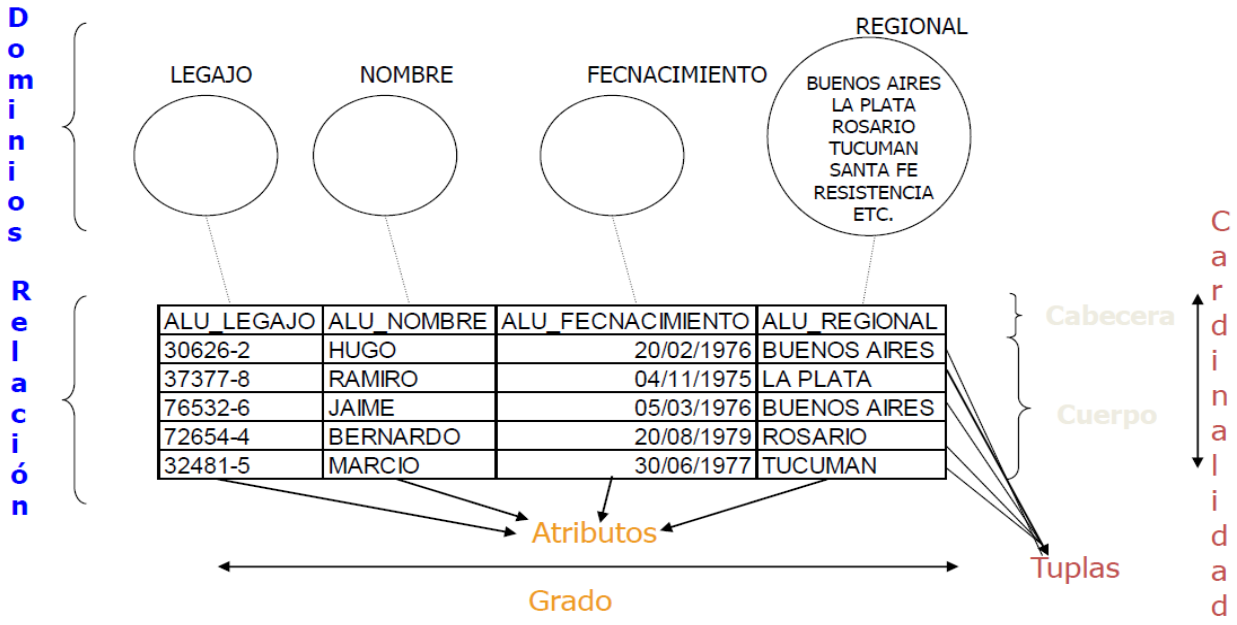
MODELO RELACIONAL

Concepto introducido por Edgar Codd que consta de: **estructura**, **manipulación de datos** e **integridad**.

- **Estructura** → recopilación de objetos o relaciones.



DOMINIO → colección de valores escalares de igual tipo, de los cuales los atributos obtienen sus valores reales.

- Menor unidad semántica de información.
- Atómicos → no se pueden descomponer sin perder significado.
- No contienen valores nulos.
- Para su implementación se requieren:
  - Restricciones de chequeo.
  - FKs.
  - Definición de tipos de datos de usuario.
  - Definición de convenciones de nombre para los atributos.

RELACIÓN → subconjunto del producto cartesiano de los dominios de los valores involucrados.

- Consta de:
  - **Cabecera** → conjunto fijo de pares atributo-dominio.
  - **Cuerpo** → conjunto de tuplas que varía con el tiempo.
  - **Tupla** → conjunto de pares atributo-valor.
  - **Tabla** → representación de una relación.
  - **BD Relacional** → BD percibida por el usuario como una colección de relaciones normalizadas de diversos grados que varía con el tiempo.
- Propiedades:
  - No hay tuplas repetidas → toda relación tiene una PK.
  - Las tuplas y los atributos no están ordenadas.
- Tipos: tablas, resultados de consultas, resultados intermedios de consultas, vistas y *snapshots*.

- **Manipulación de datos** → refiere a las operaciones sobre las relaciones (basadas en el álgebra relacional) para producir otras relaciones.

  Algunas de esas operaciones son:
  - Unión.
  - Intersección.
  - Diferencia.
  - División.
  - SELECT.
  - PROJECT.
  - Producto cartesiano.
  - JOIN.

- **Integridad** → para obtener precisión y consistencia.

PK – *PRIMARY KEY* → atributo (o conjunto de atributos) identificador único para una relación.

