Conceptos - Codd

Estructura: recopilación de objetos o relaciones
Manipulación de datos: operaciones sobre las relaciones para producir otras.
Integridad: para obtener precisión y consistencia

Dominio
Es el conjunto de valores posibles que puede tomar una columna (campo o atributo) de una tabla. Los dominios son la menor unidad de semántica de información desde el punto de vista del modelo, son atómicos, o sea que no se pueden "descomponer". En consecuencia un dominio es un conjunto de valores escalares, todos del mismo tipo. Observación: como nulo no es un valor, los dominios no contienen nulos.

- Menor unidad semántica de información.
- Atómicos (no se pueden descomponer sin perder significado).
- Conjunto de valores escalares de igual tipo.
- No contienen nulos.

Se implementa:
- Nombre de columna: debe ser definido claramente para que todos entiendan *qué* contiene ese campo
- Tipos de datos: int, datetime, varchar, etc
- Constraints: restringen los valores posibles dentro del tipo de dato.
    - NULL / NOT NULL
    - DEFAULT
    - CHECK
    - PRIMARY KEY, UNIQUE
    - CLAVES FORÁNEAS (Integridad referencial)
- Triggers: restringen el dominio mediante procedimientos que validan reglas de negocio.

<u>Implementación</u>
- Restricciones de chequeo
- Claves foráneas
- Definición de “tipos de datos de usuario”
- Definición de convenciones de nombres para los atributos

Relación
Subconjunto del producto cartesiano de los dominios de valores involucrados.
- Cabecera: conjunto fijo de pares atributo - dominio.
- Cuerpo: conjunto de tuplas que varían con el tiempo.
    - Tupla: conjunto de pares atributo - valor.
- Tabla: representación de una relación.

- Base de datos relacional: base de datos percibida por el usuario como una colección de relaciones normalizadas de diversos grados que varía con el tiempo.

Propiedades
- No hay tuplas repetidas
    - Las tuplas no están ordenadas
- Los atributos no están ordenados
- Toda relación tiene clave primaria
- Los valores de los atributos son atómicos
    - La relación está normalizada

Tipos
- Tablas - relaciones base
- Resultados de consultas
- Resultados intermedios de consultas
- Vistas
- Instantáneas o snapshots
- Tablas temporales

Integridad
Clave primaria + clave foránea -> Reglas de Integridad.
El término integridad de datos se refiere a la correctitud y completitud de la información en una base de datos.
Cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.

Clave primaria
Atributo (o conjunto de atributos) identificador único para la relación. Se parte de un conjunto de claves que se denominan candidatas. Las mismas deben cumplir minimalidad y unicidad. Aquellas no seleccionadas como primarias se denominan claves alternas.

Regla de integridad de las entidades
Ningún componente de la clave primaria de una relación base puede aceptar nulos.

- Una tupla es un elemento de una relación, que representa al mundo real.
- Las claves primarias, son el mecanismo de identificación en el modelo relacional.
- Una base de datos relacional no admite registrar información acerca de algo que no se puede identificar.

Clave foránea
Atributo (o conjunto de atributos) de una relación (R2) cuyos valores (no nulos) deben coincidir con los de la clave primaria de una relación (R1).

- La clave primaria en R1 y foránea en R2 están definidas sobre el mismo dominio
- La clave foránea puede o no ser parte de la clave primaria de R2.
- R1 y R2 no necesariamente son distintas. Si R1=R2 existe una relación autorreferencial.
- Las claves foráneas deben en ciertas ocasiones aceptar nulos.

Regla de Integridad Referencial
La base de datos no debe contener valores no nulos de clave foránea para los cuales no exista un valor concordante de clave primaria en la relación referenciada.

- Mantener un estado consistente de la base de datos.
- Determinar acciones a llevar a cabo ante operaciones que puedan violar la integridad referencial
    - Eliminación: RESTRICT / CASCADE / SET NULL
    - Modificación: RESTRICT / CASCADE / SET NULL
    - Inserción: RESTRICT

Independencia de Datos
Independencia lógica: Cada aplicación requiere una vista diferente de los mismos datos y no requiere saber todos los atributos de una tabla o el orden real de dichos atributos, ni la distribución de atributos a través de las tablas. Se podrían hacer cambios en tablas sin que afecten a las aplicaciones que no los requieren.

