1. Introducción.

Este manual está basado en el caso de la municipalidad de Coronado, San José, Costa Rica y toma curso sobre el camino del software libre más precisamente PostgreSQL.

1. Objetivo

Proporcionar una guía para el lector, promover el conocimiento básico para la administración, mantenimiento y optimización de una base de datos.

1. Descripción de la organización

Esta organización corre con el sistema de archivos DOS uno de los más usados en el mundo, hay una muy buena razón para que aún se mantenga funcionando, este está “Bien” logrado pero aun así sabemos que estos sistemas presentan grandes desventajas en comparación a los sistemas gestores de bases de datos (SGBD).

Es importante destacar que los sistemas que actualmente se usan en esta organización están en su estado “terminal” y es necesario dar paso a una implementación de un sistema gestor de bases de datos.

En nuestra reunión con los principales entes administrativos del departamento de TI en la municipalidad se nos hizo saber que esta quería comenzar a utilizar un sistema de software libre, por lo tanto recomendamos el uso de PostgreSQL.

1. Características importantes de PostgreSQL.

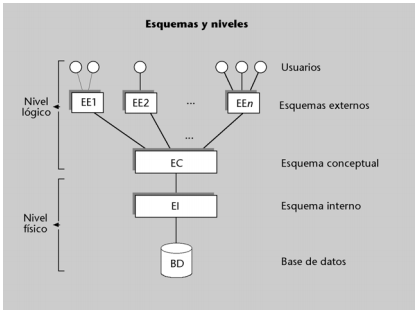
Como se mencionó anteriormente este manual está enfocado a la utilización de postgreSQL, por esta razón debemos conocer el porque es un SGBD ampliamente usado y preferido en el área del software libre. Unas de sus principales características son:

**PostgreSQL**

* Se rige bajo una licencia BSD la cual no nos limita en ningún aspecto. Se puede comercializar libremente aunque debemos poner una nota sobre la misma en nuestro software.
* No ofrece soporte directamente pero sí ofrece los medios necesarios para que la comunidad sea la que se encargue de dicha tarea. También existe una lista (corta) de empresas que brindan soporte profesional.
* PHP se acopla muy bien con este SGBD ya que ofrece funciones especializadas para aprovechar al máximo sus características. Ampliamente usado en aplicaciones web.
* Su administración se basa en usuarios y privilegios.
* Es altamente confiable en cuanto a estabilidad se refiere.
* Para la creación de estructuras (tablas, índices, etc) PostgreSQL ofrece prácticamente todas las prestaciones contempladas en el estándar, y tiene un fantástico sistema de extensión.
* Los tipos de datos y operadores que soporta son más que suficientes para su uso en la municipalidad.
* Para la administración soporta bastante bien el estándar en cuanto a la creación de usuarios y la gestión de sus privilegios con GRANT y REVOKE. Su sistema múltiple de autenticación lo hace demasiado completo en este aspecto.
* Para la gestión de copias de seguridad disponemos tanto de herramientas de volcado, como de la posibilidad de copia binaria de la base de datos.
* Posee una documentación muy extensa y muy bien desarrollada.
* Es extensible mediante FDU (Funciones definidas por el usuario) fuerte, mediana y débilmente acopladas. Además de soportar extensiones para tipo de datos compuestos o nativos así como los operadores para manejar los nuevos tipos de datos.

Como se puede ver es un SGBD muy completo y totalmente ACID (Atomicity, Consistency, Isolation and Durability) lo cual significa Atomicidad, Consistencia, Aislamiento y Durabilidad.

1. Arquitecturas



* 1. **Esquema interno o físico**

En este nivel se debe tener en cuenta que el trabajo del DBA será responder a las cuestiones de rendimiento (espacio y tiempo) planteadas al hacer el diseño físico de la BD y al ajustarlo posteriormente a las necesidades cambiantes de la municipalidad.

Debemos lograr un sistema robusto en el que la BD y los sistemas de aplicación que depende de ella no se vean comprometidos por un cambio en la estructura física debido a que un disco fallo o los recursos se están quedando cortos debido a la gran cantidad de datos procesados.

En este aspecto hay que señalar que la Municipalidad tiene en este momento un estimado de los picos de solicitudes a la base de datos lo cual hace que se tome en cuenta la cantidad de registros y la cantidad de personas que tienen acceso a la base de datos para crear una cifra muy aproximada y se pueda llegar mejor a las necesidades físicas que va a requerir la base de datos, y así la base no quede expuesta en caso de que el disco quede lleno o el procesador no pueda realizar todas las consultas que se hagan en la base de datos.

