Лекція 5. Транзакції. Тригери. Індекси

5.1. Введення в транзакції

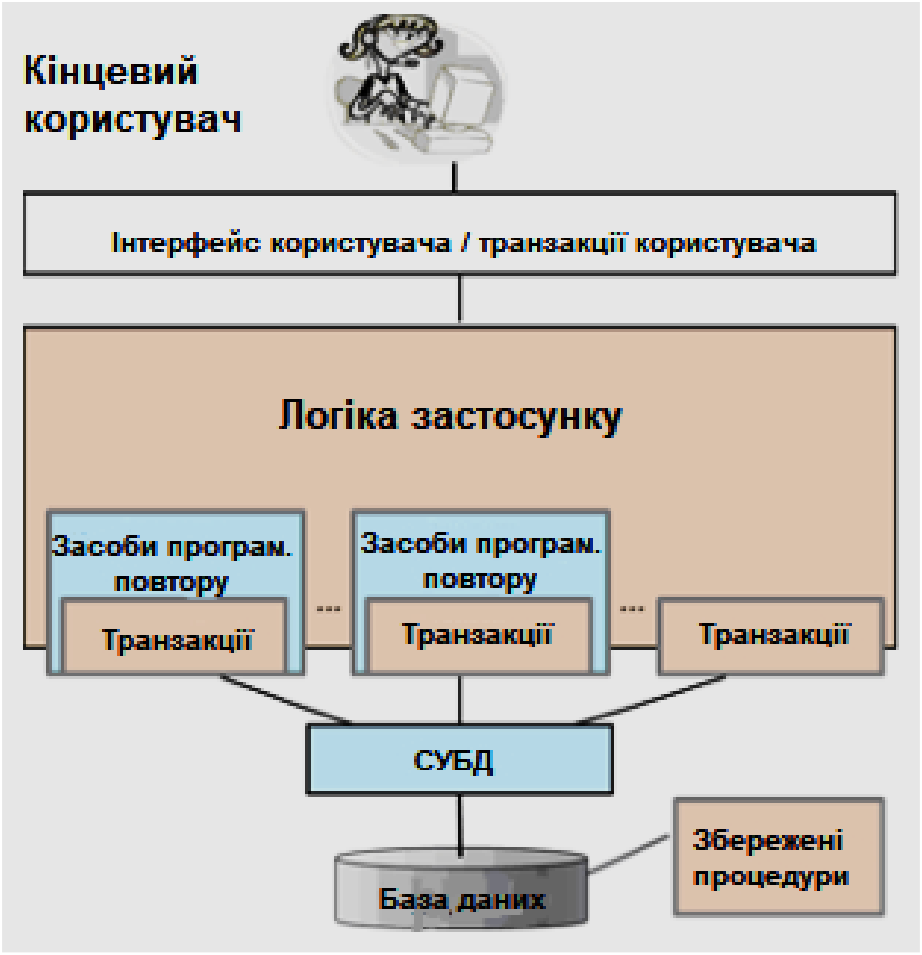
Транзакція - група послідовних операцій з базою даних, яка являє собою логічну одиницю роботи з даними.

Існує два типи транзакцій:

1.Неявние - кожен оператор, такий як INSERT, UPDATE або DELETE, виконується в транзакції.

2. Явні - група інструкцій мови Transact -SQL, початок і кінець якої позначаються такими інструкціями, як BEGIN TRANSACTION, COMMIT і ROLLBACK.

Розташування транзакцій SQL в протоколах прикладного рівня



Принцип ACID (принцип ідеальної транзакції)

Неподільність (Atomic) Транзакція має бути неподільною послідовністю операцій («все або нічого»), які або успішно виконуються до кінця і фіксуються, або виконується відкат всіх цих операцій.

Узгодженість (Consistent) Послідовність операцій буде передавати вміст бази даних з одного узгодженого стану в інше.

Ізольованість (Isolated) Звучить як «Події однієї транзакції повинні бути приховані від інших транзакцій, що виконуються одночасно».

Довговічність (Durable) Зафіксовані результати в базі даних будуть доступні на дисках, не дивлячись на можливі збої системи.

