

СУЧАСНІ СУБД

Лекція №14

Тема:

ORACLE PL/SQL

Збережені процедури,
пакети та тригери

(Продовження)

Пакет являє собою згрупований за певними правилами іменований набір елементів коду **PL/SOL**. Він забезпечує логічну структуру для організації програм і інших елементів **PL/SQL**: курсорів, типів даних і змінних. Пакети втілюють низку специфічних функціональних можливостей, зокрема можливість приховування логіки і даних та визначення глобальних даних, що існують протягом сеансу.

№	Перевага	Опис
1	Спрошення супроводу і розширення застосунків	Якість застосунків PL/SQL визначається не тільки їх продуктивністю, але і простою супроводу. Пакети забезпечують інкапсуляцію коду (зокрема, приховати команди SQL за інтерфейсом процедур), дають можливість визначати константи для літералів та «чарівних» чисел, і групувати логічно пов'язані функції. Пакетний підхід до проектування і реалізації скорочує кількість потенційних збоїв в застосунках.
2	Підвищення продуктивності застосунків.	У багатьох ситуаціях використання пакетів підвищує продуктивність і ефективність роботи застосунків. Визначення постійних структур даних рівня пакета дозволяє переводити статичні значення з бази даних в байт-код. Це дає можливість уникнути повторних запитів, а отже, значно прискорити отримання результату. Крім того, підсистема управління пам'яттю Oracle оптимізована для доступу до відкомпільоване коду пакетів.

Переваги використання пакетів

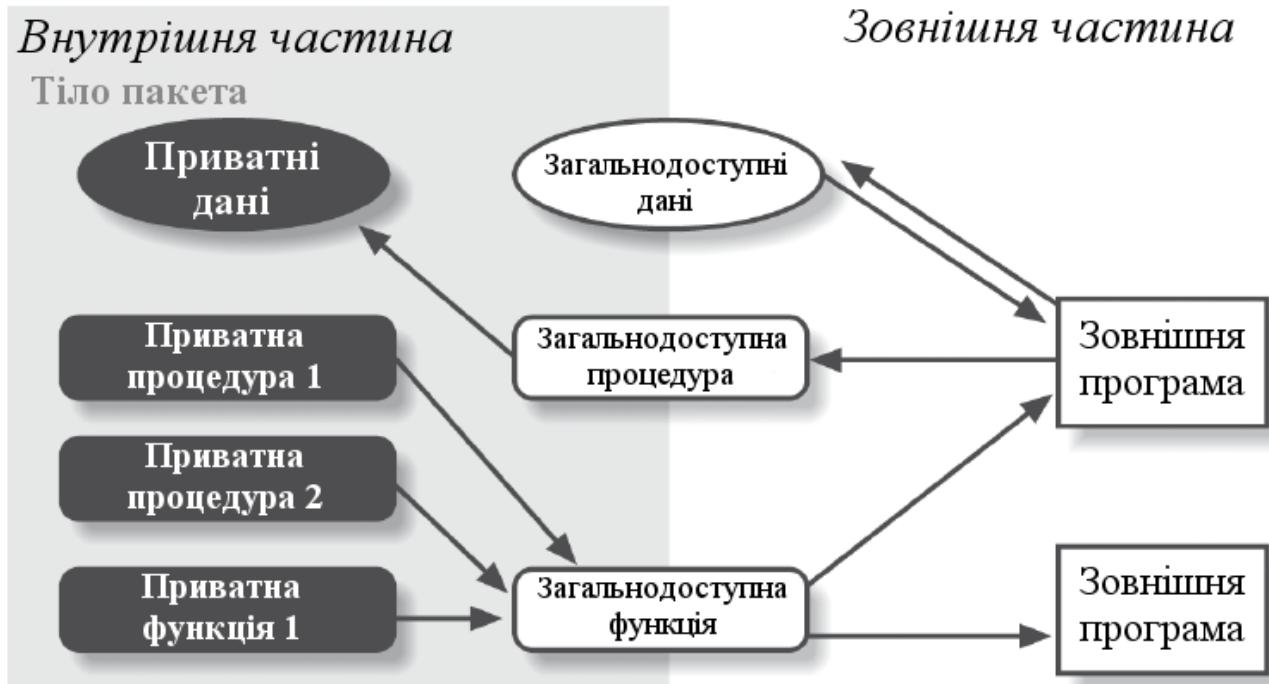
№	Перевага	Опис
3	Виправлення недоліків застосунків або вбудованих елементів	Деякі з вбудованих програмних компонентів Oracle мають недоліки. Зокрема, не кращим чином реалізовані найважливіші функції вбудованих пакетів UTL_FILE і DBMS_OUTPUT . Миритися з ними не обов'язково: можна розробити власний пакет на базі існуючого, виправивши якомога більше проблем. Наприклад, можна замінити вбудовану функцію DBMS_OUTPUT.PUT_LINE додаванням перевантаження для типу XMLType . Подібного результату можна досягти і за допомогою окремих функцій і процедур, але рішення з пакетами ефективніше..
4	Зниження необхідності в перекомпіляції коду	Пакет зазвичай складається з двох елементів: специфікації і тіла. Зовнішні програми (які не визначені в пакеті) можуть викликати тільки програми, перераховані в специфікації. Зміна і перекомпіляція тіла пакета не відбувається на роботі цих зовнішніх програм. Зниження необхідності в перекомпіляції коду є найважливішим фактором адміністрування великих обсягів програмного коду застосунка.