Conceptos y recomendaciones que debemos conocer para lograr la labor descrita:

* + 1. **Tablespaces:** Son unidades de almacenamiento lógico que permite al DBA definir las rutas en el sistema de archivos donde los ficheros que representan los objetos(tablas, índices, etc) de la BD pueden estar guardados en el disco. Esto nos brinda 2 **ventajas:**
* Si la partición o volumen donde el clúster fue inicializado se queda sin espacio y no se puede extender, podríamos crear un tablespace en otra partición y usarlo hasta que el sistema pueda ser actualizado con los discos necesarios.
* Los tablespace permite al DBA usar el conocimiento del patrón de uso de los objetos de la base de datos para mejorar el rendimiento. Por ejemplo: Consideremos un índice que es usado muy frecuentemente en la municipalidad; digamos una lista de contribuyentes. Se podría pensar en ubicarlo en un disco de mayor velocidad (SSD) y en sentido inverso ubicar los datos que son raramente usados en discos de menor prioridad y no tan caros.

La creación de tablespaces y su sintaxis se verá en la sección 3.1 del manual.

* + 1. **Discos:** Son los encargados de almacenar los datos de manera persistente pero son lentos en relación a la memoria RAM. Aumentar la memoria puede ayudar para bases de datos pequeñas ya que con más memoria tendremos que acceder menos al disco pero esta solución no aplica para bases de datos grandes. Algunas consideraciones:
* De ser posible usar una configuración RAID para lograr una mayor integridad, mayor tolerancia a fallos, mayor rendimiento y por supuesto mayor capacidad de almacenamiento. **Nota:** El nivel de RAID a utilizar debe ser elegido en base a la necesidad de la municipalidad.
* Es recomendable mantener el directorio **/pg\_xlog(**donde se encuentran los archivos WAL**)** en un disco aparte para que la escrituras de estos archivos no generen un costo mayor en el rendimiento general de la base.
* Es recomendable mantener en discos separados los índices y datos.

* + 1. **Memoria:** La memoria RAM tiene unos de los roles más importante en el funcionamiento optimizado de la base de datos. Es por esta razón que se considera importante conocer los siguientes parámetros del archivo postgresql.conf:
* **shared\_buffers** determina la memoria dedicada que usará PostgreSQL para mantener datos en caché. Es recomendable empezar probando con un ¼ de la memoria RAM disponible ya que si se empieza con un valor más alto puede ocasionar problemas en el servidor al tomar memoria que el SO utiliza para realizar sus procesos.
* **work\_mem** este parámetro sirve para establecer la cantidad de memoria que vamos a utilizar del shared\_buffers para realizar operaciones relacionadas a ordenamientos, joins, hash, etc. Esto quiere decir que se usará X veces por consulta(s) realizada por lo tanto debemos tener en cuenta el siguiente parámetro.
* **max\_connections** con ese parámetro vamos a establecer el número máximo de conexiones a la BD, si en la municipalidad se tienen muchos usuarios trabajando simultáneamente se debe tomar en consideración la cantidad de memoria de trabajo(work\_mem) asignada para que la memoria no se sature y las consultas, funciones y procedimientos trabajen de la mejor manera.
* **wal\_buffers** establece la cantidad de memoria compartida que se usará para mantener los registros antes de ser escritos en disco. La escritura se realiza antes de realizar un commit. El valor por defecto de este parámetro suele tomar 3% de memoria del shared\_buffers con esto suele ser suficiente el la mayoría de los casos. Pero puede ser ajustado a la necesidad.

