

СУЧАСНІ СУБД

Лекція №12

Тема:

ORACLE PL/SQL

Команди ітеративного розгалуження

Цикли в PL/SQL

Під командами ітеративного розгалуження необхідно розуміти такі керуючі структури PL/SQL, як цикли. Цикли призначені для багаторазового виконання певних фрагментів програмного коду. Також в Oracle-11g з'явилась команда **CONTINUE**, яка теж відноситься до команд ітеративного розгалуження. PL/SOL підтримує цикли трьох видів: прості (нескінченні), **FOR** і **WHILE**.

Три різновиди циклів потрібні для створення оптимального алгоритму розв'язанняожної конкретної задачі. У більшості випадків задача може бути вирішена за допомогою будь-якої з трьох циклічних конструкцій, але при невдалому виборі конструкції програмний код буде містити безліч зайвих рядків, що ускладнить розуміння і супровід написаних модулів.

Простий цикл починається з ключового слова **LOOP** і завершується командою **END LOOP**. Виконання циклу переривається при виконанні команди **EXIT**, **EXIT WHEN** або **RETURN** в тілі циклу (або при виникненні виключення).

Цикл FOR існує в двох формах: числовий і курсорний. Числова форма починається з ключового слова **FOR**, після якого задається початкове і кінцеве цілочисельні значення. Тіло циклу починається командою **LOOP** і завершується командою **END LOOP**. PL/SQL перебирає всі проміжні значення вказаного діапазону, після чого завершує цикл.

Цикл WHILE має багато спільного з простим циклом. Принципова відмінність полягає в тому, що умова завершення перевіряється перед виконанням чергової ітерації. Можливі ситуації, в яких тіло циклу не буде виконано жодного разу:

Цикли в PL/SQL. Приклади.

Простий цикл

```

BEGIN
  LOOP
    EXIT WHEN (l_current_year > end_year_in);
    display_total_sales(l_current_year);
    l_current_year := l_current_year + 1;
  END LOOP;
END
/

```

Цикл WHILE

```

BEGIN
  WHILE (l_current_year <= end_year_in)
  LOOP
    display_total_sales(l_current_year);
    l_current_year := l_current_year + 1;
  END LOOP;
END
/

```

Цикл FOR

```

BEGIN
  FOR l_current_year IN start_year_in .. end_year
  LOOP
    display_total_sales(l_current_year)
  END LOOP;
END
/

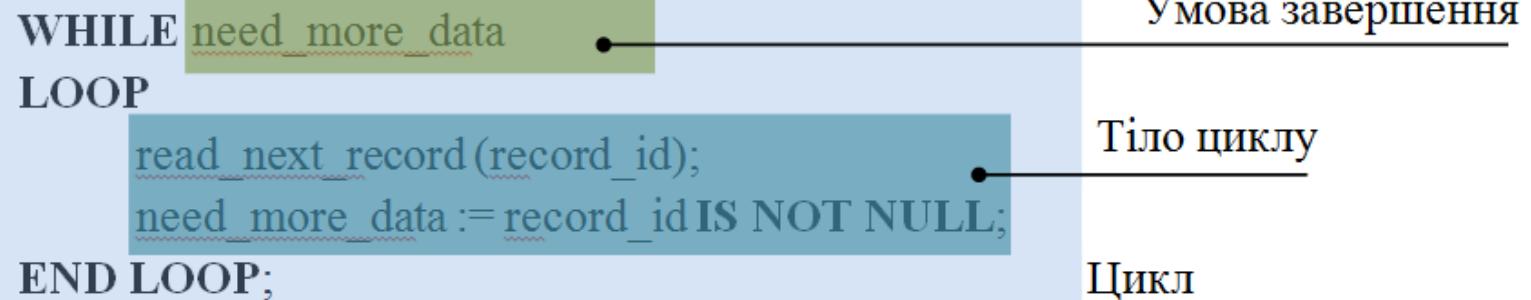
```

Структура циклів PL/SQL.

Незважаючи на відмінності між різними формами циклічних конструкцій, кожен цикл складається з двох частин: обмежувачів і тіла циклу.

Обмежувачі – ключові слова, що визначають початок циклу, умова завершення, і команда **END LOOP**, що завершує цикл.

Тіло циклу – послідовність виконуваних команд всередині границь циклу, що виконуються на кожній ітерації.



