

**Федеральное государственное автономное образовательное учреждение  
высшего профессионального образования  
«Национальный исследовательский ядерный университет «МИФИ»**

**Факультет «Информационная безопасность»**

Отчёт по

Лабораторной работе №2-3:

«SQL–аналитика»

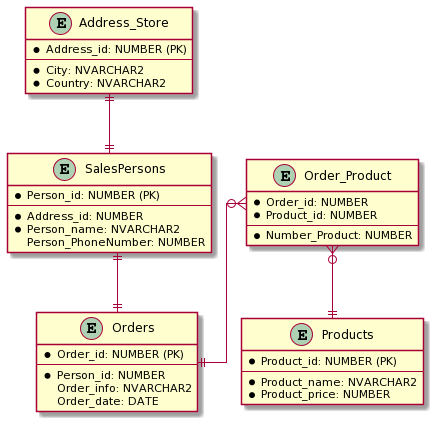
Выполнила: Чыонг Тхи Ан Хай

Группа: Б17-565

Преподаватель:  Куприяшин М.А.

Москва 2020

Поскольку база данных, использованная в предыдущих работах, не соответствует текущим лабораторным требованиям, создала новую базу данных:



Листинг использованных инструкций SQL в [1]

**Цель работы**

Изучение продвинутых аналитических возможностей Oracle Database.

**Ход работы**

***0***. ***Продемонстрировать обновление таблицы посредством операции слияния (инструкция MERGE).***

Использовать оператор MERGE для выбора строк из одного или нескольких источников для обновления или вставки в таблицу или представление. Можно указать условия, чтобы определить, обновлять или вставлять в целевую таблицу или представление.

Этот оператор - удобный способ объединения нескольких операций. Это позволяет избежать использования нескольких операторов DML INSERT, UPDATE и DELETE.

- Создайте таблицу бонусов с бонусом 50 по умолчанию для продавцов с более чем 2 заказами

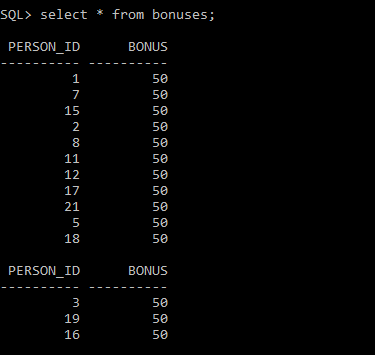
CREATE TABLE bonuses (Person\_id NUMBER (3, 0), bonus NUMBER DEFAULT 50);

INSERT INTO bonuses (Person\_id)

(SELECT Person\_id FROM

(SELECT o.Person\_id, COUNT (o.Person\_id) a FROM Orders o

GROUP by Person\_id) WHERE a>=2);



* Каждому продавцу, работающему в московском филиале, начисляется дополнительный бонус 25. Добавили и обновили в таблицу *bonuses*.

MERGE INTO bonuses b

USING (SELECT Person\_id, City FROM SalesPersons s, Address\_Store a

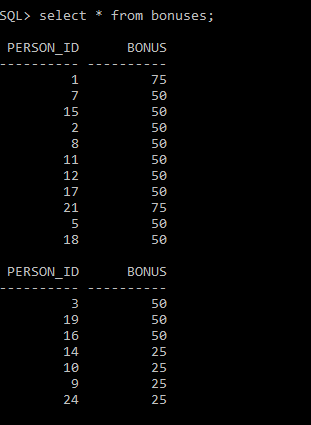
WHERE s.Address\_id = a.Address\_id and a.City = ’Moscow’) pa

ON (b.Person\_id = pa.Person\_id)

WHEN MATCHED THEN UPDATE SET b.bonus = b.bonus + 25

WHEN NOT MATCHED THEN INSERT (b.Person\_id, b.bonus)

VALUES (pa.Person\_id, 25);



***1. Спроектировать и составить отчёт по разработанной базе данных с промежуточными итогами на основе расширенной группировки GROUP BY ROLLUP;***

В дополнение к обычным результатам агрегирования, которые мы ожидаем от предложения GROUP BY, расширение ROLLUP производит групповые промежуточные итоги справа налево и общий итог. Если «n» - это количество столбцов, перечисленных в ROLLUP, будет n + 1 уровней промежуточных итогов.