Independencia física: Es posible modificar la estructura de almacenamiento, la distribución física o la técnica de acceso sin afectar las aplicaciones. (crear un índice por nombre para que se resuelva el listado más rápidamente, crear un objeto determinado asociado a una tabla , etc.).

Reglas de Codd
- Independencia entre el motor de base de datos y los programas que acceden a los datos (es posible modificar el motor de base de datos o los componentes de 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 base, no en programas de aplicación.
- Soportar información faltante mediante valores nulos (NULL)
- Proveer lenguajes para:
    - Definición de datos

    - Manipulación de datos, 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.

Terminología

- Tabla: Estructura básica de almacenamiento. Es lo que Codd denominaba relación. También se la denomina entidad.
- Fila: Todos los datos para cada ocurrencia en la tabla. Es lo que Codd denomina tupla.
- Columna: identifica un dato de la tabla. Puede ser clave o no clave. Las claves pueden ser primarias o foráneas.
- Campo: intersección de fila y columna. Pueden ser o no ser nulos.

Definición de Base de Datos

Una base de datos es un conjunto de datos persistentes e interrelacionados que es utilizado por los sistemas de aplicación de una empresa. Los mismos se encuentran almacenados en un conjunto independiente y sin redundancias o con redundancias mínimas.

Definición de RDBMS

Es un programa que permite administrar los contenidos de una/s base/s de datos almacenada en disco. También llamado Motor de Base de Datos.

El DBMS ofrece a los usuarios una percepción de la base de datos 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.

El DBMS también interpreta y ejecuta todos los comandos SQL que le son enviados.

Entre los motores de Base de Datos más utilizados podemos nombrar los siguientes: Oracle, MS SqlServer, MySql, PostgreSQL, DB2, Informix, Sybase, SQLite, entre otros.

Componentes principales de un DBMS

- Procesos
    - Controlan el motor de Base de Datos
    - Proveen funcionalidades específicas asociadas a seguridad, integridad, operación, organización interna entre otros.
- Memoria Compartida: es usada para:
    - Cachear datos del disco para tener un acceso más rápido.
    - Mantener y controlar los recursos necesarios para los procesos.

    - Proveer mecanismos de comunicación para los procesos clientes o propios del motor para conversar con otras aplicaciones y coordinar actividades.
- Unidades de Discos
    - Es la colección de una o más unidades de espacio de disco asignadas al DBMS.
    - Toda la información de las bases de datos más la información propia del sistema necesarias para mantener el DBMS están almacenadas en este componente.

Ventajas del enfoque de base de datos

- Seguridad
- Estándares de documentación y normalización de nomenclaturas.
- Redundancia mínima:
- Consistencia de datos
    - Transacciones
- Concurrencia en acceso a datos.
- Integridad de los datos
    - Restricciones
    - Objetos de BD
- Criterios y normativas para organización de datos
- Independencia

Arquitectura de un sistema de bases de datos (ANSI/SPARC)

- Nivel externo: vistas individuales. Esta es la percepción que tienen los usuarios respecto de la BD. Los usuarios pueden ser programadores, usuarios finales, o el DBA. La percepción que tenemos como usuarios de una base de datos es solamente una vista externa. Una vista externa es el contenido de una Base de datos como lo ve algún usuario en particular (para este usuario la vista es la Base de Datos).
- Nivel conceptual: vista común de la BD (DDL conceptual). Representa de una forma 'entendible' toda la información contenida en una base de datos. En lugar de describir datos personales, describe los datos de toda la organización. La vista conceptual se define mediante un esquema conceptual. Este esquema conceptual se escribe en DDL. Contiene definiciones del contenido de la base, , tipos de datos, restricciones, reglas de integridad, etc.
- Nivel interno: forma física de los datos, métodos de acceso. En este nivel se define cómo se almacenan los datos en disco, es una representación de bajo nivel de toda la base de datos. Por ejemplo, se especifican las estructuras internas de disco y memoria, se definen métodos de acceso, en qué secuencia física se encuentran los registros, entre otras.