Приховування інформації. Приховування інформації про систему або додатку зазвичай переслідує дві мети. По-перше, можливості людини по роботі зі складними системами обмежені, отже розробник звільняється від необхідності вникати в непотрібні подробиці і може зосередитися на дійсно важливих аспектах. По-друге, приховування інформації перешкоджає доступу до закритих даних. Наприклад, розробник може викликати в своєму додатку готову функцію для обчислення деякого значення, але при цьому формула обчислень може бути таємною. Крім того, в разі зміни формулі всі модифікації будуть вноситися тільки в одному місці.

Спільні і приватні елементи. Концепція спільних і приватних елементів тісно пов'язана з концепцією приховування інформації. Загальнодоступний код визначається в специфікації пакета і доступний будь-якій схемі, яка має для цього пакета привілей **EXECUTE**. Приватний код доступний тільки в межах пакету. Зовнішні програми, що працюють з пакетом, не можуть використовувати приватний код.

Специфікація пакету. Вона містить визначення всіх загальнодоступних елементів пакету, на які можна посилатися ззовні. Специфікація нагадує великий розділ оголошень; вона не містить блоків **PL/SQL** або виконуваного коду. З добре спроектованої специфікації розробник може отримати всю необхідну для використання пакета інформацію.

Пакети. Приватність.



