Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Gestión de Datos

Trabajo Práctico

1° Cuatrimestre 2025

FRBA – Fábrica de Sillones

GRUPO 83: LOS_QUE_SABEN_SABEN

Ignacio, Scarfo	214.123-1	iscarfo@frba.utn.edu.ar
Zheng, Lucas	209.241-4	lzheng@frba.utn.edu.ar
Di Bucci, Manuel	213.656-9	mdibucci@frba.utn.edu.ar
Torres Franco, Santiago Nicolas	214.191-7	storres@frba.utn.edu.ar

FECHA DE PRESENTACIÓN (Primera Entrega): 01/06/25
FECHA DE PRESENTACION (Segunda Entrega): 27/06/25

TABLA DE CONTENIDO

MODELO TRANSACCIONAL

MODELO BI

MODELO TRANSACCIONAL

1. Consideraciones Generales del Proceso de Migración

El objetivo principal de esta migración fue trasladar los datos desde la tabla gd_esquema.Maestra hacia las tablas de un esquema transaccional (LOS_QUE_SABEN_SABEN), asegurando la integridad, unicidad y consistencia de la información. Para esto, se diseñaron procedimientos almacenados para cada tabla de destino.

2. Soluciones Implementadas

Durante el proceso, nos surgieron desafíos recurrentes que se abordaron con soluciones específicas, las cuales se reflejan en el código final:

2.1. Manejo de Duplicados y Unicidad de Registros

La tabla gd_esquema.Maestra contiene gran cantidad de datos, y en muchos casos, múltiples filas de la Maestra podían representar un único registro lógico en las tablas de destino. Para evitar la inserción de datos repetidos, se emplearon principalmente dos técnicas:

- Cláusula DISTINCT: Esta cláusula se utilizó en la mayoría de los SELECT dentro de los INSERT para obtener solo los valores únicos de un conjunto de columnas que identifican un registro lógico en la tabla de destino. Por ejemplo, en Migrar_Provincias, se usa SELECT DISTINCT Sucursal_Provincia para obtener cada nombre de provincia una sola vez. De manera similar, en Migrar_Compras, se utiliza SELECT DISTINCT tm.Compra_Numero, s.ID_Sucursal, prov.ID_Proveedor, tm.Compra_Fecha, tm.Compra_Total para asegurar que cada compra (encabezado) se inserte una sola vez, incluso si en la Maestra aparece en

varias filas debido a los detalles de compra. Esto es útil cuando la clave principal ya es única en el origen (como Compra_Numero o ID_Sucursal).

- Cláusula NOT EXISTS: Esta clausula nos garantiza evitar la inserción de registros que ya existen en el destino. Antes de insertar una fila, se verifica si ya hay un registro con la misma clave o combinación de atributos únicos en la tabla de destino. Ejemplos claros de esto se ven en Migrar_Localidades, donde se verifica la combinación Nombre e Id_Provincia, en Migrar_Clientes por Dni, en Migrar_Proveedores por CUIT, o en Migrar_Sucursales por ID_Sucursal.

- Uso de ROW_NUMBER(): en dos casos especificos, esta clausula nos vino de utilidad para escenarios donde un mismo identificador en la Maestra podía tener múltiples filas/versiones del mismo dato lógico o cuando necesitábamos consolidar información. Por ejemplo, en Migrar_Sillones, se utiliza ROW_NUMBER() para seleccionar una única definición (Modelo y Medida) para cada Sillon_Codigo, asegurando que, incluso si un Sillon_Codigo aparece múltiples veces en la Maestra, solo se tome una de sus representaciones para crear el registro del sillón. También se usó en Migrar_DetallePedido para asegurar una única combinación ID_Pedido y ID_Sillon.

2.2. Manejo de Claves Primarias y Valores Nulos

Se presentaron casos donde las claves primarias de las tablas de destino no eran inicialmente IDENTITY (autoincrementales) y el campo correspondiente en la tabla Maestra contenía valores NULL.

