

## Referências Bibliográficas

- [Agrawal et al. 00] AGRAWAL, S.; CHAUDHURI, S. e NARASAYYA, V., **Automated selection of materialized views and indexes for sql databases**. Procs of the 26th VLDB International Conference, 2000, pp 496-505.
- [Ashedevi09] ASHEDEVI, B. BALASUBRAMANIAN, R. **Optimized Cost Effective Approach for Selection of Materialized Views in Data Warehousing**. Journal of Computer Science & Technology, volume 9, nº 1, 2009
- [Baril03] BARIL, X. BELLAHSÉNE, Z. **Selection of Materialized Views: A Cost-Based Approach**. Advanced Information Systems Engineering, Lecture Notes in Computer Science, Volume 2681/2003, 1031, 2003
- [Ceri91] CERI, S. WIDOM, J. **Deriving productions rules for incremental view maintenance**. Proceedings of the Seventeenth International Conference on Very Large Data Bases, 1991.
- [Gupta93] GUPTA, A. NUMICK, S. SUBRAHMANIAN, V. **Maintaining views incrementally**. Proceedings of the ACM SIGMOD international conference on Management of data, 1993
- [Harrison92] HARRISON, J. DIETRICH, S. **Maintenance of materialized view in a deductive database: An update propagation approach**. Em: Deductive Database Workshops,2.4, 1992
- [Horn01] HORN, P. **Automatic computing: IBM's perspective on the state of information technology**. Em: Computing Systems, volume 15, p. 1-40, 2001

- [Hose et al. 09] HOSE, K. KLAN, D. SATTLER, K. **Online Tuning of Aggregation Tables for OLAP**. Em: International Conference on Data Engineering - ICDE, p. 1679-1686, 2009
- [Karde10] KARDE, P. THAKARE, V. **An Efficient Materialized View Selection Approach for Query Processing in Database Management**. International Journal of Computer Science and Network Security (IJCSNS), volume.10 n° 9, 2010
- [MicrosoftTuning] **Database Engine Tuning Advisor Overview**. Acessado em 9 de março de 2011. Disponível em: <http://msdn.microsoft.com/en-us/library/ms173494.aspx>
- [Milanés04] MILANÉS, A., **Uma arquitetura para auto-sintonia global de SGBDs usando agentes**, Tese de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2004.
- [Milanés et al. 04] MILANÉS, A. LIFSCHITZ, S. SALLES, M. **Estado da Arte em Auto-Sintonia de Sistemas de Dados Relacionais**. Relatório Técnico. Pontifícia Universidade Católica do Rio de Janeiro. 2004.
- [Monteiro08]MONTEIRO, J. **Uma abordagem Não Intrusiva para a Manutenção Automática do Projeto Físico de Banco de Dados**. Tese de Doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2008.
- [Monteiro06] MONTEIRO, J., LIFSCHITZ, S. e BRAYNER, A. **Automated Selection of Materialized Views**. Monografia de Ciência da Computação, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2006.

- [Morelli et al. 09] MORELLI, E. MONTEIRO, J. M. ALMEIDA, A. C. LIFSCHITZ, S. **Reindexação Automática em SGBDs Relacionais. XXIV Simpósio Brasileiro de Banco de Dados. 2009.**
- [Morelli06a] MORELLI, E. M. T. **Recriação Automática de Índices em um SGBD Relacional.** Tese de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2006.
- [Morelli06b] MORELLI, E. T.; LIFSCHITZ, S. **Estudo dos malefícios gerados pela fragmentação de Índices em sistemas de banco de dados relacionais.** Relatório Técnico, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2006.
- [Navathe05] ELMASRI, R. NAVATHE, S.B. **Sistemas de Banco de Dados: Fundamentos e Aplicações.** Addison Wesley, 2005, 2.
- [OracleColumn] **Overview of Character Datatypes.** Acessado em 14 de janeiro de 2011. Disponível em: [http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28318/datatype.htm#i3253](http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/datatype.htm#i3253)
- [PerformanceSQLServer] **Improving Performance with SQL Server 2008 Indexed View.** Acessado em 20 de março de 2011. Disponível em: [http://msdn.microsoft.com/en-us/library/dd171921\(v=sql.100\).aspx](http://msdn.microsoft.com/en-us/library/dd171921(v=sql.100).aspx)
- [Ramakrishnan08] RAMAKRISHNAN, R. GEHRKE, J. **Sistemas de Gerenciamento de Banco de Dados.** 3ª Ed. 2008.
- [Salles04] SALLES, M. V. **Autonomic index creation in databases.** Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2004.
- [Silberschatz06] SILBERSCHATZ, A. **Sistema de Banco de Dados.** 5ª Ed. 2006

[SQLServerColumn] **Column Size**. Acessado em 14 de janeiro de 2011.  
Disponível em: [http://msdn.microsoft.com/en-us/library/ms711786\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms711786(v=vs.85).aspx)

[Weikum et al 02] WEIKUM, G.; MÖNKEBERG, A.; HASSE, C. ; ZABBACK, P.. **Self-tuning database technology and information services: from wishful thinking to viable engineering**. PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES (VLDB), p. 20{31, 2002.

[Weikum94] WEIKUM, G.; HASSE, C. MONKEBERG, A.; ZABBACK, P. **The COMFORT automatic tuning project, invited project review**. Information Systems, 19(5):381-432, 1994.

## **A**

### **Visões sobre parte de Consultas**

Com a finalidade de saber se visões que contem dados apenas de parte a consulta seria utilizado pelo otimizador SQL Server 2008. Ou seja, dada uma consulta que referencia mais de uma tabela e uma visão que tenha todos os dados referentes a um subconjunto de tabelas dessa consulta, não sobre o conjunto inteiro, queremos saber se o otimizador usaria essa visão para obter os dados da consulta mais rápido.

Para tanto listamos uma lista de consultas e visões sobre parte das consultas e verificamos o custo estimado que o otimizador, com o intuito de saber se ele usará ou não a visão.

Consultas utilizadas:

C1.

