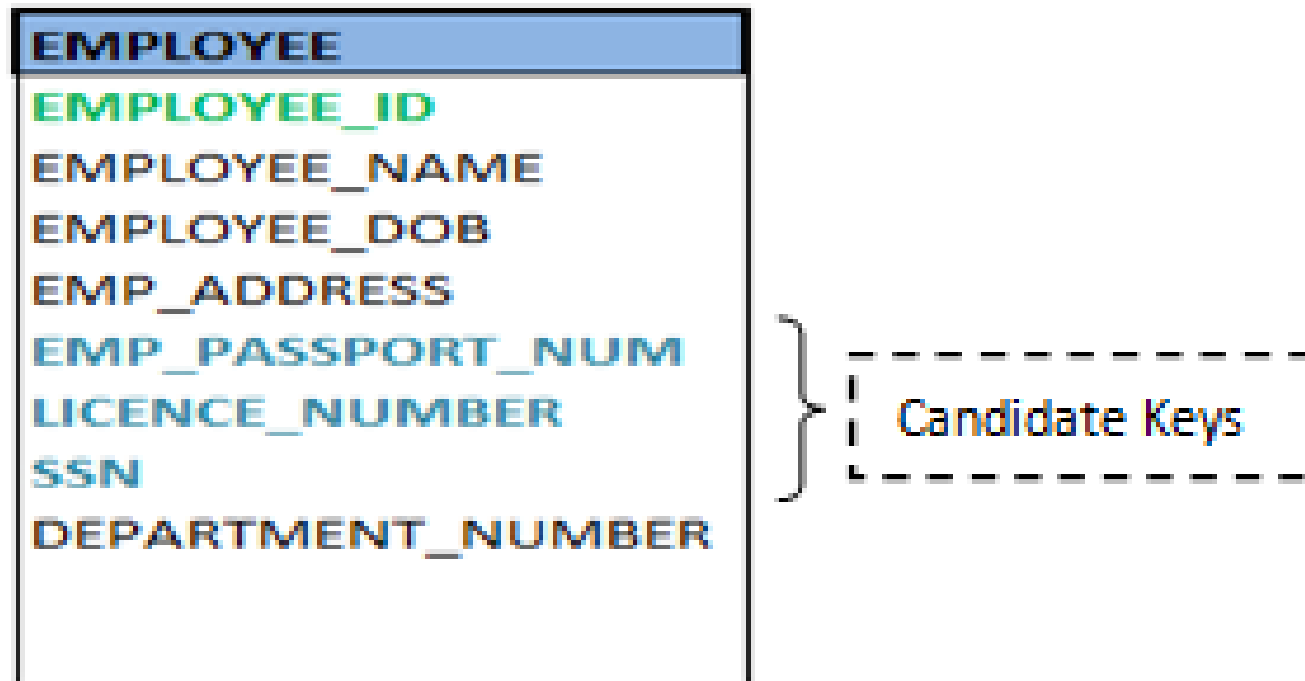
Типи реляційних ключів.

Первинний і альтернативні ключі

- ❑ **Первинний ключ** (PK, Primary Key) – це потенційний ключ, який вибраний для унікальної ідентифікації кортежів всередині відношення.
 - ❑ Відношення не обов'язково повинне мати первинний ключ, але бажано завжди визначати потенційний ключ.
 - ❑ Потенційні ключі, які не вибрані в якості первинного ключа, називаються **альтернативними ключами**.
-

Типи реляційних ключів.

Первинний і альтернативні ключі



Типи реляційних ключів.

Зовнішній ключ

- ❑ **Зовнішній ключ** (FK, Foreign Key) – це один або декілька атрибутів відношення (дочірнє відношення), які одночасно є потенційними ключами іншого відношення (батьківське відношення).
 - ❑ Зовнішній ключ, як і потенційний, може бути простим і складеним.
 - ❑ Зовнішній ключ повинен бути визначений на тих же доменах, що і відповідний первинний ключ батьківського відношення.
-

Реляційні ключі

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID	PROJ_ID
100	Joseph	Clinton Town	10	206
101	Rose	Fraser Town	20	205
102	Mathew	Lakeside Village	10	206
103	Stewart	Troy	30	204
104	William	Holland	30	202

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Значення реляційного ключа

- ☐ однозначна ідентифікація рядків таблиці;
 - ☐ попередження повторень значень атрибута;
 - ☐ прискорення виконання запитів до БД;
 - ☐ встановлення зв'язків між окремими таблицями БД;
 - ☐ використання обмежень цілісності посилань
-

Реляційна цілісність

Задаються два **правила цілісності**, які є обмеженнями для всіх допустимих станів бази даних. Ці два основних правила реляційної моделі називаються:

- ❑ цілісністю сутностей
- ❑ цілісністю посилань

СУБД вважається реляційною у повному сенсі, якщо у неї є стандартні засоби автоматичної реалізації обмежень цілісності сутностей і цілісності посилань.

