**PROYECTO DE CURSO DE BASES DE DATOS**

**INTRODUCCIÓN**

Una universidad en Bogotá D.C. ofrece 16 programas académicos en pregrado y posgrado y cuenta con equipo de unos 610 funcionarios de los cuales 80 son docentes de planta.

La Vicerrectoría de Administración y Financiera de esta institución encomendó el desarrollo de un sistema computacional soportado en una base de datos, que permita una adecuada gestión de los activos fijos tangibles de la universidad.

*Un activo fijo tangible es un bien que tiene la universidad a su disposición para cumplir con sus funciones. Se trata de un bien que no está a la venta y cuenta con una vida útil de mediano o largo plazo.*

Los activos fijos tangibles de la universidad son en su mayoría, mobiliario de las oficinas y salones, equipos electrónicos como computadores, televisores o equipos de sonido y aparatos utilizados en los laboratorios.

La gestión de estos activos es de interés para los siguientes departamentos de la universidad:

* Logística y Servicios Administrativos
* Contabilidad y Finanzas
* Gestión Humana y Desarrollo Organizacional

El siguiente organigrama resumido de la universidad, explica la relación que existe entre estas entidades:



Figura 1 - Entidades interesadas en el desarrollo

**DEFINICIÓN DEL PROBLEMA**

El Área de Compras y Almacén ha venido registrando en un documento de Excel, los activos fijos que se han comprado a lo largo del año 2020. Aun no se tiene registro de todos los activos con los que contaba la universidad antes de crear este archivo. Por esta razón, se desea conformar un equipo de auxiliares de inventarios para que recorran la universidad, y vayan completando este documento.

Sin embargo, seguir utilizando el archivo Excel tiene el inconveniente de que cada vez que se realice una actualización en el mismo, el área de Compras y Almacén tendría que compartirlo con el área de Servicios Generales y los departamentos de Contabilidad y Gestión Humana. Debido a que cada una de estas entidades se preocupa por unos datos específicos de los activos, no conviene compartir el mismo documento para todos.

Por esta razón, se desea tener una aplicación de fácil acceso y soportada en una base de datos, que contenga toda esta información, se actualice para todos los usuarios a medida que se hacen cambios e inserciones a la misma, y presente solo la información relevante para cada entidad y/o tipo de usuario.

No contar con esta herramienta en la universidad supone una serie de problemas para la universidad en términos de logística y seguridad, control contable y gestión del personal.

**LOGÍSTICA Y SEGURIDAD**

No hay un registro de la ubicación física de los activos de la universidad, lo cual representa un problema en términos de logística.

* *Por ejemplo, si el área de Gestión Humana desea realizar una actividad de bienestar laboral al aire libre, y solicita un conjunto de mesas, sillas, equipos de cómputo o de sonido, para organizar el evento; bastaría con que el área de Servicios Generales hiciera una consulta en el sistema para verificar cuáles de estos activos están disponibles y cuál es su ubicación, de manera que se coordine su traslado al lugar del evento.*

No contar con la ubicación física de los activos de más valor, también puede representar un problema en términos de costos y seguridad para la universidad. Saber dónde se encuentran facilitaría el proceso de vigilancia por parte del personal de la universidad dedicado a ello.

**CONTROL CONTABLE**

No hay información actualizada de la depreciación de los activos fijos. El departamento de Contabilidad necesita de esta información para elaborar sus informes y responder a las exigencias de la universidad y de auditorías externas.

Adicionalmente, no contar con un registro de los activos disponibles en la universidad, también puede resultar en gastos innecesarios:

* *Por ejemplo, si un funcionario necesita un reemplazo de su silla porque ya no funciona correctamente y no se puede reparar, bastaría con consultar el sistema y buscar una silla en buenas condiciones que se encuentre disponible. Esto evitaría tener que hacer una solicitud al área de Compras por una silla totalmente nueva.*

**GESTIÓN DE PERSONAL**

No hay un registro que indique la condición de los puestos de trabajo de los funcionarios. Algunos funcionarios pueden estar trabajando en puestos o con herramientas que presentan fallas físicas y técnicas, reduciendo su comodidad y eficiencia, y en algunos casos hasta afectando su salud sin siquiera saberlo. Esta información es de utilidad para el equipo encargado de la salud en el trabajo de la universidad.

Por otro lado algunos funcionarios terminan su contrato de trabajo y se retiran de la universidad, pero no se sigue un proceso de devolución del puesto de trabajo con la debida firma de paz y salvo. Contar con un sistema que registre los activos que están bajo la responsabilidad de un funcionario facilitaría este proceso, y permitiría imponer alguna sanción (si aplica) cuando se evidencie un daño en el activo fijo al momento de su devolución.

**OBJETIVO Y CARACTERÍSTICAS DEL PRODUCTO FINAL**

Desarrollar una aplicación web construida a partir de una base de datos de los activos fijos tangibles de la universidad, de manera que:

* presente información relevante para cada una de las áreas interesadas,
* soporte la gestión de este inventario y
* sirva de registro de una serie de procedimientos inherentes a esta gestión

A continuaciónse presenta la información que se debe presentar en la aplicación según cada tipo de usuario

|  |  |  |  |
| --- | --- | --- | --- |
| ***Información relevante para cada tipo de usuario según la entidad para la que trabaja*** | | | |
| *Usuario* | *Entidad* | *Información* | *Permiso* |
| Auxiliar de Inventarios | Área de Compras y Almacén | La descripción de los activos fijos, incluyendo sus datos contables y de ubicación física. | Permiso para insertar y actualizar registros.  *Es necesario a medida que realizan el inventario de los activos existentes o se adquiere un nuevo activo desde la oficina de Compras.* |
| Auxiliar de Mantenimiento | Área de Servicios Generales y Transportes | La descripción de los activos fijos que necesitan una reparación o mantenimiento, incluyendo su ubicación física. | Permiso para consultar.  *Conocer la descripción física y ubicación de los activos, permitirá identificarlos y realizar las reparaciones o mantenimientos necesarios.* |
| Personal de Servicios Generales y Transportes | Área de Servicios Generales y Transportes | La descripción de los activos fijos, incluyendo su ubicación física. | Permiso para consultar.  *Conocer la descripción física y ubicación de los activos, permitirá coordinar traslados cuando se necesiten y organizar eventos* |
| Personal de Contabilidad y Finanzas | Departamento de Contabilidad y Finanzas | Los datos de identificación de los activos fijos sin su descripción física e incluyendo los datos contables. | Permiso para actualizar y eliminar registros.  *Es posible que esta área decida que un registro no entre dentro de la categoría de activo fijo para la universidad y proceda a eliminarlo. También pueden decidir ajustar los índices de depreciación de ciertos activos.* |
| Personal de Gestión Humana y Desarrollo Organizacional | Departamento de Gestión Humana y Desarrollo Organizacional | La descripción de los activos fijos incluyendo su descripción física de los activos asignados a los funcionarios de la universidad, incluyendo detalles sobre su estado según la última revisión realizada por los auxiliares de inventarios. | Permiso para consultar.  *Así podrán tomar decisiones para preservar la comodidad, salud y seguridad de los funcionarios en sus puestos de trabajo, como imponer sanciones cuando se evidencie algún daño intencionado o por negligencia de parte de un funcionario a su puesto o herramientas de trabajo.* |

Tabla - Información relevante para cada entidad interesada

Adicional a estas características, la aplicación debe permitir a los auxiliares de inventarios, llevar registro de cuatro procedimientos relacionados con la gestión de los activos fijos de la universidad, y que permitirán responder a las necesidades de las entidades interesadas.

**Revisión:** Cuando el Departamento de Logística y Servicios Administrativos recibe una notificación de parte de algún área o funcionario de la universidad en la que se solicita una revisión de su puesto de trabajo o herramienta porque presenta alguna falla, un auxiliar de inventarios se acerca al lugar y realiza una verificación o diagnóstico. El auxiliar de inventarios decide si se debe solicitar la compra (reemplazo) del activo o si se debe coordinar cuna cita de mantenimiento con el personal de Servicios Generales. Los auxiliares de inventarios realizan rondas cada cierto verificando la consistencia de la base de datos. Cuando encuentran alguna anomalía, pueden registrarlo con este tipo de procedimiento.

**Entrega:** Cuando un funcionario es contratado en la universidad, un auxiliar de inventarios se acerca al puesto de trabajo y formaliza la entrega y asignación del mobiliario, equipo y herramientas que requiere el trabajador. Se hace el registro del proceso y se asigna el funcionario en cuestión a cada uno de los activos. Es la entrega formal de su puesto de trabajo, el cual en la mayoría de los casos consiste de una silla, un escritorio y un computador. Algunos funcionarios tendrán a su disposición otros dispositivos y herramientas según las funciones que desempeñe. Este procedimiento es de interés para el departamento de Gestión Humana, con el fin de llevar un registro de las condiciones de los puestos de trabajo y herramientas de sus funcionarios, de manera que puedan prevenir enfermedades y accidentes laborales, y realizar análisis del impacto que tiene el ambiente de trabajo en el desempeño de los trabajadores. Al realizar este procedimiento, se busca actualizar el registro de la tabla ACTIVO indicando el funcionario al cuál se le está haciendo la entrega. Este procedimiento es realizado por un auxiliar de inventarios.

**Paz y salvo:** También llamado devolución del puesto de trabajo.Cuando un funcionario se retira de la compañía, un auxiliar de inventarios se acerca al puesto de trabajo y revisa el estado de los activos que lo componen. Toma las observaciones necesarias y se determina si se puede o no firmar su paz y salvo en coordinación con el personal de Gestión Humana. El activo puede estar en buenas condiciones, necesitar un mantenimiento de parte de los auxiliares de mantenimiento o requerir ser reemplazado por uno nuevo. Los auxiliares de mantenimiento tienen acceso a la información de los activos que requieren mantenimiento y se acercan al lugar a cumplir con esta función. Asimismo se debe retirar el funcionario responsable de cada uno de los registros de los activos en cuestión, es decir formalizar la devolución del puesto de trabajo.

**Mantenimiento:** Los auxiliares de mantenimiento registran las citas de mantenimiento completadas, para cada uno de los activos reparados. Hay un espacio para escribir las observaciones del proceso. La lista de activos que necesitan ser reemplazados se actualiza en una tabla visible para el área de Compras y Almacén.

**FASES DEL PROYECTO U OBJETIVOS ESPECÍFICOS**

Las fases a seguir para alcanzar el objetivo encomendado son:

**DISEÑO**

Diseñar la estructura de la base de datos para los activos fijos. La base de datos debe ser consistente con otros sistemas ya existentes en la universidad, como es el de administración del personal, donde se encuentra el registro de todos los funcionarios de la universidad según su posición en el organigrama.

**DESARROLLO**

Desarrollar la aplicación web con el constructor de aplicaciones del motor utilizado y soportada en la base de datos ya implementada, incluyendo todas las funcionalidades y características mencionadas anteriormente en el documento.

**IMPLEMENTACIÓN**

Implementar la estructura de la base de datos en un motor de bases de datos (Oracle APEX) haciendo la respectiva inserción de datos, según la información disponible hasta el momento: activos fijos registrados en el archivo Excel hasta el momento, datos de los funcionarios, estructura organizacional y física de la universidad.

**DISEÑO DE LA BASE DE DATOS**

**Especificaciones del diseño**

La tabla más importante de la base de datos es ACTIVO, la cual contiene una descripción física de cada activo fijo, junto con su ubicación física y datos contables.

Algunos activos fijos son asignados a un funcionario responsable por medio del atributo ID\_FUNCIONARIO, el cual es una llave foránea que relaciona a la tabla ACTIVO con la tabla FUNCIONARIO.

La universidad cuenta con un registro de todos los funcionarios de la empresa actualmente, que suman 610, de los cuales 80 son docentes de planta. No se incluyen en esta cuenta a los directores de departamento, decanos de facultad ni vicerrectores.

La base de datos a diseñar, debe ser consistente con otros sistemas computacionales de la universidad, y por esta razón, se debe representar el esquema organizacional de la compañía en la base de datos. A continuación los detalles:

La universidad es dirigida por cuatro vicerrectorías. A continuación se presentan, junto con el vicerrector asignado actualmente.

* Vicerrectoría de Formación: María Claudia Rincón Mora
* Vicerrectoría de Administración y Financiera: Luz Marina Trejos Montero
* Vicerrectoría de Investigación: Carlos Humberto Angarita Preciado
* Vicerrectoría de Extensión y Proyección Social: Luis Augusto Moreno Boadas

Adicionalmente cuenta con cinco facultades. Cada facultad tiene un decano asignado.

* Facultad de Ciencias Económicas, Administrativas y Contables: Mario Alberto Rodríguez Franco
* Facultad de Ciencias Jurídicas y Sociales: Francisco Antonio Pedraza Vásquez
* Facultad de Ciencias Agrarias: Néstor Mauricio Serna Rodríguez
* Facultad de Ingeniería: María Luisa Pérez Cifuentes
* Facultad de Ciencias Básicas y de la Educación: Sandra Milena Piedrahita Buitrago

La universidad cuenta con 23 departamentos, los cuales pueden pertenecer a una vicerrectoría o a una facultad, pero no a ambos al mismo tiempo. Cada departamento tiene un director asignado.

|  |  |  |
| --- | --- | --- |
| Vicerrectoría | Departamento | Director |
| Formación | Formación Académica | Carlos Andrés Preciado Sosa |
| Educación Virtual | Carlos Arturo Durán Porras |
| Aseguramiento de la Calidad | Mario Andrés Canadas Buitrago |
| Administración y Financiera | Logística y Servicios Administrativos | Wilson Andrés Montaño Cárdenas |
| Administración Dependencia Teusaquillo | María Alejandra Azuero López |
| Administración Dependencia Facatativá | Beatriz Gaitán Oviedo |
| Mercadeo | Johanna Vanessa Duarte Jiménez |
| Secretaría General | María Helena Cadavid Quintana |
| Sistemas y Tecnología | Daniel Humberto Díaz Calero |
| Gestión Humana y Desarrollo Organizacional | María Alejandra Pinzón Ramos |
| Contabilidad y Finanzas | Carlos Alberto Bernal Tiquiza |
| Planeación | David Hernando Torres Piraquive |
| Investigación | Investigación Formativa | Adriana Martínez Méndez |
| Investigación Aplicada | Pablo Andrés Rodríguez Peña |
| Extensión y Proyección Social | Bienestar Universitario | Eduardo Fernández Wilches |
| Proyectos Educativos en Contextos Rurales | Sandra Patricia Lara Martínez |
| Educación para el Trabajo y el Desarrollo Humano | Samuel Cipriano Moreno Rodríguez |

Tabla - Dominio: Directores de departamento por vicerrectoría

|  |  |  |
| --- | --- | --- |
| Facultad | Departamento | Director |
| Ciencias Agrarias | Investigación en Ciencias Agrarias | Manuel Carlos Medina Lopera |
| Ciencias Básicas y de la Educación | Humanidades | Martín Alejandro Corrales Parra |
| Investigación en Ciencias Básicas | Diego Alonso Mancilla Perilla |
| Idiomas | Roger Miller Campbell |
| Ciencias Básicas | Myriam Adriana Rodríguez López |
| Ingeniería | Investigación en Ingeniería | Gloria Mercedes Cardona Sotelo |

Tabla – Dominio: Directores de departamento por facultad

Un nivel más abajo en el organigrama, la universidad cuenta con 56 áreas, las cuales pueden pertenecer a una facultad o a un departamento, pero no a ambos.

|  |  |
| --- | --- |
| Facultad | Área |
| Ciencias Agrarias | Jefatura del CIDT Pinares de Tenjo |
| Sección de Laboratorios de Ciencias Agrarias |
| Ciencias Básicas y de la Educación | Sección de Laboratorios de Ciencias Básicas |
| Escuela de Formación y Perfeccionamiento Docente |
| Ciencias Económicas, Administrativas y Contables | Consultorio Contable |
| Ingeniería | Sección de Laboratorios de Ingeniería |
| Ciencias Jurídicas y Sociales | Consultorio Jurídico |
| Centros de Conciliación |

Tabla - Dominio: Áreas por facultad

|  |  |
| --- | --- |
| Departamento | Área |
| Investigación Aplicada | Ética y Bioética |
| Investigación Formativa | Divulgación Científica |
| Aseguramiento de la Calidad | Programa Sembrar Paz |
| Autoevaluación |
| Registros Calificados |
| Acreditación |
| Proyectos Educativos en Contextos Rurales | Educación Permanente |
| Articulación con la Educación Media |
| Gestión Humana y Desarrollo Organizacional | Capacitación y Desarrollo de Personal |
| Seguridad y Salud en el Trabajo |
| Reclutamiento y Selección de Personal |
| Administración de Personal |
| Logística y Servicios Administrativos | Servicios Generales y Transportes |
| Compras y Almacén |
| Infraestructura |
| Sistemas y Tecnología | Soporte Sistema Institucional |
| Soporte Técnico y Telecomunicaciones |
| Ayudas Educativas |
| Administración Dependencia Facatativá | Servicios Administrativos |
| Servicios del Medio Universitario |
| Jefatura de Ingeniería Civil |
| Jefatura de Ingeniería Mecatrónica |
| Jefatura de Derecho |
| Administración Dependencia Teusaquillo | Servicios Administrativos |
| Servicios del Medio Universitario |
| Jefatura de Derecho |
| Educación Virtual | Innovación Educativa |
| Desarrollo de Programas Virtuales |
| Ambientes Virtuales de Aprendizaje |
| Bienestar Universitario | Acompañamiento Estudiantil |
| Integración con el Egresado |
| Psicología Clínica |
| Odontología |
| Arte y Cultura |
| Actividad Física y Deporte |
| Medicina Estudiantil |
| Formación Académica | Emprendimiento |
| Estudios Ambientales |
| Registro y Control |
| Sección de Bibliotecas |
| Secretaría General | Archivo y Gestión Documental |
| Planeación | Sistemas Integrados de Gestión |
| Gestión de la Información Institucional |
| Planeación Presupuestal |
| Planificación Estratégica |
| Mercadeo | Permanencia Estudiantil |
| Comunicaciones |
| Investigación de Mercados |

Tabla - Dominio: Áreas por departamento

Cada área cuenta con un jefe de área, dos coordinadores, cuatro asistentes y dos asistentes junior o aprendices.

Es decir, se tienen las siguientes cifras

* 56 jefes de área: un jefe por área
* 112 coordinadores: dos por área
* 224 asistentes: cuatro por área
* 112 asistentes junior: dos por área

Lo cual suma un total de 504 empleados en el sector administrativo de la universidad.

Adicionalmente se cuenta con 26 auxiliares:

* 3 auxiliares de inventarios – Área de Compras y Almacén
* 3 auxiliares de mantenimiento – Área de Servicios Generales y Transportes
* 12 auxiliares de laboratorio – Sección de Laboratorios de Ingeniería
* 5 auxiliares de laboratorio – Sección de Laboratorios de Ciencias Agrarias
* 3 auxiliares de laboratorio – Sección de Laboratorios de Ciencias Básicas

La universidad ofrece 16 programas académicos actualmente. Cada programa pertenece a una facultad:

|  |  |
| --- | --- |
| Facultad | Programa |
| Ciencias Agrarias | Especialización en Bienestar Animal y Etología |
| Medicina Veterinaria |
| Zootecnia |
| Ciencias Básicas y de la Educación | Licenciatura en Ciencias Naturales y Educación Ambiental |
| Ciencias Económicas, Administrativas y Contables | Administración Financiera y de Sistemas |
| Contaduría Pública |
| Especialización en Gestión de Agronegocios |
| Especialización en Sistemas de Gestión Integrada HSEQ |
| Ingeniería | Especialización en Seguridad Industrial, Higiene y Gestión Ambiental |
| Ingeniería Agroindustrial |
| Ingeniería Civil |
| Ingeniería de Alimentos |
| Ingeniería Industrial |
| Ingeniería Mecatrónica |
| Ciencias Jurídicas y Sociales | Derecho |
| Especialización en Legislación Rural y Ordenamiento Territorial |

Tabla - Dominio: Programas de la universidad

Actualmente la universidad cuenta con un plantel de 80 docentes cuyo contrato es a tiempo completo, tienen funciones administrativas en sus programas y facultades y trabajan en investigación.