- Ignorar registros con IDs nulos y datos asociados nulos: Se observó que para entidades como Factura, Compra y Envio, si el ID principal (ej., Factura_Numero, Compra_Numero, Envio_Numero) estaba en NULL en la Maestra, todos los campos asociados a esa entidad también eran NULL. Esto significa que, en esos casos, no había una compra, factura o envío real que migrar. Por lo tanto, la estrategia fue simplemente filtrar y no incluir estos registros en la migración,

usando cláusulas WHERE tm.Compra_Numero IS NOT NULL o similares. Esto evita intentar insertar NULL en una PRIMARY KEY o crear registros incompletos.

- Generación de IDs: En el caso de Envios, se discutió que Envio.ID_Envio era una clave primaria DECIMAL pero Maestra.Envio_Numero era NULL. La solución que optamos fue la generación secuencial para id_Envio mediante IDENTITY, de forma de que poder asegurar la unicidad y no nulidad

2.3. Mapeo Correcto y Consolidación de Datos

- El uso de JOINs nos fue fundamental en la migración porque la Maestra es una tabla "plana"(los datos estas desnormalizados), mientras que nuestro esquema de destino (LOS_QUE_SABEN_SABEN) está normalizado en múltiples tablas relacionadas. Los JOINs nos permiten conectar la información de la Maestra con los IDs ya existentes en nuestras tablas de destino (como ID_Cliente, ID_Sucursal, ID_Provincia, ID_Localidad, etc.). Esto nos fue crucial para poder construir los registros en las tablas que tienen claves foráneas, buscando y "traduciendo" los datos de la Maestra (ej., el DNI del cliente, el número de sucursal, el nombre de la provincia) en los IDs correspondientes de nuestras tablas ya pobladas. Este proceso garantiza que las relaciones entre los datos queden correctamente establecidas en el nuevo esquema.

- Consolidación de Encabezados con GROUP BY y MAX(): Para tablas como Pedidos y Facturas, donde la Maestra podía tener múltiples filas para el mismo encabezado (por los detalles), se utilizó un GROUP BY por el identificador único del encabezado (ej., Pedido_Numero o Factura_Numero). Se combinó esto con funciones de agregación como MAX() en los campos que deberían ser únicos por cada encabezado (ej., MAX(tm.Pedido_Fecha), MAX(tm.Pedido_Total)). Esto garantiza que se inserte una única fila para cada encabezado de pedido o factura, sin que los detalles causen duplicación de los datos del encabezado.

2.4. Manejo Específico de Tipos y Materiales

- Migración Jerárquica: Para los Materiales, se optó por una migración en etapas dentro del mismo procedimiento Migrar_Materiales. Primero, se migran los Tipo_Material (usando DISTINCT y EXCEPT para solo añadir nuevos tipos). Luego, se migran los Material generales, uniendo con los tipos ya insertados. Finalmente, se migran las subclases Tela, Madera y Relleno, cada una mapeada a su ID_Tipo_Material correspondiente y con sus atributos específicos.

- Manejo de Nulos en Unicidad de Medida, Tela, Madera, Relleno: Para tablas como Medida o las subclases de Material (Tela, Madera, Relleno) donde la unicidad puede depender de campos que permiten NULL (ej., Tela_Color, Tela_Textura), se utilizó ISNULL(campo, '') o ISNULL(campo, -1) en la cláusula NOT EXISTS. Esto permite que el NOT EXISTS compare correctamente los valores NULL con NULL (tratándolos como cadenas vacías o un valor sentinela) para evitar duplicados si los valores nulos forman parte de la clave candidata.