Отчет о доходах от продаж по продуктам, (продуктам + странам) и всем:

SELECT p.Product\_id, a.Country, SUM (p.Product\_Price\*op.Number\_Product) Sum

FROM Products p

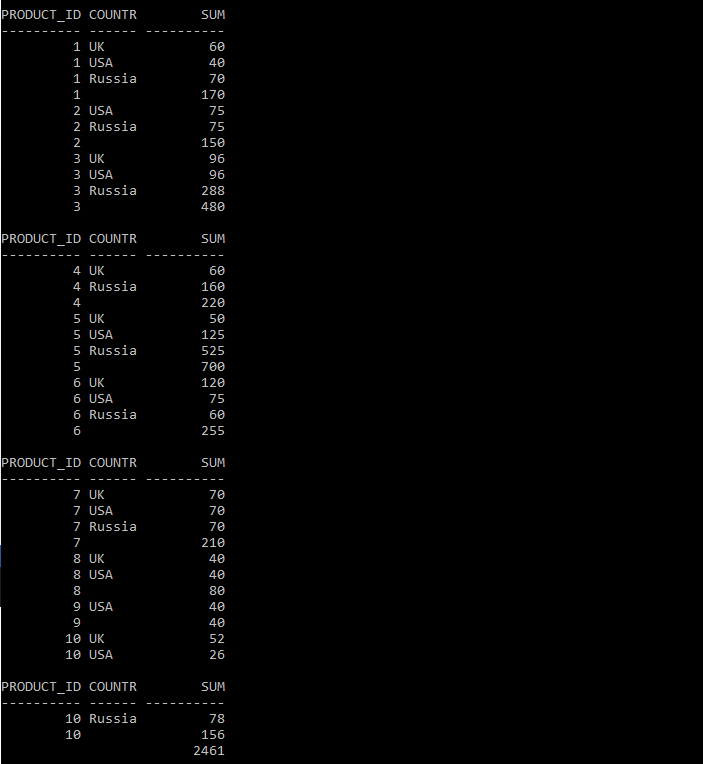
INNER JOIN Order\_Product op ON op.Product\_id = p.Product\_id

INNER JOIN Orders o ON op.Order\_id = o.Order\_id

INNER JOIN SalesPersons s ON o.Person\_id = s.Person\_id

INNER JOIN Address\_Store a ON a.Address\_id = s.Address\_id

GROUP BY ROLLUP (p.Product\_id, a.Country);



***2. Спроектировать и составить отчёт по разработанной базе данных на основе расширенной группировки «гиперкуб» (GROUP BY CUBE) или GROUPING SETS. На примере этого отчёта показать, как работают данные режимы группировки;***

- Предложение GROUPING SETS – это дальнейшее расширение предложения GROUP BY, которое можно использовать, чтобы задать несколько группирований данных.

* Отчет о доходах от продаж по (продуктам + странам) и (продуктам + продавцам)

SELECT s.Person\_id, p.Product\_id, a.Country, SUM (p.Product\_Price\*op.Number\_Product) Sum

FROM Products p

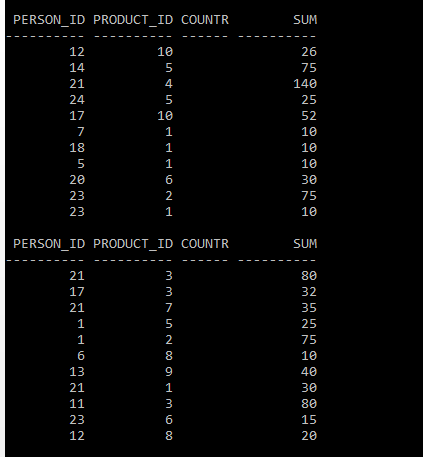
INNER JOIN Order\_Product op ON op.Product\_id = p.Product\_id

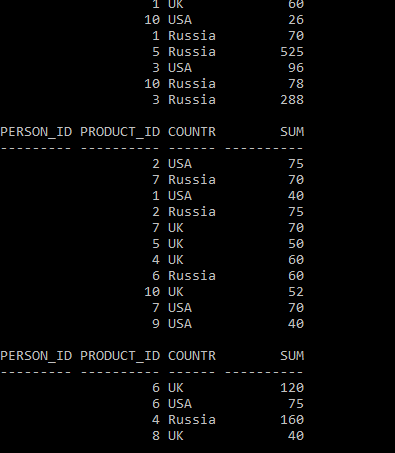
INNER JOIN Orders o ON op.Order\_id = o.Order\_id

INNER JOIN SalesPersons s ON o.Person\_id = s.Person\_id

INNER JOIN Address\_Store a ON a.Address\_id = s.Address\_id

GROUP BY GROUPING SETS ((p.Product\_id, a.Country), (p.Product\_id, s.Person\_id));





* Помимо промежуточных итогов, генерируемых расширением ROLLUP, расширение CUBE генерирует промежуточные итоги для всех комбинаций указанных измерений. Если «n» - это количество столбцов, перечисленных в CUBE, будет 2n промежуточных комбинаций.
* Отчет о доходах от продаж по продуктам, странам, продавцам, (продуктам + странам), (продуктам + продавцам), (странам + продавцам), (продуктам + странам + продавцам) и всем:

SELECT s.Person\_id, p.Product\_id, a.Country, SUM (p.Product\_Price\*op.Number\_Product) Sum