SELECT

C\_CUSTKEY,

C\_NAME,

SUM(L\_EXTENDEDPRISE \* (1 - L\_DISCOUNT)) AS REVENUE,

C\_ACCTBAL,

N\_NAME,

C\_ADDRESS,

C\_PHONE,

C\_COMMENT

FROM

CUSTOMER,

ORDERS,

LINEITEM,

NATION

WHERE

C\_CUSTKEY = O\_CUSTKEY

AND L\_ORDERKEY = O\_ORDERKEY

```
AND O_ORDERDATE >= '1992/08/01'
AND O_ORDERDATE < DATEADD(MONTH, 3, '1992/08/01')
AND L_RETURNFLAG = 'R'
AND C_NATIONKEY = N_NATIONKEY
GROUP BY
    C_CUSTKEY,
    C_NAME,
    C_ACCTBAL,
    C_PHONE,
    N_NAME,
    C_ADDRESS,
    C_COMMENT
ORDER BY
    REVENUE DESC;

C2.
SELECT
    L_ORDERKEY,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
    O_ORDERDATE,
    O_SHIPPRIORITY
FROM
    CUSTOMER,
    ORDERS,
    LINEITEM
WHERE
    C_MKTSEGMENT = 'AUTOMOBILE'
    AND C_CUSTKEY = O_CUSTKEY
    AND L_ORDERKEY = O_ORDERKEY
    AND O_ORDERDATE < '31/12/1998'
    AND L_SHIPDATE > '01/01/1991'
GROUP BY
    L_ORDERKEY,
    O_ORDERDATE,
```

```
O_SHIPRIORITY
ORDER BY
    REVENUE DESC,
    O_ORDERDATE;

C3.
SELECT
    N_NAME,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE
FROM
    CUSTOMER,
    ORDERS,
    LINEITEM,
    SUPPLIER,
    NATION,
    REGION
WHERE
    C_CUSTKEY = O_CUSTKEY
    AND L_ORDERKEY = O_ORDERKEY
    AND L_SUPPKEY = S_SUPPKEY
    AND C_NATIONKEY = S_NATIONKEY
    AND S_NATIONKEY = N_NATIONKEY
    AND N_REGIONKEY = R_REGIONKEY
    AND R_NAME = 'AMERICA'
    AND O_ORDERDATE >= '1991/08/01'
    AND O_ORDERDATE < DATEADD(YEAR, 1, '1991/08/01')

GROUP BY
    N_NAME
ORDER BY
    REVENUE DESC;

C4.
SELECT
    C_NAME,
    C_CUSTKEY,
```

```
O_ORDERKEY,  
O_ORDERDATE,  
O_TOTALPRICE,  
SUM(L_QUANTITY)  
FROM  
CUSTOMER,  
ORDERS,  
LINEITEM  
WHERE  
O_ORDERKEY IN (2, 5, 6, 7)  
AND C_CUSTKEY = O_CUSTKEY  
AND O_ORDERKEY = L_ORDERKEY  
GROUP BY  
C_NAME,  
C_CUSTKEY,  
O_ORDERKEY,  
O_ORDERDATE,  
O_TOTALPRICE  
ORDER BY  
O_TOTALPRICE DESC,  
O_ORDERDATE;  
  
C5.  
SELECT  
L_ORDERKEY,  
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE  
FROM  
ORDERS,  
LINEITEM  
WHERE  
L_ORDERKEY = O_ORDERKEY  
GROUP BY  
L_ORDERKEY  
ORDER BY
```



REVENUE DESC

C6.

```
SELECT
    L_ORDERKEY,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
    O_ORDERDATE,
    O_SHIPPRIORITY
FROM
    CUSTOMER,
    ORDERS,
    LINEITEM
WHERE
    C_CUSTKEY = O_CUSTKEY
    AND L_ORDERKEY = O_ORDERKEY
    AND O_ORDERDATE < '1998/12/31'
    AND L_SHIPDATE > '1991/01/01'
GROUP BY
    L_ORDERKEY,
    O_ORDERDATE,
    O_SHIPPRIORITY
ORDER BY
    REVENUE DESC,
    O_ORDERDATE;
```

Visões utilizadas:

V1.

```
CREATE VIEW [DBO].[VM_1] WITH SCHEMABINDING
AS
SELECT
    L_ORDERKEY,
    L_RETURNFLAG,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
```

```
        COUNT_BIG(*) AS CNT
FROM
        DBO.LINEITEM
GROUP BY
        L_ORDERKEY,
        L_RETURNFLAG
V2.
CREATE VIEW [DBO].[VM_2] WITH SCHEMABINDING AS
SELECT
        L_ORDERKEY,
        L_SHIPDATE,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
        COUNT_BIG(*) AS CNT
FROM
        DBO.LINEITEM
GROUP BY
        L_ORDERKEY,
        L_SHIPDATE
V3.
CREATE VIEW [DBO].[VM_3] WITH SCHEMABINDING
AS
SELECT
        L_ORDERKEY,
        L_SUPPKEY,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
        COUNT_BIG(*)AS CNT
FROM
        DBO.LINEITEM
GROUP BY
        L_ORDERKEY,
        L_SUPPKEY
V4.
CREATE VIEW [DBO].[VM_4] WITH SCHEMABINDING
```

```
AS
SELECT
    L_ORDERKEY,
    SUM(L_QUANTITY) AS SOM,
    COUNT_BIG(*) AS CNT
FROM
    DBO.LINEITEM
GROUP BY
    L_ORDERKEY
```

V5.

```
CREATE VIEW [DBO].[VM_5] WITH SCHEMABINDING
AS
SELECT
    L_ORDERKEY,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
    COUNT_BIG(*) AS CNT
FROM
    DBO.LINEITEM
GROUP BY
    L_ORDERKEY
```

V6.

```
CREATE VIEW [DBO].[VM_6] WITH SCHEMABINDING
AS
SELECT
    L_ORDERKEY,
    O_ORDERDATE,
    L_SHIPDATE,
    O_SHIPPRIORITY,
    O_CUSTKEY,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE,
    COUNT_BIG(*) AS CNT
FROM
    DBO.ORDERS,
    DBO.LINEITEM
```

WHERE

L\_ORDERKEY = O\_ORDERKEY

GROUP BY

L\_ORDERKEY,

O\_ORDERDATE,

L\_SHIPDATE,

O\_SHIPPRIORITY,

O\_CUSTKEY

A seguir mostramos uma tabela paralelizando as consultas que poderiam usar a visão, e qual o custo estimado do otimizador para a consulta com a visão materializada e sem a visão materializada.

