

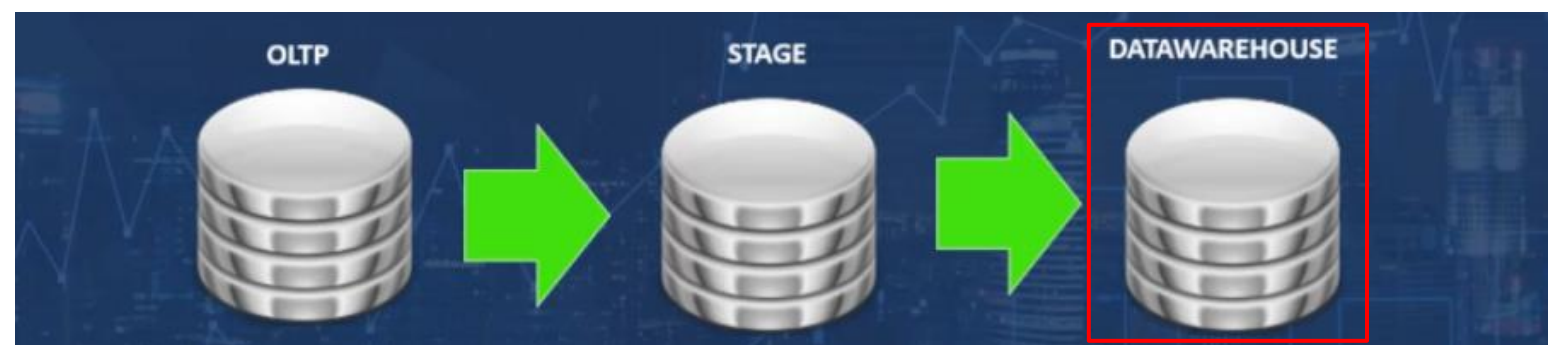
DATA WAREHOUSE – GOIÁS MARKET

O *Data Warehouse*, ou DW, é uma das partes mais importantes do projeto de *Business Intelligence*. Aqui, culmina-se todos os esforços de cargas e transformações de dados – ETL – necessários para armazenamento de dados concentrando-as no formato correto para análises, construção de *dashboards* e *reports* necessários para os analistas, gestores e tomadores de decisões no nível gerencial.

A partir do Data Warehouse, assim como os próximos projetos, irão tratar do ambiente OLAP, destinado à análises e automação de relatórios.

Um rápido paralelo aos conceitos de modelagem de um DW, pode-se citar os *Data Mart*, que são bancos de dados com a mesma função do DW, porém, com um assunto (ou Fato) específico. Como exemplo, *Data Mart* destinados ao Marketing, ou Logística, ou Financeiro, ou Comercial, etc.

Neste projeto, não serão modelados *Data Mart*. Apenas o *Data Warehouse* que irá armazenar todos os dados do projeto.



Portanto, após as etapas de ETL do sistema OLTP para a *STAGE AREA* e, em seguida, para o *Data Warehouse*, os dados já estão prontos para ingestão e análise.

Como será definido mais adiante, é importante ressaltar que o DW também possui a função de guardar o histórico de modificações das entradas do OLTP.

Uma vez que algumas informações podem ser modificadas, tal como endereço, local de residência, telefone, sexo,

OBJETIVO DO PROJETO – DATA WAREHOUSE

O projeto do *Data Warehouse* tem como objetivo o armazenamento dos dados oriundos do sistema transacional do negócio (banco OLTP), a fim de diminuir o tráfego de informações que causam lentidões, e disponibilizar esses dados de forma estruturada para serem analisados e interpretados pelos analistas e tomadores de decisões.

Para este projeto, o objetivo do DW visa responder as seguintes questões de negócio:

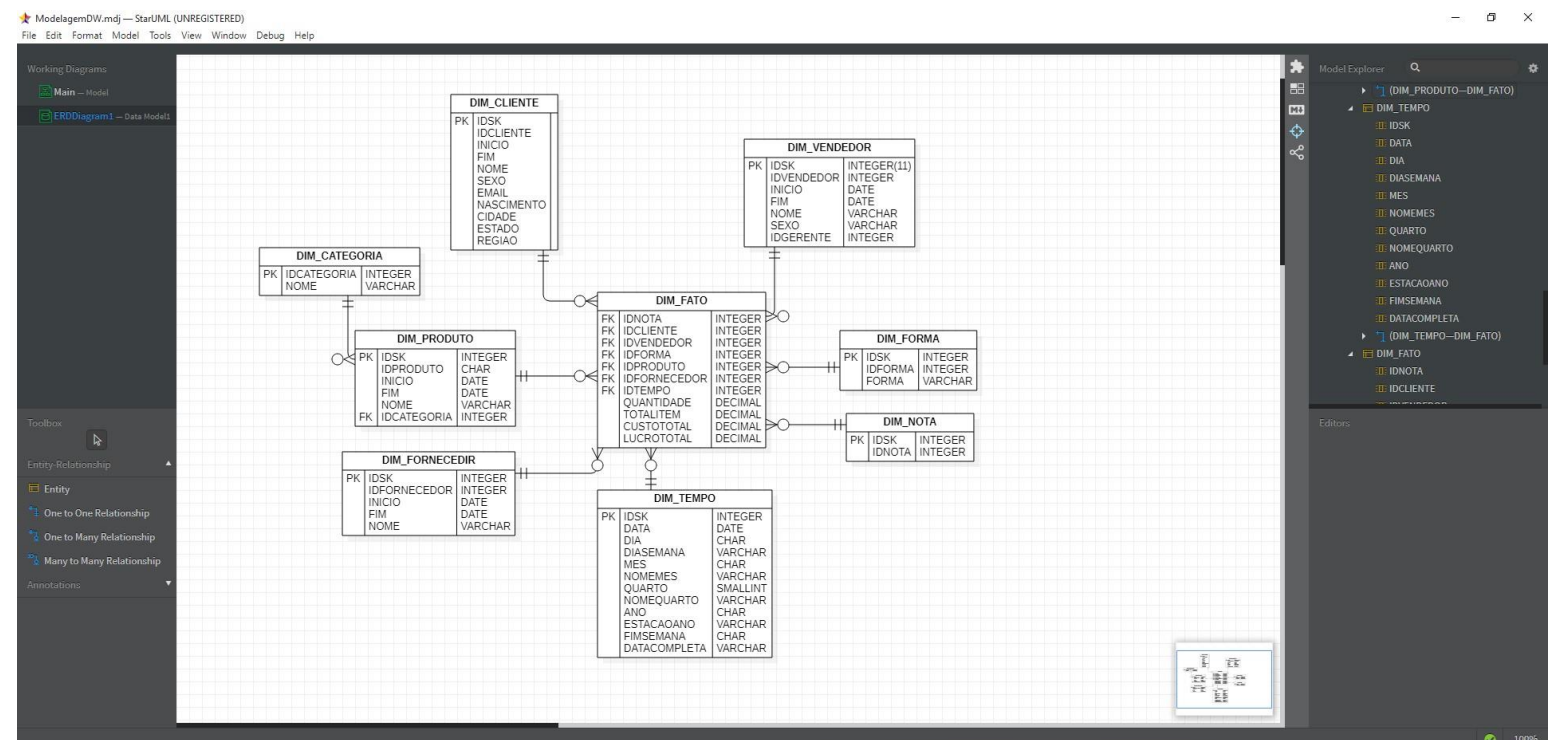
- Quem são os melhores clientes?
- Quem são os melhores vendedores?
- Qual categoria rende mais?
- Qual a minha relação com os fornecedores?
- Qual meu pior e melhor produto?
- Em qual região eu vendo mais?