- Representa unívocamente a una fila de la tabla.
- Son el mecanismo de identificación en el modelo relacional.

FK – *FOREIGN KEY* → atributo (o conjunto de atributos) de una relación R2 cuyos valores no nulos deben coincidir con los valores de una PK de una relación R1.

- La PK de R1 y la FK de R2 están definidas sobre el mismo dominio.
- La FK puede o no ser parte de la PK de R2.
- R1 y R2 no necesariamente son distintas → si R1=R2, existe una relación autorreferencial.
- Las FKs deben aceptar nulos en ciertas ocasiones.

CK – *CANDIDATE KEY* → clave que identifica unívocamente a la relación, pero que aún no es PK.

- El contexto define quién es PK y quién CK.

AK – *ALTERNATE KEY* → CK que no es PK.

- Las PKs y las FKs hacen a la integridad.

REGLA DE INTEGRIDAD DE LAS ENTIDADES → ningún componente de la PK de una relación base puede aceptar nulos.

REGLA DE INTEGRIDAD REFERENCIAL → la BD no debe contener valores no nulos de FK para los cuales no exista un valor concordante de PK en la relación referenciada.

REGLAS DE CODD

- Independencia entre el motor de BD y los programas que acceden a los datos → es posible modificar el motor de BD o los componentes de la aplicación en forma independiente.
- Representar la información de la estructura de las tablas y sus relaciones mediante un catálogo dinámico basado en el modelo relacional.
- Las reglas de integridad deben guardarse en el catálogo de la BD, no en programas de aplicación.
- Soportar información faltante mediante valores nulos (NULL).
- Proveer lenguajes para:
    - Definición de datos (DDL).
    - Manipulación de datos (DML) → donde debe haber operaciones de alto nivel para insertar, eliminar, actualizar o buscar.
    - Definición de restricciones de seguridad, restricciones de integridad, autorización y delimitar una transacción.

INDEPENDENCIA DE DATOS LÓGICA → se podrían hacer cambios en tablas sin que afecten a las aplicaciones que no los requieren.

INDEPENDENCIA DE DATOS FÍSICA → es posible modificar la estructura de almacenamiento, la distribución física o la técnica de acceso sin afectar las aplicaciones.

Terminología de BDRs:

- TABLA → estructura básica de almacenamiento.
- FILA → todos los datos para cada ocurrencia en la tabla.
- COLUMNA → identifica a un dato de la tabla.
- CAMPO → intersección entre una fila y una columna.

TABLA → relación.
FILA → tupla.
COLUMNA → atributo.

OBJETOS DE UNA BASE DE DATOS

TABLA → unidad básica de almacenamiento de datos, los cuales están almacenados en filas y columnas.

- De existencia permanente.
- Tiene un nombre identificatorio.
- Cada columna tiene: un nombre, un tipo de dato y un ancho.

TABLA TEMPORAL → tabla creada cuyos datos son de existencia temporal.

- No son registradas en las tablas del diccionario de datos.
- No es posible alterarlas.
  Sí es posible eliminarlas y crearles los índices temporales que necesite una aplicación.
- Usos:
  - Almacenamiento intermedio de consultas muy grandes.
  - Optimización de accesos a una consulta varias veces en una aplicación.
- Tablas locales (de sesión) → visibles sólo para sus creadores durante la misma sesión/conexión.
  Tablas globales → visibles para cualquier usuario y sesión.
- Tablas de creación explícita → las tablas son creadas a partir de la instrucción CREATE, donde se indican: su nombre, sus campos, tipos de datos, etc.
  Tablas de creación implícita → las tablas son creadas a partir del resultado de una consulta SELECT.

CONSTRAINTS → restricciones que sirven para resguardar la integridad de la información de una BD.

- **INTEGRIDAD DE ENTIDAD** → usada para asegurar que los datos pertenecientes a una misma tabla tengan una única manera de identificarse.
  - PRIMARY KEY CONSTRAINT.
- **INTEGRIDAD REFERENCIAL** → usada para asegurar la coherencia entre datos de dos tablas.
  - FOREIGN KEY CONSTRAINT.
- **INTEGRIDAD SEMÁNTICA** → usada para asegurar que los datos a almacenar tengan una apropiada configuración y que respeten las restricciones definidas sobre los dominios o sobre los atributos.
  - DATA TYPE → define el tipo de valor que se puede almacenar en una columna.
  - DEFAULT CONSTRAINT → valor que se inserta en una columna cuando al insertar un registro no se especifica ningún valor. Aplica a columnas no listadas en una sentencia INSERT.
  - UNIQUE CONSTRAINT → especifica, sobre una o más columnas, que la inserción o actualización de una fila contiene un valor único en esa columna o conjunto de columnas.
    - Con esta restricción podemos representar AKs (claves alternas).
  - NOT NULL CONSTRAINT → asegura que una columna contenga un valor no nulo al hacer un INSERT o un UPDATE.
  - CHECK CONSTRAINT → especifica condiciones al hacer un INSERT o UPDATE en una columna.
    - Cada fila insertada o actualizada debe cumplir con dichas condiciones.

