# Передача больших объектов XML через параметры ODAC в процедуры PLSQL

В жизни каждого программиста иногда возникают не совсем стандартные задачи. Одна из таких задач – передача в функцию или процедуру PLSQL нескольких значений в виде одного текстового упакованного поля (с последующей их распаковкой и обработкой внутри PLSQL).

При этом каждый программист придумывает собственные «велосипеды» форматов упаковки и разбора этих данных.

Обычно для этого используется длинная строка, которая записывается в переменную типа VARCHAR2, передаётся внутрь процедуры PLSQL и там обрабатывается.

Оракул, начиная с версии 9, поддерживает обработку XML, что предоставляет программистам очень удобный механизм упаковки больших структур данных в этот формат.

Для этого, в оракуле существует специальный объектный тип XMLtype, который можно использовать как внутри процедур и функций PLSQL:

declare

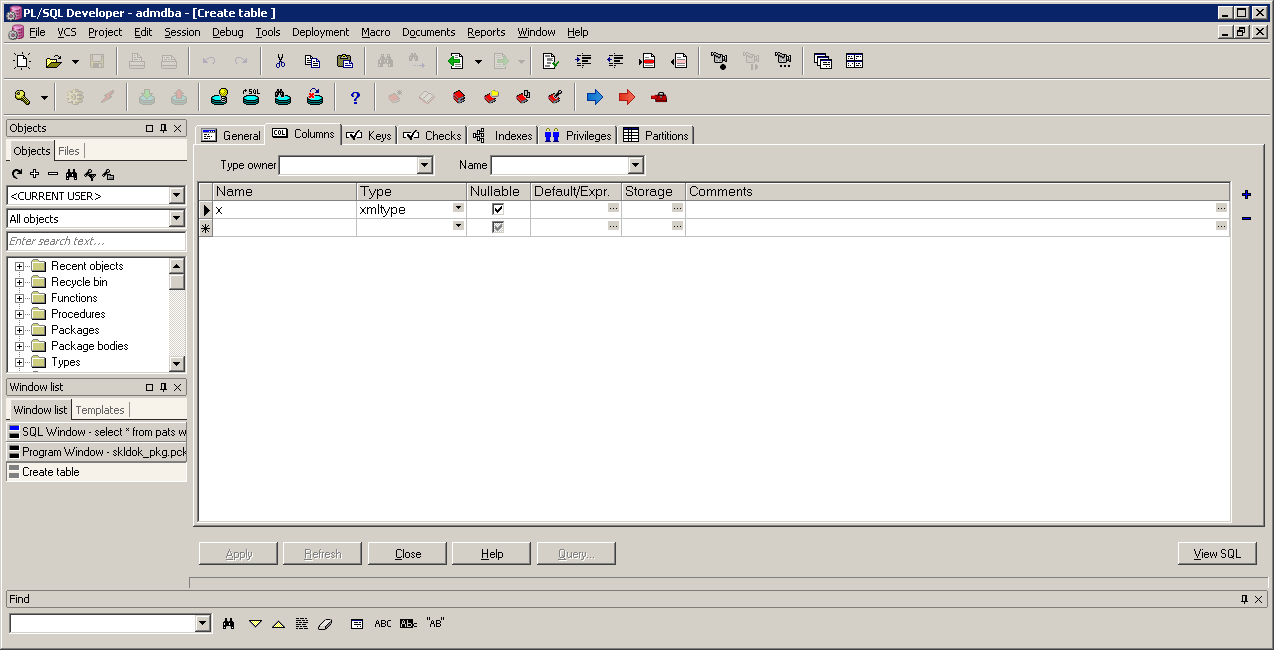
x XMLtype;

begin

null;

end;

так и непосредственно в таблицах для хранения таких данных:



При этом, «сырые» данные XML массива можно передать посредством привычного VARCHAR2 и очень просто преобразовать их в нужный тип