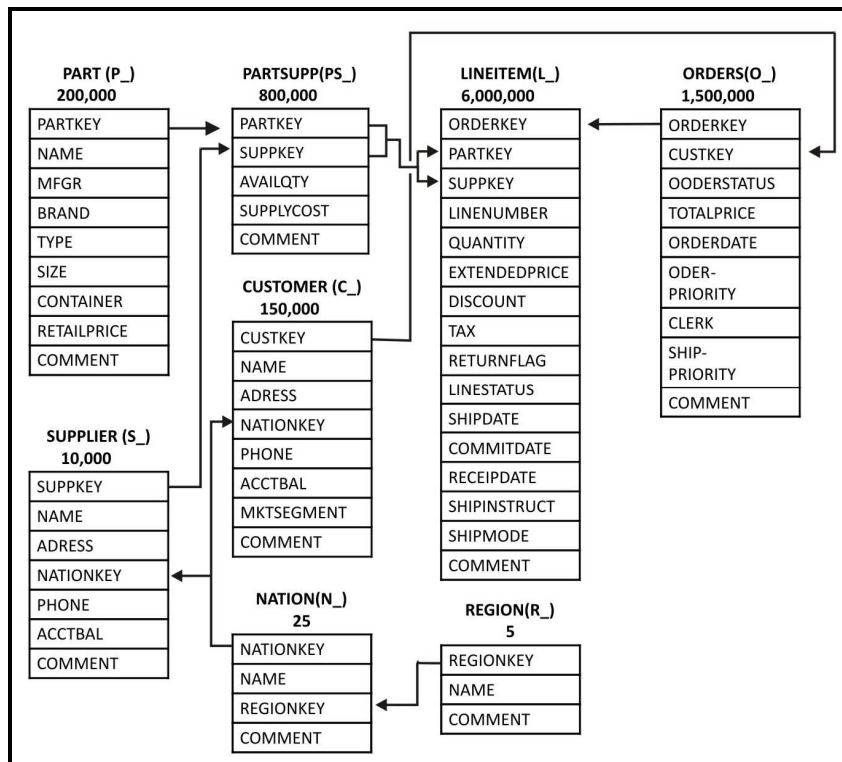
Consulta	Visão	Custo Sem Visão	Custo Com Visão
C1	V1	157.6	157.6
C2	V2	344.8	344.8
C3	V3	153.4	153.4
C4	V4	122.2	122.2
C5	V5	189.0	189.0
C6	V6	821.7	821.7

**Tabela A1 – Tabela de comparação de custo estimado utilizando visões sobre parte de consultas**

Como o conjunto de visões, sobre parte de consultas, testado não é utilizado, esse trabalho não criou visões sobre parte de consultas.

## B Benchmark TPC-H

Como descrito em [Monteiro08], o benchmark TPC-H é um benchmark de suporte a decisões que consiste em um conjunto de consultas *ad-hoc* voltadas para os negócios. As cargas de trabalho realizada nessa dissertação são submetidas a uma estrutura padrão de oito tabelas. A figura B.1(Morelli06b) ilustra o modelo do benchmark TPC-H.



**Figura B.124 - Modelo do TPC-H**

O número que aparece logo abaixo do nome da tabela representa sua cardinalidade. Estão representadas 5 regiões (continentes) que congregam vinte e cinco nações (tabelas *region* e *nation*, respectivamente). Clientes e Fornecedores (tabelas *supplier* e *customer*) estão associados às nações.

Enquanto os clientes realizam pedidos de compras (tabela *orders*), os fornecedores fornecem componentes (tabela *part*) de itens de compra. Como um fornecedor pode oferecer vários itens e um item pode ser disponibilizado por vários fornecedores, existe uma tabela para registrar esta relação  $N \times N$  (*partsupp*).

A tabela mais volumosa associa itens de compra a pedidos (tabela *lineitem*). As flechas indicam as associações entre chaves primárias e estrangeiras. As expressões entre parênteses significam os prefixos utilizados para denominar os campos da tabela em questão. Por exemplo: C\_CUSTKEY.

A base completa ocupa aproximadamente 1GB e os volumes de cada tabela são os que aparecem na figura B.1. A base de dados sofre acesso de um conjunto de consultas possuindo características *ad-hoc*, ou seja, não se conhecem nem a ordem de execução, nem os parâmetros de cada uma das 22 consultas.

### B.1 Consultas do TPC-H

Como mencionado anteriormente, o benchmark TPC-H é composto por um conjunto de consultas *ad-hoc* que simulam as atividades encontradas em um ambiente OLAP. Consideramos que a visão revenue (V1) apresentada a seguir existia no banco em todo o processo de teste dessa dissertação.

```
V1. CREATE VIEW REVENUE (SUPPLIER_NO, TOTAL_REVENUE)
AS
SELECT L_SUPPKEY,
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT))
FROM LINEITEM
WHERE L_SHIPDATE >= '1996/01/02' AND
      L_SHIPDATE < DATEADD(DAY,90,'1996/01/02')
GROUP BY L_SUPPKEY;
```

A seguir, representamos as consultas que compõem este benchmark:

```
1. SELECT L_RETURNFLAG, L_LINESTATUS,
SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS
SUM_DISC_PRICE,
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT) * (1 + L_TAX))
AS SUM_CHARGE,
AVG(L_QUANTITY) AS AVG_QTY,
```

```

    AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
    AVG(L_DISCOUNT) AS AVG_DISC, COUNT(*) AS
    COUNT_ORDER
FROM LINEITEM
WHERE L_SHIPDATE <= DATEADD(DAY, -10, '1998/12/01')
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;