Правило цілісності сутностей

- ❑ Сутність в реляційній моделі - це синонім відношення або таблиці
 - ❑ Обмеження цілісності сутностей стосується первинних ключів базових відношень (відношень, які реально існують в базі даних)
 - ❑ **Правило:** в базовому відношенні ні один атрибут первинного ключа не може містити невизначених значень, що позначаються словом NULL.
-

Правило цілісності посилань

- ❑ Обмеження цілісності посилань стосується зовнішніх ключів.
 - ❑ **Правило:** якщо у відношенні існує зовнішній ключ, то значення зовнішнього ключа повинно відповідати значенню потенційного (первинного) ключа деякого кортежу в його батьківському відношенні або задаватись словом NULL.
 - ❑ Обернене твердження є невірним, оскільки у полі зв'язку батьківської таблиці можуть бути значення, на які не посилається ні одне значення зовнішнього ключа
-

Зовнішні ключі і типи зв'язку між відношеннями

1. Зовнішній ключ не володіє властивістю унікальності. В дочірньому відношенні може бути декілька кортежів, які посилаються на один і той самий кортеж батьківського відношення. Це тип зв'язку між відношеннями «**один-до-багатьох**» (стандартний тип зв'язків зі збереженням цілісності посилань).
 2. Зовнішній ключ володіє властивістю унікальності - зв'язок між відношеннями має тип «**один-до-одного**».
-

Маніпулятивна частина реляційної бази даних

- ❑ Описує засоби, за допомогою яких:
 - 1) з даних, що зберігаються в базі даних, можна отримати вибірки або зведені результати,
 - 2) змінювати самі дані або їх структуру.
 - ❑ В сучасних промислових СУБД це мова запитів, зокрема SQL. Усі реляційні СУБД реалізують той чи інший діалект SQL.
 - ❑ Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений певною сукупністю операторів мови SQL.
-

Реляційна повнота мови SQL

Relational Query Language					
Relational Algebra			Relational Calculus		
	Operator	Symbol	Operator	Symbol	Example
Fundamental Operators	SELECT	σ	Tuple Relational Calculus (TRC)	$\{t \mid \text{condition}(t)\}$	$\{t \mid \text{EMPLOYEE}(t) \text{ AND } t.DEPT_ID = 10\}$
					$\{ \langle EMP_ID, EMP_NAME \rangle \mid \langle EMP_ID, EMP_NAME \rangle \in \text{EMPLOYEE A } DEPT_ID = 10 \}$
	PROJECT	Π	Domain Relational Calculus (DRC)	$\{ \langle a1, a2, a3, \dots, an \rangle \mid P(a1, a2, a3, \dots, an) \}$	$\{ \langle EMP_ID, EMP_NAME \rangle \mid \langle EMP_ID, EMP_NAME \rangle \in \text{EMPLOYEE A } DEPT_ID = 10 \}$
	RENAME	ρ			
	UNION	\cup			
	SET DIFFERENCE	$-$			
	CARTESIAN PRODUCT	\times			
Non-Fundamental Operators	SET INTERSECTION	\cap			
	ASSIGNMENT	\leftarrow			
	Natural JOINS	\bowtie			
	Left Outer Join				
	Right Outer Join				
	Full Outer Join				
	Division	\div			

SQL – структурована мова запитів

SQL дає можливість вирішувати наступні задачі:

- ☐ створення БД і визначення її структури
 - ☐ виконання запитів до БД
 - ☐ керування безпекою даних
-