- Transformación Externa/Conceptual: dada una determinada vista externa, esta transformación define la correspondencia entre dicha vista externa y la vista conceptual.
- Transformación Conceptual / Interna: define la correspondencia entre la vista conceptual y la base de datos almacenada, y específica la representación en el nivel interno de las filas y columnas del modelo conceptual.

Funciones del motor de base de datos y del DBMS en su conjunto

Diccionario de datos
Es un conjunto de tablas de la base de datos del sistema, que define cada uno de los objetos dentro del mismo. Es lo que se llama 'metadatos‘.
Estas tablas no pueden ser alteradas por ningún usuario de la base de datos.
Por ejemplo de una Tabla, tiene almacenado todos los parámetros propios de la definición de la tabla, sus campos y sus restricciones (constraints), así como las relaciones entre las tablas.
Algunas sentencias SQL relacionadas con esta funcionalidad

- CREATE –Creación de objetos de BD (Tablas, índices, vistas, etc.)
- ALTER –Modificación de objetos de BD
- DROP –Eliminación de objetos de BD.

Control de la seguridad
La seguridad involucra la autenticación: asegurar que el usuario existe y su clave es correcta permitiendo el acceso a la BD y la autorización determinar los permisos que tiene el usuario para ejecutar acciones sobre diferentes objetos de la BD.
Para ello los DBMS poseen lo que se llama el catálogo. El catálogo mismo está formado por entidades e interrelaciones (por lo que en un esquema relacional van a ser tablas).

<u>Entidades a tener en cuenta para administrar la seguridad</u>

- Usuarios (internos, externos, mixtos)
- Roles o Perfiles (built-in roles / user defined roles)
- Acciones posibles a realizar sobre los objetos.
- Objetos de la BD sobre los que se otorgarán permisos.

Algunas sentencias SQL relacionadas con la seguridad

- GRANT –Para otorgar permisos.
- REVOKE –Para revocar permisos

Otros objetos relacionados con la seguridad

- Vistas, Triggers, Stored Procedures, Sinónimos, Funciones.

Mecanismos para garantizar la integridad de datos
El DBMS cuenta con distintos mecanismos para poder controlar la integridad de los datos que se contienen en la/s Base/s de Datos.
Para asegurar la integridad cuenta con distintas restricciones y objetos que pueden ser creados y a partir de ello el DBMS cuenta con procesos que se encarga de controlar que se cumplan dichas restricciones asegurando la integridad de los datos.

Otros objetos relacionados con la integridad

- CONSTRAINTS
- UNIQUE, NOT NULL, CHECK, DEFAULT
- PRIMARY KEY
- FOREIGN KEY
- Triggers
- Índices (únicos)
- Views (with check option)
- Stored Procedures
- Funciones

Mecanismos para garantizar la consistencia de datos

El DBMS cuenta con distintos mecanismos para poder asegurar la consistencia de los datos que existen en la/s Base/s de Datos.

Conceptos relacionados con la consistencia de datos

- TRANSACCIONES: Es un conjunto de sentencias SQL que se ejecutan atómicamente en una unidad lógica de trabajo. Partiendo de que una transacción lleva la base de datos de un estado correcto a otro estado correcto, el motor posee mecanismos de manera de garantizar que la operación completa se ejecute o falle, no permitiendo que queden datos inconsistentes.
  Cada sentencia de alteración de datos (insert, update o delete) es una transacción en sí misma (singleton transaction).
- LOGS TRANSACCIONALES: Es un registro donde el motor almacena la información de cada operación llevada a cabo con los datos
- RECOVERY: Método de recuperación ante caídas.

Para definir una transacción debemos definir un conjunto de instrucciones precedidas por la sentencia BEGIN TRANSACTION, de esta manera las sentencias a continuación se ejecutarán de forma atómica. Pero para lo que es el concepto de transacción, una transacción puede finalizar correctamente o puede fallar; en el caso de finalizar correctamente, todos los datos se actualizarán en la base y en caso de fallar se deberían deshacer todos los cambios hasta el comienzo de la transacción. Para manejar estas acciones contamos con la sentencia COMMIT TRANSACTION para actualizar los datos en la base de datos, y con la sentencia ROLLBACK TRANSACTION para deshacer nuestra transacción.