```

```

2. SELECT S_ACCTBAL, S_NAME, N_NAME, P_PARTKEY,
    P_MFGR, S_ADDRESS, S_PHONE, S_COMMENT
FROM PART, SUPPLIER, PARTSUPP, NATION, REGION
WHERE P_PARTKEY = PS_PARTKEY AND
    S_SUPPKEY = PS_SUPPKEY
    AND P_SIZE = 20 AND P_TYPE LIKE '%COPPER' AND
    S_NATIONKEY = N_NATIONKEY AND
    N_REGIONKEY = R_REGIONKEY
    AND R_NAME = 'AMERICA' AND
    PS_SUPPLYCOST = ( SELECT MIN(PS_SUPPLYCOST)
        FROM PARTSUPP, SUPPLIER, NATION,
        REGION
        WHERE P_PARTKEY = PS_PARTKEY
        AND S_SUPPKEY = PS_SUPPKEY
        AND S_NATIONKEY =
        N_NATIONKEY AND
        N_REGIONKEY = R_REGIONKEY
        AND R_NAME = 'AMERICA' )
ORDER BY S_ACCTBAL DESC, N_NAME,
    S_NAME, P_PARTKEY;

```

```

3. SELECT L_ORDERKEY,
    SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS
    REVENUE, O_ORDERDATE, O_SHIPPRIORITY
FROM CUSTOMER, ORDERS, LINEITEM

```

```

WHERE C_MKTSEGMENT = 'AUTOMOBILE' AND
      C_CUSTKEY = O_CUSTKEY AND
      L_ORDERKEY = O_ORDERKEY
AND O_ORDERDATE < '1998/12/31' AND
      L_SHIPDATE > '1991/01/01'
GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY REVENUE DESC, O_ORDERDATE;

```

```

4. SELECT O_ORDERPRIORITY, COUNT(*) AS ORDER_COUNT
FROM ORDERS
WHERE O_ORDERDATE >= '1998/08/01' AND
      O_ORDERDATE < DATEADD(MONTH, 3, '1998/08/08') AND
      EXISTS ( SELECT L_ORDERKEY
                FROM LINEITEM
                WHERE L_ORDERKEY = O_ORDERKEY AND
                      L_COMMITDATE < L_RECEIPTDATE )
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY;

```

```

5. SELECT N_NAME,
       SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS
       REVENUE
FROM CUSTOMER, ORDERS, LINEITEM, SUPPLIER, NATION,
      REGION
WHERE C_CUSTKEY = O_CUSTKEY AND
      L_ORDERKEY = O_ORDERKEY
AND L_SUPPKEY = S_SUPPKEY AND
      C_NATIONKEY = S_NATIONKEY AND
      S_NATIONKEY = N_NATIONKEY AND
      N_REGIONKEY = R_REGIONKEY AND
      R_NAME = 'AMERICA' AND

      O_ORDERDATE >= '1991/08/01' AND

```



```
O_ORDERDATE < DATEADD(YEAR, 1, '1991/08/01')
GROUP BY N_NAME
ORDER BY REVENUE DESC;
```

```
6. SELECT SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS
    REVENUE
FROM LINEITEM
WHERE L_SHIPDATE >= '1998/07/01' AND
    L_SHIPDATE < DATEADD(YEAR, 1, '1998/07/01') AND
    L_DISCOUNT BETWEEN 2 - 0.01 AND 2 + 0.01 AND
    L_QUANTITY < 5;
```

```
7. SELECT SUPP_NATION, CUST_NATION, L_YEAR,
    SUM(VOLUME) AS REVENUE
FROM ( SELECT N1.N_NAME AS SUPP_NATION,
    N2.N_NAME AS CUST_NATION,
    YEAR(L_SHIPDATE) AS L_YEAR,
    L_EXTENDEDPRICE * (1 - L_DISCOUNT) AS
    VOLUME
FROM SUPPLIER, LINEITEM, ORDERS, CUSTOMER,
    NATION N1, NATION N2
WHERE S_SUPPKEY = L_SUPPKEY AND
    O_ORDERKEY = L_ORDERKEY AND
    C_CUSTKEY = O_CUSTKEY AND
    S_NATIONKEY = N1.N_NATIONKEY AND
    C_NATIONKEY = N2.N_NATIONKEY AND
    ((N1.N_NAME = 'ARGENTINA' AND
    N2.N_NAME = 'ARGENTINA') OR
    (N1.N_NAME = 'BRAZIL' AND
    N2.N_NAME = 'BRAZIL') ) AND
    L_SHIPDATE BETWEEN '1995/01/01' AND
    '1996/12/31' ) AS SHIPPING
GROUP BY SUPP_NATION, CUST_NATION, L_YEAR
```

ORDER BY SUPP\_NATION, CUST\_NATION, L\_YEAR;

8. SELECT O\_YEAR, SUM(CASE WHEN NATION =  
 'UNITED STATES'  
 THEN VOLUME ELSE 0 END) / SUM(VOLUME)  
 AS MKT\_SHARE  
 FROM ( SELECT YEAR(O\_ORDERDATE) AS O\_YEAR,  
 L\_EXTENDEDPRI \* (1 - L\_DISCOUNT)  
 AS VOLUME,  
 N2.N\_NAME AS NATION  
 FROM PART, SUPPLIER, LINEITEM,  
 ORDERS, CUSTOMER,  
 NATION N1, NATION N2, REGION  
 WHERE P\_PARTKEY = L\_PARTKEY AND  
 S\_SUPPKEY = L\_SUPPKEY AND  
 L\_ORDERKEY = O\_ORDERKEY AND  
 O\_CUSTKEY = C\_CUSTKEY AND  
 C\_NATIONKEY = N1.N\_NATIONKEY AND  
 N1.N\_REGIONKEY = R\_REGIONKEY AND  
 R\_NAME = 'AFRICA' AND  
 S\_NATIONKEY = N2.N\_NATIONKEY AND  
 O\_ORDERDATE BETWEEN '1995/01/01' AND  
 '1996/12/31' AND  
 P\_TYPE = 'ECONOMY BRUSHED COPPER') AS  
 ALL\_NATIONS  
 GROUP BY O\_YEAR  
 ORDER BY O\_YEAR;
9. SELECT NATION, O\_YEAR, SUM(AMOUNT) AS SUM\_PROFIT  
 FROM ( SELECT N\_NAME AS NATION,  
 YEAR(O\_ORDERDATE) AS O\_YEAR,  
 L\_EXTENDEDPRI \* (1 - L\_DISCOUNT) -  
 PS\_SUPPLYCOST \* L\_QUANTITY AS AMOUNT

```

FROM PART, SUPPLIER, LINEITEM, PARTSUPP,
     ORDERS, NATION
WHERE S_SUPPKEY = L_SUPPKEY AND
     PS_SUPPKEY = L_SUPPKEY AND
     PS_PARTKEY = L_PARTKEY AND
     P_PARTKEY = L_PARTKEY
     AND O_ORDERKEY = L_ORDERKEY AND
     S_NATIONKEY = N_NATIONKEY AND
     P_NAME LIKE '%BLUSH%' ) AS PROFIT
GROUP BY NATION, O_YEAR
ORDER BY NATION, O_YEAR DESC;