·

* 1. **Esquema conceptual (Parte lógica)**
     1. **La identificación del modelo de datos de una organización**: Es el pasos que hay que tomar en cuenta en la creación de una base de datos, se debe investigar acerca de las reglas de negocio de la organización para crear una base de datos óptima que cubra todas las necesidades específicas de la organización, efectivamente las grandes organizaciones no tuvieron las mejores bases de datos en sus comienzos, esto tiende a ser porque en el entorno de la informática podemos ver que las prácticas tanto como el software están en una constante mejora que las impulsa a tener visiones en el mercado más amplias y con esto sus bases de datos deben de ser lo suficientemente genéricas y eficientes para poder acomodar las nuevas reglas de negocio que surjan en estas mejoras.
     2. **Definición de cardinalidades**: Una vez que el modelo de datos ha sido identificado, hay que tener claro que las diferentes entidades de las bases de datos son relacionadas con otras, esto implica que en la base de datos las cardinalidades deben de ser altamente estudiadas para que un futuro no se presenten errores de integridad, un claro ejemplo de esta práctica es imaginarse un salón de clase, en este salón el alumno es quien tiene un salon?  
        O es el salon quien tiene a los estudiantes?  
        En este caso entramos a temas muy filosóficos pero si utilizamos nuestra lógica como informáticos nos damos cuenta que es más eficiente que un alumno tenga asignado el número de un salón a su tabla de cursos donde se pueda saber con certeza que ese será el lugar donde el estudiante tenga clases, de la misma manera nos encontramos con el dilema del salón teniendo una cantidad N de estudiantes, donde sí se ve de una manera, el salon en teoria podria ser capaz de tener una tabla donde estan los codigos de todos los estudiantes que asisten al mismo, de ambas maneras consideramos que pueden ser prácticas válidas, dicho sea el caso, hay que tomar en cuenta las reglas de la organización, una organización que se centra en estudiantes no tiene porque darle tanta importancia a un salón de clases, pero por otro lado si este dile fuese por seguridad y se quisiese saber exactamente qué estudiante y a que hora es cuando habita el salón podemos ver reflejada la segunda opción propuesta, todo se basa en las reglas de la organización y la manera en que nosotros como informáticos somos capaces de ver los diferentes problemas y sus respectivas soluciones.
     3. **Creación del modelo Entidad - Relación:** Para la creacion de un modelo entidad- relacion se deben de identificar las diferentes entidades que van a conformar la base de datos, las entidades son la base toda implementación de una base de datos, si como programadores podemos crear clases a partir del paradigma orientado a objetos donde cada objeto tiene atributos y funciones, una entidad pasa a ser uno de esos objetos que puede ser almacenado y utilizado en el futuro para su posterior uso en una base de datos, al mismo tiempo la identificación de sus diferentes atributos pasa a ser una parte importante porque estos pueden mantener referencias a las diferentes relaciones que se crean a partir de este modelo.  
        Este modelo tiende a ser un poco lento pero tiene una implicación altamente eficiente y se usa para calcular un nivel muy alto en la tasa de transacciones en la base de datos, una vez modeladas las entidades se deben establecer las relaciones que comparten las mismas, estas relaciones en este modelo pasan a ser tablas separadas a las entidades para confeccionar un modelo que es entendible a la vista; estas dos viniendo a ser dos de las más grandes ventajas de este modelo. Una vez creado este modelo se pueden identificar las diferentes relaciones que en si no son necesarias para la siguiente implementación, el modelo Relacional.
     4. **Creación del modelo Relacional:** El modelo relacional cumple con uno de los principales objetivos de las bases de datos, la eficiencia en sus consultas, este modelo debería ser creado a partir de un modelo entidad-relación, ya que este mantiene solo las relaciones que son puras de las reglas de negocio en la organización y las que no son necesarias se descomponen en atributos que referencian otras entidades, la aplicación de este modelo nos lleva a un punto de optimizacion mucho mas alto que el anterior modelo, aun así se recomienda empezar con el anterior para que se puedan hacer los cálculos necesarios para un mejor manejo del espacio requerido en la base de datos. Para que este modelo pueda funcionar de manera óptima se recomienda cumplir con las 4 restricciones de integridad:
* **Integridad relacional**: Esta dice “Toda relación requiere una única llave primaria” esta se refiere a que la identificación de cada una de estas entidades debería ser única e irrepetible.
* **Integridad referencial**: “No se pueden eliminar llaves primarias ligadas a llaves foráneas” esta se refiere a la eliminación de entidades que tienen un significado derivado por otra entidad, es la idea de que no es correcto deshacerse de un padre si este tiene un hijo por el cual velar, porque perderíamos integridad en la base de datos.
* **Integridad de dominio**: “Si un atributo (a) solo permite un dominio, a este se le deberían establecer sólo los valores correctos”, esta integridad es importante ya que le da la oportunidad a la base de datos de ser mas inteligente y poder atrapar excepciones por ese lado reduciendo la cantidad de programación hecha por el informático.
* **Correcta dependencia funcional**: “Conocer la regla de negocio” este es un tema importante que viene a ser la base de toda base de datos, es importante conocer el negocio para que la base de datos sea adaptada a las necesidades de la organización.
  + 1. **Normalizar el modelo Relacional**: La normalización es útil para llevar la implementación de la base de datos al siguiente nivel de implementación ya que asegura que no existan problemas de integridad en la misma.

La Primer Forma normal: Es cuando una tabla está libre de atributos repetidos y sus dominios son autónomos.

La Segunda Forma normal: Esta no debe tener dependencias parciales entre la llave primaria y los otros atributos, esto quiere decir que hay que dividir las tablas hasta que solo exista una llave primaria.

La Tercera Forma normal: Es cuando no existen dependencias transitivas entre la llave primaria y los demás atributos.

1. PL SQL
   1. **Creación de tablespaces.**

Sintaxis de la sentencia para crear un tablespace en PostgreSQL 9.3.

**Ejemplo 1:**

Para crear un tablespace llamado **espacioBD** en el directorio **/data/dbs:**

CREATE TABLESPACE **espacioBD** LOCATION **'/data/dbs'**;

**Ejemplo 2:**

Para crear un tablespace llamado **espacioIndices**en el directorio **/data/índices** perteneciente a usuario llamado **DBA:**

CREATE TABLESPACE ***espacioIndices*** OWNER **DBA** LOCATION **'/data/indices'**;

**Nota:** El parámetro **tablespace\_option** de momento solo puede tomar 2 valores **seq\_page\_cost** y **random\_page\_cost.** Definir este parámetro para un tablespace en particular hará que se sobreescriba la estimación de costo de lectura del *PLANNER* para leer las páginas de las tablas en ese tablespace. Esto puede ser útil si uno de los tablespace está ubicado en un disco que es más lento o más rápido que el resto del sistema.

* 1. **Triggers**

**Servidor FSQL-PostgreSQL**: Su función principal es capturar las sentencias escritas en lenguaje difuso FSQL [3][6], y traducir y enviar las mismas al SGBD. Utiliza todos los módulos que están soportados sobre PostgreSQL, es decir, base de datos, FMB, DIC y una serie de paquetes de funciones, procedimientos, triggers, etc., que se encuentran implementados en el lenguaje procedural PgSQL.