MODELAGEM LÓGICA

A modelagem do *Data Warehouse* possui grande foco na integração das tabelas. A integridade relacional deve ser garantida a fim de que todos os dados estejam conectados corretamente através das *Primary Key* (PK, ou chave primária) e *Surrogate Key* (SK, ou chave substituta).

No modelo a seguir, pode-se destacar a tabela DIM_FATO, que é a tabela principal do DW. Essa tabela poderia ser facilmente escrita por um Query SQL, trazendo todos os dados na seleção. Entretanto, a tabela FATO consolida a função do *Data Warehouse*, facilitando a consulta de dados.

A tabela FATO, como demonstrado abaixo, recebe os relacionamentos de cardinalidade de 1 para muitos (1 x N). Isto significa que a tabela fato recebe os registros únicos de cada tabela relacionada, bem como o seu histórico de modificação, consolidando as informações de uma PK em um único lugar.



Os dados armazenados no *Data Warehouse* já estão prontos para consultas pelos analistas, sejam elas através de *queries* diretamente do banco de dados, por queries SQL, por exportação para um sistema de análise em ambiente OLAP (como o SQL Server Analysis Services) ou por uma conexão com alguma ferramenta de *DataViz* (como o Microsoft Power BI, Tableau, Qlik Sense e etc).

Nesse ponto, é válido também a conexão com ferramentas de Data Science, como R e Python, que são capazes de acessar diretamente o DW através de bibliotecas específicas para conexão e interface com SQL.

Os pipelines em R e Python serão modelados futuramente neste mesmo projeto, para modelagem de dados, aplicação de modelos de *Machine Learning* e *Data Mining*.



Ao modelar o DW, há um novo atributo nas tabelas em relação ao banco OLTP e a *Stage Area*, são as colunas “INICIO” e “FIM”, do tipo *DATE*. Estas colunas serão responsáveis por manter o registro de alteração de algum item.

O conceito que se aplica em tabelas com capacidade de armazenar histórico por alguma alteração, são definidas como dimensões mutáveis, ou SCD (*Slowly Changing Dimensions*)

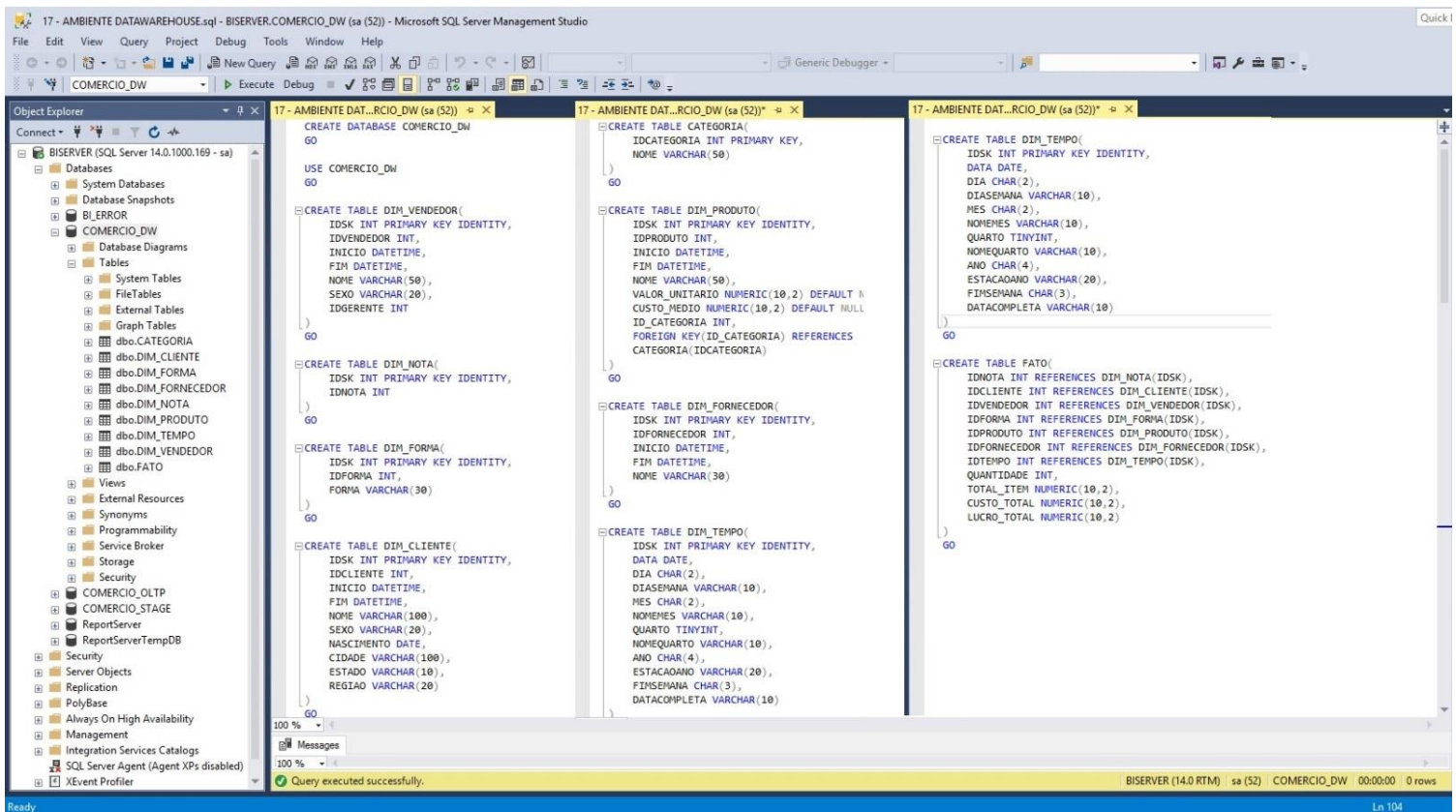
Dessa forma, é possível guardar um registro de alteração de um item ao longo tempo sem perder a rastreabilidade, pois a chave primeira (PK) se manterá igual, e os outros atributos que serão alterados terão uma data de validade, representada na coluna “FIM”, que é a data em que foi feita a alteração e este item será substituído pela sua alteração.

Feito isso, cria-se uma nova entrada no DW com a mesma *Primary Key* (PK), porém com uma *Surrogate Key* (SK) diferente, já com as devidas alterações.

As tabelas que podem sofrer esse tipo de alteração, são: DIM_PRODUTO, DIM_CLIENTE, DIM_FORNECEDOR E DIM_VENDEDOR. Todas essas tabelas podem sofrer alteração em alguma de suas entradas mediante a necessidade e serem salvas no *Data Warehouse* durante o processo de ETL. O *Integration Services* será responsável por esse processo e será exemplificado a seguir.

Com os conceitos já definidos e a modelagem lógica feita, partimos então para a modelagem física do *Data Warehouse*.

Começamos então a modelagem física no banco de dados, pelo SQL Server, com a criação do banco de dados do *Data Warehouse*. Neste projeto, o DW será chamado de COMERCIO_DW.