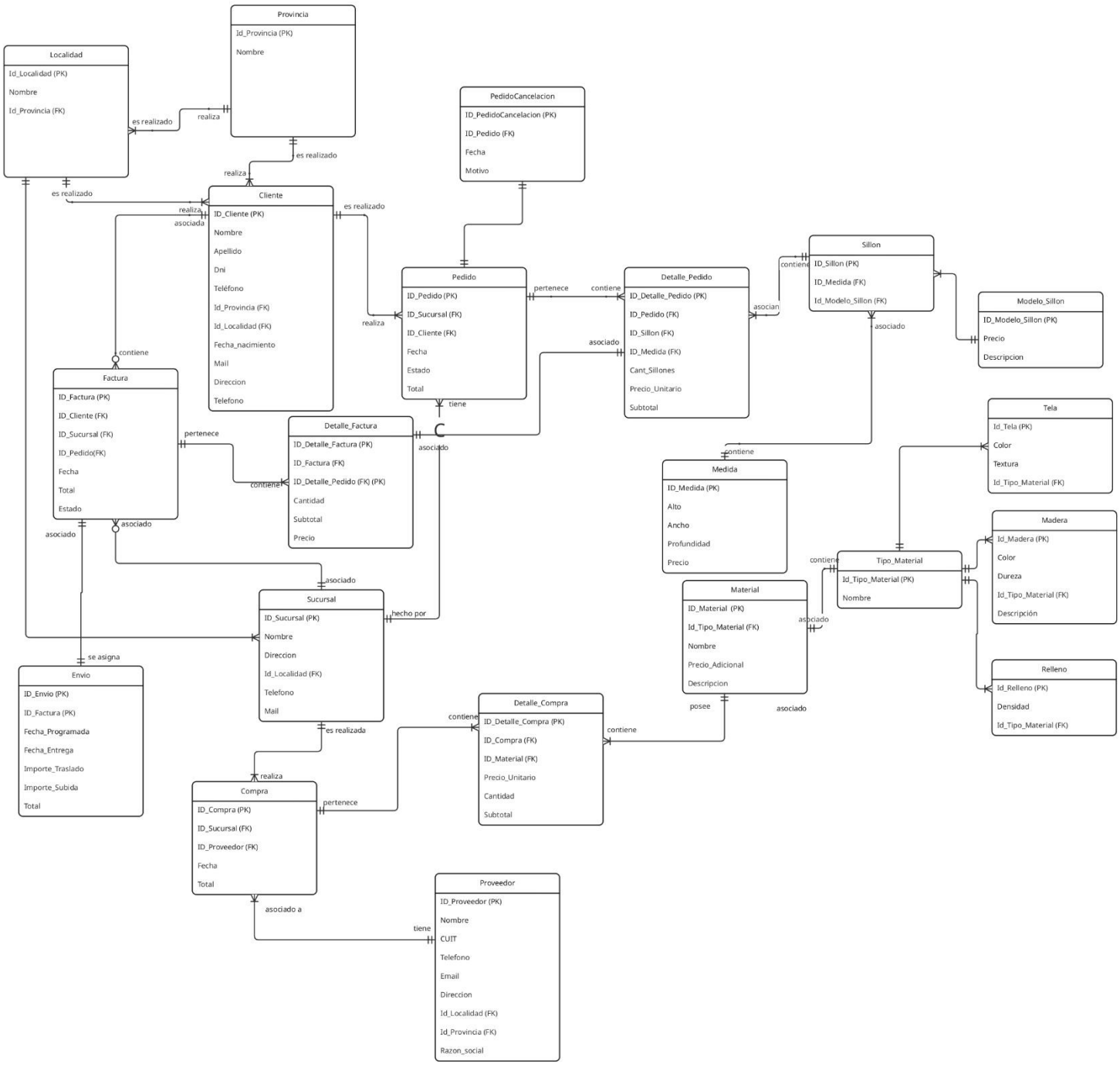
2.5. Consideraciones de Dependencia y Orden de Ejecución

Los procedimientos se hallan en un orden determinado debido a las dependencias de clave foránea. Por ejemplo:

1. Provincias y Localidades deben migrarse primero, ya que Clientes, Sucursales y Proveedores dependen de ellas.
2. Clientes, Sucursales y Proveedores deben migrarse antes que Pedidos, Facturas y Compras respectivamente.
3. Modelo_Sillon y Medida deben migrarse antes que Sillon.
4. Sillon debe migrarse antes que Detalle_Pedido.
5. Pedidos y Facturas deben migrarse antes que sus respectivos Detalle y Cancelacion.
6. Compras y Materiales deben migrarse antes que Detalle_Compra.

Este orden asegura que las claves foráneas en las tablas de destino siempre apunten a registros ya existentes, evitando errores de integridad.

3. DER Modelo Transaccional



4. Correcciones (Entrega 27/06)

4.1 Generación de IDs para la tabla Envio:

- Entrega Anterior: Mencionamos que optamos por la generación secuencial de ID_Envio usando IDENTITY porque Maestra.Envio_Numero podía ser NULL.
- Version Actual: La tabla Envio no está definida con IDENTITY para ID_Envio. En cambio, el procedimiento Migrar_Envios_Simplificado inserta directamente el valor de tm.Envio_Numero en ID_Envio, usando una cláusula WHERE tm.Envio_Numero IS NOT NULL. Esto significa que el script usa el número de envío existente en Maestra (cuando no es nulo) como clave primaria.