PL/pgSQL puede ser usado para definir procedimientos de triggers. Los procedimientos de triggers pueden ser creados con el comando CREATE FUNCTION, declarando como una función sin argumentos y retorna el tipo de trigger. Tenga en cuenta que la función debe ser declarada sin argumentos incluso si se espera recibir argumentos especificados en CREATE TRIGGER — los argumentos son enviados via TG\_ARGV, como se describen abajo.

Cuando una función PL/pgSQL es llamada como un trigger, muchas variable especiales son creadas automáticamente en el bloque de primer nivel estos son:

NEW

Tipo de dato RECORD; variable sostiene la nueva base de datos seguidos de operaciones INSERT/UPDATE en triggers a nivel de registro. Esta variable no está asignada en triggers a nivel de instrucción y por la operación DELETE.

OLD

Tipo de Dato RECORD; variable que sostiene la vieja base de datos seguidos por operaciones UPDATE/DELETE en triggers a nivel de registro. Esta variable no está asignada en triggers a nivel de instrucción por la operación INSERT.

TG\_NAME

Tipo de dato nombre; variable que contiene el nombre del trigger actualmente efectuado

TG\_WHEN

tipo de dato text; un string de BEFORE,AFTER, or INSTEAD OF, dependiendo de la definición del trigger

TG\_LEVEL

tipo de dato text; un string de ya sea ROW o STATEMENT dependiendo de la definición del trigger

TG\_OP

Tipo de dato text; un string de INSERT,UPDATE,DELETE, o TRUNCATE diciendo por qué se disparó el trigger.

TG\_RELID

Tipo de dato oid; el objeto ID de la tabla que causa la invocación del trigger

TG\_RELNAME

Tipo de dato name; el nombre de la tabla que causa la invocación del trigger.

TG\_TABLE\_NAME

Tipo de dato name; el nombre de la tabla que causa la invocación del trigger.

TG\_TABLE\_SCHEMA

Tipo de dato name; el nombre del schema que causa la invocación del trigger. .

TG\_NARGS

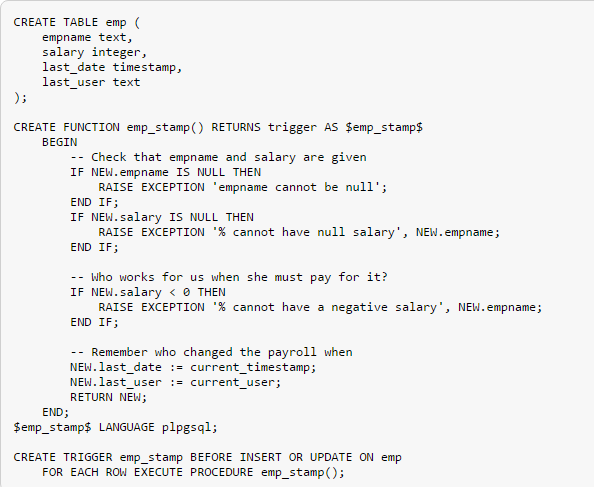
Tipo de dato integer; el número de argumentos dado que el procedimiento de activación en la sentencia CREATE TRIGGER.

TG\_ARGV[]

Tipo de datos de la matriz de texto; los argumentos de la sentencia CREATE TRIGGER. Los recuentos de índice de 0. Los índices Inválido (menor que 0 o mayor que o igual a tg\_nargs) dan como resultado un valor nulo.

Una función de disparo debe devolver NULL o un valor de record/row que tiene exactamente la estructura de la tabla el gatillo fue despedido.

Ejemplo:



1. Algebra relacional

La creación de un modelo relacional nos da un vistazo a la optimización de la base de datos aun así pueden haber maneras de incrementar la velocidad de las sentencias SQL con un entendimiento de el algebra relacional, esta es un lenguaje procedimental de alto nivel, para usuarios sofisticados de la base de datos.  
Este lenguaje consta de símbolos los cuales son llamados “Operadores” y de relaciones llamados “registros”.  
Como todo lenguaje se utilizan funciones de dos tipos, las funciones “Unitarias” y las funciones “Binarias” las cuales son utilizadas para interpretar las diferentes sentencias SQL.

Funciones Unitarias:

-Proyección (Π): Esta función es el equivalente a los “Select” donde un ejemplo para seleccionar el nombre y el apellido de la tabla de Personas viene a ser “Πnombre,apellido(Personas)” el cual es equivalente a “Select nombre, apellido from Personas”.

-Selección (σ): Esta función es equivalente a las restricciones de las consultas las cuales empiezan luego de un “where” por lo tanto si queremos hacer una selección de los nombres en la tabla de Personas donde los nombres sean iguales a “Juancho” se denota “Πnombre(σ nombre = “Juancho” (Personas))” el cual es el equivalente a una consulta “Select nombre from Personas where nombre = “Juancho”.