Давайте розглянемо наступну таблицю банківських рахунків:

```
CREATE TABLE Accounts (
acctId INTEGER NOT NULL PRIMARY KEY,
balance DECIMAL (11,2) CHECK ( balance > = 0.00)
);
```

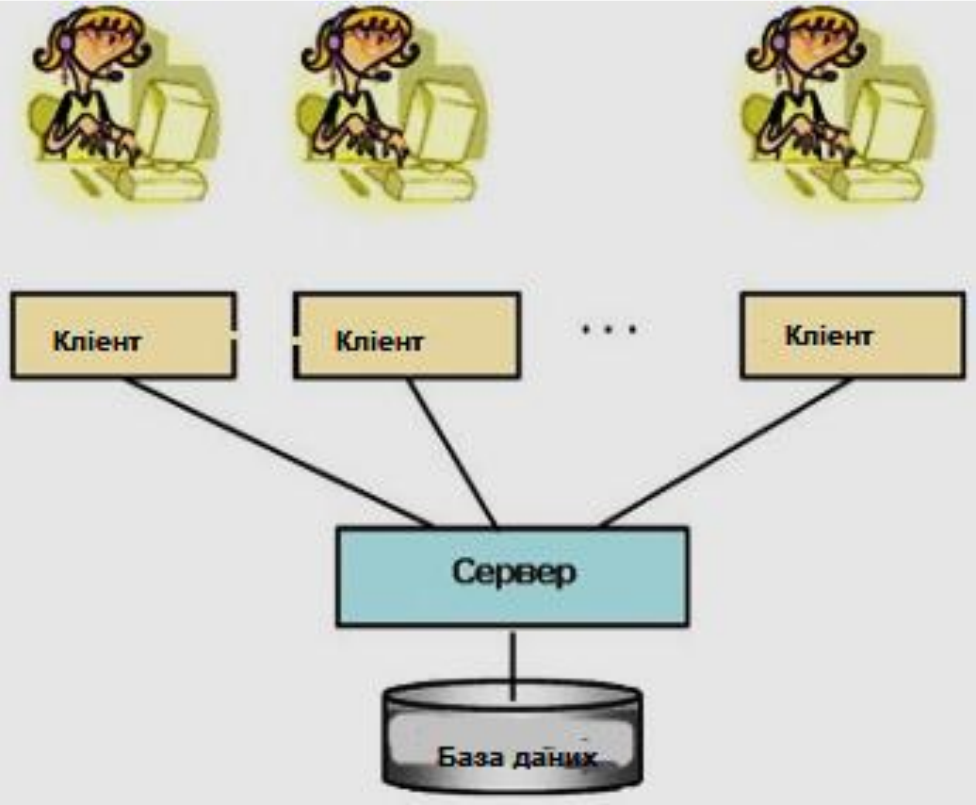
Типовим прикладом транзакцій SQL є переказ визначеної суми (наприклад, 100 євро) з одного рахунку на інший:

```
BEGIN TRANSACTION;
UPDATE Accounts SET balance = balance - 100
WHERE acctId = 101;

UPDATE Accounts SET balance = balance + 100
WHERE acctId = 202;
COMMIT;
```

Стандарт ISO SQL-89 визначив команду SQLCODE у вигляді відображення цілого числа, значення якого «0» в кінці кожної команди SQL вказує на її успішне виконання, в той час як значення 100 вказує на те, що відповідні рядки не були знайдені .

Будь-які позитивні значення вказують на попередження, а негативні значення вказують на різні помилки, на які дані пояснення у відповідних довідкових посібниках продукту.



Доступ до однієї бази даних декількох клієнтів в багатокористувацькому середовищі

Розглянемо типові проблеми (аномалії) паралелізму:

• Проблема втраченого поновлення;