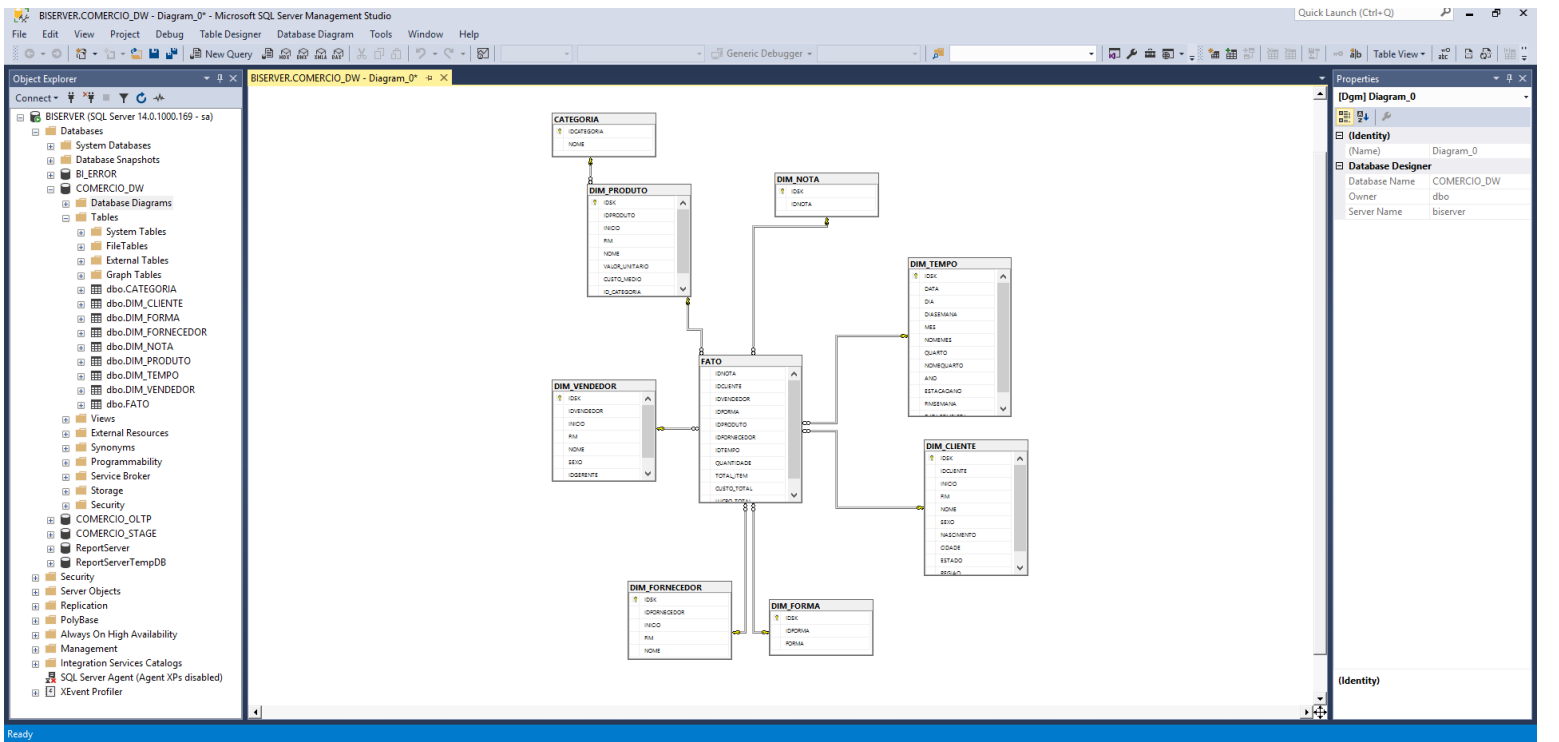
As tabelas foram criadas com o prefixo “DIM_”, indicando as dimensões do DW separadas por assunto.

As desse banco foram criadas em uma ordem específica já com os seus relacionamentos (ou *Constraints*) definidos, interligando as chaves primárias de cada uma. Apesar de que o procedimento de criação de *constraints* possa ser feito a parte, foram aplicados mediante a criação das tabelas para evitar qualquer fuga de relacionamentos.

A tabela DIM_FATO, ilustra bem a modelagem de *constraints* pois é a tabela mais importante do DW, que necessita estar totalmente referenciada com as dimensões que a compõe.

A função SQL utilizada para referenciar as tabelas durante a criação, segue na última *query*:
REFERENCES

Neste momento, a banco já está criado, como segue o diagrama gerado pelo SQL Server, e as tabelas já estão prontas para receber os dados por ETL.

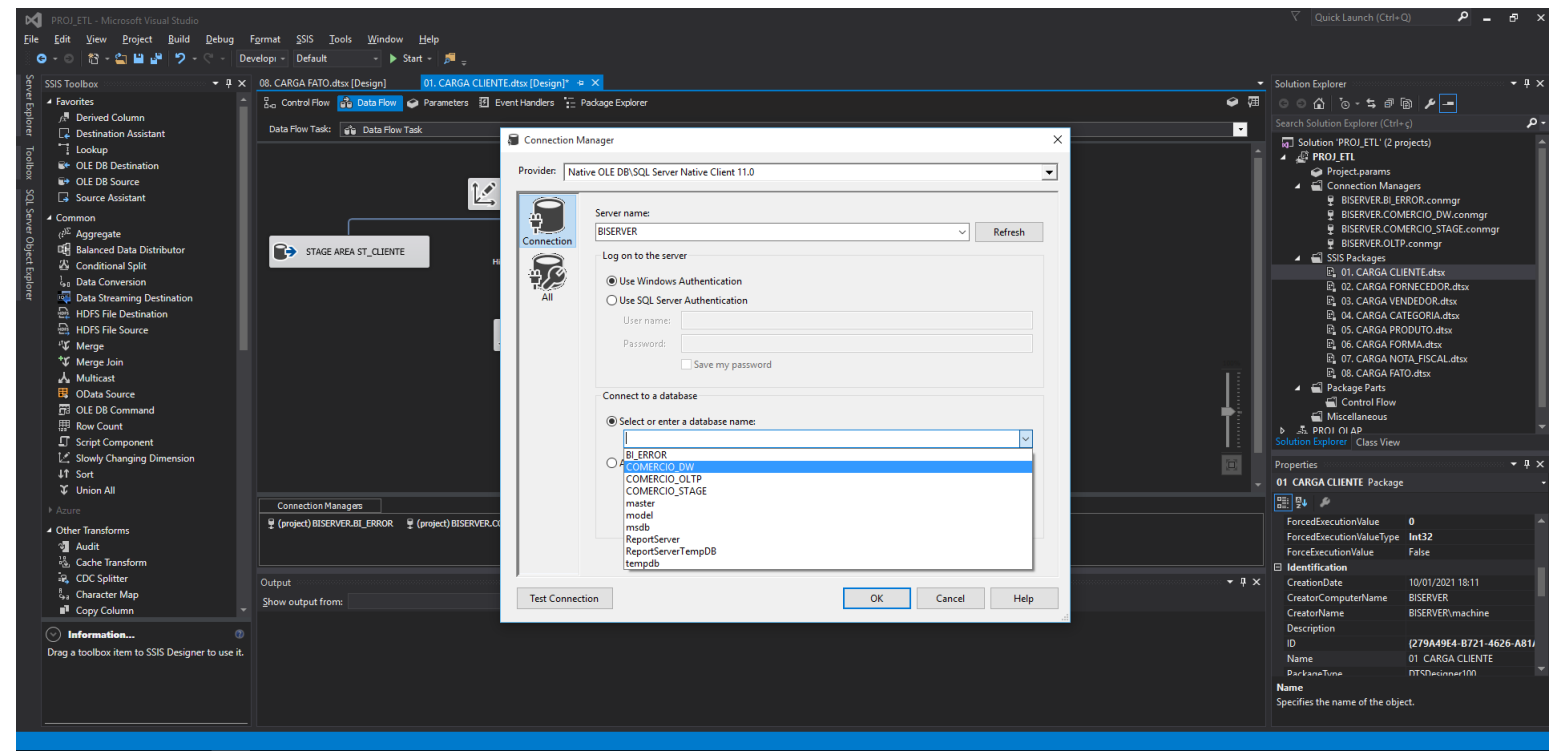


SINCRONIZAÇÃO DE ETL COM INTEGRATION SERVICES

O *Integration Service* (SSIS) é uma versátil ferramenta de ETL, construindo os pacotes de cargas e os gerenciando por projetos e por solução, como segue no menu à direita.