Los docentes están adscritos a un programa (en lugar de un área) y además cuentan con un rol entre cuatro escalafones que maneja la universidad.

|  |  |  |
| --- | --- | --- |
| **Nombre del programa** | **Rol / Escalafón** | **N°** |
| Administración Financiera y de Sistemas | Asociado | 2 |
| Instructor | 2 |
| Contaduría Pública | Titular | 2 |
| Instructor | 2 |
| Asociado | 2 |
| Derecho | Titular | 3 |
| Instructor | 2 |
| Asociado | 3 |
| Especialización en Bienestar Animal y Etología | Titular | 2 |
| Instructor | 1 |
| Asistente | 1 |
| Especialización en Gestión de Agronegocios | Asociado | 1 |
| Titular | 1 |
| Instructor | 1 |
| Especialización en Legislación Rural y Ordenamiento Territorial | Instructor | 1 |
| Asociado | 2 |
| Titular | 2 |
| Especialización en Seguridad Industrial, Higiene y Gestión Ambiental | Asociado | 1 |
| Instructor | 1 |
| Titular | 1 |
| Especialización en Sistemas de Gestión Integrada HSEQ | Titular | 3 |
| Instructor | 2 |
| Asistente | 2 |
| Ingeniería Agroindustrial | Instructor | 2 |
| Titular | 3 |
| Asistente | 2 |
| Ingeniería Civil | Asistente | 1 |
| Instructor | 1 |
| Titular | 2 |
| Ingeniería de Alimentos | Asociado | 1 |
| Instructor | 1 |
| Titular | 1 |
| Ingeniería Industrial | Asistente | 1 |
| Instructor | 1 |
| Titular | 1 |
| Ingeniería Mecatrónica | Instructor | 3 |
| Asociado | 3 |
| Titular | 3 |
| Licenciatura en Ciencias Naturales y Educación Ambiental | Asociado | 3 |
| Titular | 3 |
| Instructor | 2 |
| Medicina Veterinaria | Asociado | 1 |
| Instructor | 1 |
| Titular | 1 |
| Zootecnia | Asociado | 2 |
| Instructor | 1 |

Tabla - Número de docentes por rol para cada programa de la universidad.

El resultado obtenido fue aleatorio según la inserción de datos que se realizó posteriormente en la base de datos

Cada docente tiene un título que representa el máximo nivel académico alcanzado en sus estudios. En general, todos los docentes con el rol de Instructor tienen un título de Especialización. Los docentes con un título de Maestría pueden pertenecer a las categorías de Asistente, Asociado y Titular. Los docentes con un título de Doctorado son docentes asociados y titulares.

Así entonces, la jerarquía de entidades en la universidad está dada por el siguiente diagrama:

Facultad

Vicerrectoría

Departamento

Programa

Área

*Figura 2 – Relaciones jerárquicas entre las entidades de la universidad*

Los arcos en color naranja indican una relación opcional excluyente.

En relación a la jerarquía en los cargos de la universidad. Se tiene que los vicerrectores están en el nivel más alto junto con los decanos de facultad. Un nivel más abajo está los directores de departamento quienes pueden trabajar para una facultad o una vicerrectoría. Vicerrectores, decanos y directores no están siendo contados dentro de los 610 funcionarios de la empresa.

Este grupo de trabajadores en la empresa no son interesantes en la asignación de activos fijos, y por eso no necesitan incluirse dentro de la tabla que contenga a todos los funcionarios de la universidad.

El siguiente diagrama describe las relaciones de jerarquía en la universidad:

Asistente Jr.

Asistente

Auxiliar

Jefe de Área

Director

Vicerrector

Decano

Docente

Coordinador

*Figura 3 – Relaciones jerárquicas entre las entidades de la universidad incluyendo los cargos de los funcionarios*

Los arcos en color naranja indican una relación opcional excluyente. Los auxiliares tienen como jefe inmediato al jefe de área directamente. Algunos jefes de área son subordinados de un Decano y otros de un Director, tal como se mostró en la tabla de Áreas anteriormente.

Entre los docentes no hay una jerarquía diferente de su escalafón. Todos responden al decano de la facultad del programa al que pertenecen.

Adicional a esta estructura organizacional descrita, la base de datos debe reflejar la estructura física de la universidad según otros sistemas actuales de la universidad, con el fin de guardar la consistencia que se necesita y para registrar la ubicación física de los activos.

La universidad está dividida en cuatro sedes:

* Sede Principal
* Dependencia Facatativá
* Dependencia Teusaquillo
* CIDT Pinares de Tenjo

La sede más grande es la principal, y como tal, tiene un mayor número de edificios o bloques. A continuación se listan los edificios que pertenecen a la universidad para cada una de las sedes:

|  |  |
| --- | --- |
| Sede | Edificio |
| Sede Principal | Bloque A |
| Bloque B |
| Bloque C |
| Bloque D |
| Bloque E |
| Bloque F |
| Bloque G |
| Bloque H |
| Bloque I |
| Bloque J |
| Dependencia Facatativá | Bloque Principal |
| Bloque Carrera 15 |
| Dependencia Teusaquillo | Bloque Principal |
| Bloque Carlos Fernando Paredes Millán |
| CIDT Pinares de Tenjo | Bloque Principal |
| Bloque Administrativo |
| Bloque Laboratorios |
| Bloque Operarios |

Tabla - Dominio: Edificios de la universidad

La universidad tiene a su disposición 544 salones de clase repartidos en cada sede, así:

* Sede Principal: 262 salones
* Dependencia Teusaquillo: 92 salones
* Dependencia Facatativá: 78 salones
* CIDT Pinares de Tenjo: 112 salones

De entre los cuales hay 19 talleres, en los cuales se realizan actividades extracurriculares como danza, música, carpintería y de algunas asignaturas electivas de los programas como fotografía, pintura, etc.

Dentro de estos 544 salones también hay 14 salas de cómputo, cuya capacidad está dada por el número de computadores disponibles en dicha locación.

Además, hay 9 auditorios en total repartidos en algunos edificios de la universidad:

|  |  |  |  |
| --- | --- | --- | --- |
| Sede | Edificio | Auditorio | Capacidad |
| Sede Principal | Bloque A | Auditorio Santiago Francisco Mora Guerra | 120 |
| Bloque C | Auditorio Manuel Rafael Salas Gil | 100 |
| Bloque E | Auditorio Orlando Jerónimo Varela Salazar | 130 |
| Bloque G | Auditorio Alejandra Silvana Pinilla Vásquez | 100 |
| Bloque I | Auditorio Ricardo Felipe Romero Ortega | 130 |
| Dependencia Teusaquillo | Bloque Principal | Auditorio John Orlando Galeano Moreira | 130 |
| Dependencia Facatativá | Bloque Principal | Auditorio Jéssica Verónica Caballero Rivera | 130 |
| CIDT Pinares de Tenjo | Bloque Operarios | Auditorio Andrea Gina Vásquez Prieto | 110 |
| Bloque Principal | Auditorio Rosa Johanna Costa Gómez | 130 |

Tabla - Dominio: Auditorios de la universidad

*El resultado de los nombres para cada auditorio como de su capacidad fue obtenido aleatoriamente del procedo de inserción de datos que se describe más adelante en el documento.*

Otra parte importante de la estructura física de la universidad son las oficinas. En total, en toda la universidad hay 84 oficinas, algunas lo suficientemente grandes como para contener hasta diez puestos de trabajo o cubículos. Las más pequeñas tienen solamente tres puestos de trabajo.

La repartición de las oficinas por sede es la siguiente:

* Sede Principal: 44 oficinas
* Dependencia Teusaquillo: 13 oficinas
* Dependencia Facatativá: 11 oficinas
* CIDT Pinares de Tenjo: 16 oficinas

Cada oficina pertenece a un área en específico. Un área puede tener 1 o 2 oficinas para sus trabajadores.

Finalmente, se tiene un tipo especial de locación, el cual corresponde a los laboratorios. En total hay 19 laboratorios en la universidad. 15 de ellos se encuentran en la sede principal y 4 en el CIDT Pinares de Tenjo. Cada laboratorio tiene asignado a un auxiliar, quien se responsabiliza por una serie de tareas operativas encomendadas desde la sección de laboratorios a la que esté adscrito.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sede | Edificio | Laboratorio | Auxiliar | Área |
| Sede Principal | Bloque E | Biocombustibles | Steven José Sánchez Silva | Sección de Laboratorios de Ingeniería |
| Investigación, Calidad y Análisis | Ángel Paul Beltrán López |
| Investigación y Desarrollo Metrológico | Jaime Andrés Prieto Asensio |
| Instrumentación Analítica | Edgar Eloy Santos Montero |
| Química | Julián José Santamaría Varela | Sección de Laboratorios de Ciencias Básicas |
| Ciencias Biológicas | Nathaly Myriam Montes Campos |
| Física | Néstor Paul Usaquén Pérez |
| Fitoquímica | Silvia Nicol Segura Herrera | Sección de Laboratorios de Ciencias Agrarias |
| Bloque I | Materiales | Victor Augusto Galeano Núñez | Sección de Laboratorios de Ingeniería |
| Neumática | Alex Bernardo Valencia Flores |
| Ingredientes Naturales | Valentina Elisa Rey Peña |
| Hidráulica | Nicol Gina Molina Álvarez |
| Bioensayos y Fuentes de Alimentación No Convencionales | Alexis David Valencia Ramírez |
| Electrónica | Lorena Luz Espinoza Salas |
| Geomática | César Andrés Rubio Murillo |
| CIDT Pinares de Tenjo | Bloque Laboratorios | Mecánica de Suelos | Valentina Lina Amador Peña | Sección de Laboratorios de Ciencias Agrarias |
| Microbiología | Alberto Eloy Calderón Domínguez |
| Nutrición y Alimentación | Mónica Xiomara Segura Caballero |
| Pavimentos | Jerónimo Daniel Cabrera Arias | Sección de Laboratorios de Ingeniería |

Tabla - Dominio: Laboratorios de la universidad y Auxiliares de laboratorios.

Los nombres para cada auxiliar se generaron aleatoriamente según el proceso de inserción

Hay un auxiliar de laboratorio que aún no ha sido asignado a ningún laboratorio. Se trata de Enrique Edison Cardona Rey que trabaja para la Sección de Laboratorios de Ciencias Agrarias.

Así entonces, la estructura física de la universidad está dada por el siguiente diagrama:

Sede

Edificio

Auditorio

Salón de Clases

Oficina

Laboratorio

Taller

Sala de Cómputo

Auxiliar

Área

*Figura 4 – Diagrama que detalla la estructura física de la universidad y su relación con algunas entidades*

Finalmente se hablará de los activos en sí. Debe existir un registro por activo en la base de datos. Este registro debe indicar los datos de identificación del mismo, detalles sobre su descripción física y otros que aparezcan en la factura de compra, la información de interés para el personal de Contabilidad, su ubicación física y funcionario responsable, cuando aplique.

Habiendo definido todas estas estas especificaciones, están las condiciones dadas para inicia la metodología de diseño de la base de datos.

**Metodología de diseño de la base de datos**

***Definición de entidades***

Inicialmente se consideraron

* ACTIVO
* COMPRA
* FUNCIONARIO
* ÁREA
* DEPARTAMENTO
* FACULTAD
* PROGRAMA
* VICERRECTORÍA
* LOCACIÓN
* EDIFICIO
* SEDE

***Definición de subtipos y supertipos***

Hay dos supertipos en las entidades listadas anteriormente: FUNCIONARIO y LOCACIÓN.

Un funcionario puede ser un EMPLEADO o un DOCENTE.

Los empleados son los funcionarios que hacen parte del plantel administrativo u operativo de la universidad. Dentro de esta categoría solo se están incluyendo los jefes de área, coordinadores, asistentes y auxiliares guardando la jerarquía presentada en la figura [3]. No se incluirán vicerrectores, directores ni decanos. Los empleados están adscritos a un área específica de la universidad.

Se considerará otro subtipo llamado AUXILIAR, el cual es una especialización de EMPLEADO.

Los docentes conforman el plantel contratado en la universidad para ejercer su labor en tiempo completo, apoyar en tareas administrativas y de investigación. Están adscritos a un programa. No existe una jerarquía entre ellos diferente a su escalafón.

Una locación puede ser un SALÓN, una OFICINA o un LABORATORIO, según se muestra en la figura [4]. Los auditorios y salones de clase se considerarán como salones, es decir, como un único subtipo.

Más adelante se presentarán los conjuntos de atributos de cada una de estas entidades, que mostrarán que si se pueden hacer estas especializaciones, según los dos criterios:

* Los subtipos tienen una llave primaria común que permanecerá en su supertipo
* Los subtipos tienen un subconjunto de atributos en común que permanecerá en su supertipo

Así entonces, las entidades definitivas para los diagramas de entidad-relación son las siguientes:

* E1 – ACTIVO
* E2 – COMPRA
* E3 – FUNCIONARIO
* E4 – DOCENTE
* E5 – EMPLEADO
* E6 – AUXILIAR
* E7 – VICERRECTORÍA
* E8 – DEPARTAMENTO
* E9 – ÁREA
* E10 – FACULTAD
* E11 – PROGRAMA
* E12 – SEDE
* E13 – EDIFICIO
* E14 – LOCACION
* E15 – OFICINA
* E16 – SALÓN
* E17 – LABORATORIO

***Definición de relaciones entre entidades***

Las relaciones entre entidades, incluyendo los subtipos generados, se presentan en la siguiente tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RELACIONES FINALES | | NOMBRE | TIPO | CARDINALIDAD | PARTICIPACIÓN |
| E1 - E3 | ACTIVO - FUNCIONARIO | *USADO-POR* |  | m-1 |  |
| E1 - E6 | ACTIVO - AUXILIAR | *PROCEDIMIENTO* |  | m-m |  |
| E1 - E14 | ACTIVO - LOCACIÓN | *SE-ENCUENTA-EN* |  | m-1 |  |
| E3 - E3 | FUNCIONARIO - FUNCIONARIO | *TRABAJA-PARA* | RECURSIVA | m-1 |  |
| E6 - E17 | AUXILIAR - LABORATORIO | *AUXILIAR-LAB* |  | 1-1 |  |
| E4 - E11 | DOCENTE - PROGRAMA | *TRABAJA-EN* |  | m-1 |  |
| E5 - E9 | EMPLEADO - ÁREA | *TRABAJA-EN* |  | m-1 |  |
| E9 - E15 | ÁREA - OFICINA | *área-oficina* |  | 1-m |  |
| E9 - E10 | ÁREA - FACULTAD | *área-facultad* | ARCO | m-1 |  |
| E9 - E8 | ÁREA - DEPARTAMENTO | *área-departamento* | m-1 |  |
| E13 - E12 | EDIFICIO - SEDE | *edificio-sede* |  | m-1 |  |
| E14 - E13 | LOCACIÓN - EDIFICIO | *locación-edificio* | DEPENDENCIA | m-1 |  |
| E11 - E10 | PROGRAMA - FACULTAD | *programa-facultad* | DEPENDENCIA | m-1 |  |
| E8 - E7 | DEPARTAMENTO - VICERRECTORIA | *departamento-vicerrectoría* | ARCO | m-1 |  |
| E8 - E10 | DEPARTAMENTO - FACULTAD | *departamento-facultad* | m-1 |  |
| E1 - E2 | ACTIVO - COMPRA | *activo-compra* |  | m-1 |  |

Tabla - Relaciones entre entidades según la metodología de diseño E-R

*Relación m-m AUXILIAR-ACTIVO*

Hay una relación m-m que corresponde a AUXILIAR-ACTIVO y se refiere particularmente a los procedimientos que deben llevar a cabo los auxiliares de inventarios y de mantenimiento, es decir, las entregas de puestos de trabajo, las firmas de paz y salvo cuando un funcionario se retira y devuelve su puesto, las revisiones para verificar el estado de los activos y los mantenimientos cuando se necesita de alguna reparación.

*Relación recursiva FUNCIONARIO-FUNCIONARIO*

De acuerdo con la jerarquía mostrada en la figura [3], un asistente tiene como jefe a un coordinador, un coordinador tiene como jefe a un jefe de área. Los superiores de los jedes fe área no serán registrados, y se tomarán como la cabeza del árbol. Los auxiliares estarán supervisados por el jefe de su área directamente.

*Relaciones opcionales excluyentes entre sí o ARCOS*

Hay dos arcos presentes en el modelo. Un área puede pertenecer a un departamento o a una facultad, pero no a ambos al mismo tiempo. Un departamento puede pertenecer a una vicerrectoría o a una facultad, pero no a ambos al mismo tiempo.

*Relaciones de dependencia para generar llaves primarias compuestas*

En el modelo hay dos relaciones débiles: LOCACION y PROGRAMA

*De acuerdo con* Silberchatz, *una entidad débil es aquella que depende existencialmente de una entidad identificadora. No cuenta con una llave primaria dentro de su esquema, sino con un atributo llamado discriminante el cual es único entre el conjunto de entidades débiles que se relacionan con una misma entidad identificadora, pero puede llegar a repetirse si se compara con entidades débiles relacionadas a entidades identificadoras diferentes. La llave primaria de esta entidad es la concatenación entre la llave primaria de su entidad identificadora y el discriminante.*

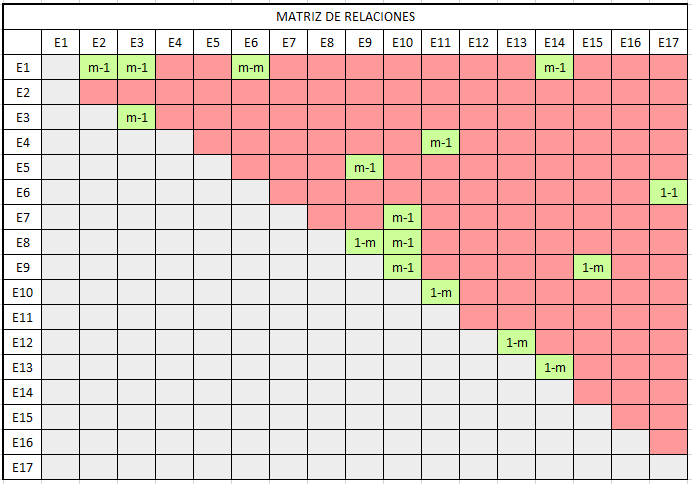
Para solucionar esta circunstancia, se hace uso de una relación de dependencia de la entidad débil a la entidad identificadora. A continuación se presentan las dos relaciones de este tipo que se tienen en el modelo.

LOCACIÓN-EDIFICIO: Cada edificio tiene un identificador para cada locación, pero estos identificadores pueden repetirse entre edificios.

PROGRAMA-FACULTAD: Cada facultad tiene un identificador para cada programa, pero estos identificadores pueden repetirse entre facultades.

Con estas dos relaciones de dependencia se genera una llave compuesta en las entidades LOCACION y PROGRAMA.

A continuación se presenta la matriz de relaciones inicial.