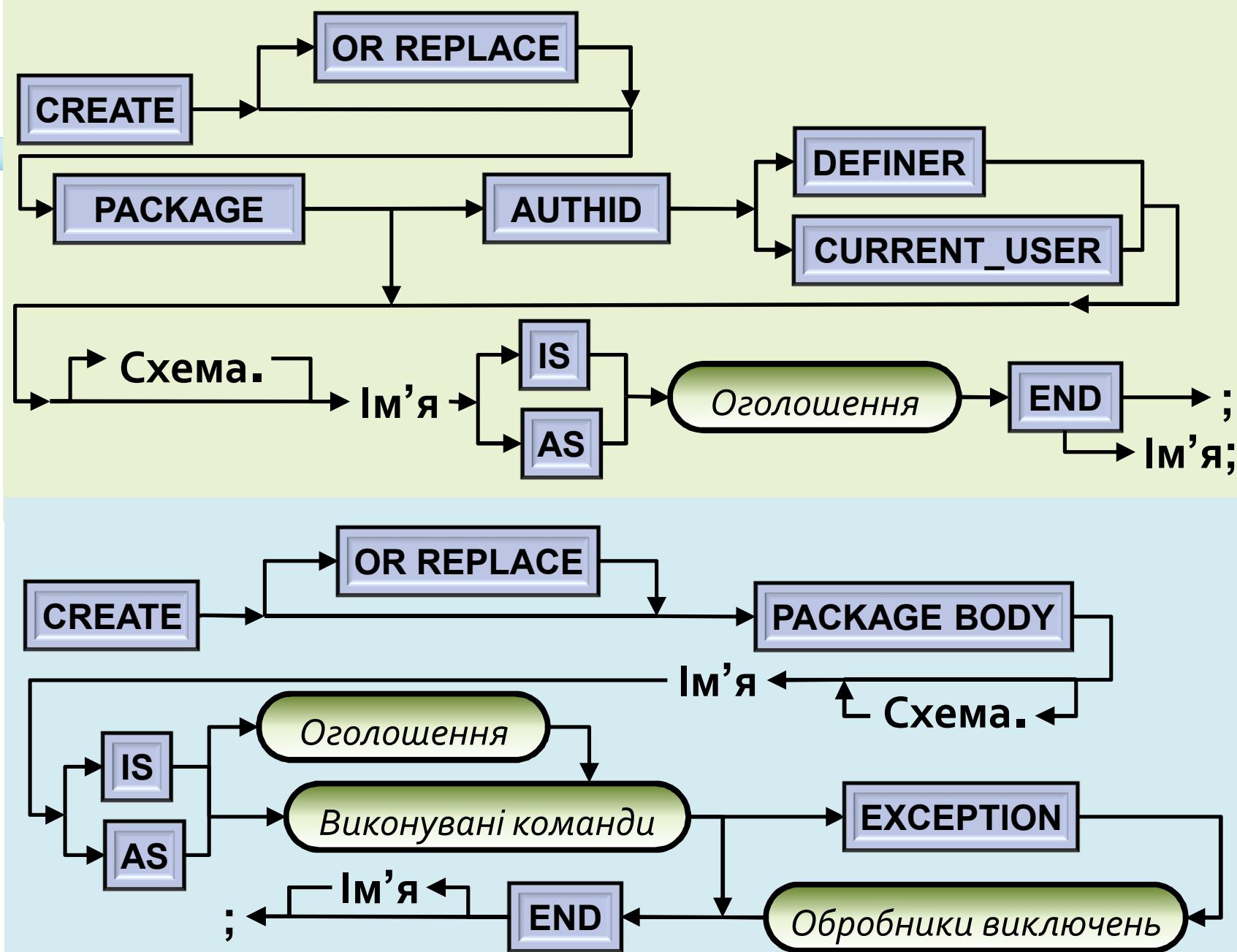
- Зовнішні програми не можуть перетинати кордон внутрішньої реалізації; інакше кажучи, зовнішня програма не може звертатися або викликати елементи, визначені в тілі пакету. Це приватні елементи, невидимі за межами пакета.
- Елементи, визначені в специфікації пакета, розташовуються по обидва боки від кордону між внутрішньою і зовнішньою частиною. Такі програми можуть викликатися зовнішньою програмою (з зовнішньої частини), вони доступні для приватних програм і в свою чергу можуть викликати або звертатися до всіх інших елементів пакету.
- Якщо виявиться, що об'єкт, який раніше був приватним (наприклад, модуль або курсор), треба зробити загальнодоступним, необхідно додати його в специфікацію і перекомпілювати пакет. Після цього об'єкт стане доступним поза межами пакета.
- Загальнодоступні елементи пакета забезпечують єдиний шлях до внутрішньої частини. В цьому відношенні специфікація пакета діє як керувальний механізм для пакета в цілому.