Mecanismos de recuperación (RECOVERY)

Es un mecanismo provisto por los motores de Base de Datos que se ejecuta en cada inicio del motor de forma automática como dispositivo de tolerancia a fallas. Sus objetivos son los siguientes:

- Retornar al Motor de Base de datos al punto consistente más reciente. (checkpoint = punto en el que el motor sincronizó memoria y disco)

- Utilizar los logs transaccionales para retornar el motor de base de datos a un estado lógico consistente, realizando un “rolling forwards” de las transacciones ocurridas con éxito luego del checkpoint más reciente y realizar un “rolling back” de las transacciones que no hayan sido exitosas.

Mecanismos de resguardo y restauración (BACKUP y RESTORE)
Los motores de BD cuentan con distintas herramientas que permiten realizar estas acciones las cuales son utilizadas generalmente en empresas medianas y chicas. Existen herramientas de terceras partes especializadas en el tema las cuales son muy utilizadas sobre todo en grandes empresas. (Por ej. Veritas, IBM Tivoli)

Backup: es la copia total o parcial de la información de una base de datos la cual debe ser guardada en algún otro sistema de almacenamiento masivo.
Restore: es la acción de tomar un back up ya realizado y restaurar la estructura y datos sobre una base dada.

De acuerdo a su clasificación podemos distinguir los siguientes tipos:
- Backup diferencial acumulativo / incremental: sólo se guardan las modificaciones a partir de cierta fecha.
- Backup completo: se guardan todos los datos.
- Backup en caliente: se realiza mientras el aplicativo está en uso.
- Backup en frío: se realiza con el aplicativo bajo.
- Backup de Logs Transaccionales: se realizan sobre los logs de transacciones.

Control de concurrencia
Los motores permiten controlar el acceso concurrente a sus recursos a través de locks y de la definición de niveles de aislamiento (isolation levels).

Locks
- Granularidad
    - Nivel de base de datos
    - Nivel de tabla
    - Nivel de página
    - Nivel de fila
    - Nivel de clave de índice
- Tipos
    - Compartido
    - Exclusivo
    - Promovible

Niveles de aislamiento

Read uncommitted

No chequea locks por *select* en la tablas a consultar, lo que mejora el rendimiento pero afecta la integridad en cuanto a que existen lecturas sucias (datos actualizados en una transacción que luego se deshacen por un rollback) y no existen lecturas repetibles (en la misma transacción poder hacer dos veces la misma consulta asegurando siempre el mismo resultado).

Pueden existir lecturas sucias, lecturas repetibles, y lecturas fantasmas (Filas que aparecen durante la transacción que fueron insertadas en otra transacción concurrente).

Read committed
El read committed asegura que sólo leerá datos confirmados por otra transacción, o sea realizará lecturas sucias.
Pero si bien el read commit ante una lectura de datos chequea la existencia de locks exclusivos en la tabla a consultar, no asegura lecturas repetibles. Ya que una vez que leyó los datos, los datos podrían ser bloqueados o modificados por otra transacción concurrente.

Lecturas no repetibles: en una misma transacción durante su transcurso puede tener dos llamadas a un mismo select, las cuales arrojaron resultados distintos.

Repeatable read
Las lecturas repetibles aseguran que no existan lecturas sucias y que las lecturas pueden ser repetibles, pero no evitan las lecturas fantasmas. Este nivel de aislamiento establece locks en los selects para todos los registros consultados, durante la duración de la transacción. Cuando otra transacción intenta modificar o borrar algún registro que está bloqueado por este select, quedará en espera.

Las lecturas fantasmas podrán aparecer, ya que la inserción de nuevos registros por una segunda transacción no asegurará el bloqueo de los mismos hasta que no sean nuevamente consultados por la transacción original.

