



# Definiowanie własnych funkcji grupujących z wykorzystaniem interfejsu ODCI

Krzysztof Jankiewicz

Instytut Informatyki

Politechniki Poznańskiej

Krzysztof.Jankiewicz@cs.put.poznan.pl

# Szablon

- Aby utworzyć funkcję grupującą należy:
  - stworzyć typ zawierający implementację funkcji grupującej, a następnie
  - funkcję agregującą wykorzystującą wcześniej zdefiniowaną implementację

# Przegląd metod typu zawierającego implementację

- **ODCIAggregateInitialize()** – inicjalizuje tzw. kontekst agregacji i instancję typu zawierającego implementację funkcji grupującej.
- **ODCIAggregateIterate()** – dokonuje iteracji przez wejściowy zbiór wierszy. Na podstawie ich wartości modyfikuje i przekazuje dalej kontekst agregacji. Jest to obowiązkowa metoda implementacji.
- **ODCIAggregateMerge()** – łączy dwa konteksty agregacji w pojedynczą instancję obiektu. Jest to obowiązkowa metoda implementacji.
- **ODCIAggregateTerminate()** – oblicza ostateczny wynik agregacji, wykonując jednocześnie operacje kończące przetwarzanie związane z przetwarzaniem (takie jak usunięcie obiektów tymczasowych itp.). Jest to obowiązkowa metoda implementacji.
- **ODCIAggregateDelete()** – usuwa wartości wejściowe z bieżącej grupy. Metoda opcjonalna.
- **ODCIAggregateWrapContext()** – integruje wszystkie zewnętrzne części bieżącego kontekstu agregacji tworząc z nich kontekst lokalny. Metoda opcjonalna.

# Metody (1/3)

## ▪ ODCIAggregateInitialize

- Składnia:

```
STATIC FUNCTION ODCIAggregateInitialize(actx IN OUT <impltype>) RETURN  
NUMBER
```

- Parametry:

- actx – kontekst agregacji zdefiniowany przez metodę. Wartość tego atrybutu jest pusta w przypadku tradycyjnych funkcji grupujących. W przypadku agregacji obejmujących okna parametr ten zawiera kontekst poprzedniego okna. Instancja obiektu przekazywana jest kolejnemu wywołaniu metody agregacji.

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

## ▪ ODCIAggregateIterate

- Składnia:

```
MEMBER FUNCTION ODCIAggregateIterate(self IN OUT <impltype>,  
val <inputdatatype>) RETURN NUMBER;
```

- Parametry:

- self – bieżący kontekst agregacji, który po zmodyfikowaniu przekazywany jest do dalszego przetwarzania
- val – agregowana wartość

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

# Metody (2/3)

## ▪ ODCIAggregateMerge

- Składnia:

```
MEMBER FUNCTION ODCIAggregateMerge(self IN OUT <impltype>,  
                                     ctx2 IN <impltype>) RETURN NUMBER;
```

- Parametry:

- self – pierwszy kontekst agregacji i jednocześnie wartość będąca wynikiem połączenia dwóch kontekstów.
- ctx2 – drugi kontekst agregacji wymagający połączenia.

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

## ▪ ODCIAggregateTerminate

- Składnia:

```
MEMBER FUNCTION ODCIAggregateTerminate(self IN <impltype>,  
                                       ReturnValue OUT <return_type>,  
                                       flags IN number) RETURN NUMBER;
```

- Parametry:

- self – kontekst agregacji
- ReturnValue – wynik agregacji
- flags – wektor binarny zawierający szereg opcji.

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

# Metody (3/3)

## ▪ ODCIAggregateDelete

- Składnia:

```
MEMBER FUNCTION ODCIAggregateDelete(self IN OUT <impltype>,  
                                     val <inputdatatype>) RETURN NUMBER;
```

- Parametry:

- self – kontekst agregacji, w wyniku metody odpowiednio modyfikowana.
- ctx2 – drugi kontekst agregacji wymagający połączenia.

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

## ▪ ODCIAggregateTerminate

- Składnia:

```
MEMBER FUNCTION ODCIAggregateTerminate(self IN <impltype>,  
                                       ReturnValue OUT <return_type>, flags IN number) RETURN NUMBER;
```

- Parametry:

- self – kontekst agregacji
- ReturnValue – wynik agregacji
- flags – wektor binarny zawierający szereg opcji.