Procedure foo(s varchar2) is

X XMLtype;

Begin

X := XMLtype(s);

End;

Также можно передать в процедуру и сам объект типа XMLtype

Procedure foo(x XMLtype) is

Begin

…

End;

# Работа с объектом XMLtype

Для работы с типом XMLtype в оракле существует несколько функций

1. Преобразование VARCHAR2 в XMLtype делается при помощи конструктора XMLtype

Declare

X XMLtype;

Begin

X := XMLtype(‘<a><b>111</b><bb>222</bb></a>’);

// или так

X := xmltype.createxml(‘<a><b>111</b><bb>222</bb></a>’);

End;

1. Прочитать значение атрибута XML

Declare

X XMLtype;

S varchar2(100);

Begin X := XMLtype(‘<a><b>111</b><bb>222</bb></a>’);

S := Extractvalue(x, ‘/a/b’); // вернёт ‘111’

S := Extractvalue(x, ‘/a/bb’); // вернёт ‘222’

S := Extractvalue(x, ‘/a/qwerty’); // вернёт NULL

1. Обращение к методам объекта XMLtype

Declare

X,xx XMLtype;

S varchar2(100);

Begin X := XMLtype(‘<a><b>111</b><bb>222</bb></a>’);

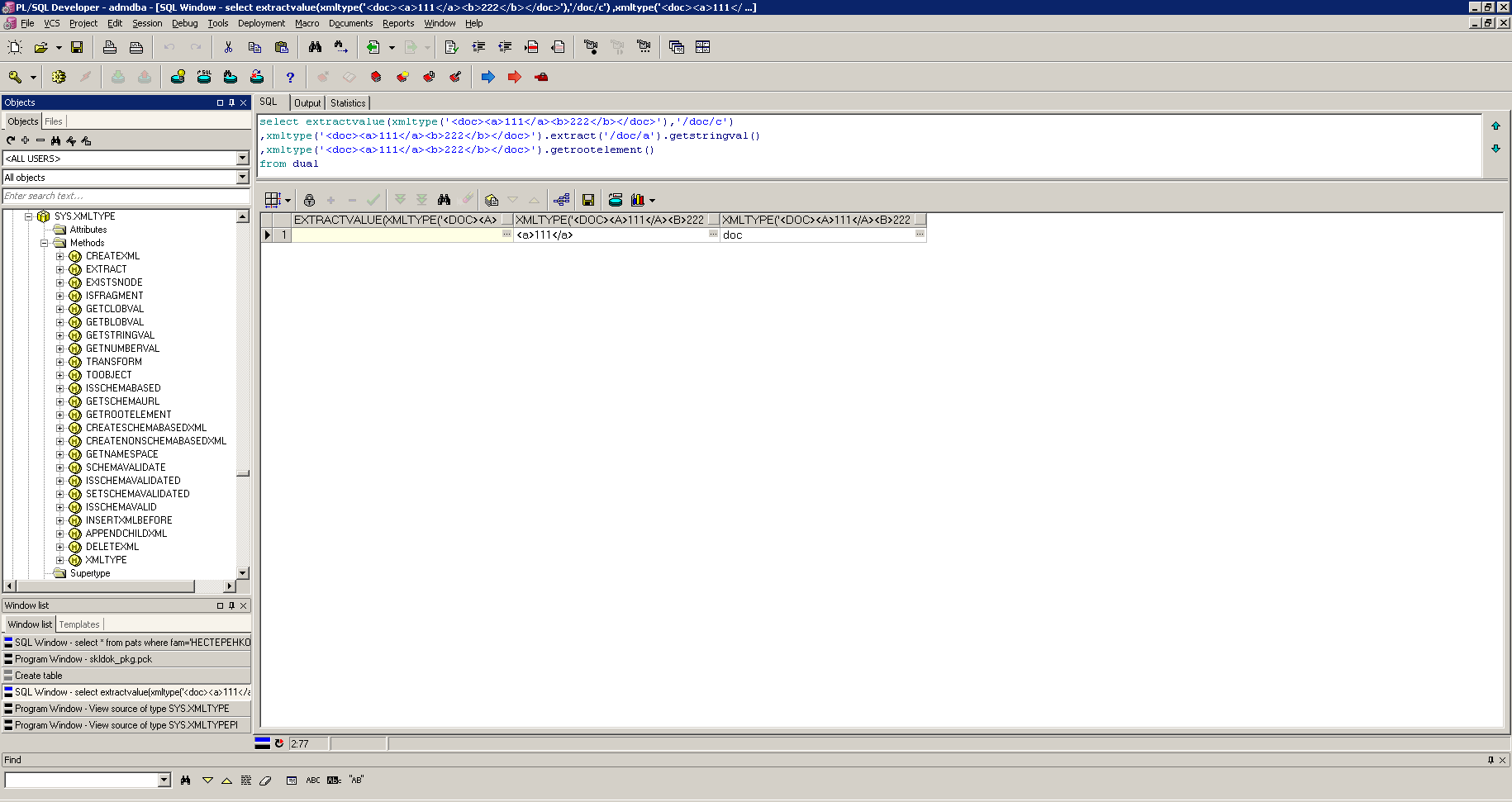
xx := X.extract(‘/a/b’); // вернёт объект XMLtype внутри которого будет ‘<b>111</b>’