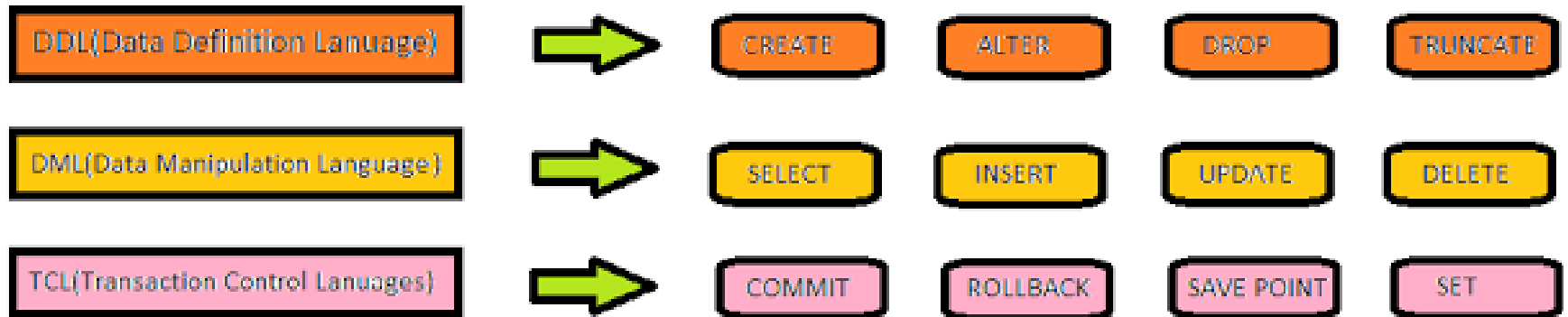
Інтерактивна та вбудована SQL

- ❑ **Інтерактивна** SQL використовується для роботи безпосередньо в СУБД, щоб опрацьовувати базу даних. При введенні команди в інтерактивній формі SQL, вона тут же виконається і виведуться результати.
 - ❑ **Вбудована** SQL складається з команд та процедур SQL, які розміщені в програмах, написаних іншими мовами програмування. Це робить такі програми більш потужними та більш ефективними.
-

Субпідрозділи (частини) SQL

- DDL (Data Definition Language) – мова визначення даних
 - DML (Data Manipulation Language) – мова маніпулювання даними
 - DQL (Data Query Language) – мова запитів
 - DCL (Data Control Language) – мова керування (адміністрування) даними
 - TCL (Transaction Control Language) – мова керування транзакціями (останні стандарти)
-

Субпідрозділи (частини) SQL



SQL Statements

Стандарти SQL

- ❑ SQL-86 або SQL-87 – перша публікація стандарту ANSI/ISO
 - ❑ SQL-89 або SQL1 - внесення невеликих змін до попереднього стандарту
 - ❑ SQL-92 або SQL2 – суттєва ревізія попередніх стандартів; донині є найбільш використовуваним
-

Стандарти SQL

- SQL-99 або SQL3 – доповнення до SQL2, зокрема такі :
 - поворот з орієнтацією на об'єкти
 - введено нові правила контролю цілісності даних
 - введення нових типів даних, в тому числі складних структурованих типів даних, які більше відповідають об'єктній орієнтації
 - додано розділ, який вводить стандарти на події та тригери
-

Стандарти SQL

- SQL:2003 доповнення до SQL3, зокрема такі :
 - додані нові типи даних
 - додано розділ стосовно використання SQL у мові програмування Java
 - додано розділ стосовно підтримки XML та роботи з XML-даними
-

Стандарти SQL

- SQL:2008 (спочатку був відомий як SQL:2006)
 - усунуто деякі неоднозначності, які були в стандарті SQL:2003
 - розширено функціональність роботи з XML-даними
 - додано нові тригери, зокрема INSTEAD OF

 - SQL:2011 - повний його текст можна придбати в організації ISO (ISO/IEC 9075:2011)
-

Загальні типи даних в SQL:2003

1. Символи
 2. Числа
 3. Логічні дані
 4. Бінарні дані
 5. Дата і час
 6. Інтервали
-

Загальні типи даних в SQL:2003

Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19
DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC(p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)
FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision
REAL	Approximate numerical, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

Символьні типи даних в SQL

- ❑ CHARACTER(n) (або **CHAR(n)** в реальних СУБД) – символьний рядок фіксованої довжини з n символів ($0 < n < 256$). Якщо n не вказане, то припускається, що рядок складається з одного символу. Якщо у стовпець такого типу вводиться $m < n$ символів, то решта позицій заповнюються пропусками.
-