Тіло пакета. Тут знаходиться весь код, який необхідний для реалізації елементів, визначених у специфікації пакета. Тіло може містити відсутні в специфікації особисті елементи, на які не можна посилатися ззовні пакета, зокрема оголошення змінних і визначення пакетних модулів. Крім того, в тілі пакета може перебувати виконуваний (ініціалізаційний) розділ, який виконується тільки один раз для ініціалізації пакету.

Ініціалізація. Концепція ініціалізації полягає в тому, що ініціалізується не окрема змінна, а весь пакет шляхом виконання коду довільної складності. При цьому Oracle стежить за тим, щоб пакет ініціалізувати тільки один раз за сесію.

Сталість протягом сесії. Концепція сталості (або зберігання) полягає в тому, що при підключені до Oracle і виконанні програми, яка присвоює значення змінній рівня пакета (тобто змінній, оголошений в пакеті поза програмами, що містяться в ньому), ця змінна зберігає значення протягом усього сесії, навіть якщо виконання програми, що привласнила це значення, завершується.

Пакети. Синтаксис.



Пакети. Приклад.

```

CREATE OR REPLACE PACKAGE q_test
AS
    TYPE vrbn_info_rct IS REF CURSOR
    RETURN vyrobnyky%ROWTYPE;
    TYPE pstvk_info IS RECORD (g postavky%ROWTYPE);
    p_pstvk pstvk_info;
    q_pstvk postavky%ROWTYPE;
    q_n VARCHAR2(40);
    PROCEDURE q_proc(n IN NUMBER, q OUT VARCHAR2);
    FUNCTION q_func(k IN NUMBER, c IN VARCHAR2)
    RETURN NUMBER;
    FUNCTION q_func_tbl(k IN NUMBER, c IN VARCHAR2)
    RETURN vrbn_info_rct;
END q_test;
/

```

```

CREATE OR REPLACE PACKAGE BODY q_test
AS
    q_pstvk_prv pstvk_info;
    q_prv VARCHAR2(200);

    PROCEDURE q_proc(n IN NUMBER, q OUT VARCHAR2)
    IS BEGIN NULL; END q_proc;

    PROCEDURE q_proc_prv(n IN NUMBER, q OUT VARCHAR2)
    IS BEGIN NULL; END q_proc_prv;

    FUNCTION q_func(k IN NUMBER, c IN VARCHAR2)
    RETURN NUMBER IS BEGIN RETURN NULL; END q_func;

    FUNCTION q_func_tbl(k IN NUMBER, c IN VARCHAR2)
    RETURN vrbn_info_rct
    IS
        q_ret vrbn_info_rct;
    BEGIN
        OPEN q_ret FOR SELECT * FROM vyrobnyky;
        RETURN q_ret;
    END q_func_tbl;

    FUNCTION q_func_prv(k IN NUMBER) RETURN NUMBER
    IS BEGIN RETURN NULL; END q_func_prv;
END q_test;
/

```

Пакети. Правила побудови. Специфікація

Елементи практично будь-якого типу - числа, виключення, типи, колекції, тощо – можуть оголошуватися на рівні пакета (тобто такі елементи не належать конкретним процедурам або функціям цього пакету). Такі дані називаються даними рівня пакетів. У загальному випадку оголошувати змінні в специфікаціях пакетів не рекомендується, хоча оголошення констант на рівні пакета цілком прийнятні. У пакеті (як в специфікації, так і в тілі) не можна оголошувати курсорні змінні (Типу **REF CURSOR**), оскільки вони не можуть зберігати своє значення протягом сеансу.

У специфікації допускається оголошення типів для будь-яких структур даних: колекцій, записів або курсорних змінних.

У специфікації можна оголошувати процедури і функції, але необхідно вказувати тільки їх заголовки тобто визначення процедури або функції до ключово-го слова **IS** або **AS**. Заголовок повинен завершуватися символом крапки з комою «;» .

У специфікацію пакета можна включати явні курсори. Вони можуть бути представлені в одній з двох форм: **SQL - запит** або *є частиною оголошення курсору*, або ховається в тілі пакета (тоді в оголошенні присутній тільки вираз **RETURN**).