FROM Products p

INNER JOIN Order\_Product op ON op.Product\_id = p.Product\_id

INNER JOIN Orders o ON op.Order\_id = o.Order\_id

INNER JOIN SalesPersons s ON o.Person\_id = s.Person\_id

INNER JOIN Address\_Store a ON a.Address\_id = s.Address\_id

GROUP BY CUBE (p.Product\_id, a.Country, s.Person\_id);

***3. Спроектировать и составить матричный отчёт по разрабатываемой базе данных с применением оператора PIVOT. В случае, если в базе данных нет подходящей таблицы, можно создать её специально. Также допустимо реализовать вместо этого запрос на базе оператора UNPIVOT. На примере этого отчёта (запроса) показать, как работает этот оператор;***

Предложение Oracle PIVOT позволяет вам написать запрос кросс-табуляции, начиная с Oracle 11g. Это означает, что вы можете агрегировать результаты и вращать строки в столбцы.

Статистика количества каждого проданного продукта по странам:

SELECT \*

FROM (SELECT p.Product\_id, a.Country, SUM (op.Number\_Product) Qty

FROM Products p

INNER JOIN Order\_Product op ON op.Product\_id = p.Product\_id

INNER JOIN Orders o ON op.Order\_id = o.Order\_id

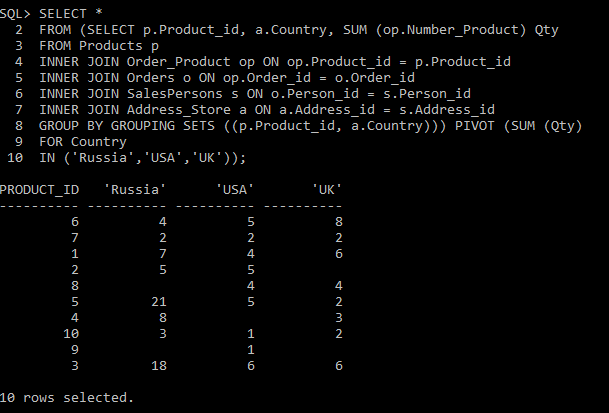
INNER JOIN SalesPersons s ON o.Person\_id = s.Person\_id

INNER JOIN Address\_Store a ON a.Address\_id = s.Address\_id

GROUP BY GROUPING SETS ((p.Product\_id, a.Country))) PIVOT (SUM (Qty)

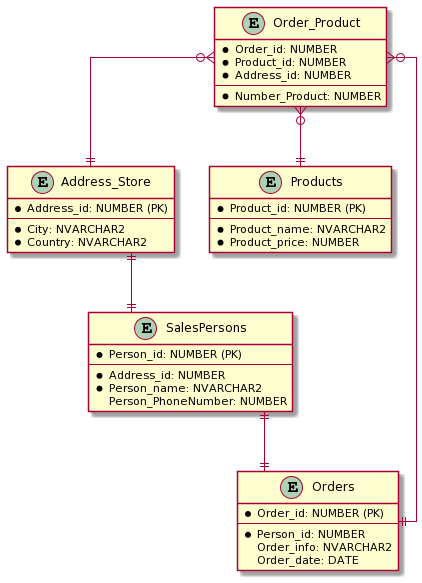
FOR Country

IN ('Russia','USA','UK'));



***4. Выделить в базе данных таблицы фактов (FACT TABLE) и таблицы измерений (DIMENSION TABLE). На основе этих таблиц создать измерения (CREATE DIMENSION) и иерархии. Использовать их для создания аналитического представления. Убедиться в работоспособности аналитического представления при помощи нескольких запросов.***

Добавить в таблицу Order\_Product: Address\_id



*Update info in table Order\_Product:*

MERGE INTO Order\_Product op

USING (SELECT s.Address\_id, o.Order\_id FROM SalesPersons s, Orders o

WHERE s.Person\_id = o.Person\_id) a

ON (op.Order\_id = a.Order\_id)

WHEN MATCHED THEN UPDATE SET op.Address\_id = a.Address\_id;

4.1 FACT TABLE and DIMENSION TABLE

|  |  |  |
| --- | --- | --- |
| Fact Table | Order\_Product | Она содержит первичные ключи измерений "Orders" и "Products", а также связанные с ними факты, такие как количество продукта. |
| Dimension Table | Address\_Store | Таблицы измерений предоставляют описательную информацию для всех измерений. |
| SalesPersons |
| Orders |
| Product |

4.2 Cоздать измерения (CREATE DIMENSION) и иерархии.

CREATE OR REPLACE ATTRIBUTE DIMENSION Address\_dim

DIMENSION TYPE STANDARD

USING Address\_Store

ATTRIBUTES (Address\_id, City, Country)

LEVEL Id

KEY Address\_id

DETERMINES (City)