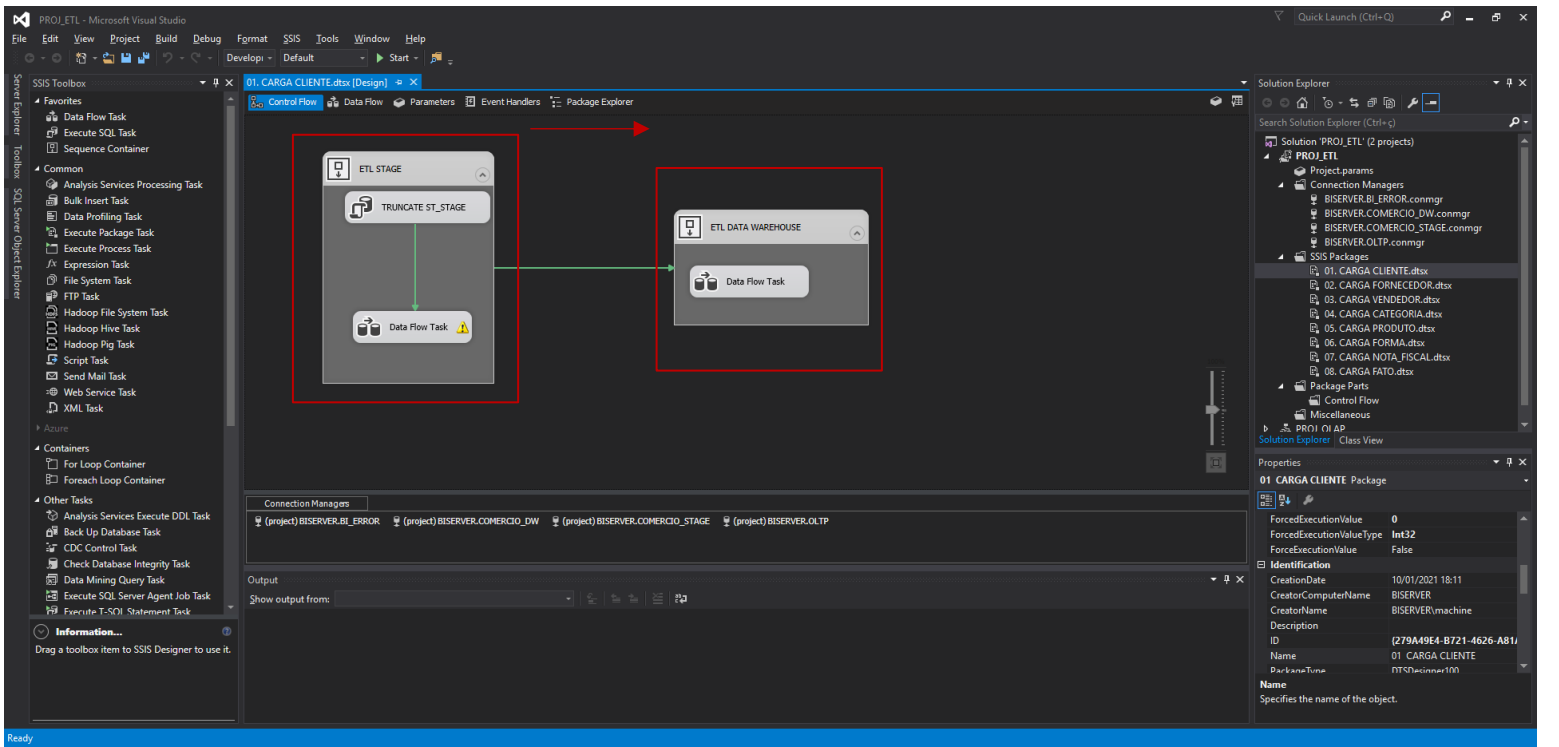
Nesse projeto, os pacotes terão o processo de ETL sincronizados com a carga da Stage Area. Isso significa que o pacote irá fazer separadamente e consequentemente as cargas dos bancos.

As conexões entre os bancos foram definidas previamente e foram compartilhadas como conexões de toda essa solução BI. Neste momento, é criada a conexão com o banco COMERCIO_DW, que é o *Data Warehouse*.



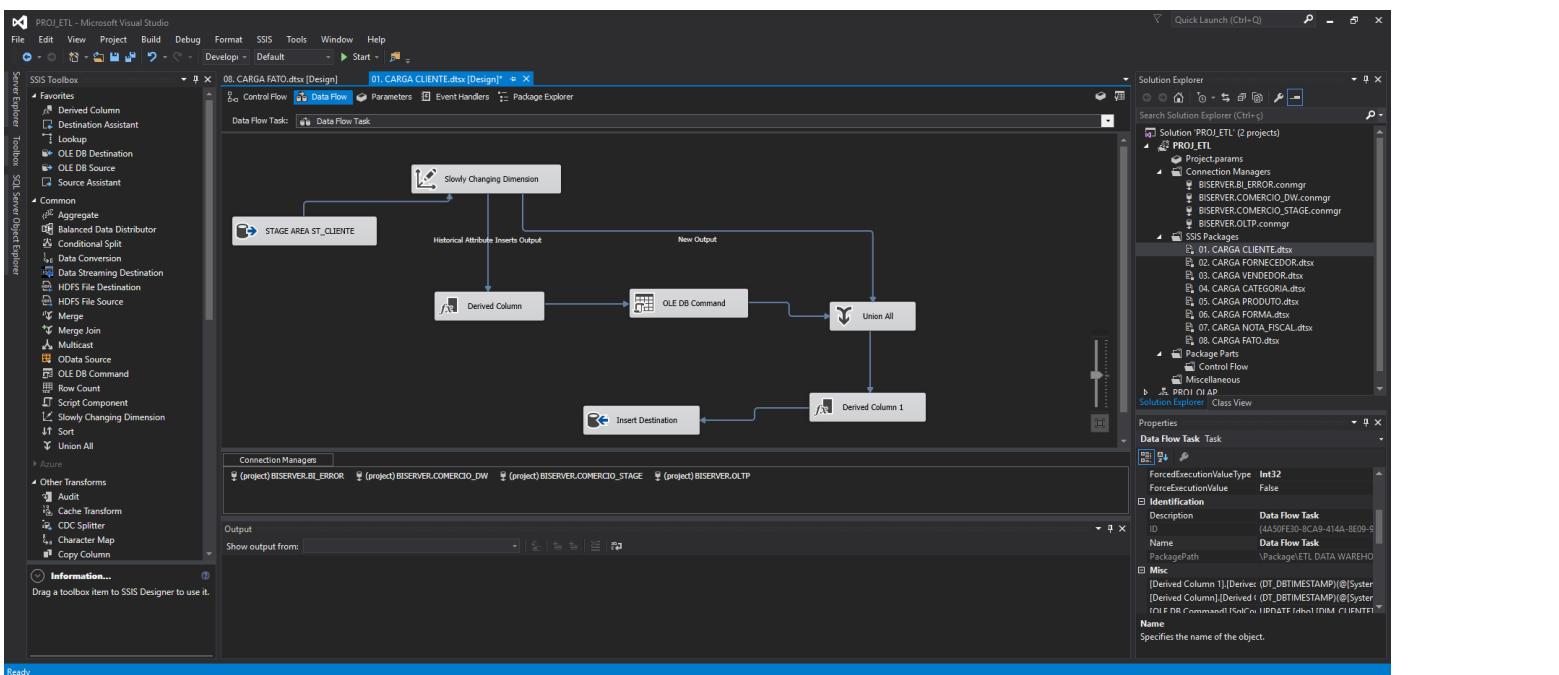
Como mostra a *screenshot* abaixo, os processos foram separados por uma ferramenta chamada ***Sequence Container***. Essa ferramenta é capaz de separar os processos de ETL da área de Stage, que é feita no primeiro momento, para então, depois de finalizado, fazer a carga do *Data Warehouse*.