*Figura 5 – Matriz de relaciones inicial según la metodología de diseño E-R*

***Definición de atributos para cada una de las entidades***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ENTIDAD | | ATRIBUTO | TIPO DE DATO | DOMINIO E INTEGRIDAD POR VALOR | EJEMPLO |
| E1 | ACTIVO | # id-activo | Integer | Enteros positivos mayores o iguales a 1 | 1294 |
| \* referencia-activo | Varchar(40) | Códigos alfanuméricos de hasta 40 caracteres | 'XE8IBEYEFPDF6FWZS6K5' |
| \* categoría-activo | Char(3) | Integridad por valor {CAE, DTS, LAB, LOG, SGT} | 'DTS' |
| \* nombre-activo | Varchar(40) | Dado por el tipo de dato Varchar(40) | 'Computador de Escritorio' |
| o material-activo | Varchar(20) | Conjunto de materiales posibles | 'Cuero' |
| \* marca-activo | Varchar(25) | Conjunto de marcas posibles | 'Ceccotti' |
| o color-activo | Varchar(20) | Conjunto de colores posibles | 'Azul Oscuro' |
| o tamaño-activo | Varchar(20) | Adjetivos de tamaño, dimensiones en alguna unidad, bidimensionales y tridimensionales | '25 pulgadas' |
| o componentes-activo | Varchar(120) | Dado por el tipo de dato Varchar(120) | 'Soporte, Kit de Repuesto' |
| \* valor-activo | Money(16, 2) | Reales positivos hasta 9999999999999999.99 con dos cifras decimales de precisión | 4003603 |
| \* depreciación-activo | Porcentaje | Reales positivos hasta 100 con dos cifras decimales de precisión | 8 |
| E2 | COMPRA | # id-compra | Integer | Enteros positivos mayores o iguales a 2020000 | 2020017 |
| \* fecha-compra | Date | Fechas de la forma 'DD/MM/YYYY' desde 01/01/2020 | 9/02/2020 |
| \* valor-compra | Money(16, 2) | Reales positivos hasta 9999999999999999.99 con dos cifras decimales de precisión | 16084234 |
| E3 | FUNCIONARIO | # id-funcionario | Integer | Enteros positivos mayores o iguales a 1017005000 | 1018005241 |
| \* nombre-funcionario | Varchar(50) | Dado por el tipo de dato Varchar(50) | 'Sergio Emilio Molina Pereira' |
| \* ini\_contrato-funcionario | Date | Fechas de la forma 'DD/MM/YYYY' desde 01/01/2020 | 1/04/2020 |
| o fin\_contrato-funcionario | Date | Fechas de la forma 'DD/MM/YYYY' desde 31/01/2020 | 30/06/2020 |
| E4 | DOCENTE | \* título-docente | Varchar(100) | {'Especialiación', 'Maestría', 'Doctorado'} | 'Especialización' |
| \* rol-docente | Varchar(20) | {'Instructor', 'Asistente', 'Titular', 'Asociado'} | 'Asociado' |
| E5 | EMPLEADO | \* cargo-empleado | Varchar(30) | {'Jefe de Área', 'Coordinador', 'Asistente', 'Asistente Jr.', 'Auxiliar'} | 'Asistente Jr.' |
| E6 | AUXILIAR | \* tipo-auxiliar | Char(3) | Integridad por valor {INV, MAN, LAB} | 'INV' |
| E7 | VICERRECTORÍA | # id-vicerrectoría | Char(3) | Integridad por valor {FOR, INV, EXT, ADM} | 'EXT' |
| \* nombre-vicerrectoría | Varchar(50) | Nombres de las 4 vicerrectorías existentes | 'Vicerrectoría de Administración y Financiera' |
| \* vicerrector-vicerrectoría | Varchar(50) | Nombres de los 4 vicerrectores existentes | 'Luis Augusto Moreno Boadas' |
| E8 | DEPARTAMENTO | # id-departamento | Integer | Entero entre 1 y 23 incluyendo extremos | 21 |
| \* nombre-departamento | Varchar(70) | Nombres de los 23 departamentos existentes | 'Idiomas' |
| \* director-departamento | Varchar(50) | Nombres de los 23 directores existentes | 'Myriam Adriana Rodríguez López' |
| E9 | ÁREA | # id-área | Integer | Entero entre 1 y 56 incluyendo extremos | 42 |
| \* nombre-área | Varchar(70) | Nombres de las 56 áreas existentes | 'Medicina Estudiantil' |
| E10 | FACULTAD | # id-facultad | Char(3) | Integridad por valor {AGR, JUR, ECO, ING, CIE} | 'JUR' |
| \* nombre-facultad | Varchar(60) | Nombres de las 5 facultades existentes | 'Ciencias Básicas y de la Educación' |
| \* decano-facultad | Varchar(50) | Nombres de los 5 decanos existentes | 'Sandra Milena Piedrahita Buitrago' |
| E11 | PROGRAMA | # id-programa | Char(10) | {'ZOO', 'AFS', 'SGI', 'ALI', 'MEC', 'EGA', 'LCNEA', 'AGR', 'DER', 'IND', 'BAT', 'VET', 'LROT', 'CIV', 'CON'} | 'SGI' |
| \* nombre-programa | Varchar(80) | Nombres de los 16 programas existentes | 'Medicina Veterinaria' |
| E12 | SEDE | # id-sede | Integer | Entero entre 1 y 4 incluyendo extremos | 4 |
| \* nombre-sede | Varchar(70) | Nombres de las 4 sedes existentes | 'Dependencia Teusaquillo' |
| E13 | EDIFICIO | # id-edificio | Integer | Entero entre 1 y 18 incluyendo extremos | 15 |
| \* nombre-edificio | Varchar(50) | Nombres de los 18 edificios existentes | 'Bloque B' |
| E14 | LOCACIÓN | # id-locación | Integer | Entero entre 1 y 56 incluyendo extremos para salones. Para las oficinas, enteros de la forma AA00N donde AA es el identificador del área al que pertenece la oficina y N es el número de la oficina, sea 1 o 2. Enteros entre 1 y 9 incluyendo extremos para los laboratorios | 34, 24002, 3 |
| \* nombre-locación | Varchar(100) | Para los salones, nombres de la forma 'Salón EN' donde E es la inicial del edificio y N es el número del salón en ese edificio. Para las oficinas, nombres de la forma 'Oficina EN' donde E es la inicial del edificio y N es el número de la oficina. Para los laboratorios, es el conjunto de nombres de los laboratorios existentes | 'Salón H34', ‘Oficina C25002’, ‘Fitoquímica’ |
| E15 | OFICINA | \* puestos-oficina | Integer | Entero entre 1 y 10 incluyendo extremos | 4 |
|  | SALÓN | \* capacidad-salón | Integer | Para las salas de cómputo {15, 16, 17, 18} , Para los auditorios {100, 110, 120, 130}, Para los salones {20, 30, 40, 50, 60} | 17, 110, 20 |
| o tipo-salón | Char(3) | Integridad por valor {AUD, TAL, COM} | 'AUD' |
| E17 | LABORATORIO | o acreditación-laboratorio | Varchar(20) | {'IEC-ISO 15189', 'IEC-ISO 17025', 'En Proceso'} | 'IEC-ISO 15189' |

Tabla - Detalle de atributos para cada entidad según la metodología de diseño E-R indicando el dominio definido para cada uno

*Observaciones:*

* No todos los activos vienen indicando el material, color, tamaño o componentes adicionales en su factura.
* Cada componente se separa por una coma y un espacio
* Algunos funcionarios son contratados a término indefinido
* Un área tiene una o dos oficinas.

Cuando se dice algo de la forma ‘*nombre de facultades existentes’* se refiere a los conjuntos de datos presentados en las especificaciones del diseño para la entidad en cuestión donde se mostraron las tablas de dominios.

*Mnemotécnicas para los valores cuyo dominio está definido por siglas:*

ACTIVO.CATEGORIA\_ACTIVO

Los activos pueden ser clasificados en cinco categorías según sea el área que los administra.

* (CAE) Centro de Ayudas Educativas: Es el área que se administra los televisores, video beams y otros dispositivos que se encuentran en los salones para apoyar las clases.
* (DTS) División de Tecnología y Sistemas: También llamada Soporte Técnico y Telecomunicaciones, es el área que administra los computadores y otros dispositivos de las salas de cómputo y de las oficinas.
* (LOG) Logística y Servicios Administrativos: Es el departamento que administra los equipos que se utilizan en eventos estudiantiles y administrativos, como por ejemplo, los equipos de sonido de los auditorios.
* (SGT) Servicios Generales y Transportes: Es el área que administra el mobiliario de la universidad. Esto es, las sillas de los salones y oficinas y demás locaciones de la universidad.
* (LAB) Secciones de Laboratorios: Las tres secciones de laboratorio administran todos los equipos usados en sus instalaciones.

AUXILIAR.TIPO\_AUXILIAR

* (INV) Auxiliar de Inventarios
* (MAN) Auxiliar de Mantenimiento
* (LAB) Auxiliar de Laboratorios

VICERRECTORIA.ID\_VICERRECTORIA

* (FOR) Vicerrectoría de Formación
* (INV) Vicerrectoría de Investigación
* (EXT) Vicerrectoría de Extensión y Proyección Social
* (ADM) Vicerrectoría de Administración y Financiera

FACULTAD.ID\_FACULTAD

* (AGR) Facultad de Ciencias Agrarias
* (JUR) Facultad de Ciencias Jurídicas y Sociales
* (ECO) Facultad de Ciencias Económicas, Administrativas y Contables
* (ING) Facultad de Ingeniería
* (ADM) Facultad de Ciencias Básicas y de la Educación

PROGRAMA.ID\_PROGRAMA

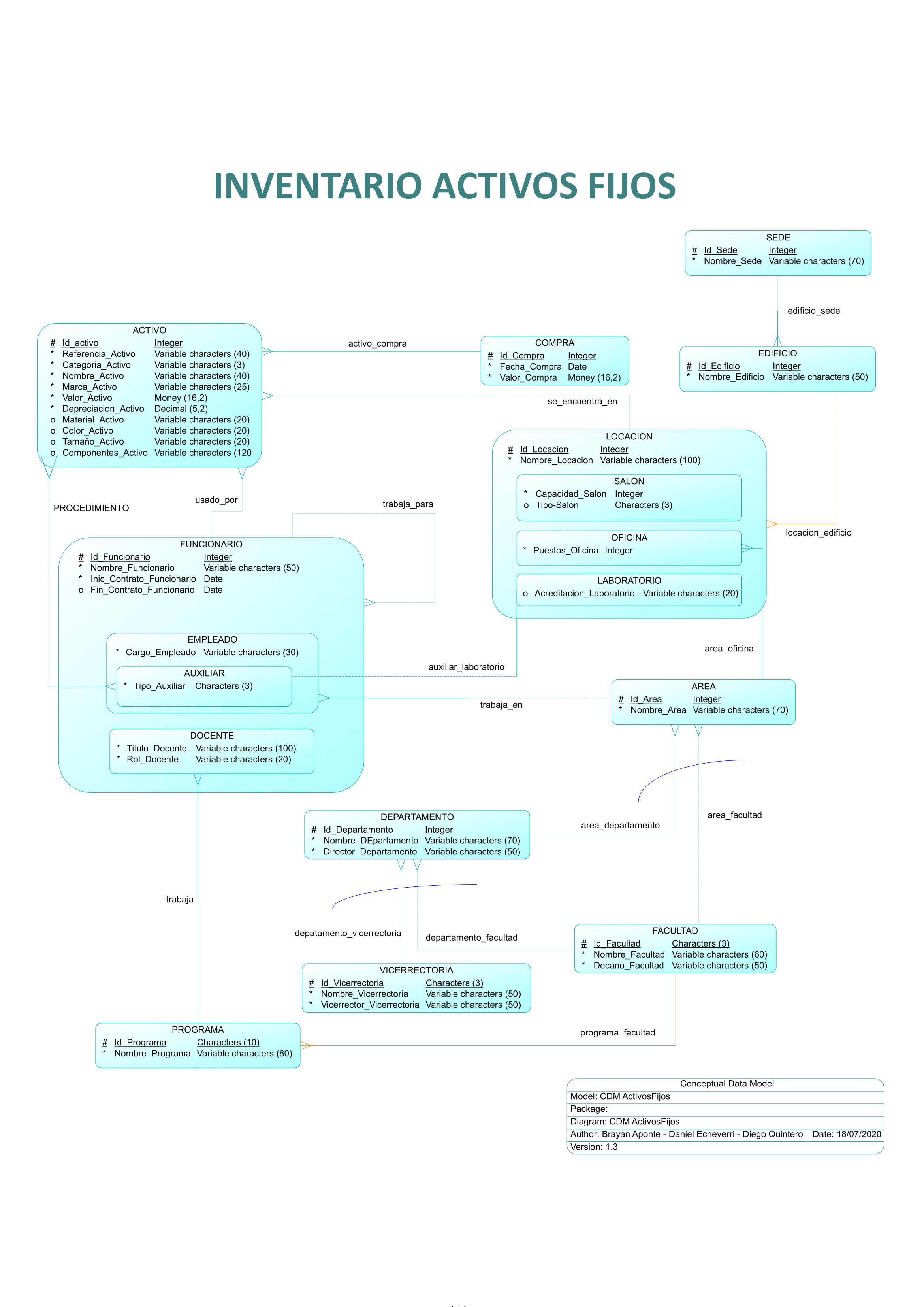
* (AFS) Administración Financiera y de Sistemas
* (AGR) Ingeniería Agroindustrial
* (ALI) Ingeniería de Alimentos
* (BAT) Especialización en Bienestar Animal y Etología
* (CIV) Ingeniería Civil
* (CON) Contaduría Pública
* (DER) Derecho
* (EGA) Especialización en Gestión de Agronegocios
* (IND) Ingeniería Industrial
* (LCNEA) Licenciatura en Ciencias Naturales y Educación Ambiental
* (LROT) Especialización en Legislación Rural y Ordenamiento Territorial
* (MEC) Ingeniería Mecatrónica
* (SGI) Especialización en Sistemas de Gestión Integrada HSEQ, Especialización en Seguridad Industrial, Higiene y Gestión Ambiental
* (VET) Medicina Veterinaria
* (ZOO) Zootecnia

SALON.TIPO\_SALON

Los salones de clase convencionales no tienen una sigla para distinguirlos. Estos salones tienen una capacidad de entre 20 y 60 personas.

* (AUD) Auditorios: Son salones con capacidad de entre 100 y 130 personas.
* (TAL) Talleres: Son salones especializados para algunas asignaturas o actividades de la universidad como por ejemplo la carpintería o la música. Su capacidad está entre las 20 y 60 personas.
* (COM) Salas de Cómputo: Son salas dedicadas a impartir clases que requieren del uso de computadores. Su capacidad está entre las 15 y 18 personas según el número de computadores disponibles.

***Construcción del modelo entidad-relación inicial o modelo de datos conceptual CDM en PowerDesigner***



*Figura 6 – Modelo de datos conceptual (CDM) o diagrama E-R inicial*

***Resolución de relaciones m-m para la generación de entidades intermedias***

En el presente modelo se tiene una única relación de cardinalidad m-m, la cual es AUXILIAR-ACTIVO. Esta relación represente una serie de procedimientos llevados a cabo por los auxiliares de inventarios y de mantenimiento.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RELACIONES m-m | | NOMBRE | CARDINALIDAD | PARTICIPACIÓN |
| E1 - E6 | ACTIVO - AUXILIAR | *PROCEDIMIENTO* | m-m |  |

Se generará una entidad intermedia llamada PROCEDIMIENTO

|  |  |  |
| --- | --- | --- |
| ENTIDAD INTERMEDIA GENERADA | | |
| E18 | PROCEDIMIENTO | Llevará un registro de las entregas de puestos de trabajo, firmas de paz y salvos, revisiones y mantenimientos de activos fijos. El tipo de procedimiento a realizar dependerá del tipo de auxiliar |

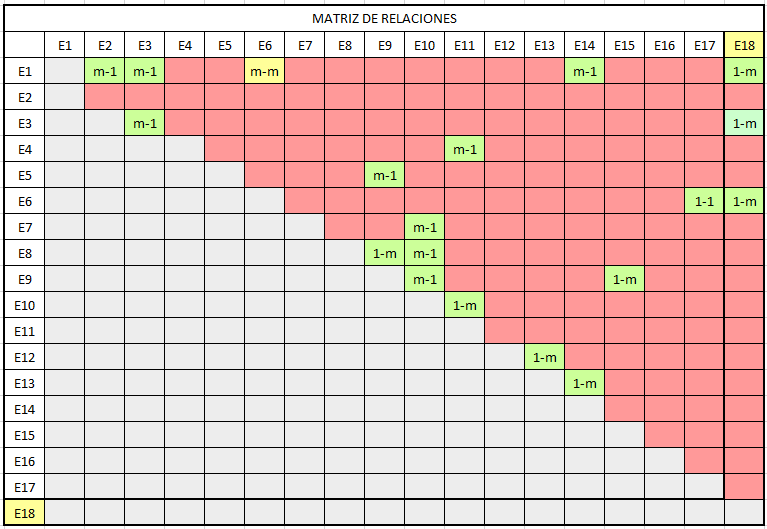
Esta nueva entidad tendrá una relación de cardinalidad m-1 con ACTIVO y AUXILIAR.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUEVAS RELACIONES | | NOMBRE | TIPO | CARDINALIDAD | PARTICIPACIÓN |
| E1 - E18 | ACTIVO - PROCEDIMIENTO | *ACTIVO-PROCEDIMIENTO* | DEPENDENCIA | 1-m |  |
| E6 - E18 | AUXILIAR - PROCEDIMIENTO | *AUXILIAR-PROCEDIMIENTO* | DEPENDENCIA | 1-m |  |

Adicionalmente, se establecerá una nueva relación entre la entidad intermedia PROCEDIMIENTO y FUNCIONARIO. La cual busca generar una llave foránea en PROCEDIMIENTO que indique cuál es el funcionario al cual se le está haciendo una entrega del puesto de trabajo, uno de los procedimientos que tienen a cargo los auxiliares de inventario.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NUEVAS RELACIONES | | NOMBRE | CARDINALIDAD | PARTICIPACIÓN |
| E3 - E18 | FUNCIONARIO - PROCEDIMIENTO | *ENTREGA* | 1-m |  |

A continuación se presenta la matriz de relaciones extendida, luego de este proceso:



*Figura 5 – Matriz de relaciones final según la metodología de diseño E-R*

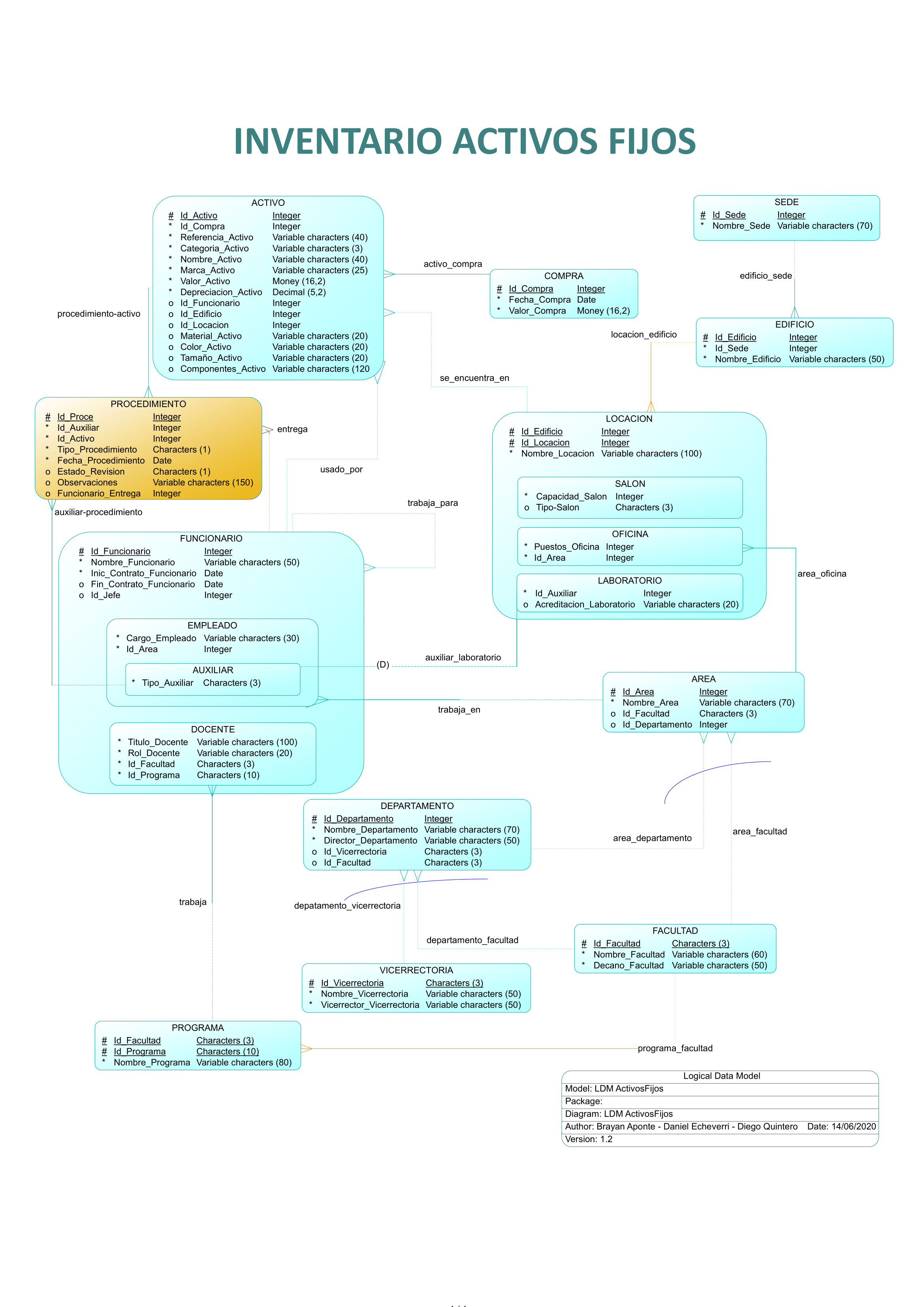
***Definición de atributos para las entidades intermedias generadas:***

No se dejará la llave primaria compuesta generada de la relación m-m entre ACTIVO y AUXILIAR, sino que se asignará un atributo adicional que cumpla esta función:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENTIDAD | | ATRIBUTO | TIPO DE DATO | INTEGRIDAD POR VALOR |
| E18 | PROCEDIMIENTO | # id-proce | Integer |  |
| # id-activo | Integer |  |
| # id-funcionario | Integer |  |
| \* tipo-procedimiento | Char(1) | [E]ntrega, [P]az y salvo, [R]evisión, [M]antenimiento |
| \* fecha-procedimiento | Date |  |
| o estado-revision | Char(1) | [R]eemplazo, [M]antenimiento |
| o observaciones-procedimiento | Varchar(150) |  |

Los conceptos de Entrega, Paz y salvo, Revisión, Mantenimiento y Reemplazo en el contexto del desarrollo fueron explicados en las especificaciones del diseño presentadas anteriormente en el documento.

