create or replace package body etl\_loader\_rnes\_pkg as

-- ----------------------------------------------------------------------

BULK\_LIMIT constant pls\_integer := 1000;

DFMT\_TZ constant varchar2(32) := 'yyyy-mm-dd"T"hh24:mi:ss.fftzh';

DFMT constant varchar2(32) := 'yyyy-mm-dd';

LOAD\_SOURCE constant varchar2(128) := 'PRIVOMA';

NO\_VALUE constant varchar2(3) := 'N/A';

-- ----------------------------------------------------------------------

g\_login varchar2(128);

g\_num\_dlm varchar2(1);

-- ----------------------------------------------------------------------

table\_not\_exist exception;

pragma exception\_init(table\_not\_exist, -942);

l\_cv varchar2(4000);

-- ----------------------------------------------------------------------

function fn (p in varchar2)

return varchar2 is

begin

return sys\_context('USERENV', 'CURRENT\_SCHEMA') || '.' || $$plsql\_unit || '.' || p;

end fn;

-- ----------------------------------------------------------------------

procedure loadSessionParameters is

begin

select substr(value, 0, 1)

into g\_num\_dlm

from nls\_session\_parameters

where parameter = 'NLS\_NUMERIC\_CHARACTERS';

end;

-- ----------------------------------------------------------------------

procedure loadContext(p\_event\_id in number) is

begin

select ete.login

into g\_login

from xxwh\_etl\_loader\_events\_t ete

where ete.id = p\_event\_id;

exception when others then

g\_login := 'ANONYMOUS';

end;

-- ----------------------------------------------------------------------

function toStr (p\_s varchar2) return varchar2 is begin l\_cv := p\_s; return p\_s; end;

function toNum (p\_s varchar2) return number is

begin l\_cv := p\_s;

return round(to\_number(translate(p\_s, '.,', g\_num\_dlm||g\_num\_dlm)), 8);

end;

function toDate(p\_s varchar2) return date is begin l\_cv := p\_s; return to\_timestamp(p\_s, DFMT\_TZ); end;

-- ----------------------------------------------------------------------

function consistentCheck (p\_event\_id in number)

return varchar2 is

--

cursor parseCursor is

select s.\*

from etl\_loader\_rnes201 s;

--

type parseTable is table of parseCursor%rowtype;

type loadTable is table of privoma\_oper\_plan\_json%rowtype;

l\_load\_table loadTable := loadTable();

l\_parse\_table parseTable;

l\_cr pls\_integer;

l\_return\_msg varchar2(4000);

--

function appendToLoad(p\_row parseCursor%rowtype)

return varchar2 is

l\_idx pls\_integer;

begin

l\_cr := p\_row."row\_id";

l\_load\_table.extend();

l\_idx := l\_load\_table.last;

l\_load\_table(l\_idx)."##origin\_name" := LOAD\_SOURCE;

l\_load\_table(l\_idx)."shipment\_id" := toNum(p\_row."shipment\_id");

l\_load\_table(l\_idx)."contractNumber" := toStr(p\_row."contractNumber");

l\_load\_table(l\_idx)."reportDate" := toDate(p\_row."reportDate");

l\_load\_table(l\_idx)."compositeKey\_id" := toNum(p\_row."compositeKey\_id");

l\_load\_table(l\_idx)."compositeKey\_reportId" := toNum(p\_row."compositeKey\_reportId");

l\_load\_table(l\_idx)."isBlock" := toNum(p\_row."isBlock");

l\_load\_table(l\_idx)."contractDate" := toDate(p\_row."contractDate");

l\_load\_table(l\_idx)."supplierTax" := toStr(p\_row."supplierTax");

l\_load\_table(l\_idx)."buyer" := toStr(p\_row."buyer");

l\_load\_table(l\_idx)."documentId" := toNum(p\_row."documentId");

l\_load\_table(l\_idx)."appendixNumber" := toStr(p\_row."appendixNumber");

l\_load\_table(l\_idx)."appendixDate" := toDate(p\_row."appendixDate");

l\_load\_table(l\_idx)."extraDocumentNumber" := toStr(p\_row."extraDocumentNumber");

l\_load\_table(l\_idx)."extraDocumentDate" := toDate(p\_row."extraDocumentDate");

l\_load\_table(l\_idx)."extraDocumentId" := toStr(p\_row."extraDocumentId");

l\_load\_table(l\_idx)."obligatoryDate" := toDate(p\_row."obligatoryDate");

l\_load\_table(l\_idx)."notificationNumber" := toStr(p\_row."notificationNumber");

l\_load\_table(l\_idx)."sellerTypeId" := toNum(p\_row."sellerTypeId");

l\_load\_table(l\_idx)."ownerId" := toStr(p\_row."ownerId");

l\_load\_table(l\_idx)."productSourceId" := toStr(p\_row."productSourceId");

l\_load\_table(l\_idx)."origin" := toStr(p\_row."origin");

l\_load\_table(l\_idx)."interdependencyId" := toNum(p\_row."interdependencyId");

l\_load\_table(l\_idx)."affiliation" := toNum(p\_row."affiliation");

l\_load\_table(l\_idx)."productTypeId" := toNum(p\_row."productTypeId");

l\_load\_table(l\_idx)."product" := toStr(p\_row."product");

l\_load\_table(l\_idx)."netWeight" := toNum(p\_row."netWeight");

l\_load\_table(l\_idx)."amountUnitId" := toNum(p\_row."amountUnitId");

l\_load\_table(l\_idx)."deliveryPoint" := toStr(p\_row."deliveryPoint");