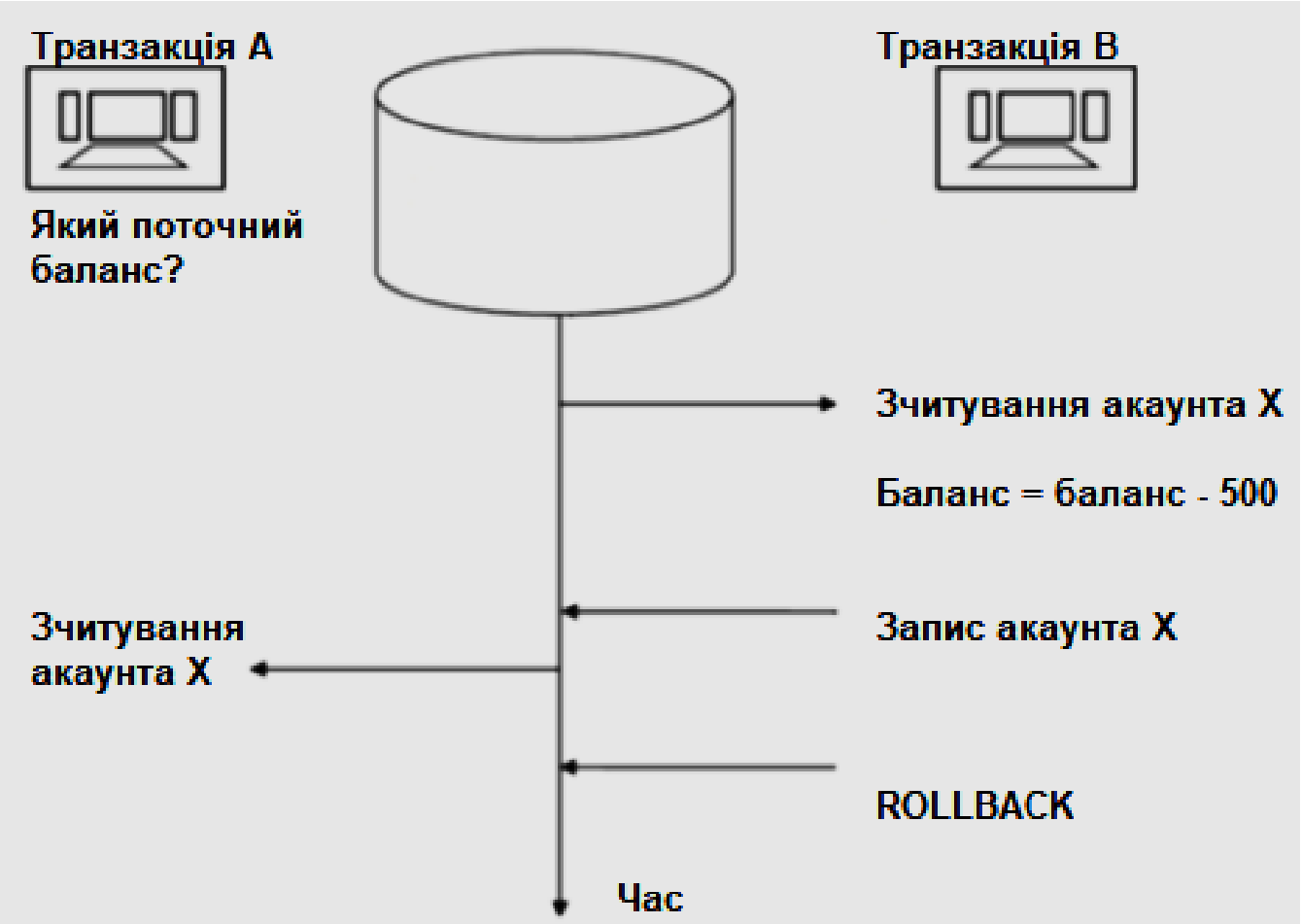
• Проблема зчитування «брудних» даних, тобто читання незафіксованих даних деяких паралельних транзакцій;

• Проблема неповторюваного читання, тобто повторюється вважаючи вання не повертає ті ж рядки і / або їх значення;