Якщо в специфікації пакета оголошуються процедури або функції або пакетний курсор без запиту, то тіло пакета повинно обов'язково включати реалізацію цих елементів.

Специфікація пакету може містити умову **AUTHID**, що визначає, як будуть вирішуватися посилання на об'єкти даних: відповідно до привілейв власника пакета (**AUTHID DEFINER**) або того, хто його викликає (**AUTHID CURRENT_USER**).

Після команди **END** в кінці специфікації пакета можна розмістити необов'язкову мітку, що ідентифікує пакет: **END my_package;**

Тіло пакета необхідно в тому випадку, якщо істинні хоча б деякі з наступних умов:

- *Специфікація пакету містить оголошення курсору з секцією RETURN.* В цьому випадку команда **SELECT** повинна бути вказана в тілі пакету.
- *Специфікація пакету містить оголошення процедури або функції.* У цьому випадку реалізація модуля повинна бути завершена в тілі пакету.
- *При ініціалізації пакету повинен виконуватися код, вказаний в ініціалізацій розділі.* Специфікація пакета не підтримує виконуваний розділ (виконувані команди в блоці BEGIN-END); ці команди можуть перебувати тільки в тілі пакету.

Пакети. Правила побудови. Тіло пакета.

Тіло пакета може містити розділ оголошень, виконуваний розділ і розділ виключення. Розділ оголошень містить повну реалізацію всіх курсорів і програм, які визначаються в специфікації, а також визначення всіх приватних елементів (не вказані в специфікації). Розділ оголошень може бути порожнім – за умови, що в тілі пакету присутній розділ ініціалізації.

Виконуваний розділ пакету також називається розділом ініціалізації; він містить додатковий код, що виконується при ініціалізації пакету в сеансі.

У розділі виключень обробляються всі виключення, ініційовані в розділі ініціалізації. Розділ виключень розташовується в кінці тіла пакета тільки в тому випадку, якщо визначено розділ ініціалізації.

Тіло пакета може мати наступну структуру: тільки розділ оголошень; тільки виконуваний розділ; виконуваний розділ і розділ виключень; розділ оголошень, виконуваний розділ і розділ виключень.

Секція **AUTHID** не може входити в тіло пакету; вона повинна розміщуватися в специфікації пакета. Все, що оголошено в специфікації, може використовуватися в тілі пакету.

Для тіла і специфікації пакета діють одні правила і обмеження оголошень структур даних – наприклад, неможливість оголошення курсорних змінних.

За командою **END** тіла пакета може слідувати необов'язкова мітка з ім'ям пакета: **END my_package;**

Пакети. Ініціалізація пакета.

Пакет може містити структури даних, що зберігаються протягом всього сеансу. Коли в ході сеансу вперше відбувається звернення до пакету (викликається оголошена в ньому програма, читається або записується значення змінної або використовується оголошений в пакеті тип), Оракл ініциалізує його, виконуючи такі дії:

- *Створення екземплярів даних рівня пакетів (значення змінних і констант).*
- *Присвоєння змінним і константам значень за замовчуванням, зазначених в оголошеннях.*
- *Виконання блоку коду, що міститься в ініціалізацій розділі.*

Оракл виконує всі ці дії тільки один раз за сеанс і тільки тоді, коли виникне безпосередня необхідність в цій інформації.



Пакет може бути повторно ініціалізований в ході сеансу, якщо він був перекомпільований з моменту останнього використання або було виконане скидання стану всього сеансу, на що вказує помилка ORA-04068.

Тригери

Тригери - це іменовані програмні блоки, які виконуються у відповідь на події, що відбуваються в базі даних. Вони відносяться до числа важливих елементів застосунків *Oracle* і використовуються для виконання дій, спрямованих на підтримку цілісності і структурної узгодженості баз даних.