- Wynik: wynikiem metody powinna być wartość ODCIConst.Success lub ODCIConst.Error

# Przykładowa implementacja (1/4)

```
CREATE TYPE Vector_TYPE AS OBJECT ( -- typ, którego wartości będą agregowane
  x_value NUMBER,
  y_value NUMBER
);
```

```
-- przykładowa tabela wraz z danymi
CREATE TABLE t1 (c1 NUMBER, c2 Vector_TYPE);
INSERT INTO T1 VALUES(1, Vector_TYPE(5,0));
INSERT INTO T1 VALUES(1, Vector_TYPE(3,0));
INSERT INTO T1 VALUES(2, Vector_TYPE(1,2));
INSERT INTO T1 VALUES(2, Vector_TYPE(2,1));
INSERT INTO T1 VALUES(2, Vector_TYPE(3,0.5));
INSERT INTO T1 VALUES(2, Vector_TYPE(3,2.5));
```

```
-- oczekiwane rezultaty
SELECT SumVector(c2) sum FROM t1;
```

```
SUM
-----
VECTOR_TYPE(17, 6)
```

```
SELECT c1, SumVector(c2) sum
FROM t1
GROUP BY c1 ORDER BY c1;
```

```
C1    SUM
-----
1 VECTOR_TYPE(8, 0)
2 VECTOR_TYPE(9, 6)
```

```
CREATE TYPE AggCtx_TYPE AS OBJECT ( -- typ wykorzystywany jako kontekst agregacji
  x NUMBER,
  y NUMBER
);
```

# Przykładowa implementacja (2/4)

```
CREATE OR REPLACE TYPE AggImpl_TYPE AS OBJECT -- specyfikacja implementacji indeksu
(
  aggCtx AggCtx_TYPE,

  STATIC FUNCTION ODCIAggregateInitialize(sctx IN OUT AggImpl_TYPE )
    RETURN PLS_INTEGER,

  MEMBER FUNCTION ODCIAggregateIterate(self IN OUT AggImpl_TYPE , arg IN Vector_TYPE)
    RETURN PLS_INTEGER,

  MEMBER FUNCTION ODCIAggregateTerminate(self IN AggImpl_TYPE , result OUT Vector_TYPE,
    flags IN NUMBER) RETURN PLS_INTEGER,

  MEMBER FUNCTION ODCIAggregateMerge(self IN OUT AggImpl_TYPE , sctx2 IN AggImpl_TYPE )
    RETURN PLS_INTEGER,

  member function ODCIAggregateDelete(self IN OUT AggImpl_TYPE , arg IN Vector_TYPE)
    RETURN PLS_INTEGER,

  MEMBER FUNCTION ODCIAggregateWrapContext(self IN OUT AggImpl_TYPE )
    RETURN PLS_INTEGER
);
```



# Przykładowa implementacja (3/4)

```
CREATE OR REPLACE TYPE BODY AggImpl_TYPE AS

    STATIC FUNCTION ODCIAggregateInitialize(sctx IN OUT AggImpl_TYPE )
        RETURN PLS_INTEGER IS
    BEGIN
        if sctx is null then
            sctx := AggImpl_TYPE (aggCtx_TYPE(0,0));
        end if;
        RETURN ODCIConst.Success;
    END;

    MEMBER FUNCTION ODCIAggregateIterate(self IN OUT AggImpl_TYPE , arg IN Vector_t)
        RETURN PLS_INTEGER IS
    BEGIN
        self.aggCtx.x := self.aggCtx.x + arg.x_value;
        self.aggCtx.y := self.aggCtx.y + arg.y_value;
        RETURN ODCIConst.Success;
    END;

    MEMBER FUNCTION ODCIAggregateTerminate(self IN AggImpl_TYPE , result OUT Vector_t,
        flags IN NUMBER) RETURN PLS_INTEGER IS
    BEGIN
        result := Vector_t(self.aggCtx.x,self.aggCtx.y);
        RETURN ODCIConst.Success;
    END;

    . . .
```

# Przykładowa implementacja (4/4)

```
. . .  
MEMBER FUNCTION ODCIAggregateMerge(self IN OUT AggImpl_TYPE , sctx2 IN AggImpl_TYPE )  
    RETURN PLS_INTEGER IS  
    BEGIN  
        self.aggCtx.x := self.aggCtx.x + sctx2.aggCtx.x;  
        self.aggCtx.y := self.aggCtx.y + sctx2.aggCtx.y;  
        RETURN ODCIConst.Success;  
    END;  
  
member function ODCIAggregateDelete(self IN OUT AggImpl_TYPE , arg IN Vector_t)  
    RETURN PLS_INTEGER IS  
    BEGIN  
        RETURN ODCIConst.Success;  
    END;  
  
MEMBER FUNCTION ODCIAggregateWrapContext(self IN OUT AggImpl_TYPE )  
    RETURN PLS_INTEGER IS  
    BEGIN  
        RETURN ODCIConst.Success;  
    END;  
  
END;
```

```
CREATE FUNCTION SumVector(arg Vector_TYPE) RETURN Vector_TYPE  
AGGREGATE USING AggImpl_TYPE ;
```