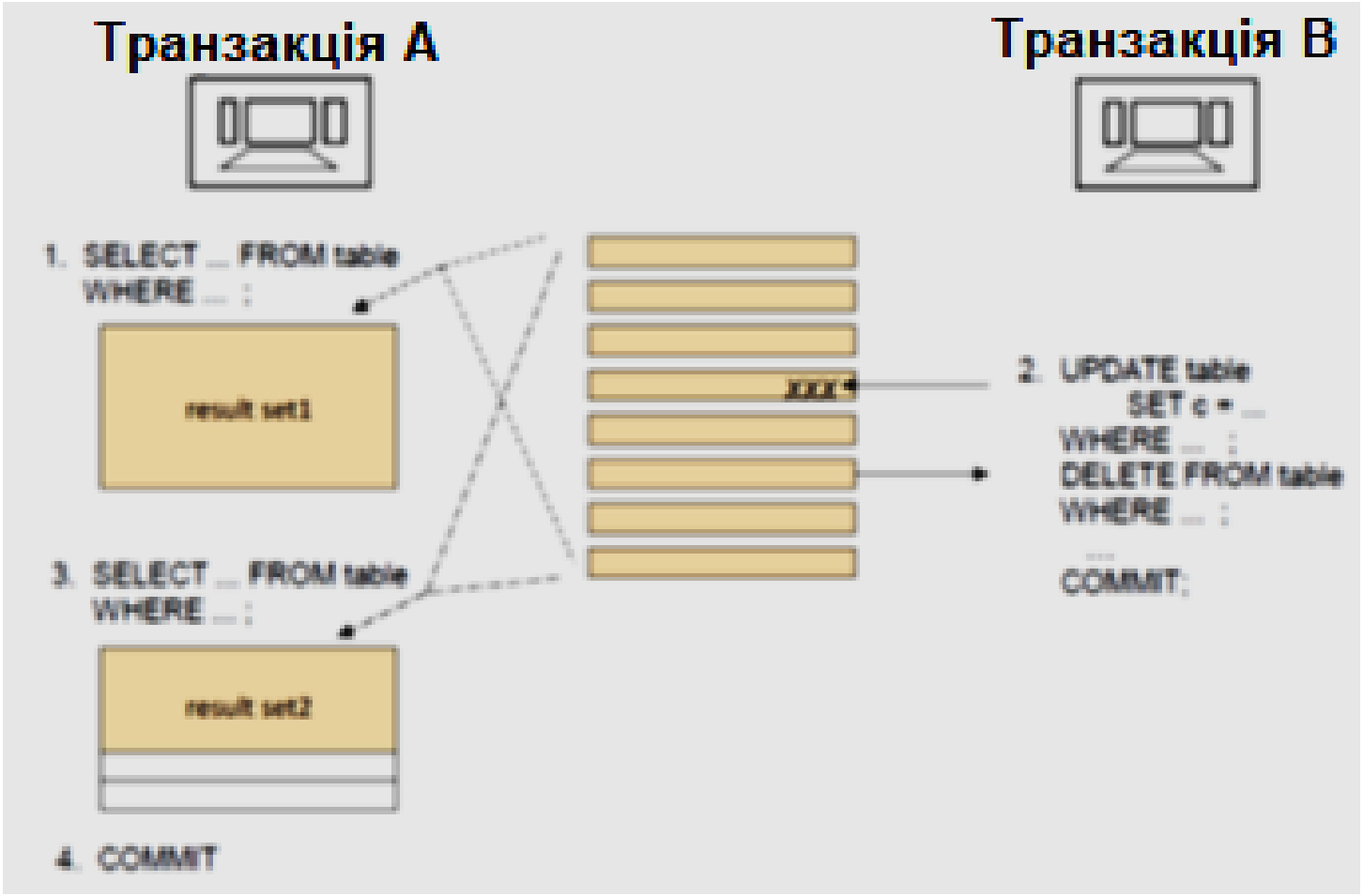
• Проблема фантомного читання, тобто під час транзакції деякі вибрані рядки можуть бути не видно транзакцією.