№	Дія	Опис
1	Перевірка внесених в таблиці змін	Оскільки логіка перевірки даних безпосередньо пов'язана з конкретним об'єктом бази даних, тригери гарантують її суворе виконання і дотримання.
2	Автоматизація супроводу бази даних	Починаючи з Oracle8i можна було використовувати тригери, які автоматично виконуються при завантаженні і вивантаженні бази даних, для виконання операцій ініціалізації і очищення. Це значно зручніше, ніж створювати для цих операцій зовнішні по відношенню до бази даних сценарії
3	Узгодження обмежень з операціями адміністрування	За допомогою тригерів можна перевірити, чи допускається виконання певної операції над конкретним об'єктом бази даних (наприклад, видалення або модифікація таблиці). Коли правила перевірки реалізовані у вигляді тригерів, обійти їх дуже важко, якщо взагалі можливо.

Тригери

Події, з якими можна зв'язувати тригери

№	Подія	Опис
1	Команди DML	Тригери DML запускаються у відповідь на внесення, оновлення та видалення запису таблиці бази даних. Їх можна використовувати з метою перевірки значень, встановлених за замовчуванням, виконання аудиту змін і навіть заборони певних команд DML .
2	Команди DDL	Тригери DDL запускаються у відповідь на виконання команд DDL - наприклад, при створенні таблиці. З їх допомогою можна виконувати аудит і забороняти певні операції.
3	Події бази даних	Тригери подій бази даних використовуються при запуску і зупинці бази даних, при підключення і відключення сервера, а також при виникненні помилок Oracle. Починаючи з Oracle8i вони також дозволяють отримувати інформацію про операції з базою даних.
4	Тригери INSTEAD OF	Тригери INSTEAD OF є альтернативою триггерам DML . Вони запускаються безпосередньо перед операціями внесення, оновлення, видалення, і їх код визначає, які дії слід виконати замість відповідної операції DML . Тригери INSTEAD OF керують операціями над представленнями, але не над таблицями. З їх допомогою можна перетворювати неоновлювані представлення в оновлювані, змінюючи при необхідності їх поведінку.

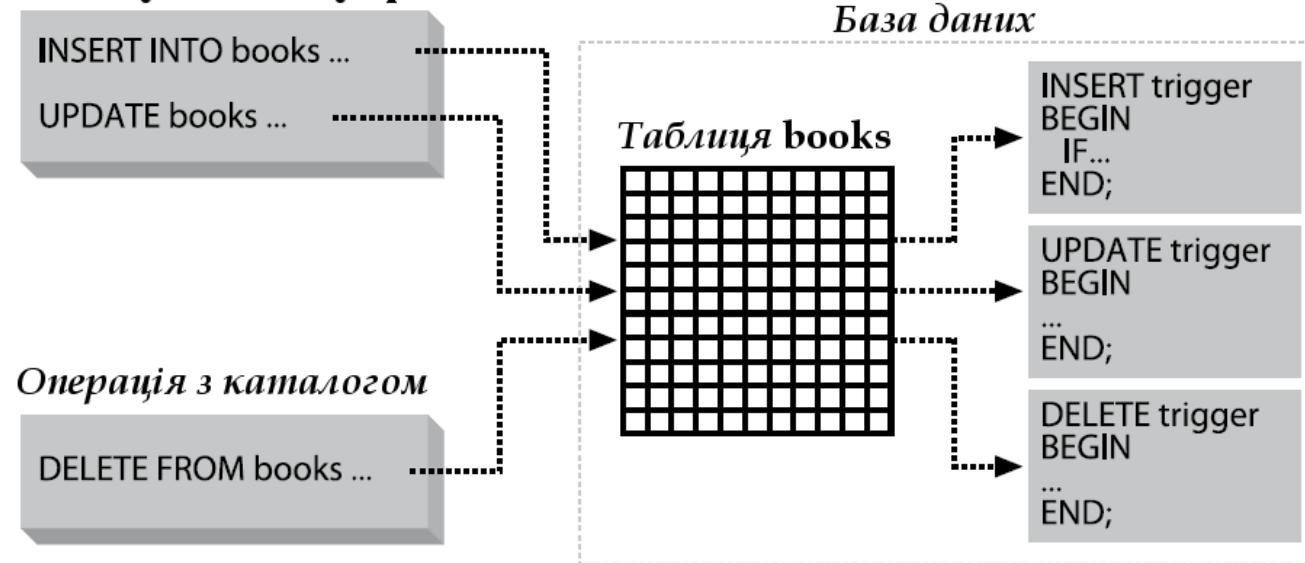
Події, з якими можна зв'язувати тригери (продовження)