ÍNDICE → estructura opcional asociada a una tabla que permite acelerar las búsquedas.

- Son lógica y físicamente independientes de los datos de la tabla asociada.
- Se puede crear o borrar un índice en cualquier momento sin afectar la tabla base u otros índices.
  Se pueden crear distintos tipos de índices sobre uno o más campos.
- **Tipos:**
  - Árbol-B → estructura de índice estándar.
  - *Hashing* → índices implementados en tablas de *hash*, en donde se indexa por los campos que son accedidos con mayor frecuencia (se basan en otros índices Árbol-B existentes para una tabla).
  - Otros, como por ejemplo: árbol-B *Cluster*, *bitmap*, *functional*, *reverse key*.

  Diferencias entre Árbol-B y *Hashing*:

	Árbol-B	***Hashing***
Velocidad en el acceso a datos	Más veloz para accesos secuenciales.	Más veloz para accesos directos.
Espacio adicional requerido	Requiere una estructura de árbol con nodos y punteros.	Requiere por lo general una única tabla de índices.
Claves duplicadas…	Acepta tanto claves duplicadas como una clave que no sea completa.	Acepta claves duplicadas.

- **¿Cuándo es recomendable usarlos? – Buenas prácticas:**
  - Indexar columnas que intervienen en JOINs.
  - Indexar columnas donde frecuentemente se realicen filtros.
  - Indexar columnas que son frecuentemente usadas en la sección ORDER BY.
  - Evitar duplicación de índices, sobre todo en columnas con pocos valores diferentes.
  - Verificar que el tamaño de índice sea pequeño en comparación con la fila.
  - Limitar la cantidad de índices en tablas que son actualizadas frecuentemente.
- **Características diferenciadoras para los índices:**
  - Unique → índice de clave única → sólo admite una fila por clave.
  - Duplicado → permite múltiples filas para una misma clave.
  - Simple → la clave está integrada por una sola columna.
  - Compuesto → la clave se compone de varias columnas.
- **Ventajas:**
  - Mejor performance al equipo → en los casos que las columnas del SELECT no formen parte del índice, no se deben hacer lecturas secuenciales, sino que se accede a través de los índices.
  - Asegura únicos valores para las filas almacenadas.
  - Mejor performance en el ordenamiento de las filas.
  - Asegura el cumplimiento de *constraints* y reglas de negocio.
- **Desventajas:**
  - Costo de espacio de disco → en algunos casos ocupan mayor espacio que los datos.
  - Costo de procesamiento → cada vez que se inserta, actualiza o borra una fila, el índice debe estar bloqueado, por lo que el sistema deberá recorrer y actualizar los diversos índices.

VISTA → representación adaptada de los datos contenidos en una o más tablas o vistas.
→ definida por la salida resultante de una consulta.

- Es tratada como una tabla.
- No almacena nada, no tiene datos.
- Tiene un nombre específico.
- **Usos:**
  - Suministrar un nivel adicional de seguridad → se restringe el acceso a un conjunto predeterminado de filas o columnas de una tabla.
  - Ocultar la complejidad de los datos.
  - Simplificar sentencias al usuario.
  - Presentar los datos desde una perspectiva diferente a los de la tabla base.
  - Aislar a las aplicaciones de los cambios de la tabla base.
- **Restricciones:**
  - Una vista depende de las tablas a las que se haga referencia a ella → si se elimina una tabla, todas las vistas que dependen de ella se borrarán o se pasará a un estado inválido, dependiendo del motor.
  - Con la opción WITH CHECK OPTION, sólo se puede insertar o actualizar siempre que se cumpla la condición del WHERE definido en la vista.
  - Al crearse una vista, el usuario debe tener permiso de SELECT sobre las columnas de las tablas involucradas.
  - Algunas vistas tienen restringidos los INSERTs, los DELETEs y los UPDATEs.