Este processo faz todo sentido uma vez que a *Stage Area* serve apenas para carga e transformações necessárias dos dados. Então, ao finalizar a carga da *Stage* pelo pacote ‘ETL_STAGE’, inicia-se o container seguinte, ‘ETL DATA WAREHOUSE’, com a carga do DW.

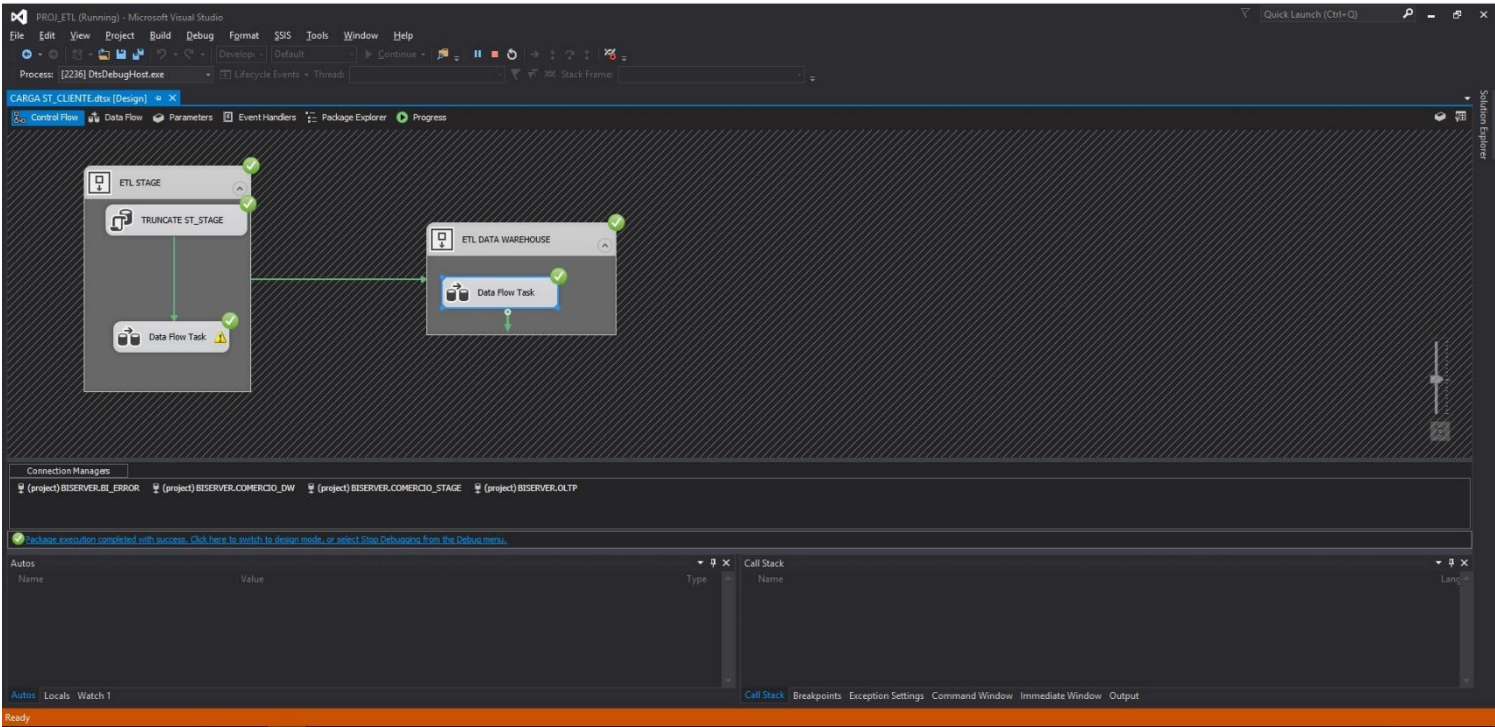


O processo interno de ETL do pacote do *Data Warehouse*, ou ***Data Flow***, possui um processo de carga um pouco diferente das demais, uma vez que esta carga se baseia em SCD (Slowly Changing Dimensions ou dimensões mutáveis) para garantir o armazenamento do histórico.

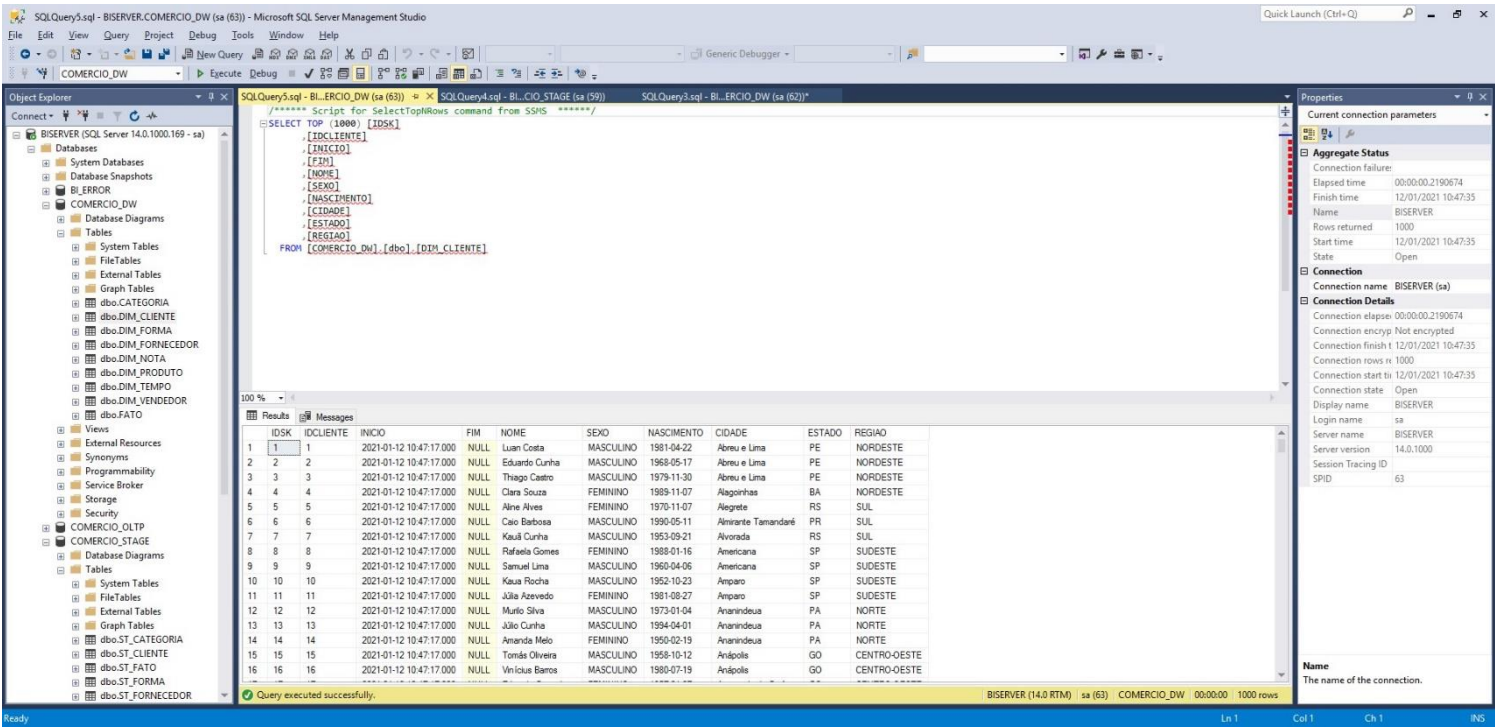
É adicionado uma ferramenta ao pacote chamada ‘Slowly Changing Dimensions’, encontrada no menu de transformações comuns do SSIS. O assistente da ferramenta define as opções para a tabela e cria automaticamente o *Data Flow*, já conectado ao source de dados (‘STAGE AREA ST_CLIENTE’).