***Construcción del modelo entidad-relación final o modelo de datos lógico LDM en PowerDesigner***

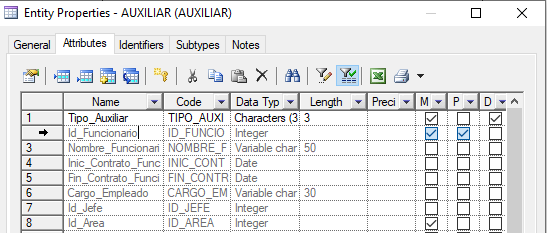


*Figura 7 – Modelo de datos lógico (LDM) o diagrama E-R final*

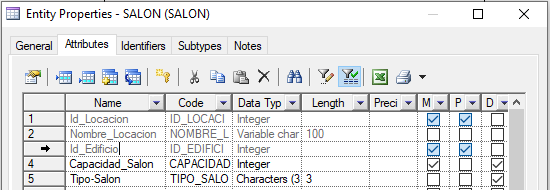
Para construir y validar este modelo de datos, fue necesario hacer las siguientes configuraciones:

Para los subtipos, es decir para AUXILIAR, EMPLEADO, DOCENTE, SALÓN, OFICINA y LABORATORIO, asegurarse que en los atributos aparezca marcado como llave primaria, el atributo destinado para ello y que pertenece al supertipo, es decir FUNCIONARIO y LOCACION.

A continuación se muestran dos ejemplos.

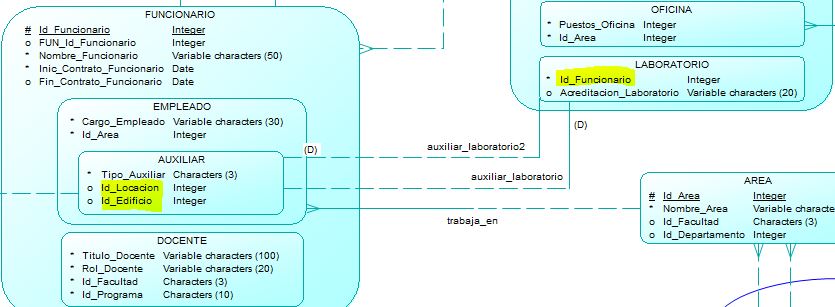


*Figura 8 – Configuración para garantizar la integridad referencial en PowerDesigner cuando hay supertipos y subtipos para el modelo de datos lógico (LDM)*



*Figura 9 – Configuración para garantizar la integridad referencial en PowerDesigner cuando hay supertipos y subtipos para el modelo de datos lógico (LDM)*

Marcar estas casillas donde aplique, debe garantizar que se puedan ver las llaves foráneas donde corresponde. Por ejemplo, a continuación se muestra como la llave foránea (compuesta) de laboratorio se presenta en AUXILIAR y la llave foránea del auxiliar se presenta en LABORATORIO.



*Figura 10 – Integridad referencial configurada apropiadamente en presencia de supertipos y subtipos para el modelo LDM*

Particularmente con esta relación 1-1 entre AUXILIAR y LABORATORIO, se debe eliminar uno de los enlaces entre las dos entidades, para dejar que la llave foránea esté en solo una de las entidades.

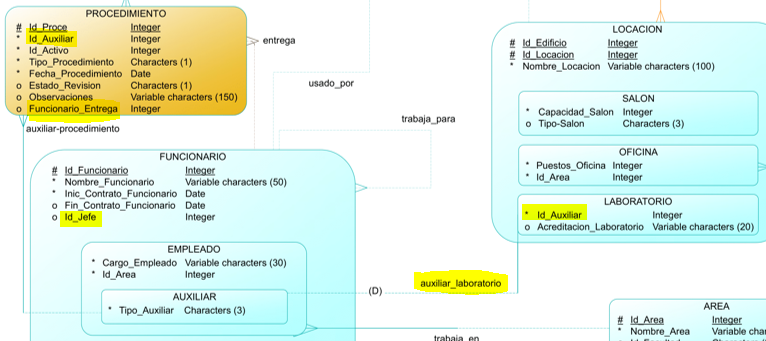
Para el presente modelo, se eliminó el enlace *auxiliar\_laboratorio* dejando así el enlace *auxiliar\_laboratorio2*, al cual posteriormente se le modificó el nombre a *auxiliar\_laboratorio* para guardar la notación. De esta manera se tiene la llave foránea en LABORATORIO únicamente, es decir, el extremo opcional de la relación, pues un AUXILIAR no necesariamente debe tener asignado un LABORATORIO, pero un LABORATORIO si debe tener asignado un AUXILIAR obligatoriamente.

Otras modificaciones realizadas corresponden a los nombres de los atributos generados como llaves foráneas en varias entidades. A continuación se mencionan:

* El atributo *id\_funcionario* en LABORATORIO y PROCEDIMIENTO se cambió a *id\_auxiliar*.
* El atributo *FUN\_id\_funcionario* en FUNCIONARIO que surgió de la relación recursiva se cambió a *id\_jefe*

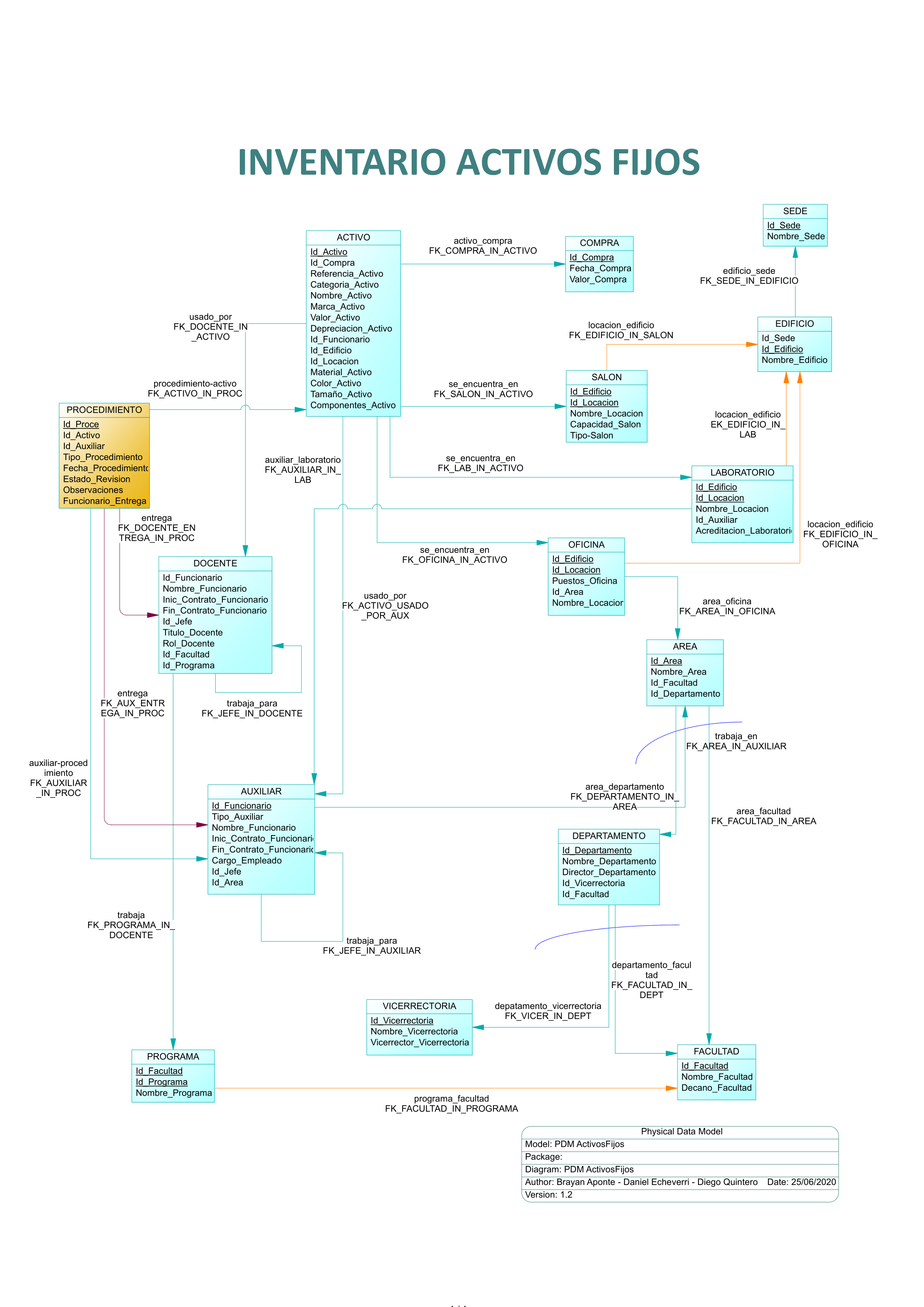
También se agregaron los atributos definidos para la relación intermedia junto con el enlace de la relación PROCEDIMIENTO-FUNCIONARIO, lo cual generó una llave foránea en PROCEDIMIENTO llamada id\_funcionario, la cual se cambió a *funcionario\_entrega*.

Todas estas modificaciones resultan en lo siguiente:



*Figura 11 – Modificaciones finales al modelo LDM*

***Generación del modelo estructura-dato o modelo de datos físico PDM en PowerDesigner***



*Figura 11 – Modelo de datos físico (PDM) o diagrama E-D*

***Generación del código de creación SQL inicial de la base de datos con PowerDesigner***

A continuación se presenta el código de creación SQL de la base de datos según es generado por PowerDesigner. La versión usada para la generación del código fue ORACLE 11g.

Todo código SQL en el documento se presenta con el presente estilo.

/\*==============================================================\*/

/\* DBMS name: ORACLE Version 11g \*/

/\* Created on: 25/06/2020 12:34:49 p. m. \*/

/\*==============================================================\*/

alter table ACTIVO

drop constraint FK\_COMPRA\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_OFICINA\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_LAB\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_SALON\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_ACTIVO\_USADO\_POR\_AUX;

alter table ACTIVO

drop constraint FK\_DOCENTE\_IN\_ACTIVO;

alter table AREA

drop constraint FK\_DEPARTAMENTO\_IN\_AREA;

alter table AREA

drop constraint FK\_FACULTAD\_IN\_AREA;

alter table AUXILIAR

drop constraint FK\_AREA\_IN\_AUXILIAR;

alter table AUXILIAR

drop constraint FK\_JEFE\_IN\_AUXILIAR;

alter table DEPARTAMENTO

drop constraint FK\_FACULTAD\_IN\_DEPT;

alter table DEPARTAMENTO

drop constraint FK\_VICER\_IN\_DEPT;

alter table DOCENTE

drop constraint FK\_PROGRAMA\_IN\_DOCENTE;

alter table DOCENTE

drop constraint FK\_JEFE\_IN\_DOCENTE;

alter table EDIFICIO

drop constraint FK\_SEDE\_IN\_EDIFICIO;

alter table LABORATORIO

drop constraint FK\_AUXILIAR\_IN\_LAB;

alter table LABORATORIO

drop constraint EK\_EDIFICIO\_IN\_LAB;

alter table OFICINA

drop constraint FK\_AREA\_IN\_OFICINA;

alter table OFICINA

drop constraint FK\_EDIFICIO\_IN\_OFICINA;

alter table PROCEDIMIENTO

drop constraint FK\_AUX\_ENTREGA\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_DOCENTE\_ENTREGA\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_AUXILIAR\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_ACTIVO\_IN\_PROC;

alter table PROGRAMA

drop constraint FK\_FACULTAD\_IN\_PROGRAMA;

alter table SALON

drop constraint FK\_EDIFICIO\_IN\_SALON;

drop index SE\_ENCUENTRA\_EN2\_FK;

drop index USADO\_POR2\_FK;

drop index ACTIVO\_COMPRA\_FK;

drop table ACTIVO cascade constraints;

drop index AREA\_DEPARTAMENTO\_FK;

drop index AREA\_FACULTAD\_FK;

drop table AREA cascade constraints;

drop index TRABAJA\_EN\_FK;

drop table AUXILIAR cascade constraints;

drop table COMPRA cascade constraints;

drop index DEPARTAMENTO\_FACULTAD\_FK;

drop index DEPATAMENTO\_VICERRECTORIA\_FK;

drop table DEPARTAMENTO cascade constraints;

drop index TRABAJA\_FK;

drop table DOCENTE cascade constraints;

drop index EDIFICIO\_SEDE\_FK;

drop table EDIFICIO cascade constraints;

drop table FACULTAD cascade constraints;

drop index LOCACION\_EDIFICIO2\_FK;

drop index AUXILIAR\_LABORATORIO\_FK;

drop table LABORATORIO cascade constraints;

drop index LOCACION\_EDIFICIO\_FK;

drop index AREA\_OFICINA\_FK;

drop table OFICINA cascade constraints;

drop index ENTREGA2\_FK;

drop index PROCEDIMIENTO\_ACTIVO\_FK;

drop index PROCEDIMIENTO2\_FK;

drop table PROCEDIMIENTO cascade constraints;

drop index PROGRAMA\_FACULTAD\_FK;

drop table PROGRAMA cascade constraints;

drop index LOCACION\_EDIFICIO3\_FK;

drop table SALON cascade constraints;

drop table SEDE cascade constraints;

drop table VICERRECTORIA cascade constraints;

/\*==============================================================\*/

/\* Table: ACTIVO \*/

/\*==============================================================\*/

create table ACTIVO

(

ID\_ACTIVO INTEGER not null,

ID\_COMPRA INTEGER not null,

REFERENCIA\_ACTIVO VARCHAR2(40) not null,

CATEGORIA\_ACTIVO VARCHAR2(3) not null

constraint CKC\_CATEGORIA\_ACTIVO\_ACTIVO check (CATEGORIA\_ACTIVO in ('CAE','DTS','LAB','LOG','SGT')),

NOMBRE\_ACTIVO VARCHAR2(40) not null,

MARCA\_ACTIVO VARCHAR2(25) not null,

VALOR\_ACTIVO NUMBER(16,2) not null,

DEPRECIACION\_ACTIVO NUMBER(5,2) not null,

ID\_FUNCIONARIO INTEGER,

ID\_EDIFICIO INTEGER,

ID\_LOCACION INTEGER,

MATERIAL\_ACTIVO VARCHAR2(20),

COLOR\_ACTIVO VARCHAR2(20),

TAMANO\_ACTIVO VARCHAR2(20),

COMPONENTES\_ACTIVO VARCHAR2(120),

constraint PK\_ACTIVO primary key (ID\_ACTIVO)

);

/\*==============================================================\*/

/\* Index: ACTIVO\_COMPRA\_FK \*/

/\*==============================================================\*/

create index ACTIVO\_COMPRA\_FK on ACTIVO (

ID\_COMPRA ASC

);

/\*==============================================================\*/

/\* Index: USADO\_POR2\_FK \*/

/\*==============================================================\*/

create index USADO\_POR2\_FK on ACTIVO (

ID\_FUNCIONARIO ASC

);

/\*==============================================================\*/

/\* Index: SE\_ENCUENTRA\_EN2\_FK \*/

/\*==============================================================\*/

create index SE\_ENCUENTRA\_EN2\_FK on ACTIVO (

ID\_LOCACION ASC,

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: AREA \*/

/\*==============================================================\*/

create table AREA

(

ID\_AREA INTEGER not null,

NOMBRE\_AREA VARCHAR2(70) not null,

ID\_FACULTAD CHAR(3),

ID\_DEPARTAMENTO INTEGER,

constraint PK\_AREA primary key (ID\_AREA)

);

/\*==============================================================\*/

/\* Index: AREA\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index AREA\_FACULTAD\_FK on AREA (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Index: AREA\_DEPARTAMENTO\_FK \*/

/\*==============================================================\*/

create index AREA\_DEPARTAMENTO\_FK on AREA (

ID\_DEPARTAMENTO ASC

);

/\*==============================================================\*/

/\* Table: AUXILIAR \*/

/\*==============================================================\*/

create table AUXILIAR

(

ID\_FUNCIONARIO INTEGER not null,

TIPO\_AUXILIAR CHAR(3) not null

constraint CKC\_TIPO\_AUXILIAR\_AUXILIAR check (TIPO\_AUXILIAR in ('INV','MAN','LAB')),

NOMBRE\_FUNCIONARIO VARCHAR2(50),

INIC\_CONTRATO\_FUNCIONARIO DATE,

FIN\_CONTRATO\_FUNCIONARIO DATE,

CARGO\_EMPLEADO VARCHAR2(30),

ID\_JEFE INTEGER,

ID\_AREA INTEGER not null,

constraint PK\_AUXILIAR primary key (ID\_FUNCIONARIO)

);

/\*==============================================================\*/

/\* Index: TRABAJA\_EN\_FK \*/

/\*==============================================================\*/

create index TRABAJA\_EN\_FK on AUXILIAR (

ID\_AREA ASC

);

/\*==============================================================\*/

/\* Table: COMPRA \*/

/\*==============================================================\*/

create table COMPRA

(

ID\_COMPRA INTEGER not null,

FECHA\_COMPRA DATE not null,

VALOR\_COMPRA NUMBER(16,2) not null,

constraint PK\_COMPRA primary key (ID\_COMPRA)

);

/\*==============================================================\*/

/\* Table: DEPARTAMENTO \*/

/\*==============================================================\*/

create table DEPARTAMENTO

(

ID\_DEPARTAMENTO INTEGER not null,

NOMBRE\_DEPARTAMENTO VARCHAR2(70) not null,

DIRECTOR\_DEPARTAMENTO VARCHAR2(50) not null,

ID\_VICERRECTORIA CHAR(3),

ID\_FACULTAD CHAR(3),

constraint PK\_DEPARTAMENTO primary key (ID\_DEPARTAMENTO)

);

/\*==============================================================\*/

/\* Index: DEPATAMENTO\_VICERRECTORIA\_FK \*/

/\*==============================================================\*/

create index DEPATAMENTO\_VICERRECTORIA\_FK on DEPARTAMENTO (

ID\_VICERRECTORIA ASC

);

/\*==============================================================\*/

/\* Index: DEPARTAMENTO\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index DEPARTAMENTO\_FACULTAD\_FK on DEPARTAMENTO (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Table: DOCENTE \*/

/\*==============================================================\*/

create table DOCENTE

(

ID\_FUNCIONARIO INTEGER not null,

NOMBRE\_FUNCIONARIO VARCHAR2(50),

INIC\_CONTRATO\_FUNCIONARIO DATE,

FIN\_CONTRATO\_FUNCIONARIO DATE,

ID\_JEFE INTEGER,

TITULO\_DOCENTE VARCHAR2(100) not null,

ROL\_DOCENTE VARCHAR2(20) not null,

ID\_FACULTAD CHAR(3) not null,

ID\_PROGRAMA CHAR(10) not null,

constraint AK\_IDENTIFIER\_1\_DOCENTE unique (ID\_FUNCIONARIO)

);