Funciones Binarias:

-Intersección(∩): Es el resultado de todas las entidades cuyos atributos pueden formarse de forma que ambos elementos puedan ser relacionados entre sí por el sistema gestor de bases de datos.  
-Producto cartesiano (x): Entrega la combinación de todas las tuplas que sean parte de las dos entidades que el usuario decida combinar.  
-División(/): Retorna todos los distintos valores de la relación entre ambas   
-Diferencia(-): Entrega todas las tuplas que estén en la primera entidad pero no en la segunda.  
-Producto Natural(⋈):Consiste en combinar las proyección, selección y producto cartesiano en una sola operación, eliminando columnas extras.

Una vez entendidas estas connotaciones recomendamos el uso del software RAT hecho por la Universidad Nacional de Costa Rica el cual es un software traslado de álgebra relacional a sentencias SQL para maximizar la velocidad de las sentencias a crear por los usuarios.

Ya que se tienen estos conceptos el DBA puede utilizar esta herramienta para crear sentencias SQL eficientes y efectivas. Donde en las asegura que no se desperdicie tanta memoria ni CPU y esos recursos se le puedan dar a otros usuarios del sistema. Esto quiere decir que para que una base sea lo mejor posible se tiene que tener en cuenta que siempre se puede mejorar, y dependiendo del equipo con el que se cuente hay que maximizar su funcionamiento.

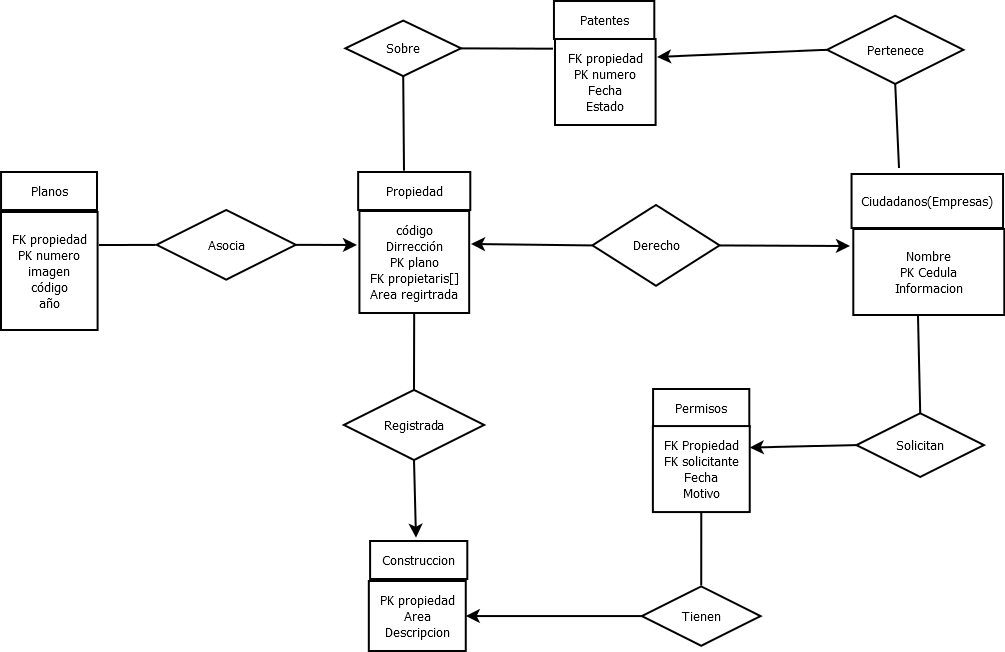
**Modelo Entidad Relación**

En este modelo se debe tener en cuenta que todas la tablas se relacionan, en este caso la relación aparece como una tabla más, la cual va a tener datos de esa relación, lo cual ayuda a entender cómo va a estar estructurada la base de datos y cómo va a estar ligada a las reglas de negocio.

En este caso la Municipalidad de Coronado nos proporcionó la información que ellos manejan lo que nos ayudó a crear el modelo Entidad Relación.

Hay que señalar que en una relación la llave primaria es donde esta el N. Esto quiere decir que por ejemplo, un alumno tiene n cursos, entonces en la relación Tiene la llave primaria sería cursos.

A continuación:



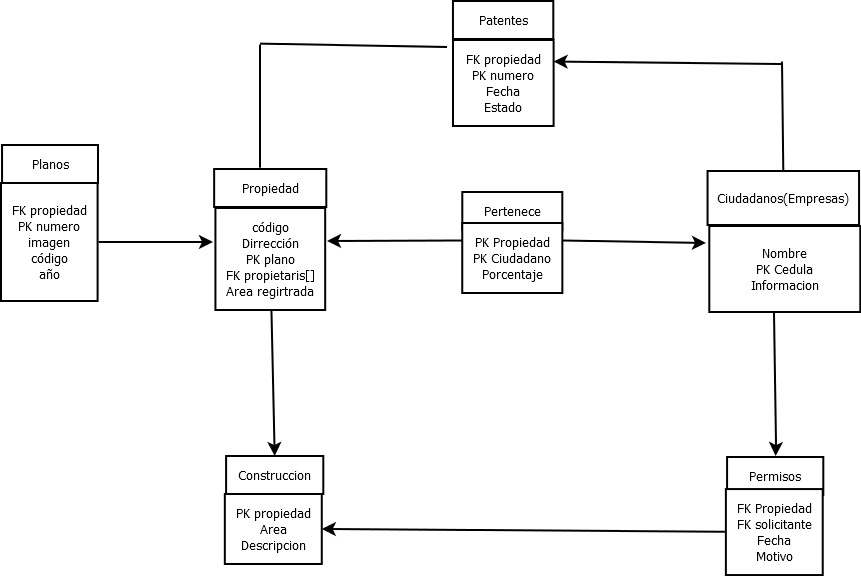
Hay que señalar que esto no es como deberia quedar la base de datos ya que se puede mejorar mucho más con el Modelo Relacional.