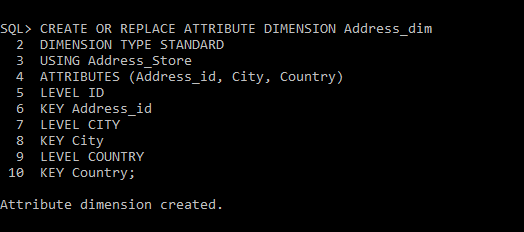
LEVEL CITY

KEY City

DETERMINES (Country)

LEVEL COUNTRY

KEY Country;



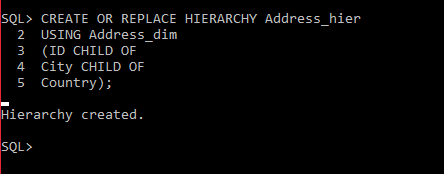
CREATE OR REPLACE HIERARCHY Address\_hier

USING Address\_dim

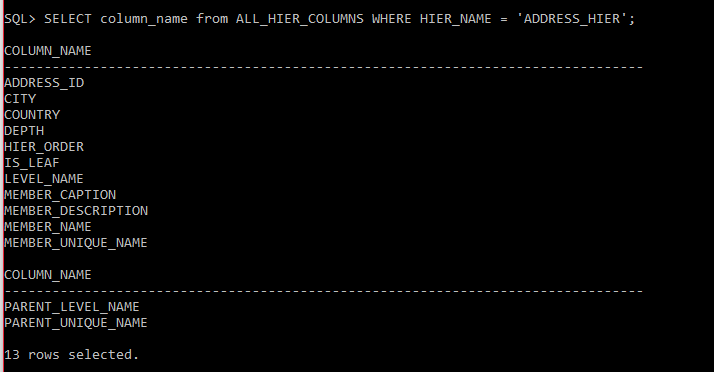
(Id CHILD OF

City CHILD OF

Country);



SELECT column\_name from ALL\_HIER\_COLUMNS WHERE HIER\_NAME = 'ADDRESS\_HIER';



4.3 Cоздания аналитического представления.

Аналитические представления обеспечивают быстрый и эффективный способ создания аналитических запросов к данным, хранящимся в существующих таблицах и представлениях базы данных.

Аналитические представления организуют данные с помощью размерной модели. Они позволяют легко добавлять агрегаты и вычисления к наборам данных и представлять данные в представлениях, которые можно запрашивать с помощью относительно простого SQL.

CREATE OR REPLACE ANALYTIC VIEW Order\_Product\_av

USING Order\_Product

DIMENSION BY (Address\_dim

KEY Address\_id REFERENCES Address\_id

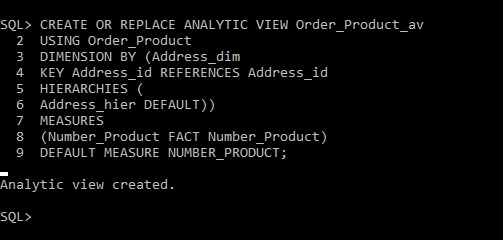
HIERARCHIES (

Address\_hier DEFAULT))

MEASURES

(Number\_Product FACT Number\_Product)

DEFAULT MEASURE NUMBER\_PRODUCT;



- Запросе аналитическое представление Order\_Product\_av включает единую иерархию

SELECT \* FROM Order\_Product\_av HIERARCHIES (Address\_hier)

ORDER BY Address\_hier.hier\_order;

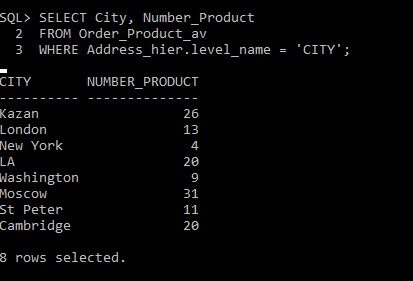


- Выбрать количество продуктов на уровне городов.

SELECT City, Number\_Product

FROM Order\_Product\_av

WHERE Address\_hier.level\_name = 'CITY';



**Залючение**

В данной работе:

* Продемонстрировали обновление таблицы посредством операции слияния (инструкция MERGE);
* Спроектировали и составили отчёт по разработанной базе данных с промежуточными итогами на основе расширенной группировки GROUP BY ROLLUP, GROUPING SETS, GROUP BY CUBE;
* Спроектировали и составили матричный отчёт по разрабатываемой базе данных с применением оператора PIVOT;
* Выделили в базе данных таблицы фактов (FACT TABLE) и таблицы измерений (DIMENSION TABLE). На основе этих таблиц создали измерения (CREATE DIMENSION) и иерархии, аналитическое представление

Ссылка:

[1] <https://github.com/TruongThiAnHai/SQL/tree/master/Lab2_3>