4.2 Consolidación de Encabezados (Pedidos y Facturas):

- Entrega Anterior: Indicamos que para tablas como Pedidos y Facturas, donde la maestra podía tener múltiples filas para el mismo encabezado, se utilizó GROUP BY con funciones de agregación como MAX() para consolidar los datos y asegurar una única fila para cada encabezado.
- Version Actual: Los procedimientos Migrar_Pedidos y Migrar_Facturas usan SELECT DISTINCT en todas las columnas del encabezado en lugar de GROUP BY con MAX(). Aunque DISTINCT puede lograr la unicidad para la combinación de campos seleccionados, la metodología específica de GROUP BY/MAX() descrita en la estrategia ya no lo implementamos en varios de los casos.

4.3 Manejo de Unicidad en Detalle_Pedido:

- Entrega Anterior: Afirmamos que se utilizó ROW_NUMBER() en Migrar_DetallePedido para asegurar una combinación única de ID_Pedido y ID_Sillon.
- Version Actual: El procedimiento Migrar_DetallePedido ya no utiliza ROW_NUMBER(). En su lugar, empleamos SELECT DISTINCT y una cláusula NOT EXISTS que verifica la unicidad basándose en la combinación de ID_Pedido, ID_Sillon, Cant_Sillones y Precio_Unitario.

4.4 Manejo de Nulos en la Unicidad de Madera:

- Entrega Anterior: Generaliza que se utilizó ISNULL(campo, '') o ISNULL(campo, -1) en las cláusulas NOT EXISTS para manejar campos que pueden ser NULL en la unicidad de tablas como Medida, Tela, Madera y Relleno.
- Version Actual: Si bien Medida, Tela y Relleno aplican esta lógica de ISNULL en sus NOT EXISTS (o verificaciones similares), la inserción de Madera dentro de Migrar_Material ya no tiene una cláusula NOT EXISTS para la unicidad de sus propios datos. En cambio, lo basamos en filtros WHERE m.Madera_Dureza IS NOT NULL AND m.Madera_Color IS NOT NULL, modificando el método de verificación de unicidad como en la aplicación del ISNULL para Madera.

MODELO BI

1. Consideraciones Generales de Business Intelligence

El modelo actual de Business Intelligence (BI) se basa en una estructura dimensional que organiza los datos del sistema transaccional en un formato optimizado para la consulta y el análisis de indicadores de gestión.

El objetivo es proveer una solución robusta y eficiente que facilite la obtención de información gerencial a través de las vistas solicitadas, conforme a lo solicitado por el enunciado.

2. Estrategia y Justificación de Implementación

A continuación, se detalla la justificación de las principales decisiones tomadas con el grupo durante el diseño y la implementación de las estructuras de datos, los procesos de migración y las vistas de negocio.

2.1. Diseño de Tablas de Dimensiones

Las tablas de dimensiones fueron diseñadas para proveer el contexto descriptivo necesario para el análisis de las métricas.

BI_Dim_Tiempo:

- Decisión de Granularidad: Optamos por una granularidad a nivel de día. Esta elección, si bien genera una tabla de mayor tamaño, ofrece una flexibilidad total para agregar la información a cualquier nivel superior requerido (mes, cuatrimestre, año). Un diseño a nivel de mes nos impediría realizar análisis más detallados si, por ejemplo, se requiriera en el futuro hipotético (escalabilidad).

- Optimización: Los atributos como Anio, Cuatrimestre y Mes decidimos almacenarlas como columnas pre-calculadas para evitar el uso de funciones de fecha en tiempo de ejecución, lo que mejora considerablemente el rendimiento de las consultas agrupadas.

BI_Dim_Ubicacion y BI_Dim_Sucursal:

- Decisión de Consolidación: Se consolidó la información de Provincia y Localidad en una única dimensión BI_Dim_Ubicacion. Esto permite reutilizar la misma estructura para analizar la geografía de las sucursales y la de los clientes (para los costos de envío), simplificando el modelo y las consultas. BI_Dim_Sucursal contiene los datos específicos de cada sucursal y se vincula a esta dimensión de ubicación.