```

```

10. SELECT C_CUSTKEY, C_NAME,
         SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT))
         AS REVENUE,
         C_ACCTBAL, N_NAME, C_ADDRESS, C_PHONE,
         C_COMMENT
FROM CUSTOMER, ORDERS, LINEITEM, NATION
WHERE C_CUSTKEY = O_CUSTKEY AND
     L_ORDERKEY = O_ORDERKEY
     AND O_ORDERDATE >= '1992/08/01' AND
     O_ORDERDATE < DATEADD(MONTH, 3, '1992/08/01') AND
     L_RETURNFLAG = 'R' AND
     C_NATIONKEY = N_NATIONKEY
GROUP BY C_CUSTKEY, C_NAME, C_ACCTBAL,
         C_PHONE, N_NAME,
         C_ADDRESS, C_COMMENT
ORDER BY REVENUE DESC;

```

```

11. SELECT PS_PARTKEY,
         SUM(PS_SUPPLYCOST * PS_AVAILQTY) AS VALUE
FROM PARTSUPP, SUPPLIER, NATION
WHERE PS_SUPPKEY = S_SUPPKEY AND

```

```

S_NATIONKEY = N_NATIONKEY
AND N_NAME = 'BRAZIL'
GROUP BY PS_PARTKEY
HAVING SUM(PS_SUPPLYCOST * PS_AVAILQTY) >
(SELECT SUM(PS_SUPPLYCOST * PS_AVAILQTY) * 2
FROM PARTSUPP, SUPPLIER, NATION
WHERE PS_SUPPKEY = S_SUPPKEY AND
S_NATIONKEY = N_NATIONKEY AND
N_NAME = 'BRAZIL' )
ORDER BY VALUE DESC;

```

```

12. SELECT L_SHIPMODE, SUM(CASE WHEN
O_ORDERPRIORITY =
'1-URGENT' OR O_ORDERPRIORITY = '2-HIGH' THEN
1 ELSE 0 END) AS HIGH_LINE_COUNT,
SUM(CASE WHEN O_ORDERPRIORITY <> '1-URGENT'
AND O_ORDERPRIORITY <> '2-HIGH'
THEN 1 ELSE 0 END) AS LOW_LINE_COUNT
FROM ORDERS, LINEITEM
WHERE O_ORDERKEY = L_ORDERKEY AND
L_SHIPMODE IN ('TRUCK', 'AIR')
AND L_COMMITDATE < L_RECEIPTDATE AND
L_SHIPDATE < L_COMMITDATE AND
L_RECEIPTDATE >= '1996/01/01' AND
L_RECEIPTDATE < DATEADD(YEAR,1,'1996/01/01')
GROUP BY L_SHIPMODE ORDER BY L_SHIPMODE;

```

```

13. SELECT C_COUNT, COUNT(*) AS CUSTDIST
FROM ( SELECT C_CUSTKEY, COUNT(O_ORDERKEY)
FROM CUSTOMER LEFT OUTER JOIN ORDERS ON
C_CUSTKEY = O_CUSTKEY AND
O_COMMENT NOT LIKE '%EVEN%DEPOSITS%'
GROUP BY C_CUSTKEY )

```

```
        AS C_ORDERS (C_CUSTKEY, C_COUNT)
GROUP BY C_COUNT
ORDER BY CUSTDIST DESC, C_COUNT DESC;
```

```
14. SELECT 100.00 * SUM(CASE WHEN P_TYPE LIKE 'PROMO%'
        THEN L_EXTENDEDPRICE * (1 - L_DISCOUNT) ELSE 0 END)
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT))
        AS PROMO_REVENUE
FROM LINEITEM, PART
WHERE L_PARTKEY = P_PARTKEY AND
        L_SHIPDATE >= '1996/01/02' AND
        L_SHIPDATE < DATEADD(MONTH,1,'1996/01/02');
```

```
15. SELECT S_SUPPKEY, S_NAME, S_ADDRESS,
        S_PHONE, TOTAL_REVENUE
FROM SUPPLIER, REVENUE
WHERE S_SUPPKEY = SUPPLIER_NO AND
        TOTAL_REVENUE = ( SELECT MAX(TOTAL_REVENUE)
        FROM REVENUE )
ORDER BY S_SUPPKEY;
```

```
16. SELECT P_BRAND, P_TYPE, P_SIZE,
        COUNT(DISTINCT PS_SUPPKEY) AS SUPPLIER_CNT
FROM PARTSUPP, PART
WHERE P_PARTKEY = PS_PARTKEY AND P_BRAND <>
'BRAND#13' AND
        P_TYPE NOT LIKE 'STANDARD%' AND
        P_SIZE IN (7, 12, 14, 16, 21, 23, 32, 43) AND
        PS_SUPPKEY NOT IN ( SELECT S_SUPPKEY
FROM SUPPLIER
```

```

WHERE S_COMMENT LIKE '%CUSTOMER%COMPLAINTS%' )
GROUP BY P_BRAND, P_TYPE, P_SIZE ORDER BY
SUPPLIER_CNT DESC, P_BRAND, P_TYPE, P_SIZE;

```

```

17. SELECT SUM(L_EXTENDEDPRICE) / 7.0 AS AVG_YEARLY
FROM LINEITEM, PART
WHERE P_PARTKEY = L_PARTKEY AND
      P_BRAND = 'BRAND#13' AND
      P_CONTAINER = 'JUMBO PKG' AND
      L_QUANTITY < ( SELECT 0.2 * AVG(L_QUANTITY)
                     FROM LINEITEM
                     WHERE L_PARTKEY = P_PARTKEY );