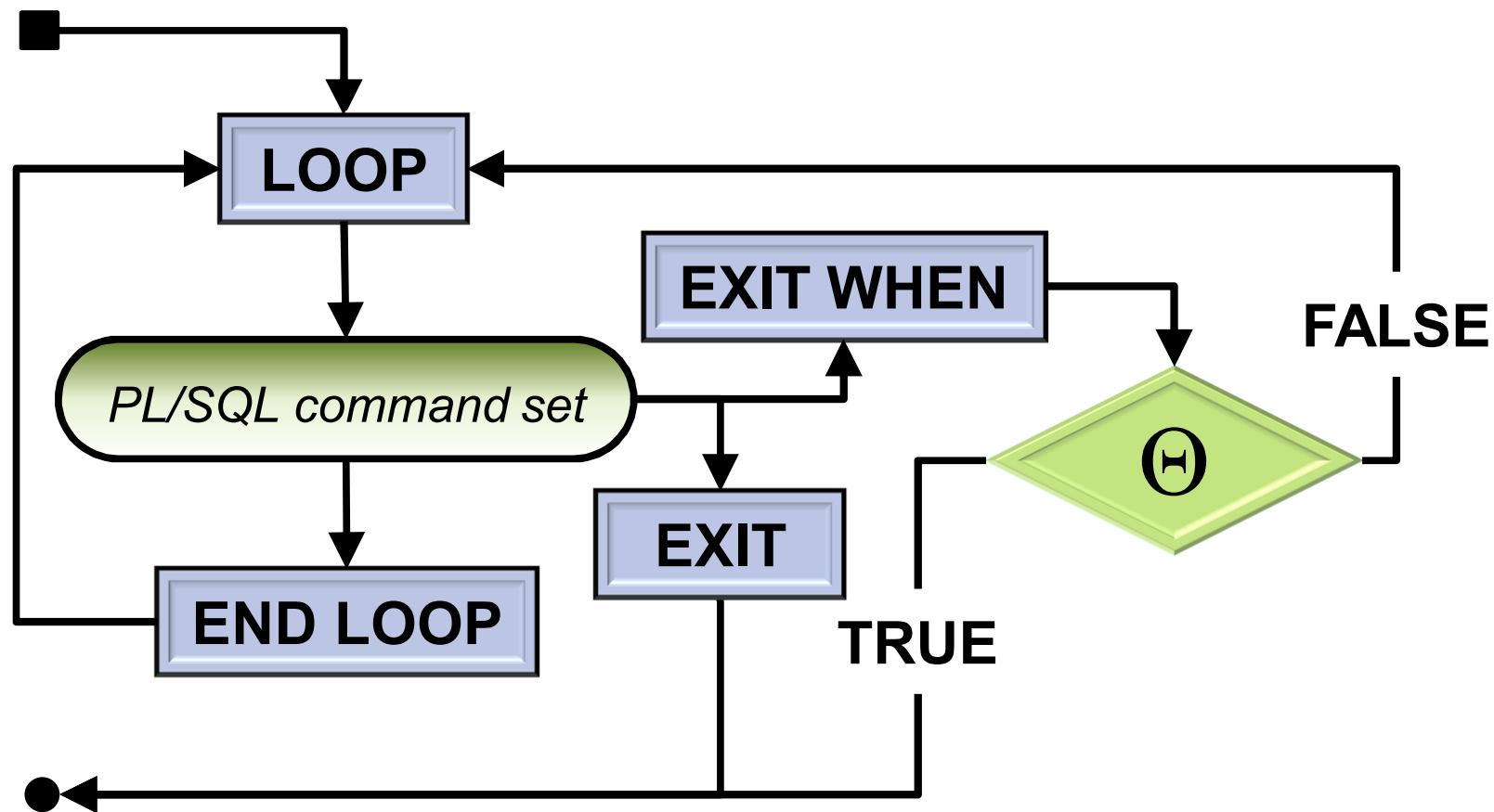
Простий цикл.

Структура простого циклу є найелементарнішій серед всіх циклічних конструкцій. Такий цикл складається з ключового слова **LOOP**, виконуваного коду (тіла циклу) і ключових слів **END LOOP**. Цикл починається командою **LOOP**, а закінчується командою **END LOOP**. Тіло циклу повинно містити як мінімум одну виконувану команду.

Властивості простого циклу

№	Властивість	Опис
1	Умова завершення циклу	Якщо в тілі циклу виконується команда EXIT . В іншому випадку цикл виконується нескінченно.
2	Позиція перевірки умови	У тілі циклу і тільки при виконанні команди EXIT або EXIT WHEN . Тіло циклу (або його частина) завжди виконується як мінімум один раз
3	Доцільність використання	<ol style="list-style-type: none"> 1) Якщо не відомо, скільки разів буде виконуватися тіло циклу; 2) Тіло циклу має бути виконане хоча б один раз

Простий цикл.



Простий цикл.Приклади.