TRIGGER → mecanismo que, ante un evento, dispara una acción (ejecuta automáticamente una sentencia SQL). El evento y la acción del *trigger* conforman una transacción.

- **Posibles eventos:**
  - Una instrucción DML (INSERT, UPDATE o DELETE) sobre tablas o vistas.
  - Una instrucción DDL (CREATE, ALTER o DROP) sobre la BD.
  - Una operación de BD (SERVERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN).
- **Posibles instantes de ejecución de las acciones:**
  - BEFORE → la acción se ejecuta antes de que el evento del *trigger* ocurra.
  - AFTER → la acción se ejecuta después de que el evento del *trigger* ocurra.
  - INSTEAD OF → la acción reemplaza el evento que disparó el *trigger* por la acción del *trigger*.
  - FOR EACH ROW → la acción se ejecuta para cada una de las filas del evento.
- **Usos:**
  - Auditoría.
  - Autorización de seguridad.
  - Replicación automática de tablas.
  - Reglas de negocio específicas.

STORED PROCEDURES → procedimientos programados que son almacenados como un objeto en la BD. Antes de ser almacenados en la BD, las sentencias son parseadas y optimizadas.

- Incluyen sentencias de SQL y sentencias de lenguaje propias.
- Luego de ser creados, pueden ser ejecutados por usuarios que posean los permisos respectivos.
- **Ventajas:**
    - Pueden reducir la complejidad en la programación (al crear SPs con las funciones más usadas).
    - Otorgan un nivel de seguridad adicional a nivel permisos.
    - Diferentes aplicaciones acceden al mismo código ya compilado y optimizado.
    - Menor tráfico a través de la red.

FUNCIONES DE USUARIO → objeto de una BD, programado en un lenguaje válido por el motor de BD, que puede recibir uno o más parámetros de entrada y devolver sólo un parámetro de salida.

Diferencias entre funciones de usuario y *stored procedures*:

- Un *stored procedure* **sí puede** actualizar datos de una o más tablas.
  Una función de usuario **no puede** actualizar nada (sí puede realizar una consulta de datos).
- Un *stored procedure* **no puede** usarse en un SELECT.
  Una función de usuario **sí puede** usarse en un SELECT, en un UPDATE o bien en un DELETE.
- Un *stored procedure* **puede** devolver 1 único parámetro como salida.
  Una función de usuario **debe** devolver sí o sí 1 único parámetro como salida.

FUNCIONES PROPIAS DEL MOTOR → funciones ya desarrolladas por el motor de BD; se clasifican en:

- Funciones agregadas → se aplican específicamente a un conjunto de valores derivados de una expresión.
    - Ejemplos: SUM, COUNT, AVG, MAX, MIN.
- Funciones escalares → se aplican a un dato específico de cada fila de una consulta o bien en una comparación en la sección WHERE.
    - Hay de varios tipos: algebraicas, trigonométricas, de fecha, de *strings*, de manejo de NULL, …
- Funciones de tablas → se aplican únicamente en la sección FROM, retornan el equivalente a una tabla.

SINÓNIMO → alias definido sobre una tabla, una vista, un *snapshot*, un *stored procedure*, una función, …

- Enmascaran el nombre y el dueño de determinado objeto.
- Simplifican las sentencias SQL para usuarios de la BD.
- Proveen una ubicación transparente para objetos remotos en una BD distribuida.
- Se usan a menudo por seguridad o por conveniencia (por ejemplo, en entornos distribuidos).

SECUENCIA → los generadores de secuencias proveen una serie de números secuenciales y únicos, especialmente usados en entornos multiusuarios, sin el *overhead* de I/O a disco o el *lockeo* transaccional.

- Los motores de BDs proveen diferentes formas de implementarlas, a través de:
    - Tipo de dato de una columna.
    - Propiedades de una columna.
    - Objeto *Sequence*.

BASES DE DATOS – *RDBMS*

BASE DE DATOS → conjunto de datos persistentes e interrelacionados que es utilizado por los sistemas de aplicación de una organización. Estos datos están almacenados en un conjunto independiente.