**Modelo Relacional**

Hay que señalar que para realizar este modelo se tiene que tener el Modelo Entidad Relación, donde, a partir de ese modelo se trata de eliminar las tablas de las relaciones, siempre y cuando se deje las relaciones de 1 a N, ya que asi queda mejor estructurada una base de datos, y se puede mantener mejor el uso de las llaves primarias.

Si al eliminar la relación no queda 1 a N, se mantiene la relación, la cual va a estabilizar la base.

En este caso basados en Modelo Entidad Relación, el Modelo Relacional quedó así:



1. Bibliografía:

Silberschatz, A., Korth, H. F., & Sudarshan, S. (1997). *Database system concepts* (Vol. 4). New York: McGraw-Hill.

BORRADOR

Desarrollo:

Arquitectura Lógica de base de datos de software libre POSTgresql

**TableSpaces**

Para esto hay que tener en cuenta que los tablespaces son los archivos donde esta guardado los datos, si se crea solo un tablespace seria un mismo archivo donde se guarda toda la información de la base de datos, lo recomendable es crear varios tablespaces para mantener la información en distintos segmentos de disco, esto va a ayudar no solo al rendimiento de la base de datos por no tener que recorrer un archivo muy grande, sino también a la seguridad en caso de que algún segmento de disco deje de funcionar, ya que se podría mantener información que estaba localizada en otros segmentos del disco, disminuyendo la pérdida de información, ya que si es solo un tablespace se perdería el 100% de la información.

También hay que recordar que a la hora de crear los tablespaces que se tiene que tener la cantidad aproximada de la cantidad de tablas y bloques se le van a meter a ese tablespace, porque esto nos va a afectar muchísimo la efectividad de la base de datos en tiempo.

**Usuarios:**

En una compañía existen diversos tipos de usuarios del sistema gestor de bases de datos y gracias a estos se genera una jerarquía donde se les da privilegios a cada uno y esto garantiza seguridad ya que los únicos que le puedan hacer cambios importantes en la base de datos son los DBA que deben estar preparados para que la base de datos no se caiga, y en rendimiento en que no todos son aptos para realizar las consultas ni las transacciones donde el que realiza las consultas debe estar seguro de que la sentencia dure lo menos posible y utilice lo menos posible de memoria y CPU.

------------------------------------------------------------------------------------------

**Usuarios normales.** Son usuarios no sofisticados que interactúan con el sistema mediante un programa de aplicación con una interfaz de formularios, donde puede rellenar los campos apropiados del formulario. Estos usuarios pueden también simplemente leer informes generados de la base de datos.

**Programadores de aplicaciones.** Son profesionales informáticos que escriben los programas de aplicación, utilizando herramientas para desarrollar interfaces de usuario, como las herramientas de **desarrollo rápido de aplicaciones (DRA)**, que facilitan crear los formularios e informes sin escribir directamente el programa.

**Usuarios sofisticados.** Interactúan con el sistema sin programas escritos, usando el lenguaje de consulta de base de datos para hacer sus consultas. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría, usando ellos las herramientas de **procesamiento analítico en línea** (**OLAP, OnLine Analytical Processing**), o herramientas de **recopilación de datos**.

**Usuarios especializados.** Son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas y adecuadas para el procesamiento de datos tradicional. Entre estas aplicaciones están los sistemas de diseño asistido por computadora, sistemas de base de conocimientos y sistemas expertos, sistemas que almacenan datos de tipos de datos complejos (como gráficos y de audio) y sistemas de modelado de entorno.

**Administradores** **de la base de datos (DBA).** Son las personas que tienen el control central del SGBD. Entre las funciones del DBA se encuentran:

* Definición del esquema de la base de datos.
* Definición de la estructura y el método de acceso.
* Modificación del esquema y la organización física.
* Concesión de autorización para el acceso a los datos.
* Mantenimiento rutinario.

----------------------------------------------------------------------------------------

### Tipos de Usuarios

Un usuario es todo aquel que tenga contacto con el sistema de bases de datos.

Se tienen 3 clases generales de usuarios:

**1. Programador de aplicaciones:** son aquellos profesionales en informática que interactúan con el sistema a través del DML(Lenguaje de Manipulación de Datos), los cuales se encuentran en un lenguaje de programación (Pascal, Cobol, etc.) Es el encargado de escribir programas de aplicación que usen Bases de Datos.