№	Подія	Опис
5	Призупинені команди	у Oracle9i введена концепція призупинених команд. Якщо в ході виконання команди виникла проблема доступності простору (недостатньо табличного простору або вичерпана квота), Oracle може перевести її в режим призупинення до тих пір, поки проблема не буде вирішена. З цією подією можна пов'язати тригер, який автоматично повідомляє про проблему або навіть самостійно усуває її.

Тригери рівня команд DML

Тригери рівня команд DML активізуються після внесення, оновлення або видалення записів конкретної таблиці. Це найпоширеніший тип тригерів. Решта тригерів використовуються переважно адміністраторами бази даних. У **Oracle11i** з'явилася можливість об'єднання декількох тригерів DML в один **складений тригер**.

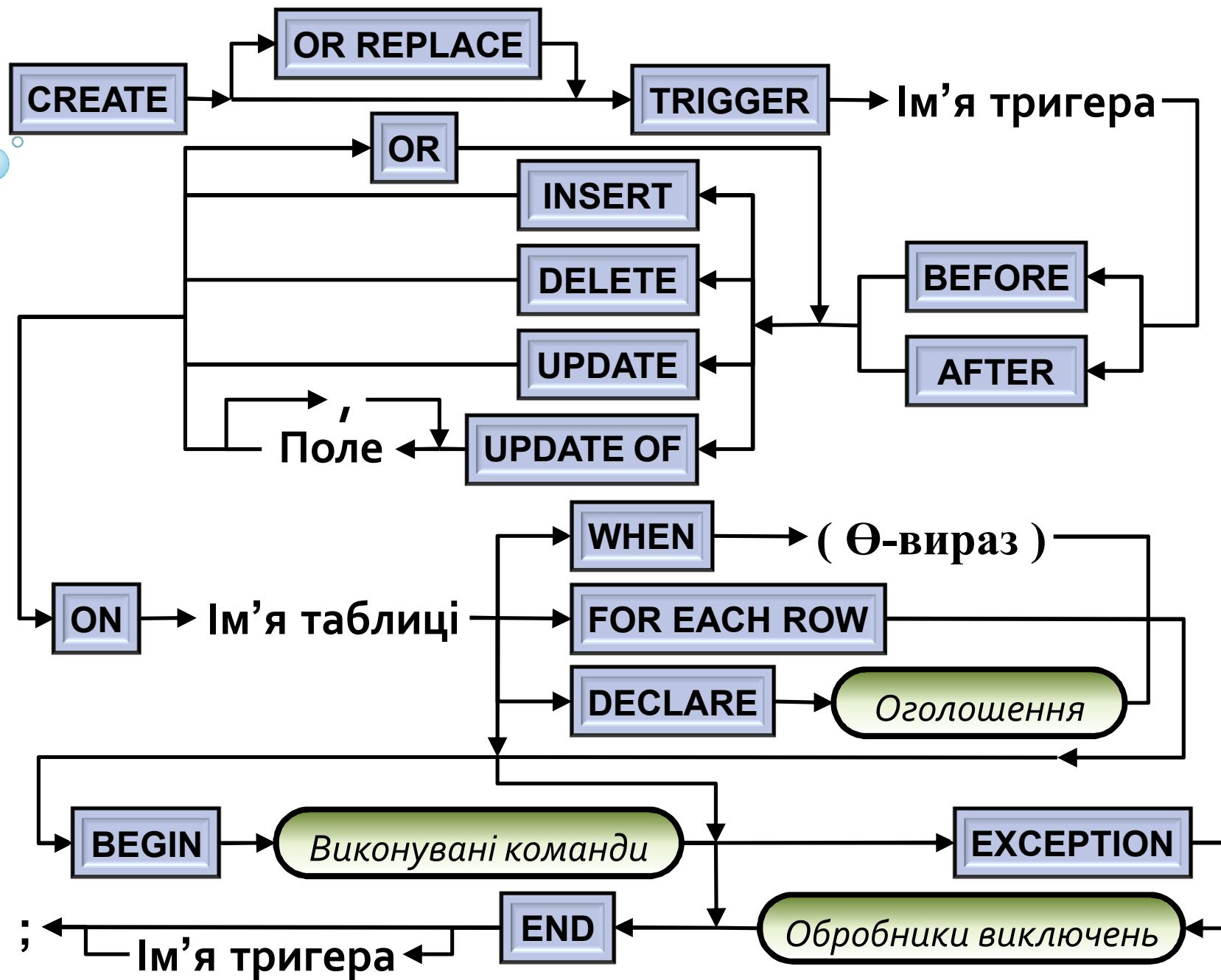
Застосунок для управління бібліотекою



При проектуванні тригера необхідно визначити:

- Як тригер буде запускатися – по одному разу для кожної команди SQL або для кожного модифікованого запису.
- Коли саме повинен викликатися тригер – до або після виконання операції над записами.
- Для яких операцій повинен спрацьовувати тригер – внесення, оновлення, видалення або їх певної комбінації.

Тригери рівня команд DML. Синтаксис.



Тригери рівня команд DML.Приклади.

```
CREATE OR REPLACE
TRIGGER bef_ins_ceo_comp
BEFORE INSERT
ON ceo_compensation
FOR EACH ROW
BEGIN
    PRAGMA AUTONOMOUS_TRANSACTION;
    INSERT INTO ceo_comp_history
    VALUES
    (
        :NEW.name,
        :OLD.compensation,
        :NEW.compensation,