BI_Dim_Rango_Etario y BI_Dim_Turno:

- Decisión sobre Datos Categóricos: Para estas dimensiones, creamos categorías fijas basadas en las reglas de negocio del enunciado.

- Manejo de Datos Incompletos: Se incluyeron categorías como 'S/D' (Sin Dato) y 'Fuera de Horario' para asegurar la integridad del proceso de migración. Esta decisión nos garantiza que todos los registros de hechos (ventas, pedidos) sean cargados, incluso si los datos originales no encajan perfectamente en las categorías definidas (ej. un cliente sin fecha de nacimiento). De esta manera, no se pierde información cuantitativa y nos analizar estos casos excepcionales.

2.2. Diseño de Tablas de Hechos

Implementamos un modelo con múltiples tablas de hechos según indica el enunciado, onde cada una representando un proceso de negocio clave. Esta separación es fundamental porque cada proceso tiene una naturaleza y un nivel de detalle distintos.

BI_Hechos_Ventas y BI_Hechos_Compras:

- Decisión de Agregación: La migración de datos para estas tablas no se realiza a nivel de detalle de ítem (factura o compra), sino que los datos se agrupan y consolidan durante el proceso de migracion. Por ejemplo, las métricas Total_Venta y Cantidad_Vendida en Hechos_Ventas son el resultado de sumar

los subtotales y cantidades de todos los ítems que comparten las mismas dimensiones. Esta pre-agregación reduce significativamente el tamaño de las tablas de hechos y acelera las consultas de negocio, ya que el cálculo pesado se realiza una sola vez.

BI_Hechos_Envios:

- Decisión de Implementacion del Flag_En_Termino: Incorporamos un campo BIT denominado Flag_En_Termino. En lugar de comparar Fecha_Entrega y Fecha_Programada en cada consulta, esta validación se realiza una única vez durante la migración y su resultado (1 o 0) se almacena directamente en la tabla.

- Justificación de Rendimiento: Este enfoque optimiza drásticamente la vista de cumplimiento de envíos. El cálculo del porcentaje se convierte en una operación aritmética simple sobre el flag (SUM(Flag_En_Termino) / COUNT(*)) en lugar de una evaluación condicional sobre todas las filas, lo cual es mucho más eficiente.

2.3. Proceso de Migración

La lógica para poblar el modelo de BI fue encapsulada en procedimientos almacenados (stored procedurs) para garantizar un proceso ordenado, modular y mantenible.

- Decisión de Modularidad: tenemos dos procedimientos principales, Migrar_Dimensiones y Migrar_Hechos, orquestados por un procedimiento maestro, Migracion_BI_Completa. Asi podemos, por ejemplo, volver a ejecutar la migración de los hechos sin necesidad de alterar las dimensiones, lo cual es útil para actualizaciones periódicas de datos.

- Secuencia de Ejecución: El orden de ejecución (primero dimensiones, luego hechos) es mandatorio. Al igual que en el modelo transaccional, las tablas de hechos utilizan las claves primarias generadas en las tablas de dimensiones como

claves foráneas. La ejecución en el orden correcto asegura la integridad referencial del modelo.

2.4. Diseño de Vistas de Negocio

Las vistas las construimos como una capa de abstracción final para responder directamente a los indicadores solicitados. Ocultan la complejidad de las uniones y presentan los datos en un formato listo para el consumo.

V_Rendimiento_Modelos: Para resolver el requerimiento de "los 3 modelos con mayores ventas", utilizamos la función analitica ROW_NUMBER() particionada por las categorías solicitadas. La consideramos la forma más eficiente en SQL para realizar rankings sin enroscarnos en consultas complejas o subóptimas.

V_Ganancias: Para asegurar que el reporte de ganancias muestre todos los meses y sucursales, incluso aquellos sin actividad, la consulta base decidimos hacerla a partir de un CROSS JOIN entre un listado único de meses/años y las sucursales. Luego, utilizamos LEFT JOIN contra los datos agregados de ventas y compras. Esto nos garantizo que no se omitan períodos de inactividad, mostrando una ganancia de 0 en lugar de no mostrar el registro, lo cual nos permite tener un análisis mas completo.

3. DER Modelo BI