**2.** **Usuario Final:** accede a la base de datos desde un equipo en el cual puede utilizar lenguaje de consulta generado como parte del sistema o acude a un programa de aplicación suministrado por un programador.

**3.** **Administrador de Bases de Datos**: es el encargado del control general del sistema.

Todo usuario que ingrese o consulte una base de datos puede clasificarse:

**Programador de Aplicaciones.**

**Usuario sofisticado**: interactúa con el sistema sin escribir programas. Generan consultas en un lenguaje de bases de datos.

**Usuario Especializado:** algunos usuarios sofisticados desarrollan aplicaciones de bases de datos especializadas. Entre estas aplicaciones se encuentran los sistemas de diseño asistido por computador.

**Usuarios ingenuos**: es el usuario final que utiliza bases de datos sin saberlo, para él es totalmente transparente como se generan las consultas de la información.

Quienes diseñan y participan en el mantenimiento de un BD se les clasifica como Actores en el escenario y Trabajadores tras bambalinas

**Actores en el escenario:** personas que su trabajo depende del uso constante una base de datos.

**DataBase Administrators(DBA):** administran 2 recursos 1. la base de datos y 2. es el SGBD y el software con el relacionado. El Administrador de Base de Datos (DBA) es quien autoriza el acceso a la base de datos, vigilar el uso y adquirir hardware y software necesarios para su uso. También es el responsable de velar por la seguridad y lentitud en el sistema.

**Diseñador de Base de Datos:** es el encargado de estructurar la arquitectura para representar y almacenar los datos. Él debe atender a los usuarios de Bases de Datos para comprender sus necesidades presentando un diseño que de respuesta a sus necesidades.

**Usuarios Finales**: son quienes requieren acceso a la base de datos para generar consultas e informes. Hay varios usuarios finales como son:

* **Usuarios finales esporádicos:** accesan de vez en cuando pero esto no significa que siempre requieren la misma información.
* **Usuarios finales simples o paramétricos:** su función gira en torno a consultas y actualizaciones de la base de datos. Todos estamos acostumbrados a tratar con estos usuarios, como los cajeros bancarias al revisar los saldos, al generar retiros y depósitos.
* **Usuarios finales avanzados**: estos son ingenieros, analistas de negocios, científicos, son quienes conocen los recursos del SGBD para satisfacer requerimientos complejos.
* **Usuarios Autónomos**: utilizan bases de datos personalizadas basadas en programas comerciales que cuentan con interfaces de fácil uso.

**Analista de sistemas y programadores de aplicaciones**: determinan los requerimientos de los usuarios finales.

**Trabajadores tras bambalinas:** están para mantener el sistema de base datos.

**Diseñadores e implementadores del SGBD:** se encarga de diseñar e implementar los módulos e interfaces de SGBD. Un Sistema de Gestión de Base de Datos consta de varios componentes y módulos.

Plsql

.Triggers

.Funciones almacenadas

.Procedimientos almacenados

.Seguridad - Usuarios

Preguntas:  
  
A cual sistema gestor de bases de datos van a implementar?  
Estimado de las tasa de transacción diarias?  
Que nuevo hardware tienen pensado?  
Un generalización del modelo de datos?

Borrador descripción

Esta organización corre con el sistema de archivos DOS uno de los más usados en el mundo, fue sorprendente cuando un grupo de investigadores de la universidad nacional los cuales están acostumbrados a trabajar con equipos de alto nivel lleguen a ver este tipo de sistemas trabajando, y hay una muy buena razón para que aún se mantenga funcionando, este esta “Bien” hecho cuenta la historia que ingenieros de TI de costa rica se dieron la tarea de crear un sistema de archivos que pudiera mantener la no tan grande ciudad de Coronado en San José, lo que la catapultó a ser un lugar lleno de historia y tecnología, los dos grandes mundos juntándose para que el progreso se diera, dicen las grandes mentes que es necesario saber del pasado para que los errores en este no se repitan en el futuro y es nuestra tarea como ingenieros en sistemas crear un manual del diseño de la base de datos de este lugar para que la universidad continúe dando un bien por la comunidad y que la municipalidad ayude a unos estudiantes con ansias de investigar y aprender a satisfacer sus sueños de convertirse en informáticos expertos.

Es importante destacar que los sistemas que actualmente se usan en esta organización están en su estado “terminal” para dar paso a una implementación de bases de datos con software libre donde nosotros podremos aprender aún más acerca de un proceso como este en las organizaciones, porque normalmente estos son sistemas que deben ser adaptados a los nuevos requerimientos que constantemente surgen y consideramos que las herramientas de software libre pueden aumentar el valor no solo de la municipalidad pero de toda la ciudad, no por el hecho de que sea gratis sino porque las nuevas mentes a tomar parte en esta organización van a poder ser capaces de abrir las puertas a ese codigo, aprender, crear, diseñar e implementar nuevas ideas que puedan satisfacer su hambre por el conocimiento y el hambre que tiene la ciudad por ser cada vez más grande y eficiente.