```

```

18. SELECT C_NAME, C_CUSTKEY, O_ORDERKEY,
          O_ORDERDATE, O_TOTALPRICE, SUM(L_QUANTITY)
FROM CUSTOMER, ORDERS, LINEITEM
WHERE O_ORDERKEY IN ( SELECT L_ORDERKEY
                      FROM LINEITEM
                      GROUP BY L_ORDERKEY
                      HAVING SUM(L_QUANTITY) > 3 ) AND
      C_CUSTKEY = O_CUSTKEY AND
      O_ORDERKEY = L_ORDERKEY
GROUP BY C_NAME, C_CUSTKEY, O_ORDERKEY,
          O_ORDERDATE, O_TOTALPRICE
ORDER BY O_TOTALPRICE DESC, O_ORDERDATE;

```

```

19. SELECT SUM(L_EXTENDEDPRICE* (1 - L_DISCOUNT))
      AS REVENUE
FROM LINEITEM, PART
WHERE ( P_PARTKEY = L_PARTKEY AND
      P_BRAND = 'BRAND#13' AND
      P_CONTAINER IN ('SM CASE', 'SM BOX', 'SM PACK',

```

```
'SM PKG') AND L_QUANTITY >= 4 AND
L_QUANTITY <= 14 AND
P_SIZE BETWEEN 1 AND 5 AND
L_SHIPMODE IN ('AIR', 'AIR REG') AND
L_SHIPINSTRUCT = 'DELIVER IN PERSON' ) OR
( P_PARTKEY = L_PARTKEY AND P_BRAND =
'BRAND#44' AND
P_CONTAINER IN ('MED BAG', 'MED BOX', 'MED PKG',
'MED PACK') AND L_QUANTITY >= 5 AND
L_QUANTITY <= 15
AND P_SIZE BETWEEN 1 AND 10 AND
L_SHIPMODE IN ('AIR', 'AIR REG') AND
L_SHIPINSTRUCT = 'DELIVER IN PERSON' ) OR
( P_PARTKEY = L_PARTKEY AND P_BRAND =
'BRAND#53' AND
P_CONTAINER IN ('LG CASE', 'LG BOX', 'LG PACK',
'LG PKG') AND L_QUANTITY >= 6 AND
L_QUANTITY <= 16 AND
P_SIZE BETWEEN 1 AND 15 AND
L_SHIPMODE IN ('AIR', 'AIR REG') AND
L_SHIPINSTRUCT = 'DELIVER IN PERSON' );
```

```
20. SELECT S_NAME, S_ADDRESS
FROM SUPPLIER, NATION
WHERE S_SUPPKEY IN
( SELECT DISTINCT (PS_SUPPKEY)
FROM PARTSUPP, PART
WHERE PS_PARTKEY=P_PARTKEY AND
P_NAME LIKE 'DIM%' AND
PS_AVAILQTY >
( SELECT 0.5 * SUM(L_QUANTITY)
FROM LINEITEM
WHERE L_PARTKEY = PS_PARTKEY
```

```

        AND L_SUPPKEY = PS_SUPPKEY
        AND L_SHIPDATE >= '1997/01/03'
        AND L_SHIPDATE <
        DATEADD(YEAR,1,'1997/01/03'))
        AND S_NATIONKEY = N_NATIONKEY
        AND N_NAME = 'ARGENTINA'
ORDER BY S_NAME;

```

```

21. SELECT S_NAME, COUNT(*) AS NUMWAIT
FROM SUPPLIER, LINEITEM L1, ORDERS, NATION
WHERE S_SUPPKEY = L1.L_SUPPKEY AND
      O_ORDERKEY = L1.L_ORDERKEY AND
      O_ORDERSTATUS = 'F' AND
      L1.L_RECEIPTDATE > L1.L_COMMITDATE AND
      EXISTS ( SELECT *
                FROM LINEITEM L2
                WHERE L2.L_ORDERKEY = L1.L_ORDERKEY AND
                      L2.L_SUPPKEY <> L1.L_SUPPKEY )
      AND NOT EXISTS ( SELECT *
                       FROM LINEITEM L3
                       WHERE L3.L_ORDERKEY = L1.L_ORDERKEY AND
                              L3.L_SUPPKEY <> L1.L_SUPPKEY AND
                              L3.L_RECEIPTDATE > L3.L_COMMITDATE )
      AND S_NATIONKEY = N_NATIONKEY
      AND N_NAME = 'BRAZIL'
GROUP BY S_NAME
ORDER BY NUMWAIT DESC, S_NAME;

```

```

22. SELECT CNTRYCODE, COUNT(*) AS NUMCUST,
      SUM(C_ACCTBAL) AS TOTACCTBAL
FROM ( SELECT SUBSTRING(C_PHONE, 1, 2) AS CNTRYCODE,
        C_ACCTBAL
      FROM CUSTOMER

```



```
WHERE SUBSTRING(C_PHONE, 1, 2) IN
      ('25', '11', '13', '14', '30', '23', '18') AND
      C_ACCTBAL > ( SELECT AVG(C_ACCTBAL)
                    FROM CUSTOMER
                    WHERE C_ACCTBAL > 0.00 AND
                          SUBSTRING(C_PHONE, 1, 2)
                          IN ('25', '11', '13', '14',
                              '30', '23', '18'))
      AND NOT EXISTS ( SELECT *
                       FROM ORDERS
                       WHERE O_CUSTKEY =
                             C_CUSTKEY)
) AS VIP
GROUP BY CNTRYCODE
ORDER BY CNTRYCODE;
```

## C

### Cargas de trabalho submetidas

As cargas de trabalho submetidas ao banco de dados do TPC-H foram sequências de consultas do TPC-H (apêndice B) escolhidas aleatoriamente. Para tanto, foi utilizado um programa em Java para sortear aleatoriamente as consultas que seriam submetidas. A seguir estão as sequências das consultas que foram utilizadas para teste nessa dissertação.