/\*==============================================================\*/

/\* Index: TRABAJA\_FK \*/

/\*==============================================================\*/

create index TRABAJA\_FK on DOCENTE (

ID\_FACULTAD ASC,

ID\_PROGRAMA ASC

);

/\*==============================================================\*/

/\* Table: EDIFICIO \*/

/\*==============================================================\*/

create table EDIFICIO

(

ID\_SEDE INTEGER not null,

ID\_EDIFICIO INTEGER not null,

NOMBRE\_EDIFICIO VARCHAR2(50) not null,

constraint PK\_EDIFICIO primary key (ID\_EDIFICIO)

);

/\*==============================================================\*/

/\* Index: EDIFICIO\_SEDE\_FK \*/

/\*==============================================================\*/

create index EDIFICIO\_SEDE\_FK on EDIFICIO (

ID\_SEDE ASC

);

/\*==============================================================\*/

/\* Table: FACULTAD \*/

/\*==============================================================\*/

create table FACULTAD

(

ID\_FACULTAD CHAR(3) not null

constraint CKC\_ID\_FACULTAD\_FACULTAD check (ID\_FACULTAD in ('AGR','JUR','ECO','ING','CIE')),

NOMBRE\_FACULTAD VARCHAR2(60) not null,

DECANO\_FACULTAD VARCHAR2(50) not null,

constraint PK\_FACULTAD primary key (ID\_FACULTAD)

);

/\*==============================================================\*/

/\* Table: LABORATORIO \*/

/\*==============================================================\*/

create table LABORATORIO

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100),

ID\_AUXILIAR INTEGER not null,

ACREDITACION\_LABORATORIO VARCHAR2(20),

constraint PK\_LABORATORIO primary key (ID\_LOCACION, ID\_EDIFICIO),

constraint AK\_IDENTIFIER\_1\_LABORATO unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: AUXILIAR\_LABORATORIO\_FK \*/

/\*==============================================================\*/

create index AUXILIAR\_LABORATORIO\_FK on LABORATORIO (

ID\_AUXILIAR ASC

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO2\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO2\_FK on LABORATORIO (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: OFICINA \*/

/\*==============================================================\*/

create table OFICINA

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

PUESTOS\_OFICINA INTEGER not null,

ID\_AREA INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100),

constraint PK\_OFICINA primary key (ID\_EDIFICIO, ID\_LOCACION),

constraint AK\_IDENTIFIER\_1\_OFICINA unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: AREA\_OFICINA\_FK \*/

/\*==============================================================\*/

create index AREA\_OFICINA\_FK on OFICINA (

ID\_AREA ASC

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO\_FK on OFICINA (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: PROCEDIMIENTO \*/

/\*==============================================================\*/

create table PROCEDIMIENTO

(

ID\_PROCE INTEGER not null,

ID\_ACTIVO INTEGER not null,

ID\_AUXILIAR INTEGER not null,

TIPO\_PROCEDIMIENTO CHAR(1) not null

constraint CKC\_TIPO\_PROCEDIMIENT\_PROCEDIM check (TIPO\_PROCEDIMIENTO in ('E','P','R','M')),

FECHA\_PROCEDIMIENTO DATE not null,

ESTADO\_REVISION CHAR(1)

constraint CKC\_ESTADO\_REVISION\_PROCEDIM check (ESTADO\_REVISION is null or (ESTADO\_REVISION in ('R','M'))),

OBSERVACIONES VARCHAR2(150),

FUNCIONARIO\_ENTREGA INTEGER,

constraint PK\_PROCEDIMIENTO primary key (ID\_PROCE)

);

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO2\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO2\_FK on PROCEDIMIENTO (

ID\_AUXILIAR ASC

);

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO\_ACTIVO\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO\_ACTIVO\_FK on PROCEDIMIENTO (

ID\_ACTIVO ASC

);

/\*==============================================================\*/

/\* Index: ENTREGA2\_FK \*/

/\*==============================================================\*/

create index ENTREGA2\_FK on PROCEDIMIENTO (

FUNCIONARIO\_ENTREGA ASC

);

/\*==============================================================\*/

/\* Table: PROGRAMA \*/

/\*==============================================================\*/

create table PROGRAMA

(

ID\_FACULTAD CHAR(3) not null,

ID\_PROGRAMA CHAR(10) not null,

NOMBRE\_PROGRAMA VARCHAR2(80) not null,

constraint PK\_PROGRAMA primary key (ID\_FACULTAD, ID\_PROGRAMA)

);

/\*==============================================================\*/

/\* Index: PROGRAMA\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index PROGRAMA\_FACULTAD\_FK on PROGRAMA (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Table: SALON \*/

/\*==============================================================\*/

create table SALON

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100),

CAPACIDAD\_SALON INTEGER not null,

TIPO\_SALON CHAR(3)

constraint CKC\_TIPO\_SALON\_SALON check (TIPO\_SALON is null or (TIPO\_SALON in ('AUD','TAL','COM'))),

constraint PK\_SALON primary key (ID\_LOCACION, ID\_EDIFICIO),

constraint AK\_IDENTIFIER\_1\_SALON unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO3\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO3\_FK on SALON (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: SEDE \*/

/\*==============================================================\*/

create table SEDE

(

ID\_SEDE INTEGER not null,

NOMBRE\_SEDE VARCHAR2(70) not null,

constraint PK\_SEDE primary key (ID\_SEDE)

);

/\*==============================================================\*/

/\* Table: VICERRECTORIA \*/

/\*==============================================================\*/

create table VICERRECTORIA

(

ID\_VICERRECTORIA CHAR(3) not null

constraint CKC\_ID\_VICERRECTORIA\_VICERREC check (ID\_VICERRECTORIA in ('FOR','INV','EXT','ADM')),

NOMBRE\_VICERRECTORIA VARCHAR2(50) not null,

VICERRECTOR\_VICERRECTORIA VARCHAR2(50) not null,

constraint PK\_VICERRECTORIA primary key (ID\_VICERRECTORIA)

);

alter table ACTIVO

add constraint FK\_COMPRA\_IN\_ACTIVO foreign key (ID\_COMPRA)

references COMPRA (ID\_COMPRA);

alter table ACTIVO

add constraint FK\_OFICINA\_IN\_ACTIVO foreign key (ID\_EDIFICIO, ID\_LOCACION)

references OFICINA (ID\_EDIFICIO, ID\_LOCACION);

alter table ACTIVO

add constraint FK\_LAB\_IN\_ACTIVO foreign key (ID\_LOCACION, ID\_EDIFICIO)

references LABORATORIO (ID\_LOCACION, ID\_EDIFICIO);

alter table ACTIVO

add constraint FK\_SALON\_IN\_ACTIVO foreign key (ID\_LOCACION, ID\_EDIFICIO)

references SALON (ID\_LOCACION, ID\_EDIFICIO);

alter table ACTIVO

add constraint FK\_ACTIVO\_USADO\_POR\_AUX foreign key (ID\_FUNCIONARIO)

references AUXILIAR (ID\_FUNCIONARIO);

alter table ACTIVO

add constraint FK\_DOCENTE\_IN\_ACTIVO foreign key (ID\_FUNCIONARIO)

references DOCENTE (ID\_FUNCIONARIO);

alter table AREA

add constraint FK\_DEPARTAMENTO\_IN\_AREA foreign key (ID\_DEPARTAMENTO)

references DEPARTAMENTO (ID\_DEPARTAMENTO);

alter table AREA

add constraint FK\_FACULTAD\_IN\_AREA foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table AUXILIAR

add constraint FK\_AREA\_IN\_AUXILIAR foreign key (ID\_AREA)

references AREA (ID\_AREA);

alter table AUXILIAR

add constraint FK\_JEFE\_IN\_AUXILIAR foreign key (ID\_JEFE)

references AUXILIAR (ID\_FUNCIONARIO);

alter table DEPARTAMENTO

add constraint FK\_FACULTAD\_IN\_DEPT foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table DEPARTAMENTO

add constraint FK\_VICER\_IN\_DEPT foreign key (ID\_VICERRECTORIA)

references VICERRECTORIA (ID\_VICERRECTORIA);

alter table DOCENTE

add constraint FK\_PROGRAMA\_IN\_DOCENTE foreign key (ID\_FACULTAD, ID\_PROGRAMA)

references PROGRAMA (ID\_FACULTAD, ID\_PROGRAMA);

alter table DOCENTE

add constraint FK\_JEFE\_IN\_DOCENTE foreign key (ID\_JEFE)

references DOCENTE (ID\_FUNCIONARIO);

alter table EDIFICIO

add constraint FK\_SEDE\_IN\_EDIFICIO foreign key (ID\_SEDE)

references SEDE (ID\_SEDE);

alter table LABORATORIO

add constraint FK\_AUXILIAR\_IN\_LAB foreign key (ID\_AUXILIAR)

references AUXILIAR (ID\_FUNCIONARIO);

alter table LABORATORIO

add constraint EK\_EDIFICIO\_IN\_LAB foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

alter table OFICINA

add constraint FK\_AREA\_IN\_OFICINA foreign key (ID\_AREA)

references AREA (ID\_AREA);

alter table OFICINA

add constraint FK\_EDIFICIO\_IN\_OFICINA foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

alter table PROCEDIMIENTO

add constraint FK\_AUX\_ENTREGA\_IN\_PROC foreign key (FUNCIONARIO\_ENTREGA)

references AUXILIAR (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_DOCENTE\_ENTREGA\_IN\_PROC foreign key (FUNCIONARIO\_ENTREGA)

references DOCENTE (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_AUXILIAR\_IN\_PROC foreign key (ID\_AUXILIAR)

references AUXILIAR (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_ACTIVO\_IN\_PROC foreign key (ID\_ACTIVO)

references ACTIVO (ID\_ACTIVO);

alter table PROGRAMA

add constraint FK\_FACULTAD\_IN\_PROGRAMA foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table SALON

add constraint FK\_EDIFICIO\_IN\_SALON foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

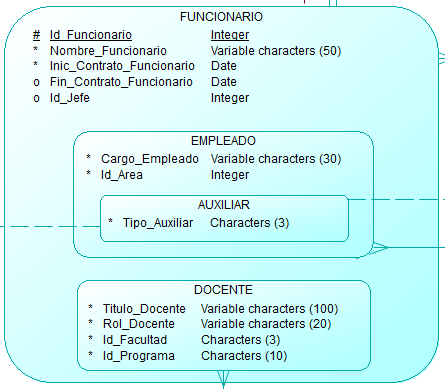
***Resolución de supertipos y subtipos:***

En el presente modelo se tienen dos supertipos: FUNCIONARIO y LOCACIÓN. Las entidades EMPLEADO, AUXILIAR, DOCENTE, SALÓN, OFICINA y LABORATORIO son especializaciones de estos supertipos.

El método a utilizar será el de generar una tabla única para cada supertipo y sus subtipos.

El primer paso consiste en diseñar una estructura general para cada supertipo, que contenga todos los atributos de sí mismo y de sus subtipos.

*Supertipo FUNCIONARIO*



*Figura 12 – Supertipo FUNCIONARIO según fue diseñado en PowerDesigner*

Se diseña una tabla FUNCIONARIO que recoja todos los atributos de la imagen anterior junto con un atributo *tipo\_funcionario*. Los atributos obligatorios de los subtipos pasarán a ser opcionales.

FUNCIONARIO

# id\_funcionario Integer

+ nombre\_funcionario Varchar2(50)

+ inic\_contrato\_funcionario Date

o fin\_contrato\_funcionario Date

o id\_jefe Integer

o tipo\_funcionario Char(1)

o cargo\_empleado Varchar2(30)

o id\_area Integer

o titulo\_docente Varchar2(100)

o rol\_docente Varchar2(20)

o id\_facultad Char(3)

o id\_programa Char(10)

o tipo\_auxiliar Char(3)

*Figura 13 – Supertipo FUNCIONARIO para la resolución de supertipos y subtipos*

La integridad por valor para el atributo *tipo\_funcionario* será ‘E’ para empleado y ‘D’ para docente.

Los atributos en color amarillo representan llaves foráneas.

*Supertipo LOCACION*



*Figura 14 – Supertipo LOCACION según fue diseñado en PowerDesigner*

Se diseña una tabla LOCACION que recoja todos los atributos anteriores y uno adicional llamado *tipo\_locacion*, Los atributos obligatorios de los subtipos pasarán a ser opcionales.

LOCACION

# id\_locacion Integer

# id\_edificio Integer

+ nombre\_locacion Varchar2(100)

o tipo\_locacion Char(1)

o capacidad\_salon Integer

o tipo\_salon Char(3)

o puestos\_oficina Integer

o id\_area Integer

o id\_auxiliar Integer

o acreditacion\_laboratorio Varchar2(20)

*Figura 15 – Supertipo LOCACION para la resolución de supertipos y subtipos*

La integridad por valor para el atributo *tipo\_locacion* será ‘S’ para salón, ‘O’ para oficina y ‘L’ para laboratorio.

Los atributos en color amarillo representan llaves foráneas.

El segundo paso consiste en proceder a realizar las modificaciones en el código de creación generado como producto del diseño.

A continuación se presenta el SQL de creación completo. Se marcarán en color rojo las líneas a eliminar y en color verde las líneas a agregar o modificar.

/\*==============================================================\*/

/\* DBMS name: ORACLE Version 11g \*/

/\* Created on: 25/06/2020 12:34:49 p. m. \*/

/\*==============================================================\*/

alter table ACTIVO

drop constraint FK\_COMPRA\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_LOCACION\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_LAB\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_SALON\_IN\_ACTIVO;

alter table ACTIVO

drop constraint FK\_ACTIVO\_USADO\_POR\_FUN;

alter table ACTIVO

drop constraint FK\_DOCENTE\_IN\_ACTIVO;

alter table AREA

drop constraint FK\_DEPARTAMENTO\_IN\_AREA;

alter table AREA

drop constraint FK\_FACULTAD\_IN\_AREA;

alter table FUNCIONARIO

drop constraint FK\_AREA\_IN\_FUNCIONARIO;

alter table FUNCIONARIO

drop constraint FK\_JEFE\_IN\_FUNCIONARIO;

alter table DEPARTAMENTO

drop constraint FK\_FACULTAD\_IN\_DEPT;

alter table DEPARTAMENTO

drop constraint FK\_VICER\_IN\_DEPT;

alter table FUNCIONARIO

drop constraint FK\_PROGRAMA\_IN\_FUNCIONARIO;

alter table DOCENTE

drop constraint FK\_JEFE\_IN\_DOCENTE;

alter table EDIFICIO

drop constraint FK\_SEDE\_IN\_EDIFICIO;

alter table LOCACION

drop constraint FK\_AUXILIAR\_IN\_LAB;

alter table LABORATORIO

drop constraint EK\_EDIFICIO\_IN\_LAB;

alter table LOCACION

drop constraint FK\_AREA\_IN\_OFICINA;

alter table LOCACION

drop constraint FK\_EDIFICIO\_IN\_LOCACION;

alter table PROCEDIMIENTO

drop constraint FK\_FUN\_ENTREGA\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_DOCENTE\_ENTREGA\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_FUNCIONARIO\_IN\_PROC;

alter table PROCEDIMIENTO

drop constraint FK\_ACTIVO\_IN\_PROC;

alter table PROGRAMA

drop constraint FK\_FACULTAD\_IN\_PROGRAMA;

alter table SALON

drop constraint FK\_EDIFICIO\_IN\_SALON;

drop index SE\_ENCUENTRA\_EN2\_FK;

drop index USADO\_POR2\_FK;

drop index ACTIVO\_COMPRA\_FK;

drop table ACTIVO cascade constraints;

drop index AREA\_DEPARTAMENTO\_FK;

drop index AREA\_FACULTAD\_FK;

drop table AREA cascade constraints;

drop index TRABAJA\_EN\_FK;

drop table FUNCIONARIO cascade constraints;

drop table COMPRA cascade constraints;

drop index DEPARTAMENTO\_FACULTAD\_FK;

drop index DEPATAMENTO\_VICERRECTORIA\_FK;

drop table DEPARTAMENTO cascade constraints;

drop index TRABAJA\_FK;

drop table DOCENTE cascade constraints;

drop index EDIFICIO\_SEDE\_FK;

drop table EDIFICIO cascade constraints;

drop table FACULTAD cascade constraints;

drop index LOCACION\_EDIFICIO2\_FK;

drop index AUXILIAR\_LABORATORIO\_FK;

drop table LABORATORIO cascade constraints;

drop index LOCACION\_EDIFICIO\_FK;

drop index AREA\_OFICINA\_FK;

drop table LOCACION cascade constraints;

drop index ENTREGA2\_FK;

drop index PROCEDIMIENTO\_ACTIVO\_FK;

drop index PROCEDIMIENTO2\_FK;

drop table PROCEDIMIENTO cascade constraints;

drop index PROGRAMA\_FACULTAD\_FK;

drop table PROGRAMA cascade constraints;

drop index LOCACION\_EDIFICIO3\_FK;

drop table SALON cascade constraints;

drop table SEDE cascade constraints;

drop table VICERRECTORIA cascade constraints;

/\*==============================================================\*/

/\* Table: ACTIVO \*/

/\*==============================================================\*/

create table ACTIVO

(

ID\_ACTIVO INTEGER not null,

ID\_COMPRA INTEGER not null,

REFERENCIA\_ACTIVO VARCHAR2(40) not null,

CATEGORIA\_ACTIVO VARCHAR2(3) not null

constraint CKC\_CATEGORIA\_ACTIVO\_ACTIVO check (CATEGORIA\_ACTIVO in ('CAE','DTS','LAB','LOG','SGT')),

NOMBRE\_ACTIVO VARCHAR2(40) not null,

MARCA\_ACTIVO VARCHAR2(25) not null,

VALOR\_ACTIVO NUMBER(16,2) not null,

DEPRECIACION\_ACTIVO NUMBER(5,2) not null,

ID\_FUNCIONARIO INTEGER,

ID\_EDIFICIO INTEGER,

ID\_LOCACION INTEGER,

MATERIAL\_ACTIVO VARCHAR2(20),

COLOR\_ACTIVO VARCHAR2(20),

TAMANO\_ACTIVO VARCHAR2(20),

COMPONENTES\_ACTIVO VARCHAR2(120),

constraint PK\_ACTIVO primary key (ID\_ACTIVO)

);

/\*==============================================================\*/

/\* Index: ACTIVO\_COMPRA\_FK \*/

/\*==============================================================\*/

create index ACTIVO\_COMPRA\_FK on ACTIVO (

ID\_COMPRA ASC

);

/\*==============================================================\*/

/\* Index: USADO\_POR2\_FK \*/

/\*==============================================================\*/

create index USADO\_POR2\_FK on ACTIVO (

ID\_FUNCIONARIO ASC

);

/\*==============================================================\*/

/\* Index: SE\_ENCUENTRA\_EN2\_FK \*/

/\*==============================================================\*/

create index SE\_ENCUENTRA\_EN2\_FK on ACTIVO (

ID\_LOCACION ASC,

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: AREA \*/

/\*==============================================================\*/

create table AREA

(

ID\_AREA INTEGER not null,

NOMBRE\_AREA VARCHAR2(70) not null,

ID\_FACULTAD CHAR(3),

ID\_DEPARTAMENTO INTEGER,

constraint PK\_AREA primary key (ID\_AREA)

);

/\*==============================================================\*/

/\* Index: AREA\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index AREA\_FACULTAD\_FK on AREA (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Index: AREA\_DEPARTAMENTO\_FK \*/

/\*==============================================================\*/

create index AREA\_DEPARTAMENTO\_FK on AREA (

ID\_DEPARTAMENTO ASC

);

/\*==============================================================\*/

/\* Table: FUNCIONARIO \*/

/\*==============================================================\*/

create table FUNCIONARIO

(

ID\_FUNCIONARIO INTEGER not null,

NOMBRE\_FUNCIONARIO VARCHAR2(50) not null,

INIC\_CONTRATO\_FUNCIONARIO DATE not null,

FIN\_CONTRATO\_FUNCIONARIO DATE ,

ID\_JEFE INTEGER ,

TIPO\_FUNCIONARIO CHAR(1) not null

constraint CKC\_TIPO\_FUN check (TIPO\_FUNCIONARIO in ('E', 'D')),

CARGO\_EMPLEADO VARCHAR2(30) ,

ID\_AREA INTEGER ,

TITULO\_DOCENTE VARCHAR2(100) ,

ROL\_DOCENTE VARCHAR2(20) ,

ID\_FACULTAD CHAR(3) ,

ID\_PROGRAMA CHAR(10) ,

TIPO\_AUXILIAR CHAR(3)

constraint CKC\_TIPO\_AUXILIAR\_FUN check (TIPO\_AUXILIAR in ('INV','MAN','LAB')),

constraint PK\_FUNCIONARIO primary key (ID\_FUNCIONARIO)