Serializable read
Es el único que asegura que no existan lecturas sucias, lecturas fantasmas y que las lecturas puedan ser repetibles. Se realizará un bloqueo de un rango de índice, según lo que se coloque en el where, y si no es posible bloqueará toda la tabla.
El problema es que se aplicará un nivel de bloqueo que puede afectar a los demás usuarios en los sistemas multiusuario.

	Dirty Read	**Repeteable Read**	**Phantom Read**
Read Uncomited	**Sí**	**No**	**Sí**
Read Committed	**No**	**No**	**Sí**

Repeatable Read	**No**	**Sí**	**Sí**
Serializable Read	**No**	**Sí**	**No**

Dirty Read (lectura sucia): Datos actualizados en una transacción concurrente que luego se deshacen por un rollback.
Phantom Read (lectura fantasma): Filas que aparecen durante la transacción que fueron insertadas en otra transacción concurrente.
Lectura Repetible: En la misma transacción poder hacer dos veces la misma consulta asegurando siempre el mismo resultado.

Facilidades de auditoría
A los DBMS se les puede activar la opción de guardar en un log un registro del uso de los recursos de la base para auditar posteriormente.

Logs del sistema
Mediante este tipo de logs, el DBA puede llegar a determinar cual fue, por ejemplo, el problema que produjo una caída del sistema.

Acomodarse a los cambios, crecimientos, modificaciones del esquema.
El motor permite realizar cambios a las tablas constantemente, casi en el mismo momento en que la tabla está siendo consultada.

Normalización / Desnormalización

Normalizar una base de datos significa transformar un conjunto de datos que tienen una cierta complejidad en su entendimiento y que su distribución en el modelo provoca problemas de lógica en las acciones de manipulación de datos, en una estructura de datos que posea un diseño claro, donde estos datos guarden coherencia y no pierdan su estado de asociación.

Objetivos de la normalización

- Reducir la redundancia de datos, y por lo tanto, las inconsistencias.
- Facilitar el mantenimiento de los datos.
- Evitar anomalías en la manipulación de datos.
- Reducir el impacto de los cambios en los datos.

Las formas normales son heurísticas o criterios que permiten resolver esas redundancias. En algunas bibliografías se habla de errores o inconsistencias, es verdad que las redundancias pueden surgir por errores en el diseño de los datos, pero veremos a continuación que esa redundancia puede ser buscada.
Cada forma normal introduce restricciones nuevas, donde la primera restricción que aplica es cumplir la forma normal anterior.

Primera Forma Normal
Una entidad que está en primera forma normal no puede tener campos repetitivos (arrays, mismo campo repetido ‘n’ veces), o campos multivaluados.

Segunda Forma Normal
Una estructura de datos está en 2FN si y sólo si no hay dependencias funcionales entre los atributos claves.

Tercera Forma Normal
Una estructura de datos está en 3FN si y sólo si no hay dependencias funcionales entre los atributos no claves.

Objetos de Bases de Datos

Tablas

Es la unidad básica de almacenamiento de datos. Los datos están almacenados en filas y columnas. Son de existencia permanente y poseen un nombre identificatorio único por esquema o por base de datos (dependiendo del motor de base de datos). Cada columna tiene entre otros datos un nombre, un tipo de datos y un ancho (este puede estar predeterminado por el tipo de dato).

Tablas temporales

Son tablas creadas cuyos datos son de existencia temporal. No son registradas en las tablas del diccionario de datos. No es posible alterar tablas temporarias. Si eliminarlas y crear los índices temporales que necesite una aplicación. Las actualizaciones a una tabla temporal podrían no generar ningún log transaccional si así se configurara.

Tipos de tablas

- De sesión (locales): son visibles sólo para sus creadores durante la misma sesión (conexión) a una instancia del motor de BD. Las tablas temporales locales se eliminan cuando el usuario se desconecta o cuando decide eliminar la tabla durante la sesión.
- Globales: están visibles para cualquier usuario y sesión una vez creadas. Su eliminación depende del motor de base de datos que se utilice.