- Primeira carga

12, 14, 17, 1, 16, 7, 9, 6, 5, 17, 4, 6, 8, 7, 6, 3, 11, 14, 13, 4, 21, 15, 7, 3, 6, 12, 16, 4, 15, 15, 15, 16, 16, 2, 16, 8, 10, 12, 9, 21, 14, 10, 10, 2, 19, 7, 13, 22, 15, 10, 6, 7, 11, 11, 2, 11, 12, 17, 1, 1, 7, 14, 12, 8, 3, 22, 17, 4, 22, 7, 11, 19, 12, 5, 14, 4, 12, 15, 14, 5, 1, 20, 19, 6, 1, 19, 22, 19, 10, 17, 7, 7, 11, 20, 9, 2, 16, 5, 16, 9, 6, 17, 14, 9, 22, 1, 9, 2, 20, 3, 18, 20, 1, 7, 8, 3, 3, 22, 16, 15, 17, 20, 2, 6, 22, 11, 15, 13, 3, 4, 8, 9, 5, 20, 1, 9, 19, 10, 14, 5, 2, 7, 15, 11, 3, 4, 4, 6, 3, 9, 8, 6, 18, 18, 4, 18, 8, 5, 11, 22, 12, 14, 11, 3, 13, 19, 13, 3, 3, 15, 8, 20, 20, 22, 4, 6, 5, 21, 18, 19, 4, 11, 22, 7, 11, 1, 1, 11, 14, 18, 8, 1, 8, 12, 17, 8, 22, 22, 10, 5, 6, 11, 20, 17, 14, 6, 8, 17, 3, 6, 6, 2, 7, 11, 11, 9, 8, 15, 15, 8, 10, 9, 15, 4, 21, 12, 9, 6, 22, 16, 4, 21, 2, 17, 12, 6, 11, 3, 9, 19, 15, 6, 5, 20, 20, 13, 8, 5, 2, 6, 13, 6, 17, 4, 4, 4, 9, 5, 7, 2, 19, 18, 10, 19, 19, 14, 12, 12, 7, 11, 18, 16, 8, 4, 5, 16, 3, 7, 5, 6, 21, 20, 3, 2, 6, 9, 21, 14, 22, 20, 13, 7, 22

- Segunda carga

16, 10, 14, 22, 21, 14, 9, 21, 20, 20, 19, 4, 1, 1, 13, 6, 8, 7, 19, 3, 12, 3, 13, 1, 16, 17, 6, 20, 20, 19, 6, 19, 1, 13, 1, 12, 22, 4, 22, 21, 21, 19, 7, 15, 16, 7, 13, 18, 18, 9, 22, 19, 2, 19, 12, 3, 16, 13, 5, 21, 12, 9, 14, 18, 7, 4, 11, 10, 10, 13, 15, 5, 11, 3, 16, 8, 11, 21, 20, 11, 15, 16, 5, 10, 22, 7, 10, 15, 3, 13, 3, 14, 5, 19, 17, 20, 5, 3, 10, 16, 6, 13, 20, 9, 6, 12, 19, 3, 13, 4, 6, 20, 14, 11, 16, 2, 5, 10, 22, 5, 15, 1, 12, 8, 10, 19, 4, 19, 18, 7, 15, 1, 7, 5, 10, 8, 1, 11, 14, 21, 12, 21, 20, 16, 19, 9, 18, 11, 2, 3, 4, 1, 9, 5, 5, 1, 7, 10, 4, 14, 9, 2, 5, 12, 16, 16, 22, 6, 3, 10, 6, 13, 3, 18, 13, 16, 11, 11, 10, 20, 11, 10, 13, 20, 20, 2, 18, 13, 3, 3, 8, 17, 14, 1, 7, 6, 18, 14, 15, 15, 15, 9, 18, 17, 14, 17, 11,

15, 9, 12, 18, 13, 9, 16, 19, 19, 18, 17, 19, 13, 5, 15, 7, 1, 8, 21, 21,  
 15, 2, 3, 7, 5, 16, 13, 15, 12, 5, 11, 12, 14, 6, 19, 12, 10, 12, 6, 1, 8,  
 20, 17, 6, 11, 21, 19, 12, 11, 13, 22, 8, 12, 15, 19, 14, 22, 15, 16, 19,  
 21, 9, 19, 5, 11, 22, 21, 17, 2, 6, 19, 19, 13, 22, 17, 16, 17, 16, 12,  
 19, 7, 3, 7, 21, 22, 18, 14, 18, 20, 3,  
 4, 17, 21, 2, 13, 4, 18, 18, 17, 16, 20, 19, 5, 14, 1, 9, 10, 13, 5, 22,  
 22, 14, 15, 2, 16, 11, 17, 14, 14, 4, 13, 17, 11, 20, 8, 14, 3, 9, 18, 16,  
 22