Символьні типи даних в SQL

- ❑ CHARACTER VARYING(n) (або **VARCHAR(n)** в реальних СУБД) – символний рядок змінної довжини, яка не перевищує n символів. Застосовується, коли дані мають різну довжину і не бажано доповнювати їх пропусками. В даному випадку є **обов'язковим** вказання максимальної кількості символів, на відміну від CHAR.
-

Символьні типи даних в SQL

- ❑ Тип даних NATIONAL CHARACTER(n) (або **NCHAR(n)** в MS SQL Server, Oracle) – символний рядок фіксованої довжини для вибраної мови.
 - ❑ Тип даних NATIONAL CHARACTER VARYING(n) (або **NVARCHAR(n)** в MS SQL Server, Oracle) – символний рядок змінної довжини для вибраної мови
-

Символьні типи даних в SQL.

CHAR/VARCHAR та NCHAR/NVARCHAR

- ❑ CHAR та VARCHAR використовують 1-байтний варіант зберігання символів, який базується на ASCII
 - ❑ NCHAR та NVARCHAR підтримують 2-байтний набір символів Unicode
 - ❑ Не залежно від вибраного набору символів, SQL-код залишається тим самим, якщо СУБД функціонує однаково для ASCII та Unicode
 - ❑ Дані наведених типів є сумісними з точки зору участі в одних і тих же символьних операціях.
-

Символьні типи даних в SQL

- CHARACTER LARGE OBJECT або CLOB – великий символьний об'єкт. Використовується для представлення дуже великих символьних рядків (наприклад, статей, книжок і т.п.). Стовпці такого типу не можуть бути первинними або зовнішніми ключами, а також об'являться як ті, що мають унікальні значення.
 - MEMO в MS Access
 - TEXT в MySQL
 - **NTEXT, TEXT** до MS SQL Server 2008. Для наступних версій слід уникати їх використання при розробці нових програм. Замість цих типів даних потрібно використовувати типи **NVARCHAR (MAX), VARCHAR (MAX) і VARBINARY (MAX)**.
-

Точні цілі числові типи даних в SQL

- ❑ **INTEGER** (або **number (long integer)** в Access, **INT** в інших СУБД) – чотирьохбайтне ціле число (до $\pm 2\,147\,483\,467$);
 - ❑ **SMALLINT** (або **number (integer)** в Access) – мале (двобайтне) ціле число (до $\pm 32\,767$);
 - ❑ **BIGINT** – велике (восьмибайтне) ціле число (до $\pm 2^{63}$);
-

Точні дійсні числові типи даних в SQL

- ❑ `NUMERIC(x,y)` – дійсне число з плаваючою комою, у якому всього `x` розрядів, з яких `y` відведено для дробової частини.
 - ❑ `DECIMAL(x,y)` – десяткове число, якому всього `x` розрядів, з яких `y` відведено для дробової частини. Дозволяє вказати максимальне число знаків і число знаків після коми. Вимагає 5-17 байтів пам'яті
-

Точні дійсні числові типи даних в SQL

Різниця NUMERIC і DECIMAL

Якщо вказано тип `NUMERIC(6,2)`, то максимальне значення числа рівне 9999.99.

Якщо в типі `DECIMAL` вказані x та y є меншими, ніж допустимі реалізацією SQL, то будуть використовуватись системні.

Наприклад, якщо вказано тип `DECIMAL(6,2)` і дозволяє реалізація SQL, то в цей стовпець можна ввести числа більші 9999.99. Цифри зліва від десяткової коми завжди будуть правильними, але тип `DECIMAL` заокруглює цифри справа від десяткової коми, якщо для їх відображення не вистачає місця.

Отже, тип `NUMERIC` жорстко задає діапазон можливих значень.

А тип `DECIMAL` є гнучким і використовується, коли є можливість вказати, скільки знаків потрібно зберегти.