Tipos de creación

- Explícita: Este tipo de creación se realiza mediante la instrucción CREATE. De manera explícita se deberá crear la tabla indicando el nombre, sus campos, tipos de datos y restricciones.
- Implícita: creadas a partir de una consulta SELECT.

Por qué utilizarlas

- Como almacenamiento intermedio de Consultas Muy Grandes: Por ejemplo, se tiene una consulta SELECT que realiza “JOINS” con ocho tablas. Muchas veces las consultas con varios “JOINS” pueden funcionar de manera poco performante. Una técnica para intentar es la de dividir una consulta grande en consultas más pequeñas. Si usamos tablas temporales, podemos crear tablas con resultados intermedios basados en consultas de menor tamaño, en lugar de intentar ejecutar una consulta única que sea demasiado grande y múltiples “JOINS”.
- Para optimizar accesos a una consulta varias veces en una aplicación: Por ejemplo, usted está utilizando una consulta que tarda varios segundos en ejecutarse, pero sólo muestra un conjunto acotado de resultados, el cual desea utilizar en varias áreas de su procedimiento almacenado, pero cada

vez que se llama se debe volver a ejecutar la consulta general. Para resolver esto, puede ejecutar la consulta una sola vez en el procedimiento, llenando una tabla temporal, de esta manera se puede hacer referencia a la tabla temporal en varios lugares en su código, sin incurrir en una sobrecarga de resultados adicional.

- Para almacenar resultados intermedios en una aplicación: Por ejemplo, usted necesita en un determinado proceso generar información que se irá actualizando y/o transformando en distintos momentos de la ejecución, sin querer actualizar o impactar a tablas reales de la BD hasta el final del procedimiento. Para resolver esto, puede crear una tabla temporal de sesión durante la ejecución del procedimiento, realizando en ella inserciones, modificaciones, borrado y/o transformación de datos. Al llegar al final del procedimiento, con los datos existentes en la tabla temporal se actualizará la o las tablas físicas de la BD que corresponda.

Constraints

Integridad de Entidad

La integridad de entidades es usada para asegurar que los datos pertenecientes a una misma tabla tienen una única manera de identificarse, es decir que cada fila de cada tabla tenga una primary key capaz de identificar unívocamente una fila y esa no puede ser nula

PRIMARY KEY CONSTRAINT: Puede estar compuesta por una o más columnas, y deberá representar unívocamente a cada fila de la tabla. No debe permitir valores nulos (depende del motor de base de datos).

Integridad Referencial

La integridad referencial es usada para asegurar la coherencia entre datos de dos tablas.

FOREIGN KEY CONSTRAINT: Puede estar compuesta por una o más columnas, y estará referenciando a la PRIMARY KEY de otra tabla.
Los constraints referenciales permiten a los usuarios especificar claves primarias y foráneas para asegurar una relación PADRE-HIJO (MAESTRO-DETALLE).

Tipos de constraints referenciales

- Cyclic referential constraint
- Self referencing constraint
- Multiple path constraint

Integridad semántica

La integridad semántica es la que nos asegura que los datos que vamos a almacenar tengan una apropiada configuración y que respeten las restricciones definidas sobre los dominios o sobre los atributos.

- DATA TYPE: Este define el tipo de valor que se puede almacenar en una columna.
- DEFAULT: Es el valor insertado en una columna cuando al insertar un registro ningún valor fue especificado para dicha columna. El valor default por default es el NULL. Se aplica a columnas no listadas en una sentencia INSERT.
  El valor por default puede ser un valor literal o una función SQL (USER, TODAY, etc.). Aplicado sólo durante un INSERT (NO UPDATE).
- UNIQUE: Específica sobre una o más columnas que la inserción o actualización de una fila contiene un valor único en esa columna o conjunto de columnas.
- NOT NULL: Asegura que una columna contenga un valor durante una operación de INSERT o UPDATE. Se considera el NULL como la ausencia de valor.
- CHECK: Especifica condiciones para la inserción o modificación en una columna.
  - Cada fila insertada en una tabla debe cumplir con dichas condiciones. Actúa tanto en el INSERT, como en el UPDATE. Es una expresión que devuelve un valor booleano de TRUE o FALSE.
  - Todas las columnas a las que referencia deben ser de la misma tabla (la corriente).
  - No puede contener subconsultas, secuencias, funciones (de fecha, usuario) ni pseudocolumnas.
  - Todas las filas existentes en una tabla deben pasar un nuevo constraint creado para dicha tabla. En el caso de que alguna de las filas no cumpla, no se podrá crear dicho constraint o se creará en estado deshabilitado.