l\_load\_table(l\_idx)."deliveryPointCode" := toStr(p\_row."deliveryPointCode");

l\_load\_table(l\_idx)."contractPrice" := toNum(p\_row."contractPrice");

l\_load\_table(l\_idx)."priceDate" := toDate(p\_row."priceDate");

l\_load\_table(l\_idx)."currency" := toStr(p\_row."currency");

l\_load\_table(l\_idx)."basisId" := toStr(p\_row."basisId");

l\_load\_table(l\_idx)."basisPrice" := toNum(p\_row."basisPrice");

l\_load\_table(l\_idx)."deliveryCondition" := toStr(p\_row."deliveryCondition");

l\_load\_table(l\_idx)."productionPlaceId" := toStr(p\_row."productionPlaceId");

l\_load\_table(l\_idx)."shipmentPlaceId" := toStr(p\_row."shipmentPlaceId");

l\_load\_table(l\_idx)."transportPrice" := toNum(p\_row."transportPrice");

l\_load\_table(l\_idx)."transportServices" := toNum(p\_row."transportServices");

l\_load\_table(l\_idx)."transportPriceDate" := toDate(p\_row."transportPriceDate");

l\_load\_table(l\_idx)."transportTypeId" := toStr(p\_row."transportTypeId");

l\_load\_table(l\_idx)."country" := toStr(p\_row."country");

l\_load\_table(l\_idx)."startDate" := toDate(p\_row."startDate");

l\_load\_table(l\_idx)."endDate" := toDate(p\_row."endDate");

l\_load\_table(l\_idx)."supplyDate" := toDate(p\_row."supplyDate");

l\_load\_table(l\_idx)."invoiceDate" := toDate(p\_row."invoiceDate");

l\_load\_table(l\_idx)."assistant" := toStr(p\_row."assistant");

l\_load\_table(l\_idx)."productGroup" := toStr(p\_row."productGroup");

l\_load\_table(l\_idx)."netPrice" := toNum(p\_row."netPrice");

l\_load\_table(l\_idx)."taxValue" := toNum(p\_row."taxValue");

l\_load\_table(l\_idx)."accessCode" := toStr(p\_row."accessCode");

l\_load\_table(l\_idx)."fromSite" := toStr(p\_row."fromSite");

l\_load\_table(l\_idx)."transportType" := toStr(p\_row."transportType");

l\_load\_table(l\_idx)."notes" := toStr(p\_row."notes");

l\_load\_table(l\_idx)."createUserId" := toNum(p\_row."createUserId");

l\_load\_table(l\_idx)."updateUserId" := toNum(p\_row."updateUserId");

l\_load\_table(l\_idx)."blockUserId" := toNum(p\_row."blockUserId");

l\_load\_table(l\_idx)."error" := toStr(p\_row."error");

l\_load\_table(l\_idx)."statusReg" := toStr(p\_row."statusReg");

l\_load\_table(l\_idx)."sentToRegDt" := toDate(p\_row."sentToRegDt");

l\_load\_table(l\_idx)."statusSentBool" := toNum(p\_row."statusSentBool");

l\_load\_table(l\_idx)."createDttm" := toDate(p\_row."createDttm");

l\_load\_table(l\_idx)."updateDttm" := toDate(p\_row."updateDttm");

l\_load\_table(l\_idx)."blockDttm" := toDate(p\_row."blockDttm");

l\_load\_table(l\_idx)."priceFca" := toNum(p\_row."priceFca");

l\_load\_table(l\_idx)."tradePolicy" := toNum(p\_row."tradePolicy");

l\_load\_table(l\_idx)."differencePrice" := toNum(p\_row."differencePrice");

l\_load\_table(l\_idx)."netWeightChanged" := toNum(p\_row."netWeightChanged");

l\_load\_table(l\_idx)."contractPriceChanged" := toNum(p\_row."contractPriceChanged");

l\_load\_table(l\_idx)."reportInvoiceDate" := toDate(p\_row."reportInvoiceDate");

l\_load\_table(l\_idx)."dealId" := toNum(p\_row."dealId");

l\_load\_table(l\_idx)."errorsJson" := toStr(p\_row."errorsJson");

l\_load\_table(l\_idx)."internalCountryID" := toNum(p\_row."internalCountryID");

l\_load\_table(l\_idx)."internalProductID" := toNum(p\_row."internalProductID");

l\_load\_table(l\_idx)."internalOwnerID" := toNum(p\_row."internalOwnerID");

l\_load\_table(l\_idx)."loadEventId" := p\_event\_id;

return

null;

end appendToLoad;

begin

loadSessionParameters;

loadContext(p\_event\_id);

savepoint before\_etl\_loader\_rnes201;

open parseCursor;

loop

fetch parseCursor bulk collect into l\_parse\_table limit BULK\_LIMIT;

exit when l\_parse\_table.count = 0;

for i in 1..l\_parse\_table.count loop

l\_return\_msg := appendToLoad(l\_parse\_table(i));

if l\_return\_msg is not null then

return l\_return\_msg;

end if;

end loop;

forall j in 1..l\_load\_table.count

insert into privoma\_oper\_plan\_json values l\_load\_table(j);

l\_load\_table.delete;

end loop;

close parseCursor;

return null;

exception

when others then

rollback to etl\_loader\_rnes201;

return 'On row '||l\_cr||' value "'||l\_cv||'" error: '||sqlerrm;

end consistentCheck;

end etl\_loader\_rnes\_pkg;

/