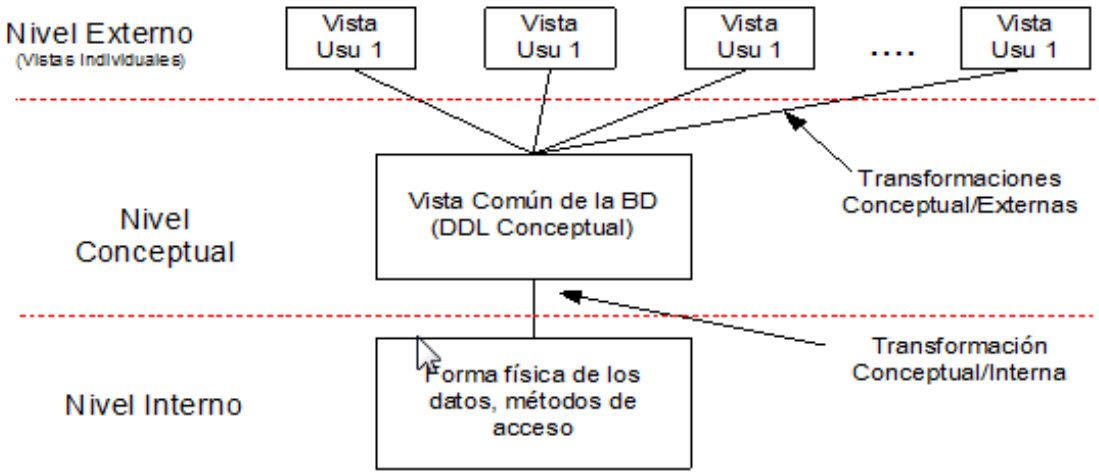
***RBDMS (Relational DataBase Management System)* · "Motor de BD"** → programa que permite administrar los contenidos de una o más BD/s almacenadas en disco.

- Ofrece a los usuarios una percepción de la BD que está en cierto modo por encima del nivel del hardware y que maneja las operaciones del usuario expresadas en el nivel más alto de percepción.
- Interpreta y ejecuta todos los comandos SQL.
- Ejemplos: Microsoft SQL Server, Oracle, MySql, PostgreSQL, DB2, Informix.
- **Componentes principales:**
    - Procesos:
        - Controlan el *RDBMS*.
        - Proveen funcionalidades específicas asociadas a seguridad, integridad, operación, organización interna y demás.
    - Memoria compartida:
        - Cachea datos del disco para tener un acceso más rápido.
        - Mantiene y controla recursos necesarios para los procesos.
    - Unidades de discos → colección de uno o más discos asignados al *RDBMS*.
        - En los discos se almacena toda la información de la/s BD junto con la información propia del sistema necesaria para mantener el *RDBMS*.
    - Usuarios → quienes se conectan a la BD.
        - Son los programadores, los DBAs, los usuarios finales, …

Enfoque de BD:

- Los archivos de datos (las tablas) residen en la BD.
- Las consultas de usuarios son enviadas al *RDBMS*, mediante algún mecanismo de comunicación estándar.
- El *RDBMS* realiza la consulta/actualización y devuelve el resultado del programa.
- El *RDBMS* se encarga de la seguridad y concurrencia.
- Las peticiones no se hacen al SO sino al *RDBMS* mediante instrucciones SQL.
- Ventajas:
    - Estándares de documentación y normalización de nomenclaturas.
    - Redundancia mínima.
    - Consistencia de datos → transacciones.
    - Concurrencia en acceso a datos.
    - Integridad de datos → restricciones y objetos de BD.
    - Criterios y normativas para organización de datos.
    - Independencia.

Arquitectura de un motor de BD (ANSI/SPARC):



- **Nivel Externo** → percepción que tienen los usuarios (usuario final, programador, DBA) respecto de la BD.
  - Una vista externa es el contenido de una BD como lo ve algún usuario en particular.
- **Nivel Conceptual** → representa de una forma "entendible" toda la información contenida en una BD.
  - Contiene definiciones del contenido de la BD, tipos de datos, restricciones, reglas de integridad, etc.
- **Nivel Interno** → representación de bajo nivel de toda la BD.
  - En este nivel se define cómo se almacenan los datos en disco.
- **Transformación externa–conceptual** → define la correspondencia entre las vistas externa y conceptual.
- **Transformación conceptual–interna** → define la correspondencia entre la vista conceptual y la BD almacenada y especifica la representación en el nivel interno de las filas y columnas del modelo conceptual.

Funciones del *RDBMS*:

- **Administrar un DICCIONARIO DE DATOS**
  - Diccionario de datos → conjunto de tablas de la BD del sistema que define cada uno de los objetos dentro de la misma. Estas tablas no pueden ser alteradas por ningún usuario de la BD.
  - Sentencias SQL relacionadas:
    - CREATE → creación de objetos de BD.
    - ALTER → modificación de objetos de BD.
    - DROP → eliminación de objetos de BD.