Secuencias

Los generadores de secuencias proveen una serie de números secuenciales, especialmente usados en entornos multiusuarios para generar una números secuenciales y únicos sin el overhead de I/O a disco o el bloqueo transaccional.

Views

Una view es un conjunto de columnas, ya sea reales o virtuales, de una misma tabla o no, con algún filtro determinado o no.

De esta forma, es una presentación adaptada de los datos contenidos en una o más tablas, o en otras vistas. Una vista toma la salida resultante de una consulta y la trata como una tabla.

Se pueden usar vistas en la mayoría de las situaciones en las que se pueden usar tablas.

Características

- Tiene un nombre específico.
- No aloca espacio de almacenamiento
- No contiene datos almacenados
- Está definida por una consulta realizada a una o varias tablas.

Usos

- Suministrar un nivel adicional de seguridad restringiendo 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 la de la tabla base.
- Aislar a las aplicaciones de los cambios en la tabla base.

Restricciones

- Al crear la view el usuario debe tener permiso de select sobre las columnas de las tablas involucradas.
- Tienen restringido los inserts, updates y deletes aquellas que tengan joins, una función agregada o un trigger instead of.
- En cuanto a updates:
    - Si en la tabla existieran campos que no permiten nulos y en la view no aparecen, los inserts fallarían.
    - Si en la view no aparece la primary key los inserts podrían fallar.
    - Se puede borrar filas desde una view que tenga una columna virtual.
    - Con la opción WITH CHECK OPTION, se puede actualizar siempre y cuando el checkeo de la opción en el where sea verdadero.

Índices

Los índices son estructuras opcionales asociadas a una tabla.
La función de los índices es la de permitir un acceso más rápido a los datos de una tabla, se pueden crear distintos tipos de índices sobre uno o más campos.
Los índices son lógica y físicamente independientes de los datos en la tabla asociada. Se puede crear o borrar un índice en cualquier momento sin afectar a las tablas base o a otros índices.

Tipos de índices

- Btree Index: estructura de índice estándar y más utilizada.
- Btree Cluster Index: este tipo de índice provoca al momento de su creación que físicamente los datos de la tabla sean ordenados por el mismo
- Bitmap Index: son utilizados para pocas claves con muchas repeticiones.
  Cada bit en el Bitmap corresponde a una fila en particular.
  Si el bit está en on significa que la fila con el correspondiente rowid tiene el valor de la clave.

- Hash Index: están implementados en tablas de hash y se basan en otros índices Btree existentes para una tabla. Si una tabla entra íntegramente en memoria, la manera más rápida de ejecutar consultas sobre ella es usando un hash index.
- Functional Index / Function based Index: son indices cuya clave deriva del resultado de una función.En general las funciones deben ser funciones definidas por un usuario.
- Reverse Key Index: invierte los bytes de la clave a indexar. Esto sirve para los índices cuyas claves son una serie constante con por ej. Crecimiento ascendente para que las inserciones se distribuyan por todas las hojas del árbol de índice.

Características

- Unique: índice de clave única. Solo 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. Facilitan múltiples joins entre columnas e incrementan la unicidad del valor de los índices.

Beneficios

- Se le provee al sistema mejor performance al equipo ya que no debe hacer lecturas secuenciales sino accede a través de los índices, sólo en los casos que las columnas del Select no formen parte del índice.
- Mejor performance en el ordenamiento de filas
- Asegura únicos valores para las filas almacenadas
- Cuando las columnas que intervienen en un JOIN tienen índices se le da mejor perfomance si el sistema logra recuperar los datos a través de ellas
- Asegura el cumplimiento de constraints y reglas de negocio (Primary key, foreign keys, unique values).