Приблизні числові типи даних в SQL

- ❑ REAL – дійсне 4-байтне число одинарної точності з плаваючою крапкою, точність залежить від реалізації SQL.
 - ❑ DOUBLE PRECISION – дійсне число подвійної точності з плаваючою крапкою. Точність залежить від реалізації SQL. Застосовується для представлення наукових даних, наприклад, дуже близьких до нуля або дуже великих.
 - ❑ FLOAT(n) – дійсне число з плаваючою крапкою і мінімальною точністю, яке займає 8 байтів, де n - кількість бітів, які використовуються для зберігання мантиси числа. Система сама вибирає точність: одинарну чи подвійну. Даний тип використовується при можливості перенесення бази даних на іншу апаратну платформу, яка відрізняється розмірами регістрів.
-

Логічні дані в SQL

- ❑ BOOLEAN має три значення – true, false та unknown (за стандартом)
 - ❑ Значення unknown (невідоме) введено для позначення результату, який отримується при порівнянні зі значенням NULL. В реальних СУБД не підтримується.
 - ❑ Результатом будь-якої операції порівняння true або false зі значенням NULL завжди є unknown. В реальних СУБД таким результатом є NULL.
 - ❑ В SQL-виразах логічні значення поміщаються в лапки, наприклад, 'true'.
-

Дата і час в SQL

- ❑ DATE - представлення значень календарної дати. Дані цього типу можуть містити будь-яку дату з 0001 року по 9999 рік.
 - ❑ TIME - представлення часу
 - ❑ TIMESTAMP - одночасне представлення дати та часу
-

Інтервали в SQL

- ❑ Інтервал – це різниця між двома значеннями типу дата-час.
 - ❑ SQL підтримує два типи інтервалів, які не можна змішувати в обчисленнях :
 - рік-місяць – кількість років і місяців між двома датами
 - день-час – кількість днів, годин, хвилин і секунд між двома моментами в межах одного місяця.
-

Приклади інтервалів в SQL

- Задання значення типу інтервал (тривалість):
`INTERVAL 'довжина' YEAR | MONTH | DAY | HOUR |
MINUTE | SECOND`

де довжина – тривалість інтервалу, після чого вказується одиниця виміру. Наприклад, для задання інтервалу тривалістю 5 днів використовується вираз: `INTERVAL '5' DAY`.

- Інтервал часу можна задати двома способами:
 - у вигляді початкового та кінцевого моментів, наприклад, `(TIME '12:25:30', TIME '14:30:00')`;
 - у вигляді початкового моменту та тривалості, наприклад, `(TIME '12:45:00', INTERVAL '5' HOUR)`.
-

Відповідні типи в SQL

- ❑ Символьні типи CHARACTER , CHARACTER VARYING, відповідні NATIONAL
- ❑ Усі числові типи
- ❑ Дата, час, дата-час, відповідні інтервали

Відповідні типи не обов'язково перетворювати один до іншого

Перетворення типів в SQL

□ Функція: `CAST(вираз AS тип);`

Наприклад

- `CAST('1234.56' AS NUMERIC(9,2));`
 - `CAST('07-09-2011' AS DATE);`
 - `CAST(CURRENT_TIMESTAMP(2) AS CHAR(20);`
-

Загальні типи даних в СУБД

Data type	Access	SQLServer	Oracle	MySQL	PostgreSQL
<i>boolean</i>	Yes/No	Bit	Byte	N/A	Boolean
<i>integer</i>	Number (integer)	Int	Number	Int Integer	Int Integer
<i>float</i>	Number (single)	Float Real	Number	Float	Numeric
<i>currency</i>	Currency	Money	N/A	N/A	Money
<i>string (fixed)</i>	N/A	Char	Char	Char	Char
<i>string (variable)</i>	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
<i>binary object</i>	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

SQL-операції

- ❑ **Арифметичні операції** в порядку спадання пріоритетів: (); *, /; +, -;
 - ❑ Символьні операції: знаком операції **конкатенації (зчіплення)** символічних значень є || (у Oracle та DB2), або + (у MS SQL Server та MS Access), або функція CONCAT() (у MySQL, Oracle та DB2). Результатом конкатенації є символічне значення з максимальною довжиною 255 символів.
-

SQL-операції

- **Операції порівняння:** =, >, <, >=, <=, <>; результат порівняння може бути або 'TRUE', або 'FALSE'.
 - **Логічні операції** в порядку спадання пріоритетів:
 - () – змінює нормальні правила пріоритетів;
 - NOT – інвертує результат логічного виразу;
 - AND – результат повинен відповідати обом умовам;
 - OR – результат повинен відповідати одній із умов.
-

Дякую за увагу