Ao finalizar o processo de ETL, podemos ver a confirmação de êxito do processo, como na *screenshot* abaixo:



Ao finalizar o pacote do SSIS, podemos consultar no SQL Server a carga dos dados, realizando um SELECT simples na tabela DIM_CLIENTE



Nesse ponto, podemos ver o *Data Warehouse* já disponível para queries e conexão com as ferramentas de BI. Claro, após a carga de todas as outras tabelas.

O processo de ETL demonstrado aqui se repete igualmente para todas as outras dimensões. Portanto, não será demonstrado aqui todas as tabelas, uma vez que o processo é igual.

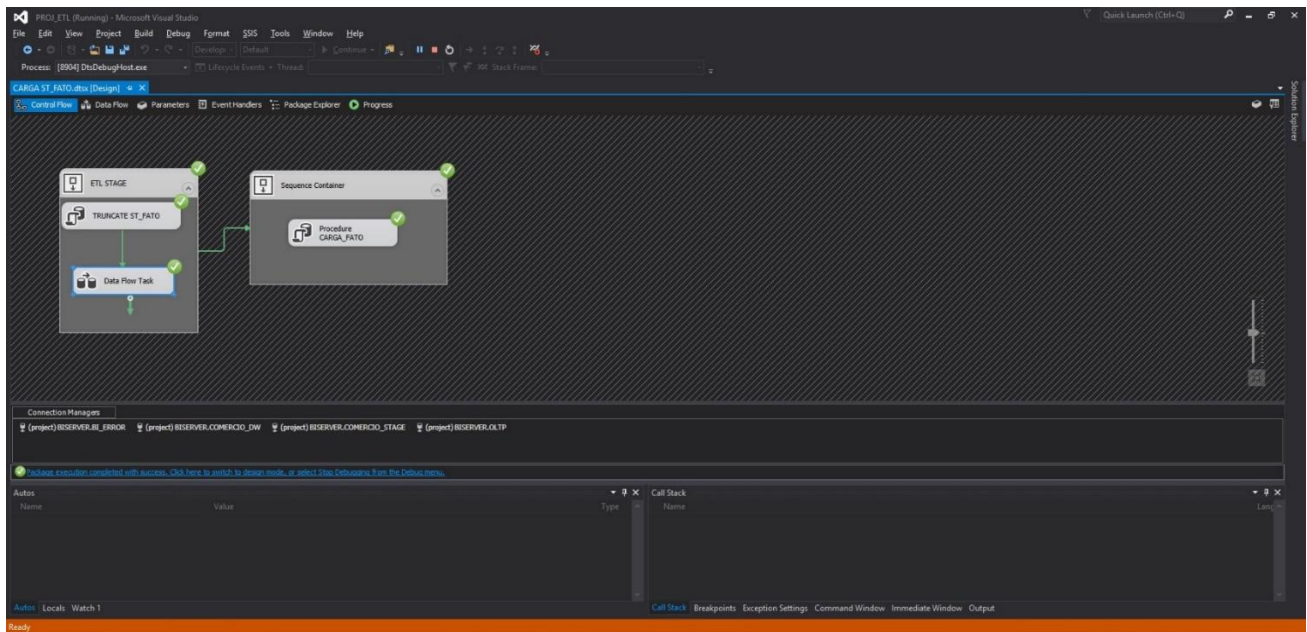
CARGA DA TABELA DIM_FATO

Por final, carregamos a tabela fato, que é a consolidação do Data Warehouse.

A tabela DIM_FATO é carregada utilizando um script em SQL que é capaz de filtrar os dados já existentes, comparando as entradas já existentes no DW com os novos resultados. Dessa forma, o ETL carrega apenas os novos dados inseridos no sistema OLTP.

Por motivos de privacidade, o script que foi fornecido pelo instrutor Felipe Mafra, não será descrito aqui nesse projeto. Entretanto, este pode ser obtido ao adquirir um de seus cursos na plataforma Udemy. O mesmo motivo é válido também para a DIM_TEMPO, que foi criada separadamente do SQL Server por um script de inteligência de data.

Os scripts, de forma genial, podem ser replicados em outros projetos de forma inteiramente funcional.



Após a conclusão do processo de carga da DIM_FATO, podemos verificar os dados através de uma query de consulta.

IDNOTA	IDCLIENTE	IDVENDEDOR	IDFORMA	IDPRODUTO	IDFORNECEDOR	IDTEMPO	QUANTIDADE	TOTAL_ITEM	CUSTO_TOTAL	LUCRO_TOTAL
1	778	6	9	58	74484	1	1150	900	250	0
2	201	21	9	111	74706	2	2600	1400	1200	0
3	201	21	9	65	74706	3	300	180	120	0
4	859	6	7	65	74674	3	300	180	120	0
5	43	19	4	121	74735	2	9600	5988	3602	0
6	450	1	7	134	74782	1	90	38	52	0
7	504	2	25	124	74534	2	3000	1800	1400	0
8	504	2	25	3	74534	3	144	72	72	0
9	958	1	11	117	74654	1	3300	2700	600	0
10	453	5	23	95	74797	3	150	75	75	0
11	453	5	23	194	74797	4	348	288	60	0
12	34	3	21	85	74806	2	5000	2600	2400	0
13	34	3	21	70	74806	1	890	500	390	0
14	171	24	9	13	74539	1	189	60	129	0
15	171	24	9	66	74539	3	1880	1155	525	0
16	944	21	6	122	74735	3	10500	8400	2100	0
17	956	18	9	25	74726	1	88	46	52	0
18	472	11	4	103	74565	3	1650	600	1050	0
19	976	3	15	71	74525	1	500	360	140	0
20	471	4	7	39	74572	1	145	78	67	0
21	283	13	13	116	74598	3	5610	3900	1710	0
22	13	11	13	3	74479	2	96	48	48	0
23	1000	21	26	165	74628	3	3300	2550	750	0
24	155	18	3	43	74493	2	290	134	156	0
25	181	1	4	158	74777	3	327	243	84	0

CONCLUSÃO DATA WAREHOUSE

O projeto de modelagem do *Data Warehouse* finalmente conclui-se com êxito em todos os processo de ETL e definição de relacionamentos.