LOOP

LOOP

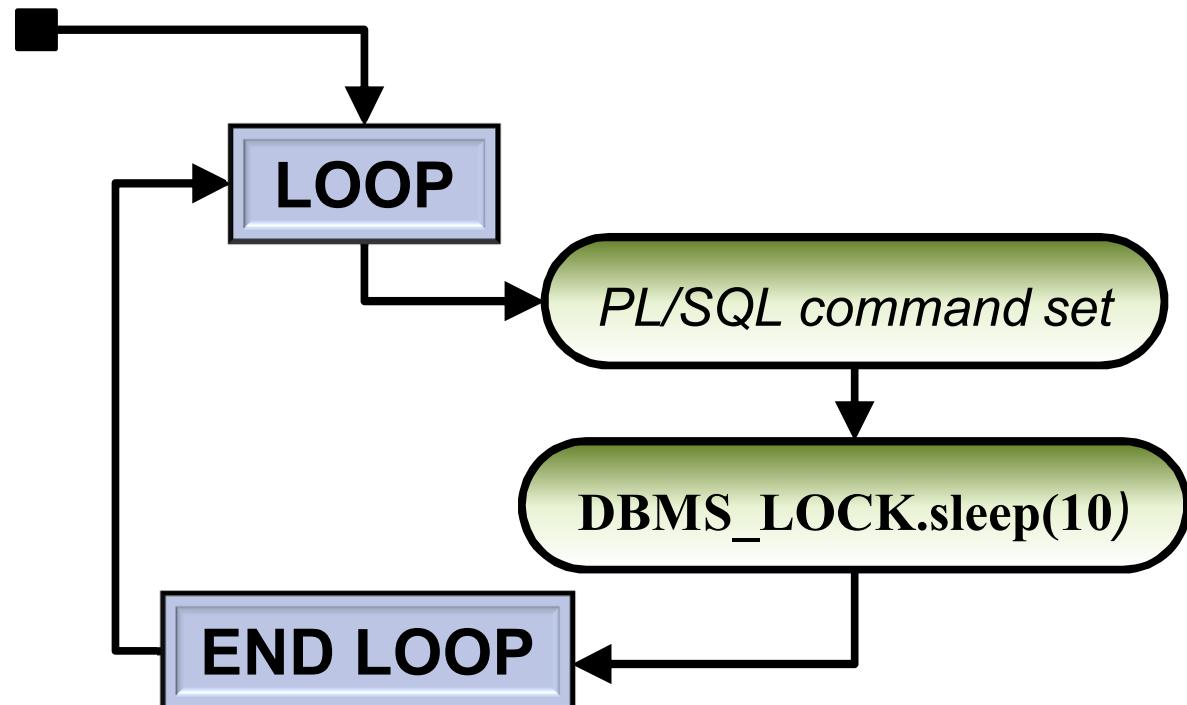
```
/* Обчислення балансу */
    balance_remaining := account_balance(account_id);
/* Умова вбудована в команду EXIT */
    EXIT WHEN balance_remaining < 1000;
/* Якщо цикл продовжує виконуватись з балансу списуються кошти */
    apply_balance(account_id, balance_remaining);
END LOOP;
```

```
balance_remaining := account_balance(account_id);
IF balance_remaining < 1000
THEN
    EXIT;
ELSE
    apply_balance(account_id, balance_remaining);
END IF;
END LOOP;
```

- Команду **EXIT WHEN** використовують, коли умова завершення циклу визначається лише одним **Θ**-виразом.
- Команду **EXIT** використовують, коли має місце кілька умов завершення циклу або якщо при виході має бути визначено значення, що повертається. Разом з **EXIT** використовують **IF** або **CASE**.

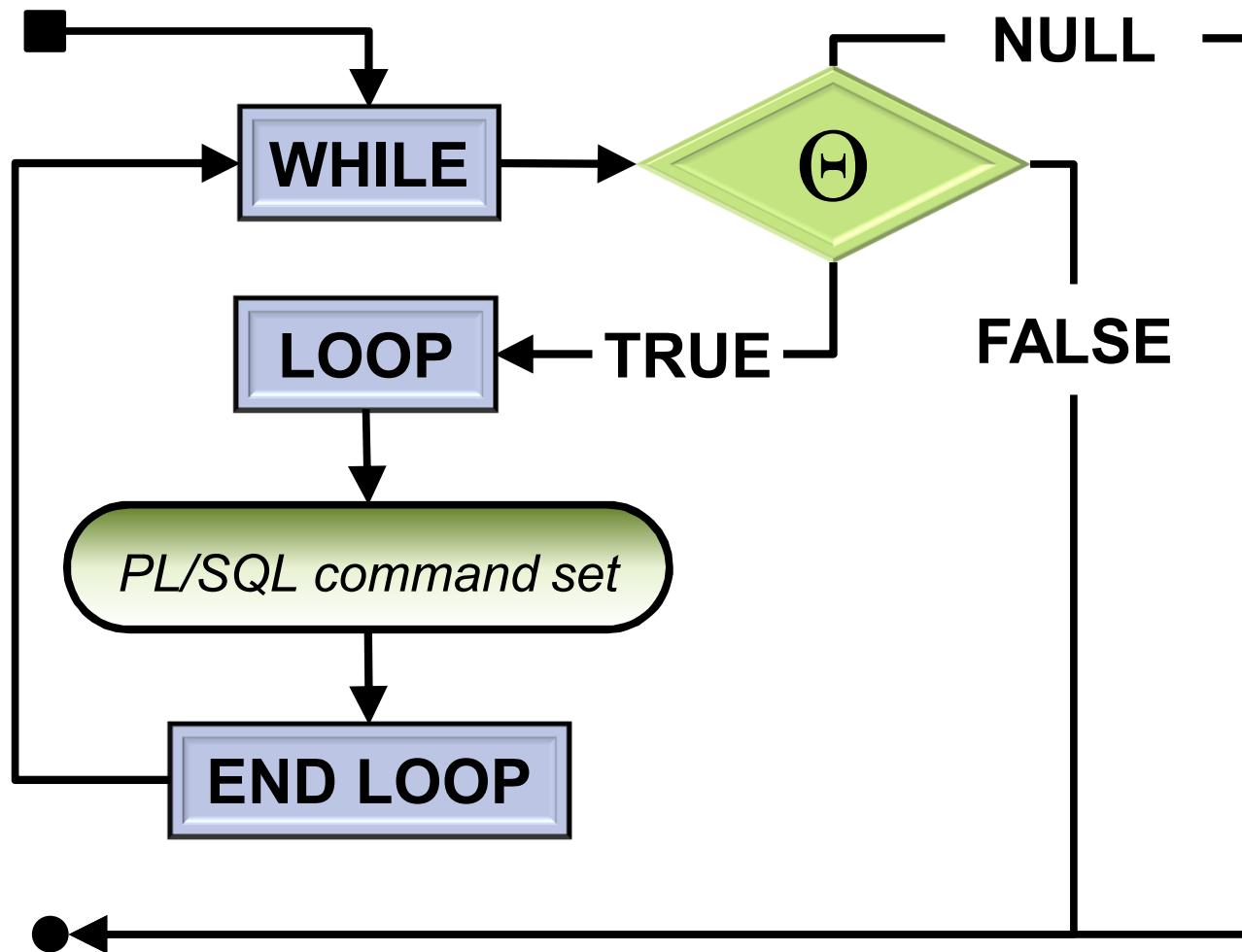
Нескінчений цикл.