- **Manejo de la INTEGRIDAD de los datos** → procesos que se encargan de controlar que se cumplan ciertas restricciones.

  Esas restricciones pueden ser:

  - Restricciones en sí mismas (*constraints*): UNIQUE, NOT NULL, CHECK, DEFAULT, PK, FK.
  - Objetos creados:
    - *Triggers* → validan integridad entre tablas ante un INSERT, por ejemplo.
    - Índices → aseguran que el dato sea único.
    - Vistas → usando la cláusula WITH CHECK OPTION, me aseguro que ante un INSERT o un UPDATE sobre una vista, el INSERT o el UPDATE solamente se realiza si se cumple la condición del WHERE definida en la vista.

- **Manejo de la SEGURIDAD de los datos** → controlar que los usuarios estén autorizados para hacer lo que intentan.

  La seguridad refiere a dos conceptos:

  - Autenticación → asegurar que el usuario sea quien dice ser.
  - Autorización → asegurar que el usuario autenticado tenga los permisos adecuados.

  Entidades a tener en cuenta para administrar la seguridad:

  - Creación de usuarios y/o roles/perfiles → internos (de la BD) o externos (del SO).
  - Acciones posibles a realizar sobre los objetos.

  Sentencias SQL relacionadas → asignación/revocación de permisos:

  - GRANT → se usa para asignar o conceder permisos.
  - REVOKE → se usa para revocar permisos que previamente habían sido concedidos.

  Objetos relacionados:

  - Vistas → para ocultar cierto contenido de las tablas, por ejemplo.
  - *Triggers* → para prohibir el acceso a usuarios en un horario en particular, por ejemplo.
  - Sinónimos → usar un nombre para ocultar la ubicación real de las tablas en un servidor.
  - Creación de usuarios y/o roles/perfiles.

- **Manejo de la CONSISTENCIA de los datos**

  Transacciones → conjunto de sentencias SQL que se ejecutan atómicamente en una unidad lógica de trabajo → permiten que el *RDBMS* pueda rehacer una operación que no se ejecutó correctamente.

  - Cada transacción cumple con las siglas ACID:
    - A → Atomicidad → las transacciones se ejecutan en forma atómica, como un todo.
    - C → Consistencia → las transacciones no van a "romper" la BD.
    - I → Aislamiento → se pueden definir distintos niveles de aislamiento para cada transacción, independientemente de cada una.
    - D → Durabilidad → la transacción persiste cuando se ejecuta correctamente.
  - Para definir una transacción se debe definir un conjunto de instrucciones precedidas por la sentencia BEGIN TRANSACTION. Las sentencias a continuación se ejecutarán en forma atómica.

    Una transacción puede finalizar correctamente o puede fallar.
    - Si finaliza correctamente, todos los datos se actualizarán en la BD.
    - Si falla, se deshacen todos los cambios hasta el momento de la transacción.

    Para manejar estas acciones, existen dos sentencias:
    - COMMIT TRANSACTION → actualiza los datos en la BD.
    - ROLLBACK TRANSACTION → deshace la transacción.

  ***Logs* transaccionales** → registros donde el motor almacena la información de cada operación llevada a cabo con los datos.
  - Sirven para hacer el *recovery*.

- **Manejo de la CONCURRENCIA** → a través de los *locks* y los niveles de aislamiento.
    - ***Locks* (bloqueos)** → bloqueos sobre estructuras.
        - Granularidad de *locks*:
            - Nivel de BD.
            - Nivel de tabla.
            - Nivel de página.
            - Nivel de fila.
            - Nivel de clave de índice.
        - Tipos de *locks*:
            - Compartidos.
            - Exclusivos.
            - Promovibles.
    - **Niveles de aislamiento** de una transacción respecto de la otra (SQL SERVER en especial):
        - RU · READ UNCOMMITTED → el más permisivo.
            - Se lee toda la tabla sin importar si las filas están *lockeadas* o no.
        - RC · READ COMMITTED → más restrictivo que RU.
            - Si no hay filas *lockeadas*, el motor lee toda la tabla.
            - Si hay filas *lockeadas*, el *RDBMS* se queda esperando hasta que se liberan los locks sobre esas filas (mediante un COMMIT o un ROLLBACK). Una vez liberados los locks, se podrá leer la tabla.
        - RR · REPEATABLE READ → más restrictivo que RC.
            - Toda lectura *lockeará* los datos que lee → *lockea* con un SELECT.
            - Ante otra lectura sobre una fila *lockeada*, el motor se queda esperando hasta que se liberen los *locks* sobre esas filas (mediante un COMMIT o un ROLLBACK). Una vez liberados los *locks*, se podrá leer la tabla.
        - SR · SERIALIZABLE READ → el más restrictivo.
            - Impide que otras transacciones actualicen o inserten filas que satisfagan los requisitos de alguna de las instrucciones ejecutadas por la transacción actual.
            - Puede afectar a los demás usuarios en los sistemas multiusuario.