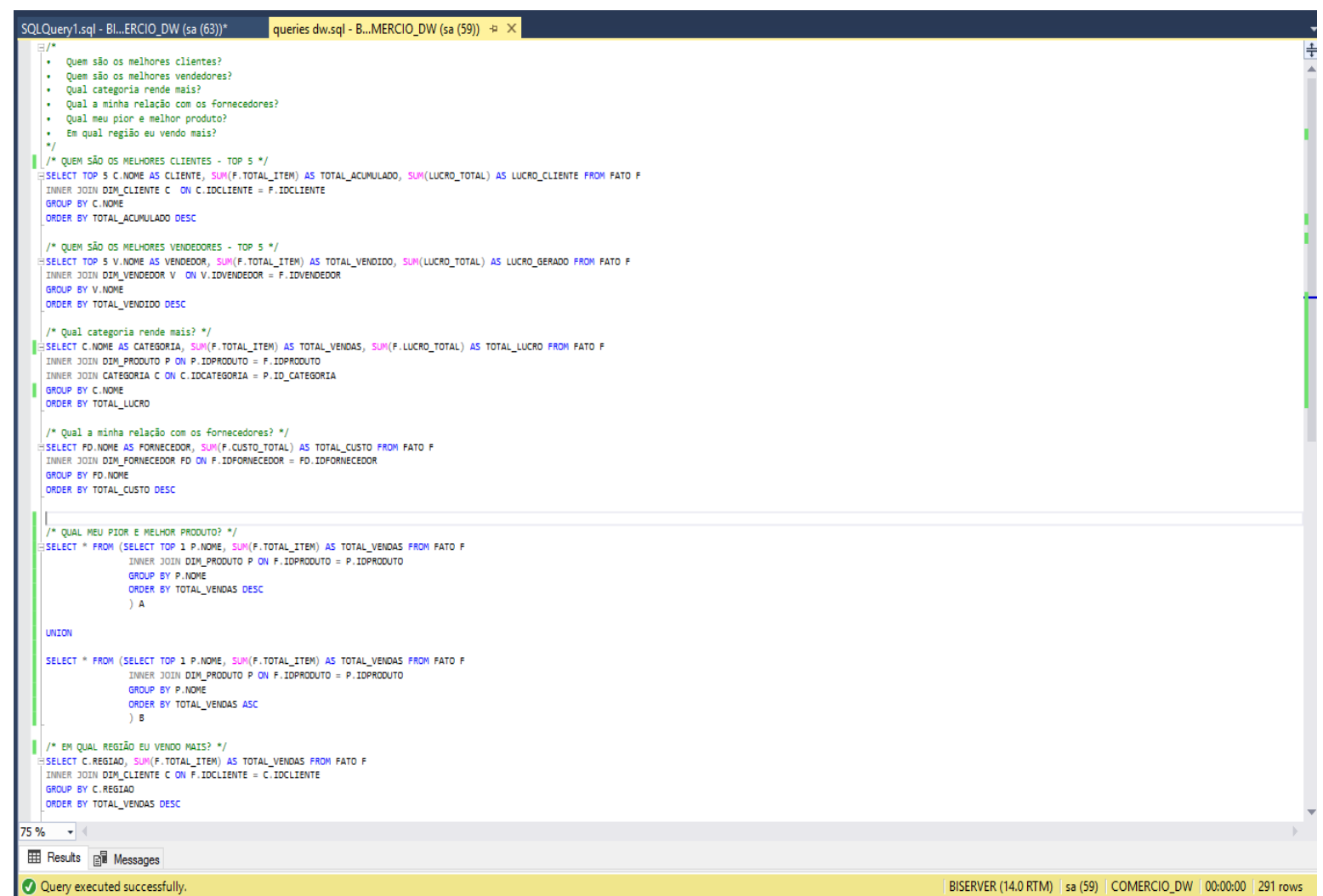
O DW está pronto para ser consultados por queries em SQL e/ou receber conexões diretamente de ferramentas de *Data Visualization* para construção de *dashboards*.

Entretanto, nesse ponto, inicia-se os projetos do sistema OLAP, utilizando o *Analysis Services* (SSAS) e o *Reporting Services* (SSRS) para criar os cubos analíticos e automação dos relatórios.

QUERIES DATA WAREHOUSE – PROPOSTA DE NEGÓCIO

Embora a proposta geral do *Data Warehouse* seja um banco de dados como fonte para análises por analistas e gestores, através de relatórios e ferramentas de *DataViz*, podemos realizar algumas *queries* diretamente no SQL Server e responder as questões iniciais que foram propostas.

Podemos então validar o *Data Warehouse* da Goiás Market respondendo ao problema de negócio proposto no escopo do projeto:



```
SQLQuery1.sql - BI...ERCIO_DW (sa (63)) * queries dw.sql - B...MERCIO_DW (sa (59)) - + X
/*
 * Quem são os melhores clientes?
 * Quem são os melhores vendedores?
 * Qual categoria rende mais?
 * Qual a minha relação com os fornecedores?
 * Qual meu pior e melhor produto?
 * Em qual região eu vendo mais?
 */
/* QUEM SÃO OS MELHORES CLIENTES - TOP 5 */
SELECT TOP 5 C.NOME AS CLIENTE, SUM(F.TOTAL_ITEM) AS TOTAL_ACUMULADO, SUM(LUCRO_TOTAL) AS LUCRO_CLIENTE FROM FATO F
INNER JOIN DIM_CLIENTE C ON C.IDCLIENTE = F.IDCLIENTE
GROUP BY C.NOME
ORDER BY TOTAL_ACUMULADO DESC

/* QUEM SÃO OS MELHORES VENDEDORES - TOP 5 */
SELECT TOP 5 V.NOME AS VENDEDORE, SUM(F.TOTAL_ITEM) AS TOTAL_VENDIDO, SUM(LUCRO_TOTAL) AS LUCRO_GERADO FROM FATO F
INNER JOIN DIM_VENDEDORE V ON V.IDVENDEDORE = F.IDVENDEDORE
GROUP BY V.NOME
ORDER BY TOTAL_VENDIDO DESC

/* Qual categoria rende mais? */
SELECT C.NOME AS CATEGORIA, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS, SUM(F.LUCRO_TOTAL) AS TOTAL_LUCRO FROM FATO F
INNER JOIN DIM_PRODUTO P ON P.IDPRODUTO = F.IDPRODUTO
INNER JOIN CATEGORIA C ON C.IDCATEGORIA = P.ID_CATEGORIA
GROUP BY C.NOME
ORDER BY TOTAL_LUCRO

/* Qual a minha relação com os fornecedores? */
SELECT FD.NOME AS FORNECEDOR, SUM(F.CUSTO_TOTAL) AS TOTAL_CUSTO FROM FATO F
INNER JOIN DIM_FORNECEDOR FD ON F.IDFORNECEDOR = FD.IDFORNECEDOR
GROUP BY FD.NOME
ORDER BY TOTAL_CUSTO DESC

/* QUAL MEU PIOR E MELHOR PRODUTO? */
SELECT * FROM (SELECT TOP 1 P.NOME, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F
INNER JOIN DIM_PRODUTO P ON F.IDPRODUTO = P.IDPRODUTO
GROUP BY P.NOME
ORDER BY TOTAL_VENDAS DESC
) A

UNION

SELECT * FROM (SELECT TOP 1 P.NOME, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F
INNER JOIN DIM_PRODUTO P ON F.IDPRODUTO = P.IDPRODUTO
GROUP BY P.NOME
ORDER BY TOTAL_VENDAS ASC
) B

/* EM QUAL REGIÃO EU VENDO MAIS? */
SELECT C.REGIAO, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F
INNER JOIN DIM_CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE
GROUP BY C.REGIAO
ORDER BY TOTAL_VENDAS DESC
```