Costos

- Costo de espacio de disco: en algunos casos suele ser mayor al que ocupan los datos.
- Costo de procesamiento y mantenimiento: cada vez que una fila es insertada o modificada o borrada, el índice debe estar bloqueado, con lo cual el sistema deberá recorrer y actualizar los distintos índices.

Guía

- Indexar columnas que intervienen en Joins
- Indexar las columnas donde frecuentemente se realizan filtros
- Indexar columnas que son frecuentemente usadas en orders by
- Evitar duplicación de índices, sobre todo en columnas con pocos valores diferentes Ej: Sexo, Estado Civil, Etc.
- Verificar que el tamaño de índice debería ser pequeño comparado con la fila

- No crear índices sobre tablas con poca cantidad de filas, no olvidar que siempre se recupera de a páginas. De esta manera evitaríamos que el sistema lea el árbol de índices.
- Limitar la cantidad de índices en tablas que son actualizadas frecuentemente, porque sobre estas tablas se estarán ejecutando Selects extras
- Tratar de usar índices compuestos para incrementar los valores únicos. Tener en cuenta que si una o más columnas intervienen en un índice compuesto, el optimizador podría decidir acceder a través de este índice aunque sea solo para la búsqueda de los datos de una columna, esto se denomina “partial key search”
- Usando cluster index se agiliza la recuperación de filas
- Uno de los principales objetivos de la Optimización de bases de datos es reducir la entrada/salida de disco. Reorganizando aquellas tablas que lo necesiten, se obtendría como resultado que las filas serían almacenadas en bloques contiguos, con lo cual facilita el acceso y reduce la cantidad de accesos ya que recupera en menos páginas los mismos datos.

Sinónimos
Un sinónimo es un alias definido, por lo general, sobre una tabla o vista. Se usan a menudo por seguridad o por conveniencia, ya que permiten enmascarar el nombre y dueño de un objeto.

Procedimientos almacenados
Es un procedimiento programado en un lenguaje permitido que es almacenado en la Base de Datos como un objeto. El mismo luego de creado, puede ser ejecutado por usuarios que posean los permisos respectivos.

Características
- Incluyen sentencias de SQL y sentencias de lenguaje propias.
- Son almacenados en la base de datos.
- Antes de ser almacenada en la base de datos las sentencias SQL son parseadas y optimizadas. Cuando el “stored procedure” es ejecutado puede que no sea necesario su optimización, en caso contrario se optimiza la sentencia antes de ejecutarse.

Ventajas
- Pueden reducir la complejidad en la programación. Creando SP con las funciones más usadas.
- Pueden ganar perfomance en algunos casos.
- Otorgan un nivel de seguridad extra.
- Pueden definirse ciertas reglas de negocio independientemente de las aplicaciones.
- Diferentes aplicaciones acceden al mismo código ya compilado y optimizado.

- En una arquitectura cliente/servidor, no sería necesario tener distribuido el código de la aplicación
- En proyectos donde el código puede ser ejecutado desde diferentes interfaces, Ud. mantiene un solo tipo de código.
- Menor tráfico en el PIPE / SOCKET, no en la cantidad de bytes que viajan sino en los ciclos que debo ejecutar una instrucción.

Funciones de usuario
Una función de usuario es un objeto de la base de datos programado en un lenguaje válido por el motor de base de datos que puede recibir uno o más parámetros de input y devolver sólo un parámetro de output.

Trigger
Es un mecanismo que ejecuta una sentencia SQL automáticamente cuando cierto evento ocurre.

Usos
- Se pueden aplicar reglas de negocio.
- Valores de columnas derivadas
- Replicación automática de tablas
- Implementación de Foreign Keys sobre tablas de otras BBDD.
- Logs de Auditoría
- Delete en Cascada
- Autorización de Seguridad

Transaccionalidad entre Evento y Acción
Tanto el evento cómo las acciones que ejecuta el trigger conforman una transacción, lo que implica que si alguno de los dos falls, se realiza un rollback automático.

Momentos (SQL Server) -> AFTER, INSTEAD OF