);

/\*==============================================================\*/

/\* Index: TRABAJA\_EN\_FK \*/

/\*==============================================================\*/

create index TRABAJA\_EN\_FK on FUNCIONARIO (

ID\_AREA ASC

);

/\*==============================================================\*/

/\* Table: COMPRA \*/

/\*==============================================================\*/

create table COMPRA

(

ID\_COMPRA INTEGER not null,

FECHA\_COMPRA DATE not null,

VALOR\_COMPRA NUMBER(16,2) not null,

constraint PK\_COMPRA primary key (ID\_COMPRA)

);

/\*==============================================================\*/

/\* Table: DEPARTAMENTO \*/

/\*==============================================================\*/

create table DEPARTAMENTO

(

ID\_DEPARTAMENTO INTEGER not null,

NOMBRE\_DEPARTAMENTO VARCHAR2(70) not null,

DIRECTOR\_DEPARTAMENTO VARCHAR2(50) not null,

ID\_VICERRECTORIA CHAR(3),

ID\_FACULTAD CHAR(3),

constraint PK\_DEPARTAMENTO primary key (ID\_DEPARTAMENTO)

);

/\*==============================================================\*/

/\* Index: DEPATAMENTO\_VICERRECTORIA\_FK \*/

/\*==============================================================\*/

create index DEPATAMENTO\_VICERRECTORIA\_FK on DEPARTAMENTO (

ID\_VICERRECTORIA ASC

);

/\*==============================================================\*/

/\* Index: DEPARTAMENTO\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index DEPARTAMENTO\_FACULTAD\_FK on DEPARTAMENTO (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Table: DOCENTE \*/

/\*==============================================================\*/

create table DOCENTE

(

ID\_FUNCIONARIO INTEGER not null,

NOMBRE\_FUNCIONARIO VARCHAR2(50),

INIC\_CONTRATO\_FUNCIONARIO DATE,

FIN\_CONTRATO\_FUNCIONARIO DATE,

ID\_JEFE INTEGER,

TITULO\_DOCENTE VARCHAR2(100) not null,

ROL\_DOCENTE VARCHAR2(20) not null,

ID\_FACULTAD CHAR(3) not null,

ID\_PROGRAMA CHAR(10) not null,

constraint AK\_IDENTIFIER\_1\_DOCENTE unique (ID\_FUNCIONARIO)

);

/\*==============================================================\*/

/\* Index: TRABAJA\_FK \*/

/\*==============================================================\*/

create index TRABAJA\_FK on DOCENTE (

ID\_FACULTAD ASC,

ID\_PROGRAMA ASC

);

/\*==============================================================\*/

/\* Table: EDIFICIO \*/

/\*==============================================================\*/

create table EDIFICIO

(

ID\_SEDE INTEGER not null,

ID\_EDIFICIO INTEGER not null,

NOMBRE\_EDIFICIO VARCHAR2(50) not null,

constraint PK\_EDIFICIO primary key (ID\_EDIFICIO)

);

/\*==============================================================\*/

/\* Index: EDIFICIO\_SEDE\_FK \*/

/\*==============================================================\*/

create index EDIFICIO\_SEDE\_FK on EDIFICIO (

ID\_SEDE ASC

);

/\*==============================================================\*/

/\* Table: FACULTAD \*/

/\*==============================================================\*/

create table FACULTAD

(

ID\_FACULTAD CHAR(3) not null

constraint CKC\_ID\_FACULTAD\_FACULTAD check (ID\_FACULTAD in ('AGR','JUR','ECO','ING','CIE')),

NOMBRE\_FACULTAD VARCHAR2(60) not null,

DECANO\_FACULTAD VARCHAR2(50) not null,

constraint PK\_FACULTAD primary key (ID\_FACULTAD)

);

/\*==============================================================\*/

/\* Table: LABORATORIO \*/

/\*==============================================================\*/

create table LABORATORIO

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100),

ID\_AUXILIAR INTEGER not null,

ACREDITACION\_LABORATORIO VARCHAR2(20),

constraint PK\_LABORATORIO primary key (ID\_LOCACION, ID\_EDIFICIO),

constraint AK\_IDENTIFIER\_1\_LABORATO unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO2\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO2\_FK on LABORATORIO (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: LOCACION \*/

/\*==============================================================\*/

create table LOCACION

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100) not null,

TIPO\_LOCACION CHAR(1) not null

constraint CKC\_TIPO\_LOCACION check (TIPO\_LOCACION in ('S','O','L')),

CAPACIDAD\_SALON INTEGER ,

TIPO\_SALON CHAR(3)

constraint CKC\_TIPO\_SALON check (TIPO\_SALON is null or (TIPO\_SALON in ('AUD','TAL','COM'))),

PUESTOS\_OFICINA INTEGER ,

ID\_AREA INTEGER ,

ID\_AUXILIAR INTEGER ,

ACREDITACION\_LAB VARCHAR2(20) ,

constraint PK\_LOCACION primary key (ID\_EDIFICIO, ID\_LOCACION)\_

constraint AK\_IDENTIFIER\_1\_OFICINA unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: AUXILIAR\_LABORATORIO\_FK \*/

/\*==============================================================\*/

create index AUXILIAR\_LABORATORIO\_FK on LOCACION (

ID\_AUXILIAR ASC

);

/\*==============================================================\*/

/\* Index: AREA\_OFICINA\_FK \*/

/\*==============================================================\*/

create index AREA\_OFICINA\_FK on LOCACION (

ID\_AREA ASC

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO\_FK on LOCACION (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: PROCEDIMIENTO \*/

/\*==============================================================\*/

create table PROCEDIMIENTO

(

ID\_PROCE INTEGER not null,

ID\_ACTIVO INTEGER not null,

ID\_AUXILIAR INTEGER not null,

TIPO\_PROCEDIMIENTO CHAR(1) not null

constraint CKC\_TIPO\_PROCEDIMIENT\_PROCEDIM check (TIPO\_PROCEDIMIENTO in ('E','P','R','M')),

FECHA\_PROCEDIMIENTO DATE not null,

ESTADO\_REVISION CHAR(1)

constraint CKC\_ESTADO\_REVISION\_PROCEDIM check (ESTADO\_REVISION is null or (ESTADO\_REVISION in ('R','M'))),

OBSERVACIONES VARCHAR2(150),

FUNCIONARIO\_ENTREGA INTEGER,

constraint PK\_PROCEDIMIENTO primary key (ID\_PROCE)

);

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO2\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO2\_FK on PROCEDIMIENTO (

ID\_AUXILIAR ASC

);

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO\_ACTIVO\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO\_ACTIVO\_FK on PROCEDIMIENTO (

ID\_ACTIVO ASC

);

/\*==============================================================\*/

/\* Index: ENTREGA2\_FK \*/

/\*==============================================================\*/

create index ENTREGA2\_FK on PROCEDIMIENTO (

FUNCIONARIO\_ENTREGA ASC

);

/\*==============================================================\*/

/\* Table: PROGRAMA \*/

/\*==============================================================\*/

create table PROGRAMA

(

ID\_FACULTAD CHAR(3) not null,

ID\_PROGRAMA CHAR(10) not null,

NOMBRE\_PROGRAMA VARCHAR2(80) not null,

constraint PK\_PROGRAMA primary key (ID\_FACULTAD, ID\_PROGRAMA)

);

/\*==============================================================\*/

/\* Index: PROGRAMA\_FACULTAD\_FK \*/

/\*==============================================================\*/

create index PROGRAMA\_FACULTAD\_FK on PROGRAMA (

ID\_FACULTAD ASC

);

/\*==============================================================\*/

/\* Table: SALON \*/

/\*==============================================================\*/

create table SALON

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100),

CAPACIDAD\_SALON INTEGER not null,

TIPO\_SALON CHAR(3)

constraint CKC\_TIPO\_SALON\_SALON check (TIPO\_SALON is null or (TIPO\_SALON in ('AUD','TAL','COM'))),

constraint PK\_SALON primary key (ID\_LOCACION, ID\_EDIFICIO),

constraint AK\_IDENTIFIER\_1\_SALON unique (ID\_LOCACION)

);

/\*==============================================================\*/

/\* Index: LOCACION\_EDIFICIO3\_FK \*/

/\*==============================================================\*/

create index LOCACION\_EDIFICIO3\_FK on SALON (

ID\_EDIFICIO ASC

);

/\*==============================================================\*/

/\* Table: SEDE \*/

/\*==============================================================\*/

create table SEDE

(

ID\_SEDE INTEGER not null,

NOMBRE\_SEDE VARCHAR2(70) not null,

constraint PK\_SEDE primary key (ID\_SEDE)

);

/\*==============================================================\*/

/\* Table: VICERRECTORIA \*/

/\*==============================================================\*/

create table VICERRECTORIA

(

ID\_VICERRECTORIA CHAR(3) not null

constraint CKC\_ID\_VICERRECTORIA\_VICERREC check (ID\_VICERRECTORIA in ('FOR','INV','EXT','ADM')),

NOMBRE\_VICERRECTORIA VARCHAR2(50) not null,

VICERRECTOR\_VICERRECTORIA VARCHAR2(50) not null,

constraint PK\_VICERRECTORIA primary key (ID\_VICERRECTORIA)

);

alter table ACTIVO

add constraint FK\_COMPRA\_IN\_ACTIVO foreign key (ID\_COMPRA)

references COMPRA (ID\_COMPRA);

alter table ACTIVO

add constraint FK\_LOCACION\_IN\_ACTIVO foreign key (ID\_EDIFICIO, ID\_LOCACION)

references LOCACION (ID\_EDIFICIO, ID\_LOCACION);

alter table ACTIVO

add constraint FK\_LAB\_IN\_ACTIVO foreign key (ID\_LOCACION, ID\_EDIFICIO)

references LABORATORIO (ID\_LOCACION, ID\_EDIFICIO);

alter table ACTIVO

add constraint FK\_SALON\_IN\_ACTIVO foreign key (ID\_LOCACION, ID\_EDIFICIO)

references SALON (ID\_LOCACION, ID\_EDIFICIO);

alter table ACTIVO

add constraint FK\_ACTIVO\_USADO\_POR\_FUN foreign key (ID\_FUNCIONARIO)

references FUNCIONARIO (ID\_FUNCIONARIO);

alter table ACTIVO

add constraint FK\_DOCENTE\_IN\_ACTIVO foreign key (ID\_FUNCIONARIO)

references DOCENTE (ID\_FUNCIONARIO);

alter table AREA

add constraint FK\_DEPARTAMENTO\_IN\_AREA foreign key (ID\_DEPARTAMENTO)

references DEPARTAMENTO (ID\_DEPARTAMENTO);

alter table AREA

add constraint FK\_FACULTAD\_IN\_AREA foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table FUNCIONARIO

add constraint FK\_AREA\_IN\_FUNCIONARIO foreign key (ID\_AREA)

references AREA (ID\_AREA);

alter table FUNCIONARIO

add constraint FK\_JEFE\_IN\_FUNCIONARIO foreign key (ID\_JEFE)

references FUNCIONARIO (ID\_FUNCIONARIO);

alter table DEPARTAMENTO

add constraint FK\_FACULTAD\_IN\_DEPT foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table DEPARTAMENTO

add constraint FK\_VICER\_IN\_DEPT foreign key (ID\_VICERRECTORIA)

references VICERRECTORIA (ID\_VICERRECTORIA);

alter table FUNCIONARIO

add constraint FK\_PROGRAMA\_IN\_FUNCIONARIO foreign key (ID\_FACULTAD, ID\_PROGRAMA)

references PROGRAMA (ID\_FACULTAD, ID\_PROGRAMA);

alter table DOCENTE

add constraint FK\_JEFE\_IN\_DOCENTE foreign key (ID\_JEFE)

references DOCENTE (ID\_FUNCIONARIO);

alter table EDIFICIO

add constraint FK\_SEDE\_IN\_EDIFICIO foreign key (ID\_SEDE)

references SEDE (ID\_SEDE);

alter table LOCACION

add constraint FK\_AUXILIAR\_IN\_LAB foreign key (ID\_AUXILIAR)

references FUNCIONARIO (ID\_FUNCIONARIO);

alter table LABORATORIO

add constraint EK\_EDIFICIO\_IN\_LAB foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

alter table LOCACION

add constraint FK\_AREA\_IN\_OFICINA foreign key (ID\_AREA)

references AREA (ID\_AREA);

alter table LOCACION

add constraint FK\_EDIFICIO\_IN\_LOCACION foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

alter table PROCEDIMIENTO

add constraint FK\_FUN\_ENTREGA\_IN\_PROC foreign key (FUNCIONARIO\_ENTREGA)

references FUNCIONARIO (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_DOCENTE\_ENTREGA\_IN\_PROC foreign key (FUNCIONARIO\_ENTREGA)

references DOCENTE (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_FUNCIONARIO\_IN\_PROC foreign key (ID\_AUXILIAR)

references FUNCIONARIO (ID\_FUNCIONARIO);

alter table PROCEDIMIENTO

add constraint FK\_ACTIVO\_IN\_PROC foreign key (ID\_ACTIVO)

references ACTIVO (ID\_ACTIVO);

alter table PROGRAMA

add constraint FK\_FACULTAD\_IN\_PROGRAMA foreign key (ID\_FACULTAD)

references FACULTAD (ID\_FACULTAD);

alter table SALON

add constraint FK\_EDIFICIO\_IN\_SALON foreign key (ID\_EDIFICIO)

references EDIFICIO (ID\_EDIFICIO);

/\*==============================================================\*/

/\* CHECK TABLE FUNCIONARIO \*/

/\*==============================================================\*/

alter table FUNCIONARIO

add constraint FUNCIONARIO\_CHK\_TABLE

CHECK (

(TIPO\_FUNCIONARIO = 'E'

AND CARGO\_EMPLEADO IS NOT NULL

AND ID\_AREA IS NOT NULL

AND TITULO\_DOCENTE IS NULL

AND ROL\_DOCENTE IS NULL

AND ID\_PROGRAMA IS NULL

AND ID\_FACULTAD IS NULL) OR

(TIPO\_FUNCIONARIO = 'D'

AND CARGO\_EMPLEADO IS NULL

AND ID\_AREA IS NULL

AND TIPO\_AUXILIAR IS NULL

AND TITULO\_DOCENTE IS NOT NULL

AND ROL\_DOCENTE IS NOT NULL

AND ID\_PROGRAMA IS NOT NULL

AND ID\_FACULTAD IS NOT NULL)

);

/\*==============================================================\*/

/\* CHECK TABLE LOCACION \*/

/\*==============================================================\*/

alter table LOCACION

add constraint LOCACION\_CHK\_TABLE

CHECK (

(TIPO\_LOCACION = 'S'

AND CAPACIDAD\_SALON IS NOT NULL

AND PUESTOS\_OFICINA IS NULL

AND ID\_AREA IS NULL

AND ID\_AUXILIAR IS NULL

AND ACREDITACION\_LAB IS NULL) OR

(TIPO\_LOCACION = 'O'

AND CAPACIDAD\_SALON IS NULL

AND TIPO\_SALON IS NULL

AND PUESTOS\_OFICINA IS NOT NULL

AND ID\_AREA IS NOT NULL

AND ID\_AUXILIAR IS NULL

AND ACREDITACION\_LAB IS NULL) OR

(TIPO\_LOCACION = 'L'

AND CAPACIDAD\_SALON IS NULL

AND TIPO\_SALON IS NULL

AND PUESTOS\_OFICINA IS NULL

AND ID\_AREA IS NULL

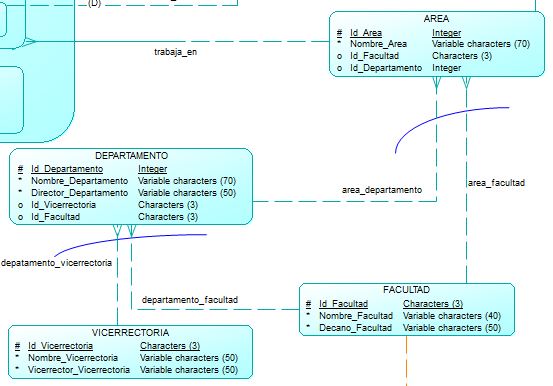
AND ID\_AUXILIAR IS NOT NULL

AND ACREDITACION\_LAB IS NOT NULL)

);

***Resolución de arcos con restricciones de chequeo***

Para el presente modelo, no es posible solucionar los arcos presentados con un supertipo, pues no es posible hacer una generalización entre las entidades FACULTAD y DEPARTAMENTO para eliminar el arco que forman con AREA y para las entidades FACULTAD y VICERRECTORÍA para eliminar el arco que forman con DEPARTAMENTO.



*Figura 16 – ARCOS presentes en el diseño de la base de datos según se mostró en los diagramas generados en PowerDesigner*

Se usará el siguiente código SQL para garantizar la integridad referencial entre estas relaciones.

alter table DEPARTAMENTO

add constraint DEPARTAMENTO\_ARC\_CHK

check (

(id\_vicerrectoria IS NOT NULL

and id\_facultad IS NULL) OR

(id\_vicerrectoria IS NULL

and id\_facultad IS NOT NULL)

);

alter table AREA

add constraint AREA\_ARC\_CHK

check (

(id\_facultad IS NOT NULL

and id\_departamento IS NULL) OR

(id\_facultad IS NULL

and id\_departamento IS NOT NULL)

);

***Modificaciones adicionales al código de creación SQL de la base de datos***

Debido a que la tabla PROCEDIMIENTO cuenta con un atributo llamado *tipo\_procedimiento*, y en función del valor que tome ese atributo, algunos atributos son necesarios y otros no, se agregará una restricción de chequeo por medio de una alteración de tabla para esa estructura. El código a agregar es el siguiente:

alter table PROCEDIMIENTO

add constraint PROCEDIMIENTO\_CHK\_TABLE

CHECK (

(TIPO\_PROCEDIMIENTO = 'R'

AND FUNCIONARIO\_ENTREGA IS NULL) OR

(TIPO\_PROCEDIMIENTO = 'P'

AND FUNCIONARIO\_ENTREGA IS NULL) OR

(TIPO\_PROCEDIMIENTO = 'M'

AND FUNCIONARIO\_ENTREGA IS NULL) OR

(TIPO\_PROCEDIMIENTO = 'E'

AND ESTADO\_REVISION IS NULL

AND FUNCIONARIO\_ENTREGA IS NOT NULL)

);

Para la relación de cardinalidad 1-1 entre FUNCIONARIO y LOCACION, se generará una restricción de tipo UNIQUE que garantice que para cada laboratorio se asigne un auxiliar de laboratorio único. El código es el siguiente indicando en color verde la línea agregada:

/\*==============================================================\*/

/\* Table: LOCACION \*/

/\*==============================================================\*/

create table LOCACION

(

ID\_EDIFICIO INTEGER not null,

ID\_LOCACION INTEGER not null,

NOMBRE\_LOCACION VARCHAR2(100) not null,

TIPO\_LOCACION CHAR(1) not null

constraint CKC\_TIPO\_LOCACION check (TIPO\_LOCACION in ('S','O','L')),

CAPACIDAD\_SALON INTEGER ,

TIPO\_SALON CHAR(3)

constraint CKC\_TIPO\_SALON check (TIPO\_SALON is null or (TIPO\_SALON in ('AUD','TAL','COM'))),

PUESTOS\_OFICINA INTEGER ,

ID\_AREA INTEGER ,

ID\_AUXILIAR INTEGER ,

ACREDITACION\_LAB VARCHAR2(20) ,

constraint PK\_LOCACION primary key (ID\_EDIFICIO, ID\_LOCACION),

constraint UK\_LOCACION unique (ID\_AUXILIAR)

);

Para la estructura que surge de la entidad intermedia PROCEDIMIENTO, se eliminarán los índices generados en PowerDesigner por las dos llaves foráneas que resultaron de la relación m-m entre las entidades AUXILIAR y ACTIVO en su momento. Esto es porque el motor de bases de datos genera un índice propio para esa estructura y los dos índices adicionales generados resultan innecesarios. En color rojo aparece el código a eliminar del código de creación.