Проблема втраченого поновлення

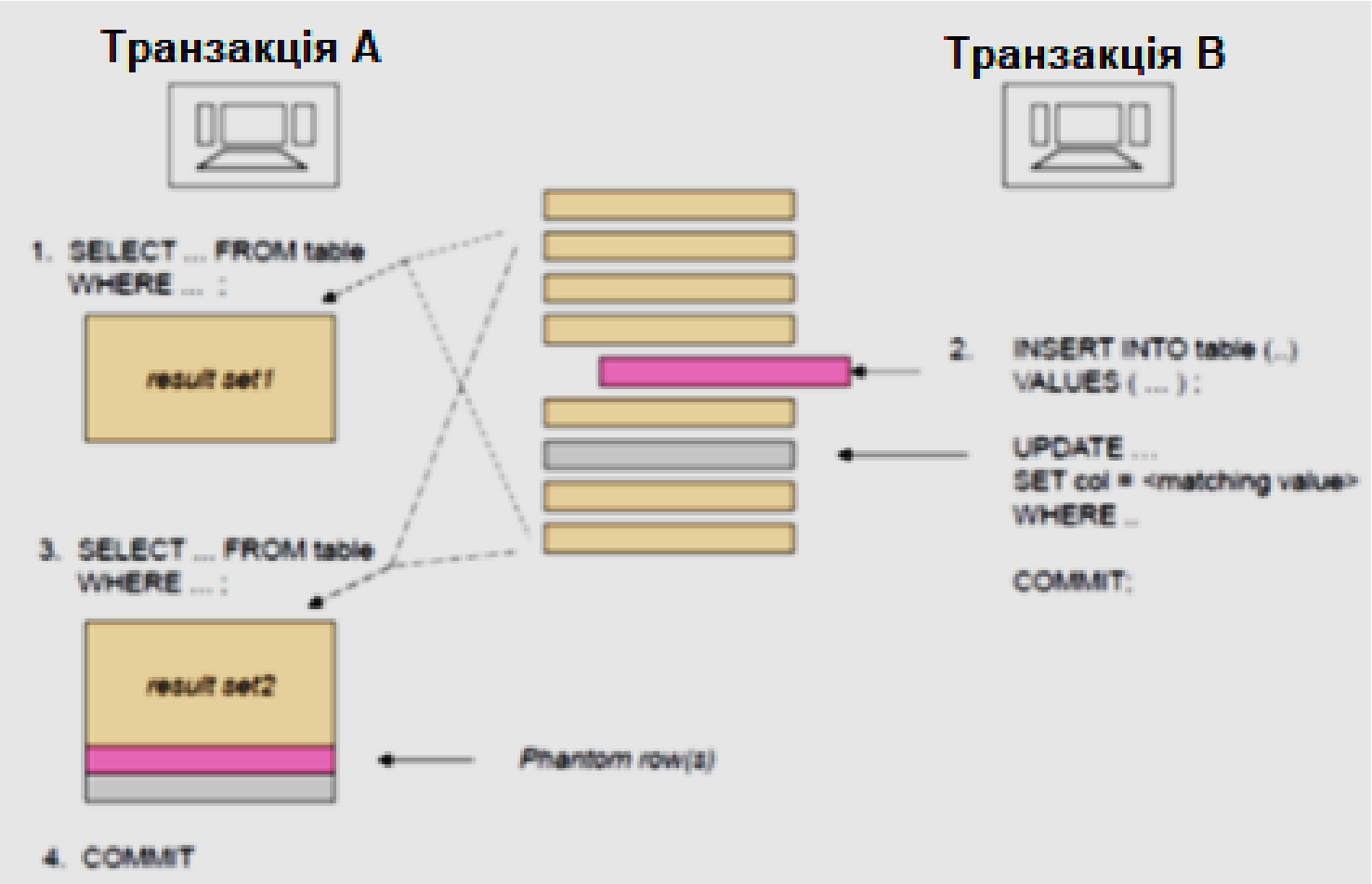
Проблема зчитування «брудних» даних



Проблема неповторюваного читання



Проблема фантомного читання



Рівні ізольованості

Проблеми / Рівні ізольованості	Утрачене поновлення	"Брудне" читання	Неповторюване читання	Фантомне читання
READ UNCOMMITTED	Запобігає	Не запобігає	Не запобігає	Не запобігає
READ COMMITTED	Запобігає	Запобігає	Не запобігає	Не запобігає
REPEATABLE READ	Запобігає	Запобігає	Запобігає	Не запобігає
SERIALIZABLE	Запобігає	Запобігає	Запобігає	Запобігає

5.2. Тригери

Тригер - це особливий різновид збереженої процедури, яка виконується автоматично при виникненні події на сервері бази даних.

Існують тригери на події DDL і DML.

Тригери DDL активуються у відповідь на різні події мови опису даних. Ці події, перш за все, відповідають інструкціям Transact - SQL CREATE, ALTER, DROP і деяким системним збереженим процедурам, які виконують схожі з DDL операції.

Тригери DML виконуються, коли користувач намагається змінити дані за допомогою подій мови обробки даних (DML). Подіями DML є процедури INSERT, UPDATE або DELETE, що застосовуються до таблиці або подання.

- Тригери DML дозволяють каскадно проводити зміни через пов'язані таблиці в базі даних; але ці зміни можуть здійснюватися більш ефективно з використанням каскадних обмежень посилальної цілісності.
- Для запобігання випадкових або невірних операцій INSERT, UPDATE і DELETE і реалізації інших більш складних обмежень, ніж ті, які визначені за допомогою обмеження CHECK.
- На відміну від обмежень CHECK, DML-тригери можуть посилатися на стовпці інших таблиць.
- Кілька DML-тригерів однакового типу (INSERT, UPDATE або DELETE) для таблиці дозволяють зробити кілька різних дій у відповідь на одну інструкцію зміни даних.
- Обмеження можуть повідомляти про помилки тільки за допомогою відповідних стандартних системних повідомлень.
- При використанні тригерів DML може статися відкат змін, що порушують кількість посилань цілісність, що приводить до заборони модифікації даних.
- Якщо в таблиці тригерів існують обмеження, то їх перевірка здійснюється між виконанням тригерів INSTEAD OF і AFTER. У разі порушення обмежень виконується відкат дій тригерів INSTEAD OF, а тригер AFTER не спрацьовує.

