Procedury zapamiętane w Interbase - samodzielne programy napisane w specjalnym języku (właściwym dla serwera baz danych Interbase), który umożliwia tworzenie zapytań, pętli, instrukcji warunkowych itp.; można je wywoływać zarówno z linii SQL jak i z aplikacji zewnętrznej; mogą posiadać dane wejściowe (parametry) jak i dane wyjściowe. Umożliwiają przeniesienie skomplikowanych (czasochłonnych) obliczeń na serwer i odciążenie stacji klientów, redukcję ruchu w sieci. Różne aplikacje pracujące na tej samej bazie danych mogą korzystać z jednej procedury zapamiętanej. Zmiana takiej procedury następuje tylko w jednym miejscu (w bazie danych) i jest widoczna na wszystkich stacjach pracujących z tą bazą. Procedury mogą być **wybierające** (muszą zwracać przynajmniej jedną wartość – wywoływane przez **SELECT**) lub **wykonywalne** (wykonują operacje na danych w bazie, mogę również zwracać wartości, wywoływane przez **EXECUTE PROCEDURE**).

Definicja procedury zapamiętanej:

```
CREATE PROCEDURE nazwa procedury                    nagłówek
(parametr1 typ danych, parametr2 typ danych, …)
RETURNS ( parametr1 typ danych, parametr2 typ danych …)
AS
DECLARE VARIABLE zmienna1 typ danych;               ciało
DECLARE VARIABLE zmienna2 typ danych;
…
BEGIN
….
END;
```

W ciele procedury można używać instrukcji SQL-a do manipulowania danymi typu INSERT, UPDATE, DELETE, SELECT, funkcji użytkownika (UDF), funkcji i operatorów SQL, generatorów, wyjątków, rozszerzeń języka SQL.

Wyzwalacze (triggery) w Interbase - mechanizmy powiązane z tabelą lub widokiem, które wykonują **automatycznie** określoną akcję przed lub po **wstawieniu**, **modyfikacji**, **usunięciu** wiersza z danej tabeli (widoku). Operują w obrębie **aktywnej transakcji** – jej wycofanie powoduje wycofanie zmian wprowadzonych przez wyzwalacze. Umożliwiają automatyzację pewnych operacji wykonywanych w bazie danych (dzienniki zmian dla tabeli, sprawdzanie poprawności wprowadzanych danych, itp.).

Definicja wyzwalacza:

```
CREATE TRIGGER nazwa FOR tabela | widok          nagłówek
[ACTIVE | INACTIVE]
BEFORE | AFTER  DELETE | INSERT | UPDATE
[POSITION numer]
AS
DECLARE VARIABLE zmienna1 typ danych;            ciało
DECLARE VARIABLE zmienna2 typ danych;
…
```

BEGIN

….

END;

W ciele wyzwalacza można używać oprócz poleceń i rozszerzeń SQL takich, jak w procedurach zapamiętanych, także zmiennych kontekstowych:

NEW.kolumna – nowa wartość w kolumnie,

OLD.kolumna – bieżąca (dotychczasowa) wartość w kolumnie.

Rozszerzenia języka SQL stosowane w procedurach zapamiętanych i wyzwalaczach w Interbase

Rozszerzenie	**Opis**
BEGIN … END	blok instrukcji
zmienna = *wyrażenie*	przypisanie wyrażenia do zmiennej, zmienną może być zmienna lokalna, parametr wejściowy lub wyjściowy
/* *tekst komentarza*/	komentarz
EXCEPTION nazwa	wywołanie wyjątku, który może zostać obsłużony poprzez WHEN
EXECUTE PROCEDURE *nazwa_procedury*[*zmienna*[,*zmienna* …]] [RETURNING_VALUES *zmienna* [, *zmienna* …]]	wywołanie procedury
EXIT	skok do końca procedury
FOR *select* DO *instrukcja/blok instrukcji*	powtarza instrukcję lub blok instrukcji dla każdego wiersza otrzymanego z zapytania *select* (SELECT z klauzulą INTO)
IF (*warunek*) THEN *instrukcja/blok instrukcji* [ELSE *instrukcja/blok instrukcji*]	instrukcja warunkowa
POST_EVENT *zdarzenie*	wywołanie zdarzenia
SUSPEND	stosowane tylko w procedurach wybierających, zawiesza wykonywanie procedury do momentu wykonania kolejnej operacji pobrania wierszy (przez aplikację użytkownika), zwraca wartości wyjściowe
WHILE (*warunek*) DO *instrukcja/blok instrukcji*	pętla, sprawdza wartość warunku, jeżeli jest on spełniony, wykonywane są instrukcje występujące po słowie DO
WHEN {*błąd* [, *błąd* …] | ANY} DO *instrukcja/blok instrukcji*	obsługa błędów, gdy wystąpi jeden z wymienionych błędów, nastąpi wywołanie instrukcji znajdujących się po słowie DO, *przez błąd* rozumiemy - EXCEPTION *nazwa_wyjątku,* SQLCODE *numer_błędu* lub GDSCODE *numer_błędu* ANY- obsługa dowolnego błędu