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO2\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO2\_FK on PROCEDIMIENTO (

ID\_AUXILIAR ASC

);

/\*==============================================================\*/

/\* Index: PROCEDIMIENTO\_ACTIVO\_FK \*/

/\*==============================================================\*/

create index PROCEDIMIENTO\_ACTIVO\_FK on PROCEDIMIENTO (

ID\_ACTIVO ASC

);

También deben eliminarse las líneas que borran estos índices al inicio del código.

drop index PROCEDIMIENTO\_ACTIVO\_FK;

drop index PROCEDIMIENTO2\_FK;

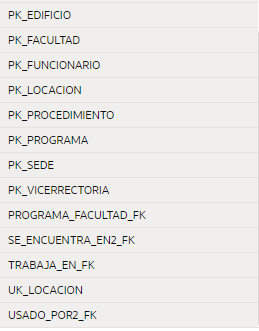
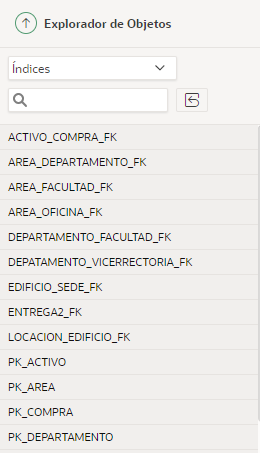
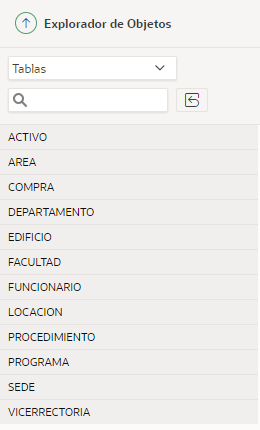
**IMPLEMENATACIÓN DE LA BASE DE DATOS**

La fase de implementación de la base de datos consiste únicamente en ejecutar el código de creación en un motor de bases de datos. Para este proyecto se escogió la versión de ORACLE 11g con el motor ORACLE APEX.

Una vez ejecutado el código en dicho motor, se procederá a generar el código de inserción de datos según las especificaciones dadas para el diseño y las restricciones de integridad por valor y referencial surgidas del diseño de la base de datos en PowerDesigner.

***Ejecución del código de creación SQL en el motor de bases de datos***

Se generaron los siguientes objetos dentro del motor:



*Figura 17 – Objetos generados en ORACLE APEX al ejecutar el código de creación de la base de datos*

Y se tienen las siguientes restricciones de integridad, que pueden ser consultadas usando el comando:

SELECT TABLE\_NAME, CONSTRAINT\_NAME, CONSTRAINT\_TYPE, SEARCH\_CONDITION FROM user\_constraints ORDER BY TABLE\_NAME, CONSTRAINT\_NAME

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE\_NAME** | **CONSTRAINT\_NAME** | **CONSTRAINT\_TYPE** | **SEARCH\_CONDITION** |
| ACTIVO | CKC\_CATEGORIA\_ACTIVO\_ACTIVO | C | CATEGORIA\_ACTIVO in ('CAE','DTS','LAB','LOG','SGT') |
| ACTIVO | FK\_ACTIVO\_USADO\_POR\_FUN | R | - |
| ACTIVO | FK\_COMPRA\_IN\_ACTIVO | R | - |
| ACTIVO | FK\_LOCACION\_IN\_ACTIVO | R | - |
| ACTIVO | PK\_ACTIVO | P | - |
| ACTIVO | SYS\_C0096531894 | C | "ID\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531895 | C | "ID\_COMPRA" IS NOT NULL |
| ACTIVO | SYS\_C0096531896 | C | "REFERENCIA\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531897 | C | "CATEGORIA\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531898 | C | "NOMBRE\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531899 | C | "MARCA\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531900 | C | "VALOR\_ACTIVO" IS NOT NULL |
| ACTIVO | SYS\_C0096531901 | C | "DEPRECIACION\_ACTIVO" IS NOT NULL |
| AREA | AREA\_ARC\_CHK | C | (id\_facultad IS NOT NULL and id\_departamento IS NULL) OR (id\_facultad IS NULL and id\_departamento IS NOT NULL) |
| AREA | FK\_DEPARTAMENTO\_IN\_AREA | R | - |
| AREA | FK\_FACULTAD\_IN\_AREA | R | - |
| AREA | PK\_AREA | P | - |
| AREA | SYS\_C0096531904 | C | "ID\_AREA" IS NOT NULL |
| AREA | SYS\_C0096531905 | C | "NOMBRE\_AREA" IS NOT NULL |
| COMPRA | PK\_COMPRA | P | - |
| COMPRA | SYS\_C0096531914 | C | "ID\_COMPRA" IS NOT NULL |
| COMPRA | SYS\_C0096531915 | C | "FECHA\_COMPRA" IS NOT NULL |
| COMPRA | SYS\_C0096531916 | C | "VALOR\_COMPRA" IS NOT NULL |
| DEPARTAMENTO | DEPARTAMENTO\_ARC\_CHK | C | (id\_vicerrectoria IS NOT NULL and id\_facultad IS NULL) OR (id\_vicerrectoria IS NULL and id\_facultad IS NOT NULL) |
| DEPARTAMENTO | FK\_FACULTAD\_IN\_DEPT | R | - |
| DEPARTAMENTO | FK\_VICER\_IN\_DEPT | R | - |
| DEPARTAMENTO | PK\_DEPARTAMENTO | P | - |
| DEPARTAMENTO | SYS\_C0096531918 | C | "ID\_DEPARTAMENTO" IS NOT NULL |
| DEPARTAMENTO | SYS\_C0096531919 | C | "NOMBRE\_DEPARTAMENTO" IS NOT NULL |
| DEPARTAMENTO | SYS\_C0096531920 | C | "DIRECTOR\_DEPARTAMENTO" IS NOT NULL |
| EDIFICIO | FK\_SEDE\_IN\_EDIFICIO | R | - |
| EDIFICIO | PK\_EDIFICIO | P | - |
| EDIFICIO | SYS\_C0096531922 | C | "ID\_SEDE" IS NOT NULL |
| EDIFICIO | SYS\_C0096531923 | C | "ID\_EDIFICIO" IS NOT NULL |
| EDIFICIO | SYS\_C0096531924 | C | "NOMBRE\_EDIFICIO" IS NOT NULL |
| FACULTAD | CKC\_ID\_FACULTAD\_FACULTAD | C | ID\_FACULTAD in ('AGR','JUR','ECO','ING','CIE') |
| FACULTAD | PK\_FACULTAD | P | - |
| FACULTAD | SYS\_C0096531926 | C | "ID\_FACULTAD" IS NOT NULL |
| FACULTAD | SYS\_C0096531927 | C | "NOMBRE\_FACULTAD" IS NOT NULL |
| FACULTAD | SYS\_C0096531928 | C | "DECANO\_FACULTAD" IS NOT NULL |
| FUNCIONARIO | CKC\_TIPO\_AUXILIAR\_FUN | C | TIPO\_AUXILIAR in ('INV','MAN','LAB') |
| FUNCIONARIO | CKC\_TIPO\_FUN | C | TIPO\_FUNCIONARIO in ('E', 'D') |
| FUNCIONARIO | FK\_AREA\_IN\_FUNCIONARIO | R | - |
| FUNCIONARIO | FK\_JEFE\_IN\_FUNCIONARIO | R | - |
| FUNCIONARIO | FK\_PROGRAMA\_IN\_FUNCIONARIO | R | - |
| FUNCIONARIO | FUNCIONARIO\_CHK\_TABLE | C | (TIPO\_FUNCIONARIO = 'E' AND CARGO\_EMPLEADO IS NOT NULL AND ID\_AREA IS NOT NULL AND TITULO\_DOCENTE IS NULL AND ROL\_DOCENTE IS NULL AND ID\_PROGRAMA IS NULL AND ID\_FACULTAD IS NULL) OR (TIPO\_FUNCIONARIO = 'D' AND CARGO\_EMPLEADO IS NULL AND ID\_AREA IS NULL AND TIPO\_AUXILIAR IS NULL AND TITULO\_DOCENTE IS NOT NULL AND ROL\_DOCENTE IS NOT NULL AND ID\_PROGRAMA IS NOT NULL AND ID\_FACULTAD IS NOT NULL) |
| FUNCIONARIO | PK\_FUNCIONARIO | P | - |
| FUNCIONARIO | SYS\_C0096531907 | C | "ID\_FUNCIONARIO" IS NOT NULL |
| FUNCIONARIO | SYS\_C0096531908 | C | "NOMBRE\_FUNCIONARIO" IS NOT NULL |
| FUNCIONARIO | SYS\_C0096531909 | C | "INIC\_CONTRATO\_FUNCIONARIO" IS NOT NULL |
| FUNCIONARIO | SYS\_C0096531910 | C | "TIPO\_FUNCIONARIO" IS NOT NULL |
| LOCACION | CKC\_TIPO\_LOCACION | C | TIPO\_LOCACION in ('S','O','L') |
| LOCACION | CKC\_TIPO\_SALON | C | TIPO\_SALON is null or (TIPO\_SALON in ('AUD','TAL','COM')) |
| LOCACION | FK\_AREA\_IN\_OFICINA | R | - |
| LOCACION | FK\_AUXILIAR\_IN\_LAB | R | - |
| LOCACION | FK\_EDIFICIO\_IN\_LOCACION | R | - |
| LOCACION | LOCACION\_CHK\_TABLE | C | (TIPO\_LOCACION = 'S' AND CAPACIDAD\_SALON IS NOT NULL AND PUESTOS\_OFICINA IS NULL AND ID\_AREA IS NULL AND ID\_AUXILIAR IS NULL AND ACREDITACION\_LAB IS NULL) OR (TIPO\_LOCACION = 'O' AND CAPACIDAD\_SALON IS NULL AND TIPO\_SALON IS NULL AND PUESTOS\_OFICINA IS NOT NULL AND ID\_AREA IS NOT NULL AND ID\_AUXILIAR IS NULL AND ACREDITACION\_LAB IS NULL) OR (TIPO\_LOCACION = 'L' AND CAPACIDAD\_SALON IS NULL AND TIPO\_SALON IS NULL AND PUESTOS\_OFICINA IS NULL AND ID\_AREA IS NULL AND ID\_AUXILIAR IS NOT NULL AND ACREDITACION\_LAB IS NOT NULL) |
| LOCACION | PK\_LOCACION | P | - |
| LOCACION | SYS\_C0096531931 | C | "ID\_EDIFICIO" IS NOT NULL |
| LOCACION | SYS\_C0096531932 | C | "ID\_LOCACION" IS NOT NULL |
| LOCACION | SYS\_C0096531933 | C | "NOMBRE\_LOCACION" IS NOT NULL |
| LOCACION | SYS\_C0096531934 | C | "TIPO\_LOCACION" IS NOT NULL |
| LOCACION | UK\_LOCACION | U | - |
| PROCEDIMIENTO | CKC\_ESTADO\_REVISION\_PROCEDIM | C | ESTADO\_REVISION is null or (ESTADO\_REVISION in ('R','M')) |
| PROCEDIMIENTO | CKC\_TIPO\_PROCEDIMIENT\_PROCEDIM | C | TIPO\_PROCEDIMIENTO in ('E','P','R','M') |
| PROCEDIMIENTO | FK\_ACTIVO\_IN\_PROC | R | - |
| PROCEDIMIENTO | FK\_FUNCIONARIO\_IN\_PROC | R | - |
| PROCEDIMIENTO | FK\_FUN\_ENTREGA\_IN\_PROC | R | - |
| PROCEDIMIENTO | PK\_PROCEDIMIENTO | P | - |
| PROCEDIMIENTO | PROCEDIMIENTO\_CHK\_TABLE | C | (TIPO\_PROCEDIMIENTO = 'R' AND FUNCIONARIO\_ENTREGA IS NULL) OR (TIPO\_PROCEDIMIENTO = 'P' AND FUNCIONARIO\_ENTREGA IS NULL) OR (TIPO\_PROCEDIMIENTO = 'M' AND ESTADO\_REVISION IS NULL AND FUNCIONARIO\_ENTREGA IS NULL) OR (TIPO\_PROCEDIMIENTO = 'E' AND ESTADO\_REVISION IS NULL AND FUNCIONARIO\_ENTREGA IS NOT NULL) |
| PROCEDIMIENTO | SYS\_C0096531939 | C | "ID\_PROCE" IS NOT NULL |
| PROCEDIMIENTO | SYS\_C0096531940 | C | "ID\_ACTIVO" IS NOT NULL |
| PROCEDIMIENTO | SYS\_C0096531941 | C | "ID\_AUXILIAR" IS NOT NULL |
| PROCEDIMIENTO | SYS\_C0096531942 | C | "TIPO\_PROCEDIMIENTO" IS NOT NULL |
| PROCEDIMIENTO | SYS\_C0096531943 | C | "FECHA\_PROCEDIMIENTO" IS NOT NULL |
| PROGRAMA | FK\_FACULTAD\_IN\_PROGRAMA | R | - |
| PROGRAMA | PK\_PROGRAMA | P | - |
| PROGRAMA | SYS\_C0096531947 | C | "ID\_FACULTAD" IS NOT NULL |
| PROGRAMA | SYS\_C0096531948 | C | "ID\_PROGRAMA" IS NOT NULL |
| PROGRAMA | SYS\_C0096531949 | C | "NOMBRE\_PROGRAMA" IS NOT NULL |
| SEDE | PK\_SEDE | P | - |
| SEDE | SYS\_C0096531951 | C | "ID\_SEDE" IS NOT NULL |
| SEDE | SYS\_C0096531952 | C | "NOMBRE\_SEDE" IS NOT NULL |
| VICERRECTORIA | CKC\_ID\_VICERRECTORIA\_VICERREC | C | ID\_VICERRECTORIA in ('FOR','INV','EXT','ADM') |
| VICERRECTORIA | PK\_VICERRECTORIA | P | - |
| VICERRECTORIA | SYS\_C0096531954 | C | "ID\_VICERRECTORIA" IS NOT NULL |
| VICERRECTORIA | SYS\_C0096531955 | C | "NOMBRE\_VICERRECTORIA" IS NOT NULL |
| VICERRECTORIA | SYS\_C0096531956 | C | "VICERRECTOR\_VICERRECTORIA" IS NOT NULL |

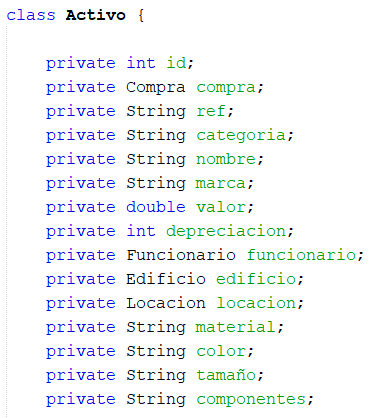
Tabla - Restricciones registradas en ORACLE APEX tras la ejecución del código de creación de la base de datos

***Generación de código SQL para la inserción de datos en el esquema de la base de datos***

El código SQL para insertar datos en la base de datos se generó por medio de un script escrito en el lenguaje programación Java, aprovechando sus características como lenguaje para la programación orientada a objetos.

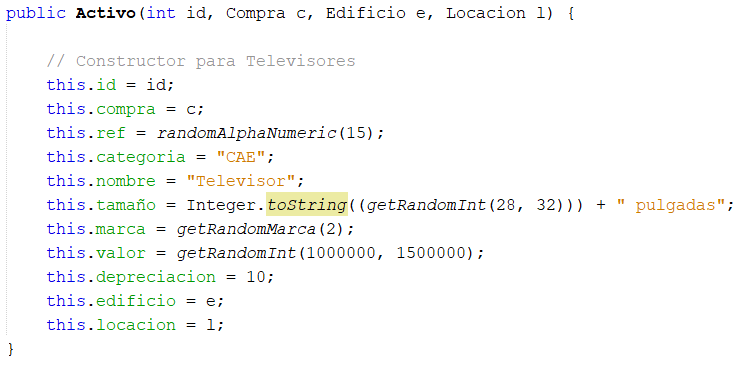
En resumen, se creó una clase por cada una de las tablas de la base de datos. Se agregaron los atributos correspondientes para cada una, donde las llaves foráneas se tomaron como referencias a otros objetos de las clases de las otras estructuras.

A continuación un ejemplo:



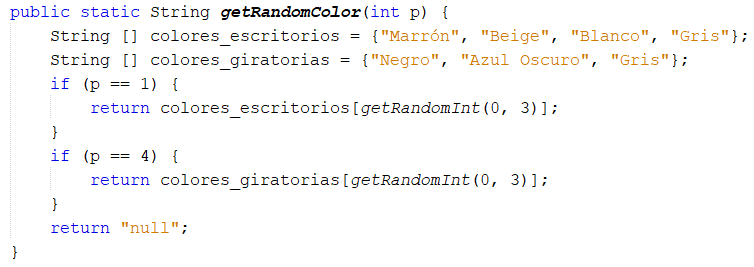
*Figura 18 – Definición de una tabla de la base de datos como clase como metodología para la generación del código de inserción SQL*

De esta manera se fueron instanciando objetos por medio de ciclos y utilizando algunos constructores ideados para ciertos registros. Este es un ejemplo de uno para generar objetos que se convertirán en registros de Televisores:



*Figura 19 – Ejemplo de constructor que permite generar registros para una tabla definida como clase usando el lenguaje de programación Java*

Se usaron algunos métodos de apoyo para generar valores aleatorios, entre ellos un método para generar enteros, para los activos: marcas, materiales, tamaños, colores y para los funcionarios nombres con sus apellidos.



*Figura 20 – Ejemplo de método implementado para generar valores aleatorios para ciertos registros de la base de datos*

Los registros de las siguientes tablas se generaron según los valores de los dominios especificados durante el diseño:

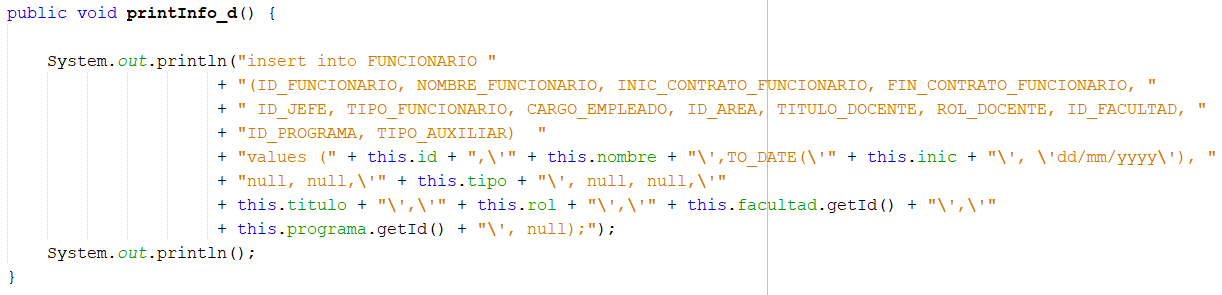
VICERRECTORIA, DEPARTAMENTO, FACULTAD, AREA, PROGRAMA, SEDE y EDIFICIO

En cuanto a las siguientes tablas, se generaron registros aleatorios guardando las condiciones de los dominios que aparecen en la *tabla 12* del presente documento

COMPRA, ACTIVO, LOCACION y FUNCIONARIO

El número de registros generados para ACTIVO, LOCACION (excepto los laboratorios) y FUNCIONARIO puede variar cada vez que se ejecute el script. Eso significa que no siempre se obtendrán 610 funcionarios con 80 de ellos docentes, la misma cantidad de activos ni de salones u oficinas.

Cada clase cuenta con un método que imprime una sentencia de inserción por cada tipo de constructor, tal como se muestra a continuación:



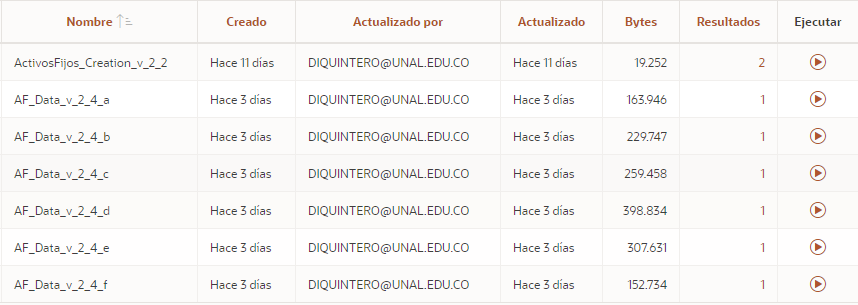
*Figura 21 – Ejemplo de método implementado para generar una sentencia de inserción SQL*

El anterior método imprime una sentencia de inserción de un registro de docente en la tabla FUNCIONARIO.