Типи тригерів

Тригери AFTER або FOR. Тригери AFTER виконуються після виконання дій інструкції INSERT, UPDATE, MERGE або DELETE. Тригери AFTER ніколи не виконуються, якщо відбувається порушення обмеження, тому ці тригери не можна використовувати для будь-якої обробки, яка могла б запобігти порушенню обмеження.

Тригер INSTEAD OF - тригери INSTEAD OF скасовують стандартні дії інструкції, що викликає тригер. Тому вони можуть використовуватися для перевірки на наявність помилок або перевірки значень на одному або декількох стовпцях і виконання додаткових дій, перш ніж вставити, оновленням або видаленням однієї або декількох рядків.

Інструкції тригерів DML використовують дві особливі таблиці: DELETED і INSERTED. SQL Server автоматично створює ці таблиці і керує ними. Ці тимчасові таблиці, що знаходяться в оперативній пам'яті, використовуються для перевірки результатів змін даних і для установки умов спрацьовування тригерів DML.

У таблиці DELETED знаходяться копії рядків, з якими працювали інструкції DELETE або UPDATE. При виконанні інструкції DELETE або UPDATE відбувається видалення рядків з таблиці тригера і їх перенесення в таблицю deleted. У таблиці DELETED зазвичай немає загальних рядків до таблиці тригера.

У таблиці INSERTED знаходяться копії рядків, з якими працювали інструкції INSERT або UPDATE. При виконанні транзакції вставки або поновлення відбувається одночасне додавання рядків в таблицю тригера і в таблицю inserted . Рядки таблиці INSERTED є копіями нових рядків таблиці тригера.

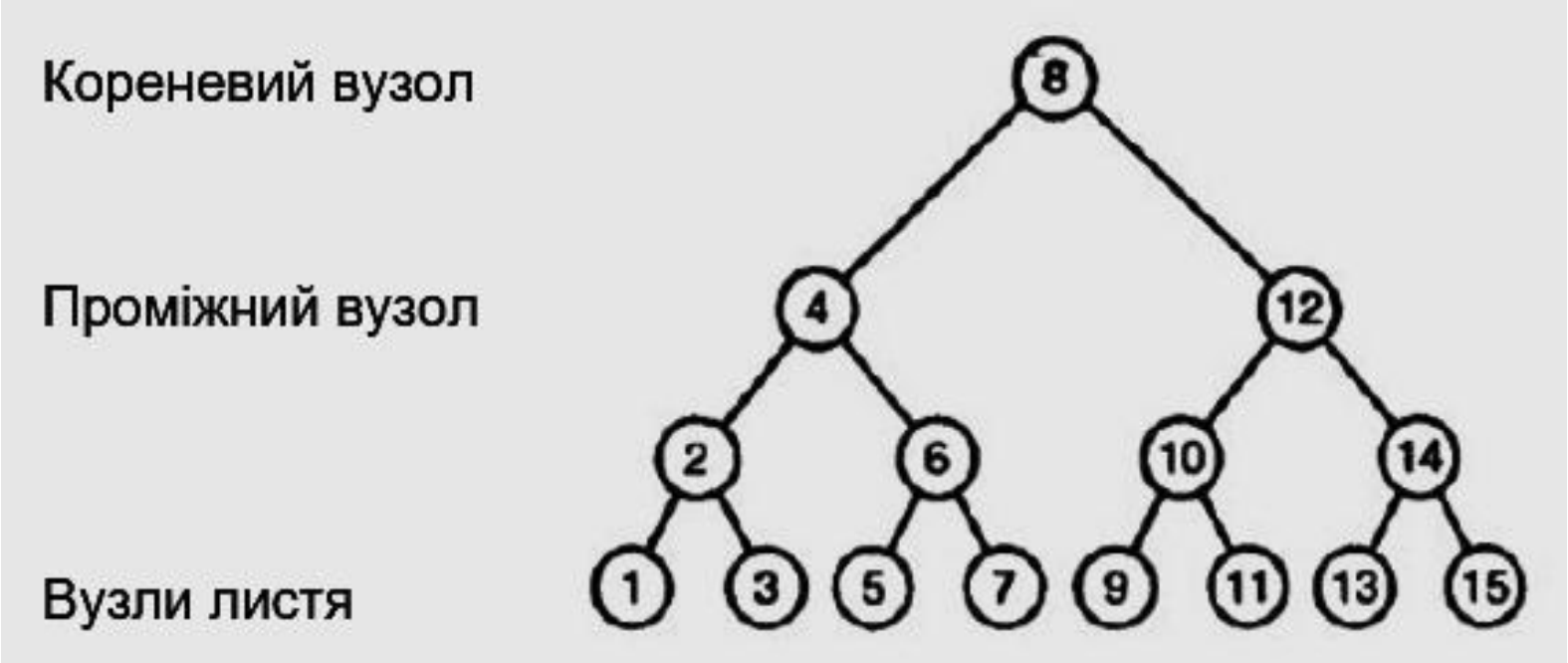
Синтаксис створення тригера

```
CREATE [ALTER] TRIGGER [ schema_name ] trigger_name
ON {table | view}
[WITH < dml_trigger_option > [... n]]
{FOR | AFTER | INSTEAD OF}
{[INSERT] [,] [UPDATE] [,] [DELETE]}
[WITH APPEND]
[NOT FOR REPLICATION]
AS { sql_statement   [; ] [, ... n] | EXTERNAL NAME <method
specifier [; ]>}
 < Dml_trigger_option >: : =
   [ENCRYPTION ]
   [EXECUTE AS Clause]

< Method_specifier >: : =
   assembly_name.class_name.method_name
```