Деякі програми (наприклад, засоби спостереження за станом системи) розраховані на безперервне виконання з накопиченням необхідної інформації. У таких випадках можна навмисно використовувати нескінчений цикл. Нескінчений цикл зазвичай поглинає значну частину ресурсів процесора. Проблема вирішується припиненням виконання між ітераціями командою DBMS_LOCK.sleep(*t NUMBER*). Під час призупинення програма практично не витрачає ресурси процесора.



Цикл WHILE.

Умовний цикл WHILE виконується до тих пір, поки визначена в циклі умова залишається рівною TRUE. А оскільки можливість виконання циклу залежить від умови і не обмежується фіксованою кількістю повторень, він використовується саме в тих випадках, коли кількість повторень циклу заздалегідь невідома.



Цикл WHILE.

Умова – логічна змінна або вираз (Θ -вираз), результатом перевірки якого є логічне значення TRUE, FALSE або NULL. Умова перевіряється при кожній ітерації циклу. Якщо результат виявляється рівним TRUE, тіло циклу виконується. Якщо ж результат дорівнює FALSE або NULL, то цикл завершується, а управління передається виконуваної команді, наступної за командою END LOOP.

Властивості циклу WHILE

№	Властивість	Опис
1	Умова завершення циклу	Якщо значенням логічного виразу циклу є FALSE або NULL.
2	Позиція перевірки умови	Перед першим та кожним наступним виконанням тіла циклу. Таким чином, не гарантується навіть одноразове виконання тіла циклу WHILE
3	Доцільність використання	<ol style="list-style-type: none"> 1) Якщо не відомо, скільки разів буде виконуватися тіло циклу; 2) можливість виконання циклу повинна визначатися умовою; 3) тіло циклу може не виконуватися жодного разу.

Цикл WHILE. Приклад.

WHILE

mask_index <= mask_count

AND

NOT dateConverted

LOOP

BEGIN

/* Спроба перетворення рядка по масці в записі таблиці */

retval := TO_DATE (value_in, fmts (mask_index));

dateConverted := TRUE;

EXCEPTION WHEN OTHERS

THEN

mask_index := mask_index + 1;