// чтобы увидеть это, можно сделать так:

S := X.extract(‘/a/b’).GetStringVal(); // вернёт текст ‘<b>111</b>’

End;

1. А вообще, можно в девелопере найти тип XMLtype и посмотреть как устроено его писание



# Как это выглядит в Builder

Если размер XML данных не слишком большой, то в принципе всё выглядит стандартно. Например так:

std::auto\_ptr<TOraQuery> Q(new TOraQuery(0));

Q->SQL->Text = "begin foo(XMLtype(:1)); end;";

Q->Prepare();

AnsiString S = “<doc><a>111</a><b>222</b></doc>”;

Q->ParamByName("1")->AsString = S;

Q->ExecSQL();

Но есть нюансы.

Через параметр ->AsString не получится передать более 4000 символов за раз. Это ограничение ODAC. Лишние символы просто обрежутся и узнаете вы об этом только в тот момент, когда попытаетесь присвоить этот обрезанный varchar2 объекту XMLtype, который упадёт с ошибкой парсинга данных.

Ситуацию можно немного поправить, если использовать ->AsMemo вместо ->AsString.

Такая простейшая модификация отодвигает максимальную границу до 32к. (точнее 32515). Это ограничение уже не связано с ODAC, а является фундаментальным ограничением типа VARCHAR2.

Q->ParamByName("1")->**AsMemo** = S;

Q->ExecSQL();

Для большинства задач это может быть достаточно. Однако в некоторых случаях может потребоваться ещё больше (не забывайте, что формат XML помимо полезных данных хранит в себе теги и различные спецпоследовательности для возможности хранения спецсимволов)

Поэтому, иногда, при большом количестве тегов, или разветвлённой структуре может потребоваться больше 32к символов за раз. Это тоже можно победить, но требует применения немного дополнительной магии.

Для этой цели можно использовать параметр типа TOraClob. Но просто написать:

Q->ParamByName("1")->**AsOraClob->AsString** = S;

Q->ExecSQL();

Недостаточно. Сразу «из коробки» работать не будет. Необходимо перед этим добавить следующие заклинания:

Q->ParamByName("1")->Options->TemporaryLobUpdate = true;

Q->ParamByName("1")->ParamType = ptInput;

Q->ParamByName("1")->AsOraClob->AsString = S;

Q->ExecSQL();

Теперь можно передавать за раз очень много. Ну очень много. Сколько захотите.

Вот и всё. Детишки ☺