75 %

Results Messages

Query executed successfully. BISERVER (14.0 RTM) | sa (59) | COMERCIO_DW | 00:00:00 | 291 rows

SQLQuery1.sql - BI...ERCIO_DW (sa (63))*

queries dw.sql - B...MERCIO_DW (sa (59))*

```
/* QUEM SÃO OS MELHORES CLIENTES - TOP 5 */
SELECT TOP 5 C.NOME AS CLIENTE, SUM(F.TOTAL_ITEM) AS TOTAL_ACUMULADO, SUM(LUCRO_TOTAL) AS LUCRO_CLIENTE FROM FATO F
INNER JOIN DIM_CLIENTE C ON C.IDCLIENTE = F.IDCLIENTE
GROUP BY C.NOME
ORDER BY TOTAL_ACUMULADO DESC
```

100 %

Results Messages

	CLIENTE	TOTAL_ACUMULADO	LUCRO_CLIENTE
1	Gustavo Goncalves	263374.00	63456.00
2	Laura Cavalcanti	220024.00	55170.00
3	Beatriz Correia	209600.00	46020.00
4	Julian Sousa	207155.00	58037.00
5	Kauã Fernandes	200472.00	51118.00

Query executed successfully.

BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 5 rows

SQLQuery1.sql - BI...ERCIO_DW (sa (63))*

queries dw.sql - B...MERCIO_DW (sa (59))*

```
/* QUEM SÃO OS MELHORES VENDEDORES - TOP 5 */
SELECT TOP 5 V.NOME AS VENDEDOR, SUM(F.TOTAL_ITEM) AS TOTAL_VENDIDO, SUM(LUCRO_TOTAL) AS LUCRO_GERADO FROM FATO F
INNER JOIN DIM_VENDEDOR V ON V.IDVENDEDOR = F.IDVENDEDOR
GROUP BY V.NOME
ORDER BY TOTAL_VENDIDO DESC
```

100 %

Results Messages

	VENDEDOR	TOTAL_VENDIDO	LUCRO_GERADO
1	PILAR SANCHES	3051968.00	793920.00
2	HERMES MACEDO	3006790.00	775751.00
3	TERESA CRISTINA	2977228.00	809442.00
4	RENE CHIARI	2940815.00	775899.00
5	DEBORA ALMEIDA	2939122.00	789056.00

Query executed successfully.

BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 5 rows

SQLQuery1.sql - BI...ERCIO_DW (sa (63)) * queries dw.sql - B...MERCIO_DW (sa (59)) *

```
/* Qual categoria rende mais? */
SELECT C.NOME AS CATEGORIA, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS, SUM(F.LUCRO_TOTAL) AS TOTAL_LUCRO FROM FATO F
INNER JOIN DIM_PRODUTO P ON P.IDPRODUTO = F.IDPRODUTO
INNER JOIN CATEGORIA C ON C.IDCATEGORIA = P.ID_CATEGORIA
GROUP BY C.NOME
ORDER BY TOTAL_LUCRO
```

100 %

Results Messages

	CATEGORIA	TOTAL_VENDAS	TOTAL_LUCRO
1	LIVROS	357641.00	161895.00
2	DVDS	566304.00	247326.00
3	GAMES	1580031.00	772081.00
4	PASSAGENS	7477640.00	825914.00
5	MOVEIS	4082000.00	923843.00
6	ELETRODOMESTICOS	8004991.00	1768005.00
7	CELULARES	6333874.00	2154946.00
8	TV E AUDIO	16652726.00	3387537.00
9	INFORMATICA	21893839.00	7351751.00

Query executed successfully. BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 9 rows

SQLQuery1.sql - BI...ERCIO_DW (sa (63)) * queries dw.sql - B...MERCIO_DW (sa (59)) *

```
/* Qual a minha relação com os fornecedores? */
SELECT FD.NOME AS FORNECEDOR, SUM(F.CUSTO_TOTAL) AS TOTAL_CUSTO FROM FATO F
INNER JOIN DIM_FORNECEDOR FD ON F.IDFORNECEDOR = FD.IDFORNECEDOR
GROUP BY FD.NOME
ORDER BY TOTAL_CUSTO DESC
```

100 %

Results Messages

	FORNECEDOR	TOTAL_CUSTO
1	SONY	7682110.00
2	SAMSUNG	4697920.00
3	LG	3415010.00
4	BRASIL MADEIRAS	2807373.00
5	MICROSOFT	2683695.00
6	LINHAS INTERNACIONAIS EURO	2627155.00
7	PANASONIC	2571250.00
8	HP	2376030.00
9	APPLE	2060070.00
10	BRASTEMP	2005522.00
11	ACER	1889148.00
12	PHILIPS	1752974.00
13	AMERICA LINES EXPRESS	1721501.00
14	LINHAS LATINAS DE AVIÕES	1711870.00
15	GANCA	1698575.00
16	CCE	1181845.00
17	LENOVO	934250.00
18	MULTILASER	882602.00
19	MOTOROLA	778000.00
20	ARNO	737496.00
21	GFORCE	438796.00
22	NOKIA	375210.00
23	MOVEIS VAREJÃO	350784.00
24	LINHAS AÉREAS BRAZUCA	301780.00
25	LINHAS BRASIL EXPRESS	289420.00
26	WALLITA	195242.00
27	BRAZUCA JOGOS	168758.00
28	GENIUS	129636.00

Query executed successfully. BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 41 rows

```
/* QUAL MEU PIOR E MELHOR PRODUTO? */  
SELECT * FROM (SELECT TOP 1 P.NOME, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F  
    INNER JOIN DIM_PRODUTO P ON F.IDPRODUTO = P.IDPRODUTO  
    GROUP BY P.NOME  
    ORDER BY TOTAL_VENDAS DESC  
    ) A  
  
UNION  
  
SELECT * FROM (SELECT TOP 1 P.NOME, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F  
    INNER JOIN DIM_PRODUTO P ON F.IDPRODUTO = P.IDPRODUTO  
    GROUP BY P.NOME  
    ORDER BY TOTAL_VENDAS ASC  
    ) B
```

100 %

Results Messages

	NOME	TOTAL_VENDAS
1	Leitor De Cartoes Multilaser	6888.00
2	Notebook Mft Hibr	1560000.00

Query executed successfully.

BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 2 rows

```
/* EM QUAL REGIÃO EU VENDO MAIS? */  
SELECT C.REGIAO, SUM(F.TOTAL_ITEM) AS TOTAL_VENDAS FROM FATO F  
    INNER JOIN DIM_CLIENTE C ON F.IDCLIENTE = C.IDCLIENTE  
    GROUP BY C.REGIAO  
    ORDER BY TOTAL_VENDAS DESC
```

100 %

Results Messages

	REGIAO	TOTAL_VENDAS
1	SUDESTE	34305715.00
2	NORDESTE	11939891.00
3	SUL	8791921.00
4	CENTRO-OESTE	8191901.00
5	NORTE	3719618.00

Query executed successfully.

BISERVER (14.0 RTM) sa (63) COMERCIO_DW 00:00:00 5 rows