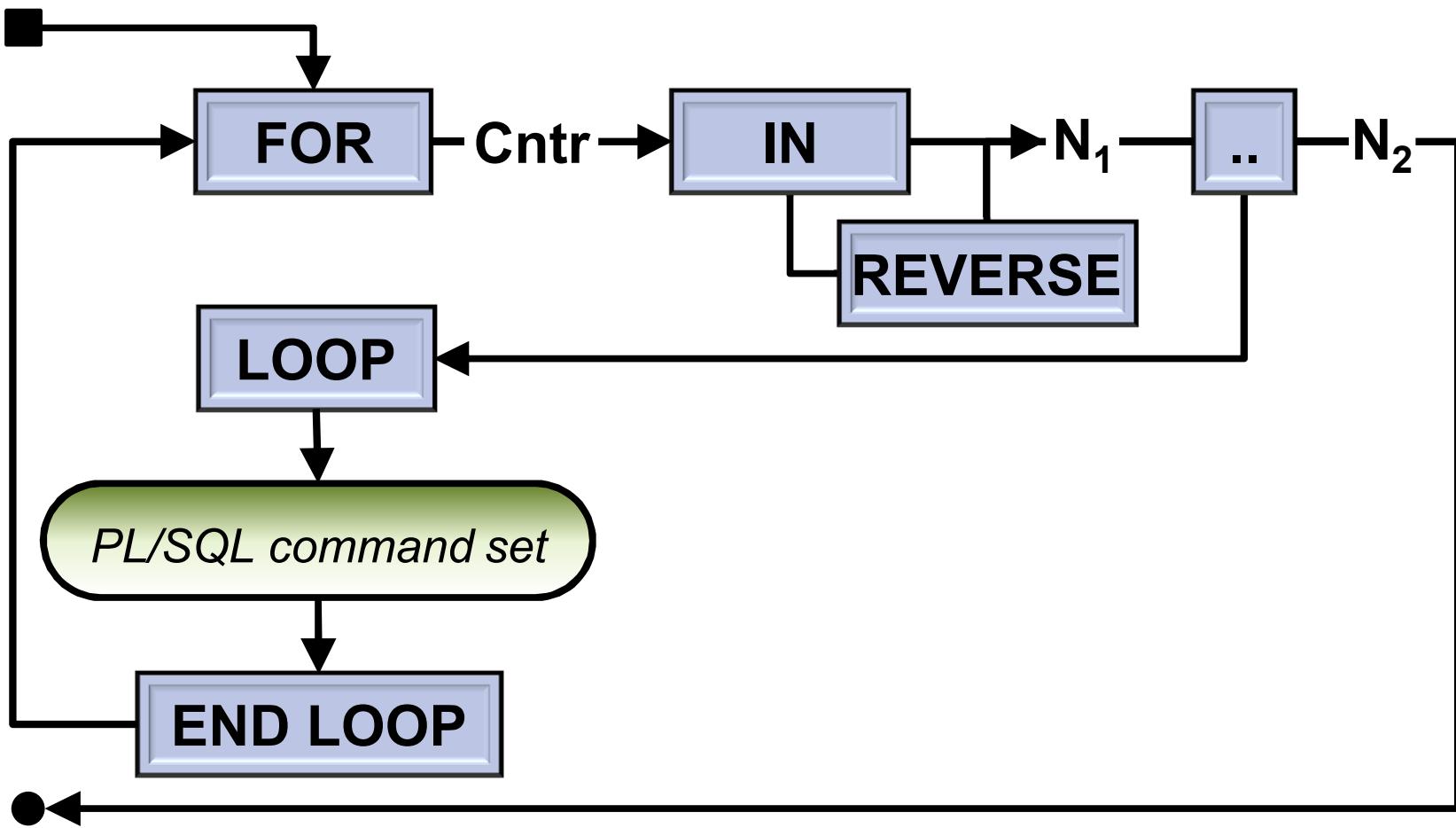
END;

END LOOP;

- Вся інформація, необхідна для обчислення умови, повинна задаватися перед первішим виконанням циклу.
- Може виявитися, що цикл WHILE не буде виконано жодного разу.

Цикл FOR з лічильником

Цикл з лічильником - це традиційний цикл FOR, підтримуваний в більшості мов програмування. Кількість ітерацій цього циклу відомо ще до його початку і задається діапазоном, певним між ключовими словами FOR і LOOP. Діапазон неявно оголошує керуючу змінну циклу (якщо вона не була явно оголошена раніше), визначає початкове і кінцеве значення діапазону, а також задає напрямок зміни лічильника (по зростанню або по спадаючій).



Цикл FOR з лічильником

Властивості циклу FOR

№	Властивість	Опис
1	Умова завершення циклу	Числовий цикл FOR безумовно завершується при виконанні кількості ітерацій, визначеного діапазоном значень лічильника. (Цикл може завершуватися і командою EXIT, але робити цього не рекомендується)
2	Позиція перевірки умови	Після кожного виконання тіла циклу PL/SQL перевіряє значення лічильника. Якщо воно виходить за межі заданого діапазону, виконання циклу припиняється. Якщо початкове значення більше кінцевого, то тіло циклу не виконується жодного разу.
3	Доцільність використання	Якщо тіло циклу повинно бути виконано певну кількість разів, а виконання не повинно перериватися передчасно

- **Не оголошуйте лічильник циклу.** PL/SQL автоматично неявно оголошує локальну змінну з типом даних INTEGER. Область дії цієї змінної збігається з границями циклу. Звертатися до лічильника за межами циклу не можна.

- **Вирази, які використовуються при визначенні діапазону, обчислюються один раз.** Вони не перераховуються в ході виконання циклу. Якщо змінити всередині циклу змінні, використовувані для визначення діапазону значень лічильника, його кордони залишаться колишніми.

- **Не міняйте значення лічильника і меж діапазону всередині циклу.** Компілятор PL/SQL або видасть повідомлення про помилку, або проігнорує зміни. В будь-якому випадку виникнуть проблеми.