	RU	**RC**	**RR**	**SR**
Lecturas sucias	*Pueden ocurrir*	***NO HAY***	***NO HAY***	***NO HAY***
Lecturas no repetibles	*Pueden ocurrir*	*Pueden ocurrir*	***NO HAY***	***NO HAY***
Lecturas fantasmas	*Pueden ocurrir*	*Pueden ocurrir*	*Pueden ocurrir*	***NO HAY***

Una **lectura sucia** ocurre cuando se le permite a una transacción leer una fila que ha sido modificada por otra transacción concurrente pero que todavía no ha sido confirmada.

Una **lectura no repetible** ocurre cuando, durante una transacción, una fila se lee dos veces y los valores no coinciden.

Una **lectura fantasma** ocurre cuando, durante una transacción, se ejecutan dos consultas iguales y los resultados de la segunda son distintos a los de la primera.

- **Mecanismos de resguardo y de restauración (*BACKUP* y *RESTORE)***
    - **Resguardo (*BACKUP*)** → copia total o parcial de información de una BD, la cual debe ser guardada en algún otro sistema de almacenamiento masivo.
        - *Backup* completo → se guardan todos los datos.
        - *Backup* diferencial → solamente se guardan las modificaciones a partir…
            - Diferencial acumulativo → … del último *backup* completo.
            - Diferencial incremental → … del último *backup*, sea o no completo.
        - *Backup* en caliente → se realiza mientras el sistema está en uso.
        - *Backup* en frío → se realiza mientras el sistema no está en uso.
        - *Backup* de *logs* transaccionales → se realizan sobre los *logs* transaccionales.
    - **Restauración *(RESTORE)*** → acción de tomar un *backup* ya realizado y restaurar la estructura y los datos sobre una BD dada.

- **Mecanismos de recuperación (*RECOVERY)*** → se ejecutan en cada inicio del motor de forma automática como dispositivo de tolerancia a fallos.
    - Retorna al *RDBMS* al punto consistente más reciente: al último *checkpoint* → punto en el cual el *RDBMS* sincronizó memoria y disco.
    - Utiliza los *logs* transaccionales para retornar el motor de BD a un estado lógico consistente:
        - realizando un "*roll forward*" de las transacciones ocurridas con éxito después del *checkpoint* más reciente, o bien;
        - realizando un "*roll back*" de las transacciones que no hayan sido exitosas después del *checkpoint* más reciente.

- **Optimizador de consultas** → intenta determinar la forma más eficiente de ejecutar una consulta SQL.
    - Antes de ejecutar la consulta, el *RDBMS* la optimiza. El resultado de dicha optimización es lo que se conoce como **plan de ejecución**.
    - El principal objetivo del plan de ejecución es mejorar la *performance* de las consultas.

- **Facilidades de AUDITORÍA** → los *RDBMS* tienen la opción de registrar el uso de los recursos de la BD para posteriores auditorías (cuándo se prendió el motor, cuándo se apagó, cuándo un usuario se *loggeó*, …).

- **Creación de *triggers*, *stored procedures* y funciones**.

- ***Mirroring* de discos**.

- ***Event alarms*** → útiles para enviar mails o mensajes de WhatsApp, por ejemplo.

- **Logs del sistema** → mediante ellos el DBA puede determinar la causa de una caída del sistema, por ejemplo.

- **Adaptación a cambios/crecimientos/modificaciones del esquema** → el *RDBMS* permite realizar cambios a las tablas constantemente, casi en el mismo instante en que la tabla está siendo consultada.

- ***Data Replication*** → útil para aumentar la disponibilidad del sistema en general.

- ***Data Encryption*** → la encriptación es de suma utilidad para aumentar la seguridad de los datos.