5.3. Індекси

Індекс являє собою окрему фізичну структуру даних, яка дозволяє отримувати швидкий доступ до однієї або кількох рядках даних.



Існує два типи індексів: **кластерізовані** та **некластерізовані**.

Кластерізовані індекси.

Кластерізований індекс визначає фізичний порядок даних в таблиці. Компонент Database Engine дозволяє створювати для таблиці лише один кластерізований індекс, тому що рядки таблиці можна впорядкувати фізично більш ніж одним способом.

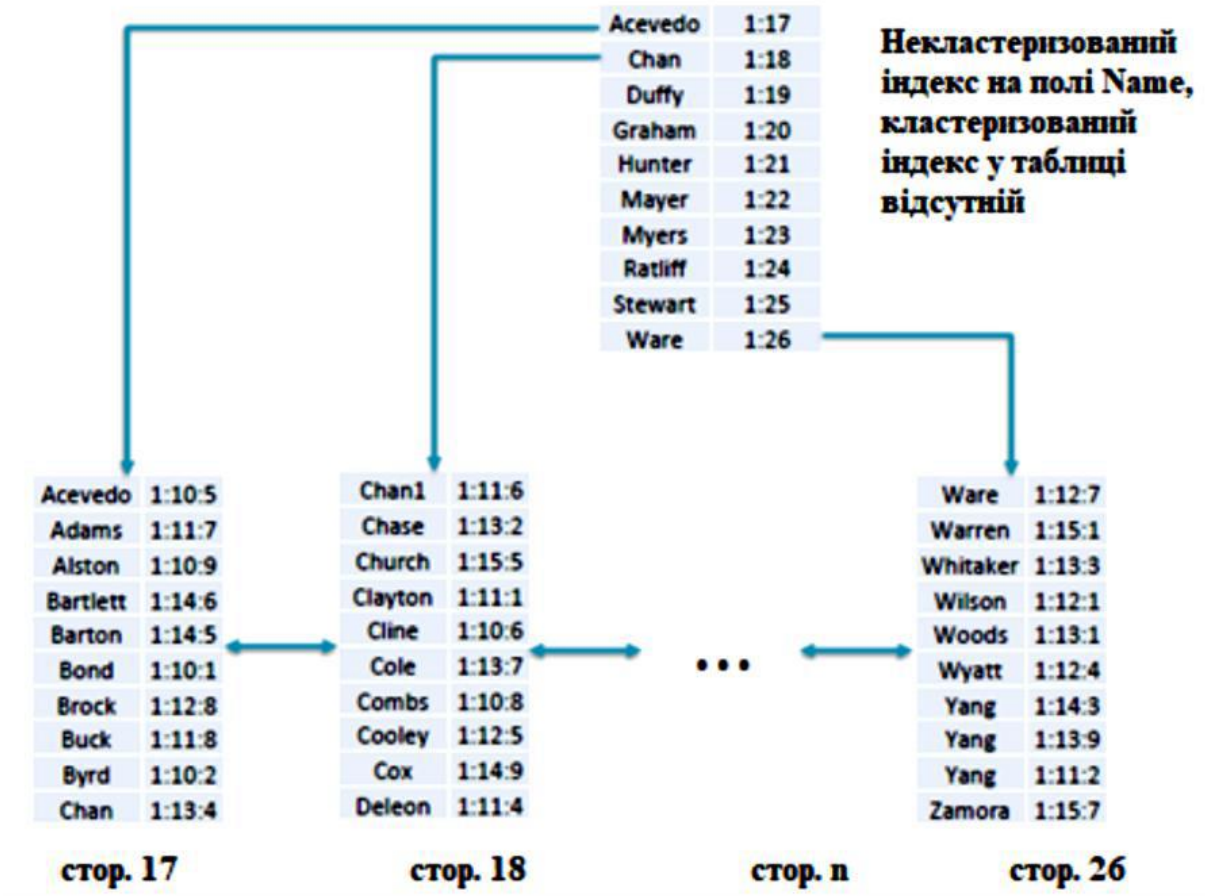
Фізична структура кластеризованих індексів