- **Щоб значення лічильника зменшувалися в напрямку від кінцевого до початкового, використовуйте ключове слово REVERSE.** При цьому перше значення у визначенні діапазону (пачаткове_значення) має бути менше другого (конечное_значение). Не міняйте порядок проходження значень - просто поставте ключове слово REVERSE.

Цикли FOR з числовим лічильником. Приклади.

*Цикл виконується 10 раз;
лічильник збільшується від 1 до 10*

```
FOR loop_counter IN 1 .. 10
LOOP
    ... виконувані_команди...
END LOOP;
```

*Цикл виконується 10 раз;
лічильник зменшується від 10 до 1*

```
FOR loop_counter IN REVERSE 1 .. 10
LOOP
    ... виконувані_команди...
END LOOP;
```

Цикл не виконується жодного разу

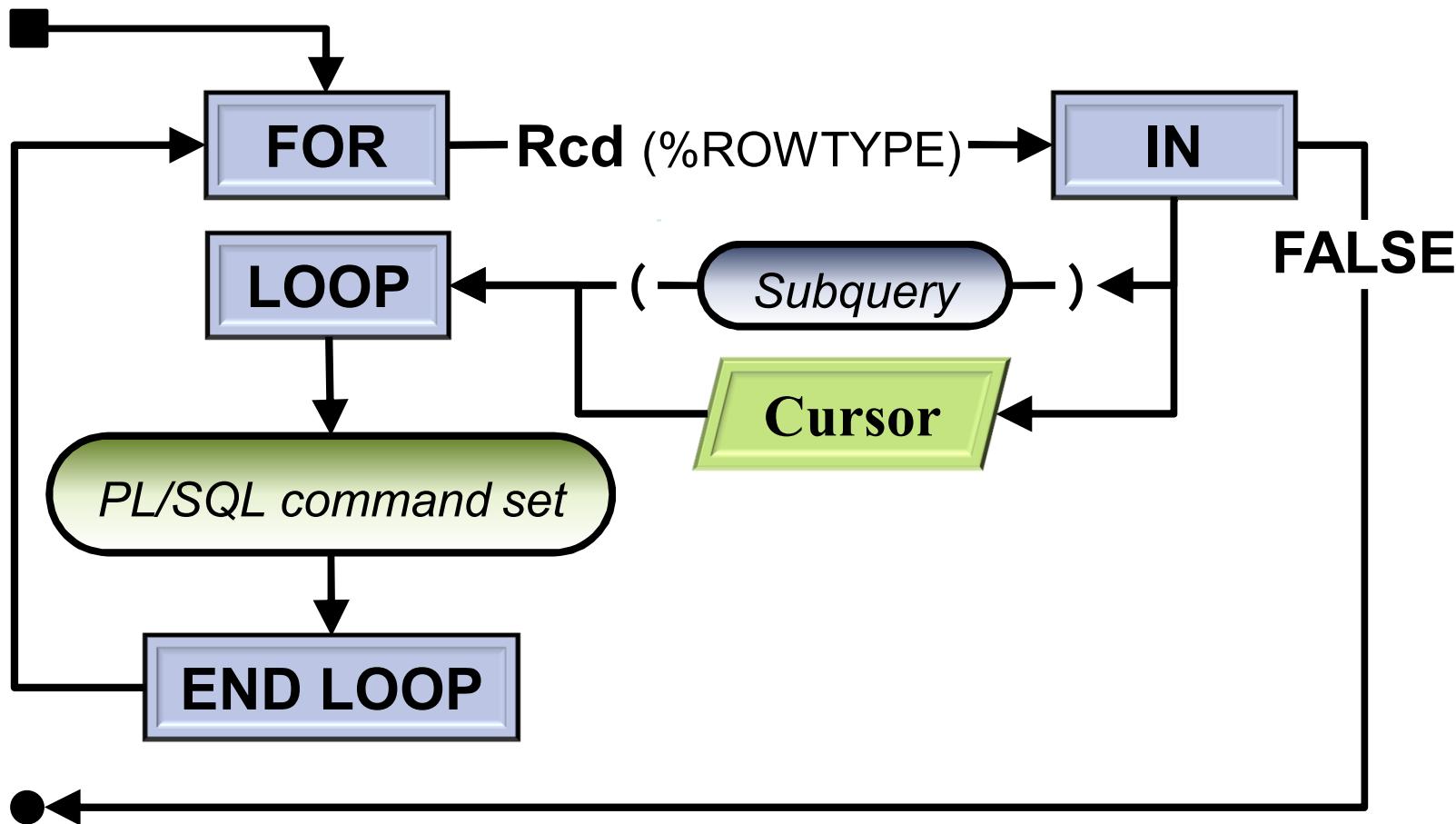
```
FOR loop_counter IN REVERSE 10 .. 1
LOOP
    ... виконувані_команди...
END LOOP;
```

