**Лабораторная работа**

**Разработка хранимого кода. Часть 2**

***Цель работы: закрепить навыки разработки и отладки хранимого программного кода, освоить использование курсоров в хранимом коде***

***Порядок выполнения работы***

1. Напишите хранимую функцию (единственный входной параметр «код рейса»), которая возвращает строку, содержащую название маршрута, по которому идет этот рейс, и все остановки этого маршрута (например: Маршрут: список названий остановок через пробел или запятую). Для проверки этой функции (назовём её ***route\_points***) можно написать запрос типа

***Select cod\_trip, route\_points(cod\_trip) from trips***

или ещё добавить в этот запрос where с каким-нибудь условием – вывод будет компактным и информативным для пассажиров.

Один из способов решения задачи – использовать курсор, содержащий названия остановок заданного рейса.

Подготовьте хотя бы два различных варианта кода этой функции, учитывая разнообразные возможности работы с курсорами в Oracle. Используйте в качестве примера функцию из файла «Пример хранимой функции с использованием курсора» на портале.

Функцию (два, а можно и три варианта) надо сдать преподавателю, т.к. в проверяющей системе такого задания пока нет. Также поместите её в отчёт по лабораторной работе.

2. Очень важная функция диспетчерской службы Автовокзала – «закрытие рейса» в момент отправки автобуса по этому рейсу. При закрытии рейса обнуляется количество проданных билетов (столбец tickets таблицы trips), а перед этим информация об отправленном рейсе заносится в архив (в том числе, такой важный показатель, как процент наполненности автобуса при отправке рейса).

Сейчас наша задача – отладить процедуру отправки («закрытия») рейса и убедиться, что информация об отправленном рейсе появилась в архиве.

Прежде чем заниматься самой процедурой, давайте подготовим архив к корректному приёму информации об отправленных рейсах. Напомним его схему:

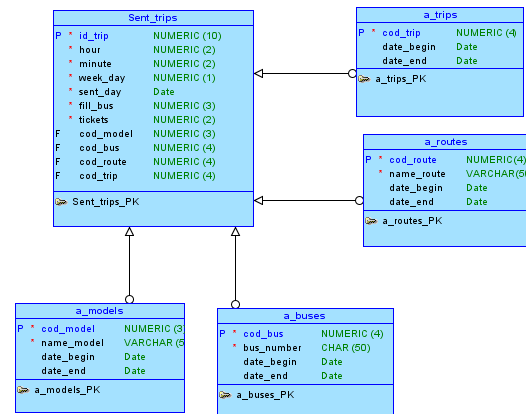


Схема хранилища данных «Автовокзал»

Таблица фактов **Sent\_trips** (**Отправленные рейсы)** содержит столбцы:

id\_trip **–** уникальный идентификатор отправленного рейса (PK)

сod\_trip (FK) – идентификатор рейса (уникален только в пределах недели, но полезен для выполнения группировки)

сod\_route (FK) – код маршрута

cod\_bus (FK) – код автобуса, назначенного на рейс

cod\_model (FK) – код модели этого автобуса

hour – часы отправления

minute – минуты отправления

week\_day – день недели

sent\_day – дата отправки рейса (при занесении рейса в хранилище это текущая дата)

fill\_bus – наполняемость автобуса в процентах (этот атрибут полезно вычислять при занесении рейса в хранилище)

tickets – количество проданных билетов (может понадобиться при формировании отчетов)

Справочники (таблицы размерностей) в основном совпадают с соответствующими таблицами оперативной БД, но дополнены полями **date\_begin** и **date\_end,** поскольку с течением времени происходит обновление и автобусного парка, и маршрутов, и рейсов – при этом записи в справочнике теряют актуальность, но не удаляются.

Скорее всего, сейчас архив у вас пустой, в том числе, и справочные таблицы a\_models, a\_buses, a\_routes и a\_trips. Наполним эти таблицы данными из оперативной базы данных – в ней есть таблицы models, buses, routes, trips и они наполнены. Например, для того, чтобы заполнить таблицу a\_models, можно написать такой запрос:

Insert into a\_models

Select cod\_model, name\_model, sysdate, null from models

Убедитесь, что таблица a\_models заполнилась данными (столбец date\_begin получит значение sysdate, а столбец date\_end должен быть пустым, т.к. все модели автобусов сейчас актуальны).

Аналогичными запросами заполните остальные таблицы-справочники хранилища - a\_buses, a\_routes и a\_trips.

Теперь можно отлаживать процедуру отправки («закрытия») рейса.

Алгоритм такой – проверяем, продан ли на заданный рейс хотя бы один билет (допустим, диспетчер нечаянно второй раз нажимает на кнопку **Закрыть рейс**). Если количество проданных билетов >0, то сначала отправляем сведения об отправляемом рейсе в Хранилище данных, а затем обнуляем столбец tickets (количество проданных билетов) в таблице trips, т.е. начинаем новый процесс продажи билетов с нуля, но уже на тот рейс, который отправляется через неделю.

Поясним наполнение таблицы sent\_trips – как раз в неё переносятся данные об отправленном рейсе.

Столбец id\_trip – суррогатный ключ, уникальный для каждого отправленного рейса (не путать с кодом рейса!), можно заполнить знакомым способом – последовательность и триггер. Последовательность создадим – от этого никуда не деться. Выполните команду:

**create sequence sent\_trips\_seq**

А без триггера можно обойтись, если заполнение id\_trip выполнять непосредственно в процедуре отправки рейса. Посмотрите, как это сделано в тексте процедуры.

Один из возможных вариантов процедуры **send\_trip** (можно предложить множество других, примерно, равноценных):

create or replace procedure send\_trip(cod\_t trips.cod\_trip%type, err out number)

as

tickets\_number number;

begin

select tickets into tickets\_number

from trips

where trips.cod\_trip = cod\_t;

if tickets\_number=0 then

err:=1;

else

insert into sent\_trips (id\_trip, cod\_trip, cod\_route, cod\_bus, cod\_model, hour, minute, week\_day, sent\_day, fill\_bus, tickets)

select sent\_trips\_seq.nextval, cod\_trip, cod\_route, cod\_bus, cod\_model, hour, minute, week\_day, sysdate, tickets/places\*100, tickets

from trips join buses using(cod\_bus) join models using(cod\_model)

where cod\_trip=cod\_t;

update trips set tickets = 0

where trips.cod\_trip = cod\_t;

err:=0;

end if;

exception

when NO\_DATA\_FOUND then

err:=2;

when OTHERS then

err:=3;

end;

Для проверки процедуры закроем (отправим) рейс с кодом 1

declare err number;

begin

send\_trip(1,err);

dbms\_output.put\_line('err=' || err);

end;

Можно проверить, что в таблице sent\_trips появилась строка об отправленном рейсе с сегодняшней датой:

select \* from sent\_trips

а количество проданных билетов в таблице trips обнулилось:

select \* from trips where cod\_trip=1

Эту процедуру также сдать преподавателю.

**3. Продолжайте работу над реализацией 2-3 хранимых процедур/функций для вашей индивидуальной базы данных. Ориентируйтесь на наиболее востребованные функции вашей предметной области. Обсудите этот вопрос с преподавателем, отладьте хранимый код в Apex и поместите в отчёт.**