En nuestra reunión con los principales entes administrativos del departamento de TI en la municipalidad se nos hizo saber que esta quería comenzar a utilizar bases de datos con software libre, por lo tanto como en nuestro curso hemos tenido la oportunidad de utilizar postgreSQL para implementar nuestras bases de datos consideramos que sería óptimo recomendar dicho software, este debe correr en un ambiente UNIX, al mismo tiempo tener los paquetes necesarios para que pueda correr la aplicación, posteriormente se debe de instalar el software, y por último se deben de crear espacios para las librerías compartidas de postgreSQL para que los ingenieros de la municipalidad puedan crear sus propias funciones y procedimientos, que más adelante indagaremos en cómo estas funcionalidades pueden llegar a ser muy útiles en una organización.

Cabe mencionar que un servidor los suficientemente poderoso para atender una amplia cantidad de consultas diarias es esencial para el óptimo funcionamiento de la base de datos.

El resguardo de la información en estos ambientes es sumamente importante, para satisfacer esa necesidad de seguridad de la información existen registros llamados **bitácoras** que son capaces de registrar toda transacción realizada y aprobada, lo cual es útil para poder revertir posibles errores o una manipulación indebida en la base de datos, que podrían resultar en grandes pérdidas para la organización.

Grandes mejoras pueden provenir de esta nueva implementación en las bases de datos de esta compañía, cabe mencionar que los problemas de redundancia, los cuales vienen a ser que existan cantidades de datos que estén en lugares repetidos en la organización son anulados por los programas gestores de bases de datos, también los problemas de aislamiento de datos, los cuales consisten en mucha información con formatos diferentes ya no serían un problema ya que estos sistemas se encargan de mantener una jerarquía a lo largo de sus diferentes implementaciones, así mismo los problemas de integridad pueden ser resueltos con los “CONSTRAINTS” que se encargan de dar una inteligencia más compleja en el almacenamiento de la información que vino a ser la solución que sustituyó viejos sistemas de archivos, y por último probablemente uno de los problemas más grandes en los sistemas de archivos es el principio de la atomicidad de su información, este principio se basa en que los datos son “Atómicos”, estos son procesos que deben de ser terminados para que se pueda procesar la información de lo contrario no deberían de ser capaces de efectuar, un claro ejemplo de este principio son las transacciones bancarias, donde una transacción no puede haber sido realizada por el usuario pero esta no se da en el sistema gestor de información porque primeramente para el negocio presenta pérdidas millonarias, y la integridad de la información se vería afectada, dando inconsistencias y como consecuencia un fallo crítico que puede presentar la diferencia entre una organización exitosa y una con problemas serios.

Funciones Almacenadas

Un procedimiento almacenado se puede definir como un programa, procedimiento ó función, el cual está almacenado en la base de datos y listo para ser usado.

Existen dos ventajas evidentes al utilizar procedimientos almacenados en nuestro sistema:

* La ejecución del procedimiento ocurre en el servidor de bases de datos. Esto probablemente aumentará el rendimiento de nuestra aplicación al no tenerse que mandar datos entre el cliente y el servidor, y no tener que procesar resultados intermedios en el cliente para obtener el resultado final.
* Al tener la *lógica de la aplicación* implementada en la base de datos no tendremos que implementarla en los clientes, con el consiguiente ahorro de líneas de código redundante y complejidad.
* Si tenemos diferentes tipos de clientes implementados en diferentes sistemas o lenguajes de programación y accediendo a la misma base de datos, no tendremos que programar la misma lógica en todos, al estar esta disponible en la base de datos. Tendremos una API a la lógica de la aplicación lista para usarse desde diferentes clientes

Un procedimiento almacenado en PostgreSQL se puede escribir en múltiples lenguajes de programación. En una instalación por defecto de PostgreSQL podremos tener disponibles los siguientes lenguajes: PL/pgSQL, PL/Perl, PL/Tcl y PL/Python.

El único lenguaje que está disponible automáticamente en PL/pgSQL. Para utilizar PL/Perl, PL/Tcl o PL/Python tendremos que haber configurado/compilado PostgreSQL con estos parámetros --with-perl --with-tcl --with-python.

También existen muchos otros lenguajes disponibles como módulos adicionales, entre ellos, PL/Java, PL/PHP, PL/R, PL/Ruby, PL/Sheme y PL/sh, pero estos tienen que descargarse e instalarse por separado.

Suponiendo que tenemos PostgreSQL instalado con los lenguajes que queremos utilizar tendremos que realizar dos operaciones para poder empezar a utilizar un procedimiento almacenado en nuestra base de datos:

1. Instalar, si no lo tenemos instalado, el lenguaje que vayamos a utilizar para programar nuestro procedimiento (solamente necesitamos hacer esto una sola vez por base de datos)
2. Programar nuestro procedimiento e instalarlo en la base de datos

<https://es.scribd.com/doc/60030648/SEGURIDAD-Bases-de-Datos-de-Tipo-PostgreSQL>