Цикл виконується в діапазоні значень змінної та виразу

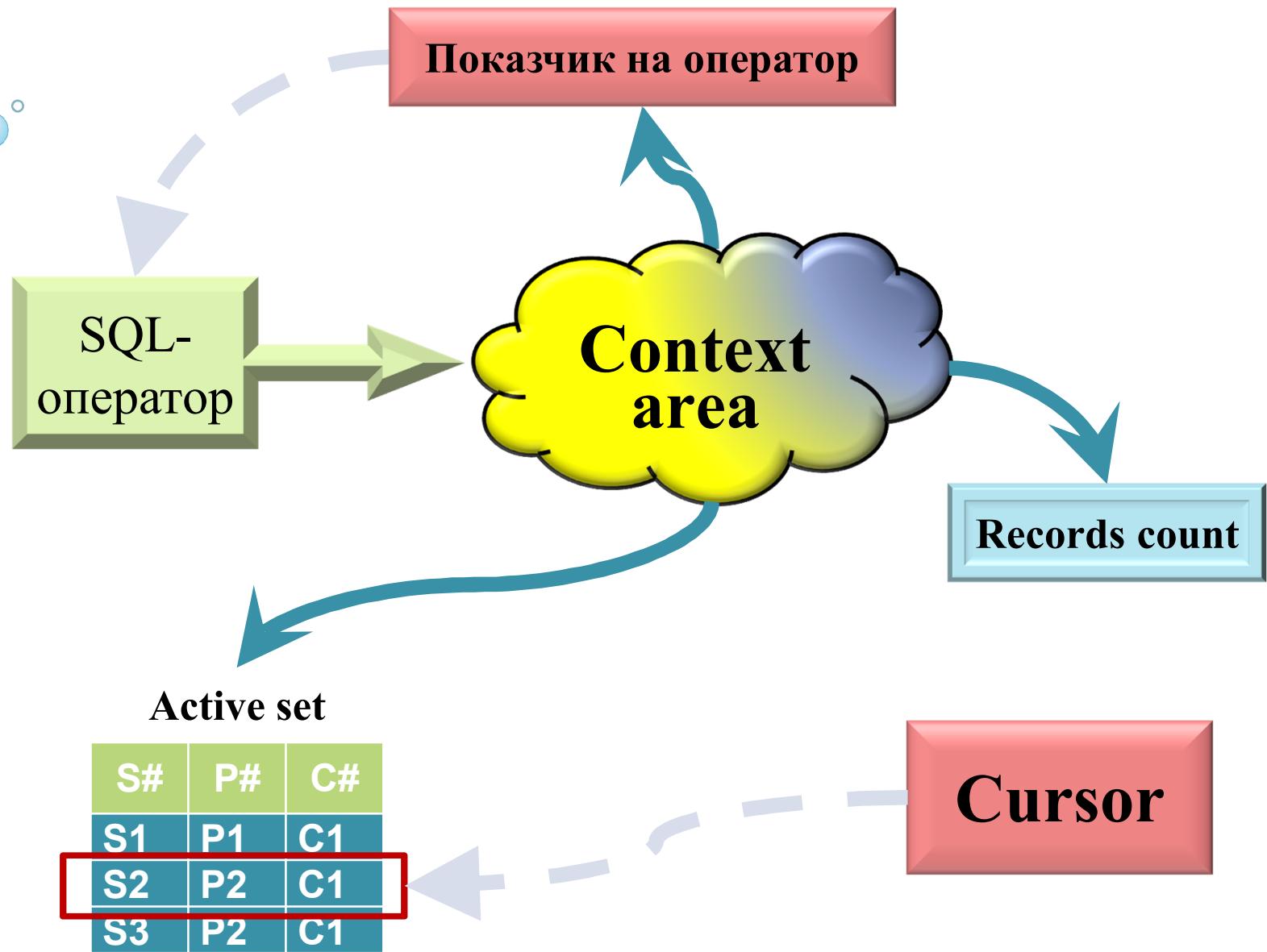
```
FOR calc_index IN
    start_period_number .. LEAST (end_period_number, current_period)
LOOP
    ... виконувані_команди...
END LOOP;
```

Цикли FOR з курсором.

Курсорна форма циклу **FOR** визначається явно заданим курсором або інструкцією **SELECT**, заданої безпосередньо в межах циклу. Цю форму використовують в тому випадку, якщо необхідно вилучити і обробити всі записи вибірка даних. Цикл **FOR** з курсором забезпечує ефективну інтеграцію процедурних конструкцій з безпосереднім доступом до баз даних. Його застосування помітно скорочує обсяг коду, та зменшує імовірність виникнення помилок при циклічній обробці даних.