BUSINESS INTELLIGENCE

- Proceso que utiliza los datos internos/externos de una organización para analizar los posibles escenarios y proyectar futuras situaciones.
- Conjunto de metodologías, herramientas y estructuras de almacenamiento que permiten la transformación DATOS → INFORMACIÓN → CONOCIMIENTO con la intención de optimizar el proceso de toma de decisiones de la organización.
- Es un factor estratégico para la organización que genera una ventaja competitiva, que se utilizará para captar nuevos mercados, clientes o para la producción de nuevos productos o servicios.

BD MULTIDIMENSIONALES

→ BD donde la información se almacena en forma dimensional, no relacional.

- Las dimensiones determinan la estructura de la información almacenada.
- La información reunida se muestra como variables que, a la vez, se determinan por una o más dimensiones. De esa manera, y a partir de la intersección de esas dimensiones, se almacena el valor.
- Permiten observar cada uno de los patrones de interés que se evaluarán como una dimensión determinada.
- Contribuye a que el usuario obtenga una visión analítica de la información → facilita el procesamiento.

HIPERCUBO → único cubo como estructura de almacenamiento de datos; es una estrategia para evitar la dispersión de datos (celdas vacías en las BDs multidimensionales).

- Permiten que se introduzcan los valores de los datos a través de la combinación de dimensiones.
- El acceso a la información es veloz → la información está almacenada en forma contigua.

TECNOLOGÍAS OLTP–OLAP

- Son utilizadas simultáneamente por las organizaciones que buscan una correcta planificación de su futuro.
- Trabajan en conjunto con un *DATA WAREHOUSE*, de donde extraen datos.

OLTP → orientados al procesamiento de transacciones.

- Focalizado en resolver requerimientos de aplicaciones específicas.
- Las BDs se optimizan para la actualización de las transacciones.
- Sistemas no integrados → cada área del negocio tiene su propio modelo.

OLAP → orientados al procesamiento analítico de la información sobre determinados patrones de interés.

- Focalizado en resolver requerimientos de análisis estratégico de la organización.
- Las BDs se optimizan para el análisis.
- Toda la información de interés se integra → análisis de la información en su conjunto.
- Se utiliza para:
  - Analizar información.
  - Elaborar reportes administrativos.
  - Analizar rentabilidad.
  - Reportes de calidad.
  - Otras aplicaciones que necesiten una visión exhaustiva del negocio.

MODELO *STAR* → técnica de modelado de datos que se utiliza para hacer corresponder un modelo multidimensional sobre una BD relacional.

DATA WAREHOUSE

- BD corporativa que integra y filtra información de una o más fuentes para luego ser procesada para su análisis desde diferentes puntos de vista y con una gran velocidad de respuesta.
- Colección de datos históricos e integrados diseñada para soportar el procesamiento informático.
- Herramienta excelente para la toma de decisiones estratégicas que no se utilizan para la operatoria diaria.
- Es el primer paso para la implementación de una solución completa y sustentable de *BUSINESS INTELLIGENCE*.
- Características:
    - Está orientado a sujetos.
    - Es integrado.
    - Es temático.
    - Es variante en el tiempo.
    - Es simple de manejar.
    - No es volátil

DATA MART → vista multidimensional de cada área del *DATA WAREHOUSE*.

A diferencia del *DATA WAREHOUSE*, los *DATA MARTS*:

- Son pequeños sistemas de almacenamiento de datos.
- Se ajustan mejor a las necesidades que tiene una parte específica de la organización.
- Optimizan la distribución de información útil para la toma de decisiones.
- Se enfocan en el manejo de datos resumidos o de muestras (no en *la historia* en detalle).
- Tienen un alcance más limitado que los DW.
- Son ideales para trabajar con objetivos puntuales y equipos de trabajo específicos.

DATA MINING → conjunto de técnicas avanzadas utilizadas para la obtención de información implícita de las grandes BDs.

- Busca patrones de interés ocultos, que son los que permiten la anticipación de futuros acontecimientos. Estas predicciones guiarán la toma de decisiones.
- Las herramientas del *DATA MINING*:
    - Encuentran la información que los datos ocultan (mediante una "explosión" de datos) a través de la combinación de todos los patrones de interés entre sí.
    - Predicen tendencias y comportamientos que contribuyen a una toma de decisiones proactiva que beneficia el desempeño de los negocios.
    - Responden preguntas de negocios cuya resolución conlleva tradicionalmente un tiempo considerable → usualmente los usuarios no están dispuestos a aceptar tales respuestas.
    - Exploran las BDs en busca de patrones ocultos, encontrando información predecible que un experto no podría encontrar porque está fuera de sus expectativas.