- Terceira carga

17, 4, 6, 8, 7, 6, 3, 11, 14, 13, 12, 14, 17, 1, 16, 7, 9, 6, 5, 4, 21, 15,  
 7, 3, 6, 12, 16, 4, 15, 15, 15, 16, 16, 2, 16, 8, 10, 12, 9, 21, 14, 10,  
 10, 2, 19, 7, 13, 22, 15, 10, 6, 7, 11, 11, 2, 11, 12, 17, 1, 1, 7, 14, 12,  
 8, 3, 22, 17, 4, 22, 7, 11, 19, 12, 5, 14, 4, 12, 15, 14, 5, 1, 20, 19, 6,  
 1, 19, 22, 19, 10, 17, 7, 7, 11, 20, 9, 2, 16, 5, 16, 9, 6, 17, 14, 9, 22,  
 1, 9, 2, 20, 3, 18, 20, 1, 7, 8, 3, 3, 22, 16, 15, 17, 20, 2, 6, 22, 11, 15,  
 13, 3, 4, 8, 9, 5, 20, 1, 9, 19, 10, 14, 5, 2, 7, 15, 11, 3, 4, 4, 6, 3, 9, 8,  
 6, 18, 18, 4, 18, 8, 5, 11, 22, 12, 14, 11, 3, 13, 19, 13, 3, 3, 15, 8, 20,  
 20, 22, 4, 6, 5, 21, 18, 19, 4, 11, 22, 7, 11, 1, 1, 11, 14, 18, 8, 1, 8,  
 12, 17, 8, 22, 22, 10, 5, 6, 11, 20, 17, 14, 6, 8, 17, 3, 6, 6, 2, 7, 11,  
 11, 9, 8, 15, 15, 8, 10, 9, 15, 4, 21, 12, 9, 6, 22, 16, 4, 21, 2, 17, 6,  
 21, 20, 3, 2, 6, 9, 21, 14, 22, 20, 13, 7, 22, 12, 6, 11, 3, 9, 19, 15, 6,  
 5, 20, 20, 13, 8, 5, 2, 6, 13, 6, 17, 4,  
 4, 4, 9, 5, 7, 2, 19, 18, 10, 19, 19, 14, 12, 12, 7, 11, 18, 16, 8, 4, 5,  
 16, 3, 7, 5, 17, 4, 6, 8, 7, 6, 3, 11, 14, 13, 12, 14, 17, 1, 16, 7, 9, 6, 5,  
 4, 21, 15, 7, 3, 6, 12, 16, 4, 15, 15, 15, 16, 16, 2, 16, 8, 10, 12, 9, 21,  
 14, 10, 10, 2, 19, 7, 13, 22, 15, 10, 6, 7, 11, 11, 2, 11, 12, 17, 1, 1, 7,  
 14, 12, 8, 3, 22, 17, 4, 22, 7, 11, 19, 12, 5, 14, 4, 12, 15, 14, 5, 1, 20,  
 19, 6, 1, 19, 22, 19, 10, 17, 7, 7, 11, 20, 9, 2, 16, 5, 16, 9, 6, 17, 14,  
 9, 22, 1, 9, 2, 20, 3, 18, 20, 1, 7, 8, 3, 3, 22, 16, 15, 17, 20, 2, 6, 22,  
 11, 15, 13, 3, 4, 8, 9, 5, 20, 1, 9, 19, 10, 14, 5, 2, 7, 15, 11, 3, 4, 4, 6,  
 3, 9, 8, 6, 18, 18, 4, 18, 8, 5, 11, 22, 12, 14, 11, 3, 13, 19, 13, 3, 3,  
 15, 8, 20, 20, 22, 4, 6, 5, 21, 18, 19, 4, 11, 22, 7, 11, 1, 1, 11, 14, 18,  
 8, 1, 8, 12, 17, 8, 22, 22, 10, 5, 6, 11, 20, 17, 14, 6, 8, 17, 3, 6, 6, 2,  
 7, 11, 11, 9, 8, 15, 15, 8, 10, 9, 15, 4, 21, 12, 9, 6, 22, 16, 4, 21, 2,  
 17, 6, 21, 20, 3, 2, 6, 9, 21, 14, 22, 20, 13, 7, 22, 12, 6, 11, 3, 9, 19,  
 15, 6, 5, 20, 20, 13, 8, 5, 2, 6, 13, 6, 17, 4, 4, 4, 9, 5, 7, 2, 19, 18, 10,  
 19, 19, 14, 12, 12, 7, 11, 18, 16, 8, 4, 5, 16, 3, 7, 5

## D

### Visões Sugeridas

Este apêndice apresenta todas as visões que foram geradas através do componente que implementa a heurística de seleção de visões hipotéticas e as visões que foram sugeridas pela ferramenta *Database Engine Tuning Advisor*.

```
V1. CREATE VIEW MV_1 WITH SCHEMABINDING
AS
SELECT L_PARTKEY, SUM(L_EXTENDEDPRICE),
       SUM(L_QUANTITY), COUNT_BIG(*)
FROM LINEITEM
GROUP BY L_PARTKEY
```

```
V2. CREATE VIEW MV_2 WITH SCHEMABINDING
AS
SELECT L_PARTKEY, L_SUPPKEY, SUM(L_QUANTITY),
       COUNT_BIG(*)
FROM LINEITEM
WHERE ( L_SHIPDATE >= '01/03/1997' AND
       L_SHIPDATE < '01/03/1998')
GROUP BY L_PARTKEY, L_SUPPKEY
```

```
V3. CREATE VIEW MV_3 WITH SCHEMABINDING
AS
SELECT L_ORDERKEY, SUM(L_QUANTITY), COUNT_BIG(*)
FROM LINEITEM
GROUP BY L_ORDERKEY
```

```
V4. CREATE VIEW MV_4 WITH SCHEMABINDING
AS
```

```
SELECT NATION.N_NAME , PARTSUPP.PS_PARTKEY ,
       SUM( PS_SUPPLYCOST*PS_AVAILQTY) , COUNT_BIG(*)
FROM SUPPLIER, PARTSUPP, NATION
WHERE S_SUPPKEY = PS_SUPPKEY AND
       S_NATIONKEY = N_NATIONKEY
GROUP BY N_NAME, PS_PARTKEY
```

V5. CREATE VIEW MV\_5 WITH SCHEMABINDING

```
AS
SELECT O_CUSTKEY , COUNT_BIG(*)
FROM ORDERS
GROUP BY O_CUSTKEY
```

V6. CREATE VIEW MV\_6 WITH SCHEMABINDING

```
AS
SELECT S_SUPPKEY , S_COMMENT , COUNT_BIG(*)
FROM SUPPLIER
GROUP BY S_SUPPKEY, S_COMMENT
```

V7. CREATE VIEW MV\_7 WITH SCHEMABINDING

```
AS
SELECT L_RETURNFLAG, L_LINESTATUS, L_SHIPDATE,
       SUM(L_QUANTITY), SUM(L_EXTENDEDPRICE),
       SUM(L_EXTENDEDPRICE * ( 1 - L_DISCOUNT ) ),
       SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)*(1+ L_TAX )),
       SUM(L_DISCOUNT), COUNT_BIG(*)
FROM LINEITEM
GROUP BY L_RETURNFLAG, L_LINESTATUS, L_SHIPDATE
```

V8. CREATE VIEW MV\_8 WITH SCHEMABINDING

```
AS
SELECT L_PARTKEY, SUM(L_QUANTITY), COUNT_BIG(*)
FROM LINEITEM
GROUP BY L_PARTKEY
```