Некластерізовані індекси.

Структура некластерізованого індексу точно така ж, як і кластерізованного , але з двома важливими відмінностями:

1. некластерізованний індекс не змінює фізичне упорядкування рядків таблиці;
2. сторінки вузлів некластерізованний індексу складаються з ключів індексу і закладок.

Якщо для таблиці визначити один або більше некластерізованних індексів, фізичний порядок рядків цієї таблиці не буде змінений.



Структура некластерізованого індексу

Створення індексів

Індекс для таблиці створюється за допомогою інструкції CREATE INDEX. Ця інструкція має наступний синтаксис :

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX index_name
   ON table_name (column1 [ASC | DESC ], ...)
      [INCLUDE ( column_name [...])]
 [WITH
   [FILLFACTOR = n]
   [[ ,] PAD_INDEX = {ON | OFF}]
   [[ ,] DROP_EXISTING = {ON | OFF}]
   [[ ,] SORT_IN_TEMPDB = {ON | OFF}]
   [[ ,] IGNORE_DUP_KEY = {ON | OFF}]
   [[ ,] ALLOW_ROW_LOCKS = {ON | OFF}]
   [[ ,] ALLOW_PAGE_LOCKS = {ON | OFF}]
   [[ ,] STATISTICS_NORECOMPUTE = {ON | OFF}]
   [[ ,] ONLINE = {ON | OFF}]]
   [ON file_group | "default"]
```

1. PRIMARY KEY за замовчуванням створює унікальний кластерний індекс, але можна створити первинний ключ, який задасть некластерний індекс.

```
CREATE TABLE Persons
(Id INTEGER PRIMARY KEY,
Name VARCHAR ( 255)
);
```

Якщо у таблиці вже є кластерний індекс, то ми можемо створити некластерний індекс.

```
CREATE TABLE Persons
(Id INTEGER PRIMARY KEY NONCLUSTERED,
Name VARCHAR ( 255)
);
```

2. Обмеження UNIQUE по замовчуванням створює унікальний некластерний індекс

```
CREATE TABLE Persons
(Id INTEGER,
Name VARCHAR ( 255) UNIQUE
);
```

але можна створити UNIQUE, який задасть кластерний індекс.

```
CREATE TABLE Persons
(Id INTEGER,
Name VARCHAR (255) UNIQUE CLUSTERED
);
```

3. Створюючи індекси НЕ через PRIMARY KEY або UNIQUE, можна створити НЕ унікальний індекс (кластерний або некластерний).

За замовчуванням створюється НЕ унікальний некластерние індекс:

```
CREATE INDEX index _ name
ON table _ name ( column _ name )
```

Унікальний некластерние індекс :

```
CREATE UNIQUE INDEX index_name
ON table_name ( column_name )
```

Неунікальний кластерний індекс

```
CREATE CLUSTERED INDEX index_name
ON table_name ( column_name )
```

Унікальний кластерний індекс

```
CREATE UNIQUE CLUSTERED INDEX index_name
ON table_name ( column_name )
```

Рекомендації щодо створення і використання індексів

- Для кластерного індексу найбільш відповідним стовбцем буде унікальний стовбець якый не підтримує NULL значення та котрий не буде оновлюватися. Ось чому первинний ключ часто використовується як кластерний індекс.
- Створюйте індекси на стовпці, по яким відбувається пошук, сортування, угруповання, з'єднання таблиць.
- Не створюйте індекси на полях, які рідко використовуються в запитах.
- Не створюйте індекси на полях, які містять кілька унікальних значень, наприклад, стовбець, що містить тільки значення чоловічої або жіночий підлогу. Чим більше дублікатів в стовпці, тим гірше працює індекс.
- Для невеликих таблиць пошук по індексу може зайняти більше часу, ніж просте сканування всіх рядків.
- Для таблиць які часто оновлюються використовуйте як можна менше індексів.