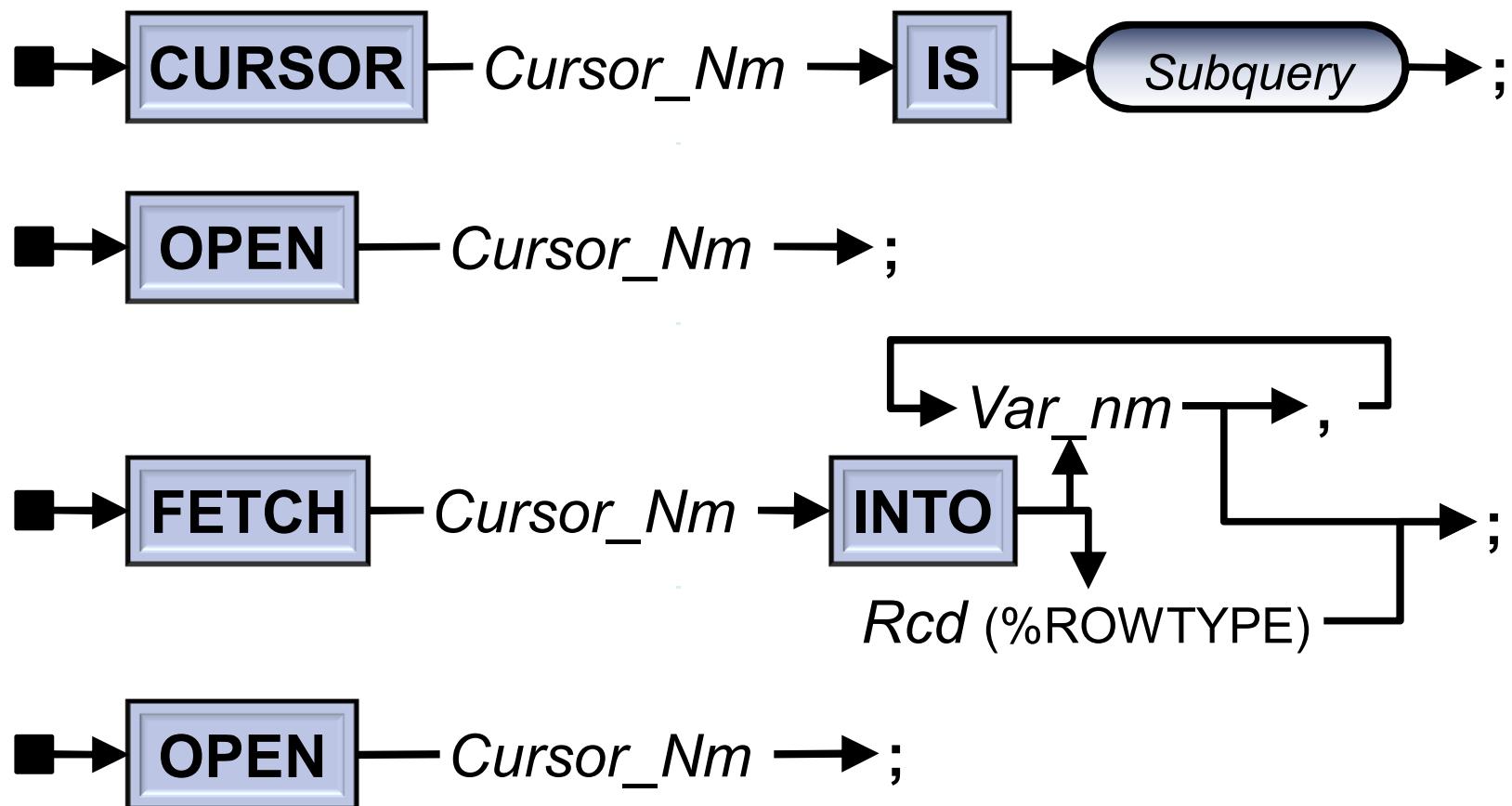
Поняття курсора



Створення і обробка курсора

Для створення і обробки явного курсору в PL/SQL необхідно:

1. Оголосити курсор.
2. Відкрити курсор для запиту.
3. Вибрати результати в змінні PL/SQL.
4. Закрити курсор.



Курсорний цикл FOR. Приклади.

DECLARE

```
CURSOR occupancy_cur IS
    SELECT pet_id, room_number
    FROM occupancy
    WHERE occupied_dt = TRUNC (SYSDATE);
    occupancy_rec  occupancy_cur%ROWTYPE;
```

BEGIN

```
OPEN occupancy_cur;
LOOP
```

```
    FETCH occupancy_cur INTO occupancy_rec;
    EXIT WHEN occupancy_cur%NOTFOUND;
    update_bill(occupancy_rec.pet_id, occupancy_rec.room_number);
```

END LOOP;

CLOSE occupancy_cur;

END;

DECLARE

```
CURSOR occupancy_cur IS
    SELECT pet_id, room_number
    FROM occupancy
    WHERE occupied_dt = TRUNC (SYSDATE);
```

BEGIN

```
FOR occupancy_rec IN occupancy_cur
LOOP
```

```
    update_bill(occupancy_rec.pet_id, occupancy_rec.room_number);
```

END LOOP;

END;

BEGIN

```
FOR occupancy_rec IN
    (

```

```
        SELECT pet_id, room_number
        FROM occupancy
        WHERE occupied_dt = TRUNC (SYSDATE)
```

)

LOOP

```
    update_bill(occupancy_rec.pet_id, occupancy_rec.room_number);
```

END LOOP;

END;