        'AFTER INSERT', SYSDATE
    );
    COMMIT;
END;
```

```
CREATE OR REPLACE
TRIGGER validate_employee_changes
AFTER INSERT OR UPDATE
ON employees
FOR EACH ROW
BEGIN
    check_date(:NEW.hire_date);
    check_email(:NEW.email);
END;
```

Основні концепції тригерів.

Тригер BEFORE. Викликається до внесення будь-яких змін (наприклад, BEFORE INSERT).

Тригер AFTER. Виконується після окремої команди SQL, яка може обробляти одну або більше записів бази даних (наприклад, AFTER UPDATE).

Тригер рівня команди. Виконується для команди SQL в цілому (яка може обробляти одну або кілька рядків бази даних).

Тригер рівня запису. Виконується для окремого запису оброблюваної командою SQL. Якщо, припустимо, таблиця містить 1000 рядків, то наступна команда UPDATE модифікує всі ці рядки. І якщо для таблиці визначено тригер рівня запису, він буде виконаний 1000 разів.

Псевдозапис NEW. Структура даних з ім'ям NEW має такий же самий вигляд і (майже) такі ж властивості, як запис PL/SQL. Цей псевдозапис доступний тільки всередині тригерів поновлення і вставки; він містить значення модифікованого запису після внесення змін.

Псевдозапис OLD. Структура даних з ім'ям OLD має такий же самий вигляд і (майже) такі ж властивості, як запис PL/SQL. Цей псевдозапис доступний тільки всередині тригерів поновлення і вставки; він містить значення модифікованого запису до внесення змін.

Секція WHEN. Частина тригера DML, що визначає умови виконання коду тригера (і дозволяє уникнути зайвих операцій).