Wyjątek w Intebase - **komunikat o błędzie**, który może zostać wywołany z poziomu procedury lub wyzwalacza. Po wywołaniu wyjątku **wstrzymane** jest wykonywanie procedury, z której wyjątek został wywołany - chyba że jest obsłużony za pomocą instrukcji **WHEN.**

tworzenie

CREATE EXCEPTION nazwa 'komunikat';

modyfikacja

ALTER EXCEPTION nazwa ‘nowy komunikat'

usuwanie

DROP EXCEPTION nazwa;

wywołanie wyjątku z poziomu procedury lub wyzwalacza

EXCEPTION nazwa;

Rodzaje błędów, które można obsłużyć w procedurze zapamiętanej (wyzwalaczu):

wyjątki wygenerowane przez **EXCEPTION,**

błędy SQL’a - **SQLCODE,**

błędy Interbase’a – **GDSCODE.**

Przykłady procedur zapamiętanych.

- użycie zmiennej lokalnej

```
SET TERM ## ;
CREATE PROCEDURE PODWYZKA AS
DECLARE VARIABLE kwota integer;
BEGIN
  kwota=100;
  UPDATE employee SET salary=salary+:kwota;
END ##
SET TERM ; ##
```

wywołanie :EXECUTE PROCEDURE podwyzka

- użycie parametrów

```
SET TERM ## ;
CREATE PROCEDURE dodaj_towar (nr INTEGER, nazwa CHAR(40), cena decimal(6,2))
AS
BEGIN
  INSERT INTO towar (nr, nazwa_towaru,cena_netto) VALUES (:nr, :nazwa, :cena);
END ##
SET TERM ; ##
```

wywołanie: EXECUTE PROCEDURE dodaj_towar(1,'LG 795',1000);

- użycie instrukcji FOR select DO

```
SET TERM !! ;
CREATE PROCEDURE GET_EMP_PROJ (EMP_NO SMALLINT)
RETURNS (EMP_PROJ SMALLINT) AS
BEGIN
  FOR SELECT PROJ_ID
      FROM EMPLOYEE_PROJECT
      WHERE EMP_NO = :EMP_NO
      INTO :EMP_PROJ
    DO
      SUSPEND;
END !!
```

wywołanie: SELECT * FROM GET_EMP_PROJ(24);

Przykłady wyzwalaczy.

- użycie generatora CUST_NO_GEN do utworzenia **unikalnego numeru klienta** CUST_NO w tabeli CUSTOMER

```
SET TERM !! ;
CREATE TRIGGER SET_CUST_NO FOR CUSTOMER
BEFORE INSERT AS
BEGIN
  NEW.CUST_NO = GEN_ID(CUST_NO_GEN, 1);
END !!
SET TERM ; !!
```

- użycie wyzwalacza w celu utworzenia **historii zmian** zarobków

```
SET TERM !! ;
CREATE TRIGGER SAVE_SALARY_CHANGE FOR EMPLOYEE
AFTER UPDATE AS
BEGIN
  IF (old.salary <> new.salary) THEN
     INSERT INTO SALARY_HISTORY (EMP_NO, CHANGE_DATE,
     UPDATER_ID, OLD_SALARY, PERCENT_CHANGE)
     VALUES (old.emp_no, 'now', USER, old.salary,
     (new.salary - old.salary) * 100 / old.salary);
END !!
SET TERM ; !!
```

- przykładowa **obsługa błędu SQL** w procedurze zapamiętanej

```
SET TERM ^ ;
CREATE PROCEDURE wstaw_towar (A INTEGER, B VARCHAR(25))
RETURNS (E CHAR(60)) AS
BEGIN
   E=' ';
   INSERT INTO towar(numer, nazwa) VALUES (:A, :B);
   WHEN SQLCODE -803 DO
   E = 'Błąd – naruszenie unikalności kolumny';
END^

CREATE EXCEPTION nieznany_nr_prac 'Nieznany numer pracownika'^

CREATE  PROCEDURE prac_proj
( EMP_NO SMALLINT, PROJ_ID CHAR(5)) AS
```

```
BEGIN
BEGIN
INSERT INTO employee_project (emp_no, proj_id) VALUES   (:emp_no, :proj_id);
WHEN SQLCODE -530 DO
EXCEPTION nieznany_nr_prac;
END
SUSPEND;
END^
/* SQLCODE -530         naruszenie klucza obcego*/
SET TERM ; ^
```