***Ejecución del código de inserción SQL en el motor de bases de datos***

Para ejecutar el código de inserción en el motor de bases de datos, se necesitó dividirlo en archivos más pequeños, es decir con menor número de sentencias para que sea aceptado por ORACLE APEX, pues esta herramienta impone un tamaño límite por archivo que ronda los 500 KB aproximadamente.

Así se dividió el código de inserción generado del script en Java en seis diferentes archivos que se ejecutaron en el motor, tal como se muestra a continuación:



*Figura 22 – Subdivisión del código de inserción para poder ser cargado en ORACLE APEX según las limitaciones de este motor*

La letra al final de los archivos AF\_Data\_v\_2\_4 indica el orden en que se ejecutaron estos seis archivos de inserción comenzando desde el archivo con letra ‘a’ hasta el archivo con letra ‘f’, siguiendo el orden del abecedario.

Luego de haber generado y ejecutado los códigos de inserción SQL, se encontraron algunos errores. Por ejemplo, cuando se escribió el script en Java, no se consideraron los tipos de salón: Taller y Salas de Cómputo, y se generaron únicamente salones de clase normales y Auditorios.

Hay que tener en cuenta además que la capacidad de las salas de cómputo está determinada por los computadores que se encuentran en dichos salones. Para solucionar este problema, se ubicaron los salones que contaban con computadores usando la siguiente consulta:

SELECT ID\_EDIFICIO, ID\_LOCACION, COUNT(\*) FROM ACTIVO NATURAL JOIN LOCACION WHERE NOMBRE\_ACTIVO = 'Computador de Escritorio' AND TIPO\_LOCACION = 'S' GROUP BY ID\_EDIFICIO, ID\_LOCACION

Se tomó nota de cada uno de estos salones como del número de computadores, para luego usar un comando de actualización sobre cada uno de estos registros, así:

UPDATE LOCACION

SET CAPACIDAD\_SALON = ##, TIPO\_SALON = 'COM'

WHERE ID\_EDIFICIO = ## AND ID\_LOCACION = ##

Así por cada salón se ejecutó uno de estos comandos de actualización, cambiando las identificaciones del mismo en WHERE y definiendo la capacidad de cada uno según el número de computadores encontrado en el mismo con el primer comando.

Contando con estas salas de cómputo registradas debidamente en la base de datos, se procedió a generar algunos talleres con el siguiente comando:

UPDATE LOCACION

SET TIPO\_SALON = 'TAL'

WHERE MOD(ID\_LOCACION, 25) = 0 AND TIPO\_LOCACION = 'S' AND TIPO\_SALON IS NULL

De esta manera se decidió de manera arbitraria que los salones de clase cuya identificación fuera 25 o 50 se cambiaran a talleres.

***Validación de la inserción de datos en la base de datos ejecutando consultas de SQL***

A continuación se presentan 18 consultas de ejemplo sobre la base de datos para verificar la funcionalidad de la base de datos según fue diseñada y los registros insertados.

Para cada consulta se presenta su equivalente en álgebra relacional y en SQL, a excepción de algunas consultas de funciones agregadas y de proyección generalizada que por su extensión, no conviene expresarlas en álgebra relacional. También se creará una vista para poder consultar los resultados en el motor de bases de datos cuando se desee.

*Consulta 1: Selección*

Desplegar los auditorios, talleres y salas de cómputo de la universidad con todo su esquema.

CREATE VIEW consulta\_1 AS

SELECT \* FROM LOCACION WHERE TIPO\_SALON IS NOT NULL ORDER BY TIPO\_SALON, ID\_EDIFICIO, ID\_LOCACION

*Consulta 2: Proyección*

Desplegar la identificación, el nombre, el título y el identificador de programa de los docentes titulares de la universidad.

CREATE VIEW consulta\_2 AS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, TITULO\_DOCENTE, ID\_PROGRAMA

FROM FUNCIONARIO

WHERE ROL\_DOCENTE = 'Titular'

ORDER BY ID\_PROGRAMA, TITULO\_DOCENTE

*Consulta 3: Producto Natural basado en Producto Cartesiano*

Desplegar la identificación y el nombre de los jefes de área indicando el nombre del área de la cual son jefes.

CREATE VIEW consulta\_3 AS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, NOMBRE\_AREA

FROM FUNCIONARIO F, AREA A

WHERE (F.ID\_AREA = A.ID\_AREA) AND

CARGO\_EMPLEADO = 'Jefe de Área'

*Consulta 4: Producto Natural*

Desplegar el identificador, la referencia, el tamaño y la ubicación de los televisores de marca Sony de la sede principal de la universidad.

CREATE VIEW consulta\_4 AS

SELECT ID\_ACTIVO, REFERENCIA\_ACTIVO, TAMANO\_ACTIVO, NOMBRE\_LOCACION

FROM ACTIVO NATURAL JOIN LOCACION NATURAL JOIN EDIFICIO NATURAL JOIN SEDE

WHERE NOMBRE\_ACTIVO = 'Televisor' AND MARCA\_ACTIVO = 'Sony' AND ID\_SEDE = 1

ORDER BY NOMBRE\_LOCACION

*Consulta 5: Unión*

Desplegar el nombre de todos los directivos de la universidad, es decir, vicerrectores, decanos y directores, indicando la entidad y el nombre de la entidad a la que pertenecen.

CREATE VIEW consulta\_5 AS

SELECT VICERRECTOR\_VICERRECTORIA NOMBRES\_DIRECTIVOS, 'Vicerrectoría' ENTIDAD, NOMBRE\_VICERRECTORIA NOMBRE\_ENTIDAD FROM VICERRECTORIA UNION

(SELECT DECANO\_FACULTAD, 'Facultad', NOMBRE\_FACULTAD FROM FACULTAD) UNION

(SELECT DIRECTOR\_DEPARTAMENTO, 'Departamento', NOMBRE\_DEPARTAMENTO FROM DEPARTAMENTO)

*Consulta 6: Diferencia*

Desplegar la identificación y el nombre de los funcionarios a los cuales no se les ha hecho entrega formal de su puesto de trabajo.

CREATE VIEW consulta\_6 AS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO FROM FUNCIONARIO MINUS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO FROM ACTIVO NATURAL JOIN FUNCIONARIO

*Consulta 7: Intersección*

Desplegar los talleres que cuentan con un pupitre para el docente y un televisor al mismo tiempo, indicando su identificación (llave compuesta ID\_EDIFICIO, ID\_LOCACION) y capacidad.

CREATE VIEW consulta\_7 AS

SELECT ID\_EDIFICIO, ID\_LOCACION,NOMBRE\_LOCACION, CAPACIDAD\_SALON FROM LOCACION NATURAL JOIN ACTIVO WHERE NOMBRE\_ACTIVO = 'Pupitre' AND TIPO\_SALON ='TAL' INTERSECT

(SELECT ID\_EDIFICIO, ID\_LOCACION, NOMBRE\_LOCACION, CAPACIDAD\_SALON FROM LOCACION NATURAL JOIN ACTIVO WHERE NOMBRE\_ACTIVO = 'Televisor' AND TIPO\_SALON ='TAL')

*Consulta 8: División*

Desplegar todo el nombre de las salas de computación en las cuales hay computadores de todas las marcas de computadores existentes en el inventario.

Se harán dos asignaciones; una para el dividendo y otra para el divisor:

CREATE VIEW divisor AS

SELECT DISTINCT MARCA\_ACTIVO FROM ACTIVO WHERE NOMBRE\_ACTIVO = 'Computador de Escritorio'

CREATE VIEW dividendo AS

SELECT NOMBRE\_LOCACION, MARCA\_ACTIVO FROM LOCACION NATURAL JOIN ACTIVO WHERE TIPO\_SALON = 'COM'

CREATE VIEW consulta\_8 AS

SELECT NOMBRE\_LOCACION FROM dividendo MINUS (SELECT NOMBRE\_LOCACION FROM (SELECT \* FROM (SELECT NOMBRE\_LOCACION FROM dividendo), divisor MINUS (SELECT \* FROM dividendo)))

*Consulta 9: Proyección Generalizada*

Desplegar el código de cada compra sin la parte inicial que indica el año y presentar la fecha subdividida en día, mes y año.

CREATE VIEW consulta\_9 AS

SELECT TO\_NUMBER(SUBSTR(TO\_CHAR(id\_compra),5, 3)) cod\_compra,

TO\_CHAR(fecha\_compra, 'yyyy') año\_compra,

TO\_CHAR(fecha\_compra, 'mm') mes\_compra,

TO\_CHAR(fecha\_compra, 'dd') dia\_compra,

valor\_compra

FROM COMPRA

*Consulta 10: Proyección Generalizada*

Desplegar el nombre de los activos que hacen parte de las secciones de laboratorios indicando el porcentaje de su valor con respecto al de la compra (en la cual se incluyeron otros activos de otras áreas). Mostrar el valor del activo y de la compra con formato de moneda.

CREATE VIEW consulta\_10 AS

SELECT NOMBRE\_ACTIVO,

(ROUND((VALOR\_ACTIVO \* 100)/VALOR\_COMPRA, 2) || '%') PORCENTAJE\_COMPRA,

(TO\_CHAR(VALOR\_ACTIVO,'$999,999,999.99') || ' COP') VALOR\_ACTIVO,

(TO\_CHAR(VALOR\_COMPRA,'$999,999,999.99') || ' COP') VALOR\_COMPRA

FROM ACTIVO NATURAL JOIN COMPRA

WHERE CATEGORIA\_ACTIVO = 'LAB'

*Consulta 11: Proyección Generalizada*

Desplegar el esquema de la tabla DEPARTAMENTO pero con los nombres de los directores subdivididos en primer nombre, segundo nombre, primer apellido y segundo apellido. Algunos de estos directores tienen solo el primer apellido registrado en la base de datos.

CREATE VIEW consulta\_11 AS

SELECT id\_departamento,

nombre\_departamento,

REGEXP\_SUBSTR(director\_departamento, '(\w+)\s', 1) primer\_nombre\_direc,

REGEXP\_SUBSTR(director\_departamento, '(\w+)\s', INSTR(director\_departamento, ' ', 1)) segundo\_nombre\_direc,

REGEXP\_SUBSTR(director\_departamento, '(\w+)\s', INSTR(director\_departamento, ' ', INSTR(director\_departamento, ' ', 1)+1)) primer\_apellido\_direc,

SUBSTR(director\_departamento, INSTR(director\_departamento, ' ', INSTR(director\_departamento, ' ', 1, 2)+1)) segundo\_apellido\_direc,

id\_vicerrectoria,

id\_facultad

FROM departamento WHERE (length(director\_departamento) - length(replace(director\_departamento, ' ', '')) +1) = 4

UNION

SELECT id\_departamento,

nombre\_departamento,

REGEXP\_SUBSTR(director\_departamento, '(\w+)\s', 1) primer\_nombre\_direc,

REGEXP\_SUBSTR(director\_departamento, '(\w+)\s', INSTR(director\_departamento, ' ', 1)) segundo\_nombre\_direc,

SUBSTR(director\_departamento, INSTR(director\_departamento, ' ', INSTR(director\_departamento, ' ', 1)+1)) primer\_apellido\_direc,

null segundo\_apellido\_direc,

id\_vicerrectoria,

id\_facultad

FROM departamento WHERE (length(director\_departamento) - length(replace(director\_departamento, ' ', '')) +1) = 3

*Consulta 12: Funciones Agregadas*

Desplegar el número de funcionarios por área a los cuales no se les ha hecho entrega formal de su puesto de trabajo.

CREATE VIEW consulta\_12 AS

SELECT NOMBRE\_AREA, COUNT(\*) NUM\_SIN\_ENTREGA

FROM (SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, ID\_AREA FROM FUNCIONARIO MINUS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, ID\_AREA FROM ACTIVO NATURAL JOIN FUNCIONARIO)

NATURAL JOIN AREA

GROUP BY NOMBRE\_AREA

ORDER BY NUM\_SIN\_ENTREGA desc

*Consulta 13: Funciones Agregadas, with clause, Inner Join*

Indicar el número de puestos entregados por cargo, junto con el número de empleados por cargo y el porcentaje de avance en la entrega.

CREATE VIEW consulta\_13 AS

WITH puestos\_asignados AS

(SELECT CARGO\_EMPLEADO, COUNT(CARGO\_EMPLEADO) PUESTOS\_ASIGNADOS

FROM (SELECT ID\_FUNCIONARIO FROM ACTIVO WHERE ID\_FUNCIONARIO IS NOT NULL GROUP BY ID\_FUNCIONARIO)

NATURAL JOIN FUNCIONARIO WHERE CARGO\_EMPLEADO IS NOT NULL GROUP BY CARGO\_EMPLEADO),

empleados\_por\_cargo AS

(SELECT CARGO\_EMPLEADO, COUNT(\*) EMPLEADOS\_EXISTENTES

FROM FUNCIONARIO WHERE CARGO\_EMPLEADO IS NOT NULL GROUP BY CARGO\_EMPLEADO)

SELECT puestos\_asignados.CARGO\_EMPLEADO,

PUESTOS\_ASIGNADOS,

EMPLEADOS\_EXISTENTES,

(ROUND((PUESTOS\_ASIGNADOS\*100)/EMPLEADOS\_EXISTENTES, 2) || '%') PORCENTAJE\_ASIGNADO

FROM puestos\_asignados INNER JOIN empleados\_por\_cargo

ON puestos\_asignados.CARGO\_EMPLEADO = empleados\_por\_cargo.CARGO\_EMPLEADO

*Consulta 14: with clause, Right Outer Join*

Desplegar una tabla en la que se concatenen dos relaciones. Una en la que se muestren los funcionarios a los cuales ya se les asignó un computador y otra que muestre los funcionarios a los que ya se les asignó un escritorio. Mostrar las equivalencias de ambas relaciones con los valores nulos al lado derecho.

CREATE OR REPLACE VIEW consulta\_14 AS

WITH funcionarios\_con\_computador AS (SELECT ID\_FUNCIONARIO ID\_FUNCIONARIO\_1, NOMBRE\_FUNCIONARIO NOMBRE\_FUNCIONARIO\_1, ID\_ACTIVO COMPUTADOR FROM FUNCIONARIO NATURAL JOIN ACTIVO WHERE NOMBRE\_ACTIVO = 'Computador de Escritorio'),

funcionarios\_con\_escritorio AS (SELECT ID\_FUNCIONARIO ID\_FUNCIONARIO\_2, NOMBRE\_FUNCIONARIO NOMBRE\_FUNCIONARIO\_2, ID\_ACTIVO ESCRITORIO FROM FUNCIONARIO NATURAL JOIN ACTIVO WHERE NOMBRE\_ACTIVO = 'Escritorio')

SELECT \* FROM funcionarios\_con\_computador RIGHT OUTER JOIN funcionarios\_con\_escritorio ON funcionarios\_con\_computador.ID\_FUNCIONARIO\_1 = funcionarios\_con\_escritorio.ID\_FUNCIONARIO\_2

ORDER BY COMPUTADOR desc

Debido a que los puestos se habían asignado completos, es decir, computador, escritorio y silla. Se eliminaron actualizaron algunas tuplas para hacer la consulta anterior, con el siguiente comando:

UPDATE ACTIVO

SET ID\_FUNCIONARIO = NULL

WHERE MOD(ID\_FUNCIONARIO, 40) = 0 AND NOMBRE\_ACTIVO = 'Computador de Escritorio'

*Consulta 15: Consultas Anidadas*

Desplegar la identificación y el nombre de los funcionarios a los cuales no se les ha entregado puesto de trabajo.

CREATE VIEW consulta\_15 AS

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO FROM FUNCIONARIO WHERE ID\_FUNCIONARIO NOT IN (SELECT ID\_FUNCIONARIO FROM FUNCIONARIO NATURAL JOIN ACTIVO)

*Consulta 16: Consultas Anidadas*

Verificar que los valores de los activos que pertenecen a una misma compra no superan el valor total de la compra. De ser así, desplegar los datos de la compra.

Para asegurarse de que no haya totales mayores al valor de una compra se usó la siguiente sentencia de actualización:

UPDATE COMPRA

SET VALOR\_COMPRA = (WITH sumas\_valores\_activos AS (SELECT ID\_COMPRA, SUM(VALOR\_ACTIVO) SUMA\_VALORES\_ACTIVOS FROM ACTIVO GROUP BY ID\_COMPRA)

SELECT sva.SUMA\_VALORES\_ACTIVOS FROM sumas\_valores\_activos sva WHERE sva.ID\_COMPRA = COMPRA.ID\_COMPRA)

La siguiente consulta permite verificar que se obtuvo el resultado correcto:

WITH a AS

(SELECT ID\_COMPRA, SUM(VALOR\_ACTIVO) FROM ACTIVO GROUP BY ID\_COMPRA),

b AS

(SELECT ID\_COMPRA, VALOR\_COMPRA FROM COMPRA)

SELECT \* FROM a INNER JOIN b ON a.ID\_COMPRA = B.ID\_COMPRA

Ahora modificamos deliberadamente dos registros de la tabla COMPRA

UPDATE COMPRA

SET VALOR\_COMPRA = VALOR\_COMPRA – 245000

WHERE ID\_COMPRA = 2020041

UPDATE COMPRA

SET VALOR\_COMPRA = VALOR\_COMPRA + 500000

WHERE ID\_COMPRA = 2020033

La consulta es la siguiente:

CREATE VIEW consulta\_16 AS

WITH sumas AS (SELECT ID\_COMPRA, SUM(VALOR\_ACTIVO) SUMA FROM ACTIVO GROUP BY ID\_COMPRA)

SELECT ID\_COMPRA, SUMA, (SUMA – (SELECT VALOR\_COMPRA FROM COMPRA C WHERE C.ID\_COMPRA = sumas.ID\_COMPRA)) EXCESO

FROM sumas

WHERE SUMA > (SELECT VALOR\_COMPRA FROM COMPRA WHERE ID\_COMPRA = sumas.ID\_COMPRA)

La siguiente consulta busca las compras cuyo valor es menor al que apareció en la factura:

WITH sumas AS (SELECT ID\_COMPRA, SUM(VALOR\_ACTIVO) SUMA FROM ACTIVO GROUP BY ID\_COMPRA)

SELECT ID\_COMPRA, SUMA, ((SELECT VALOR\_COMPRA FROM COMPRA C WHERE C.ID\_COMPRA = sumas.ID\_COMPRA)-SUMA) FALTANTE

FROM sumas

WHERE SUMA < (SELECT VALOR\_COMPRA FROM COMPRA WHERE ID\_COMPRA = sumas.ID\_COMPRA)

*Consulta 17: Consultas Anidadas*

Desplegar el detalle de las compras realizadas que han superado el promedio del valor de compras en más de 25 millones de pesos.

CREATE VIEW consulta\_17 AS

SELECT ID\_COMPRA, VALOR\_COMPRA, FECHA\_COMPRA, ID\_ACTIVO, NOMBRE\_ACTIVO, VALOR\_ACTIVO FROM COMPRA NATURAL JOIN ACTIVO WHERE (VALOR\_COMPRA - (SELECT AVG(VALOR\_COMPRA) FROM COMPRA)) >= 25000000 ORDER BY ID\_COMPRA

*Consulta 18: Consultas Anidadas, Proyección Generalizada*

Desplegar una relación en la cual se muestre el identificador, el nombre, el valor por el cual se compró cada activo, la fecha de la compra, el índice de depreciación, los días trascurridos desde la compra, el índice de depreciación acumulado y el valor que tiene hoy cada activo

CREATE VIEW consulta\_18 AS

SELECT S.\*, ROUND(VALOR\_ACTIVO-(VALOR\_ACTIVO\*S.DEPRECIACION\_TRANSCURRIDA)/100, 0) VALOR\_ACTIVO\_HOY FROM

(SELECT SQ.\*, ROUND(((SQ.DIAS\_TRANSCURRIDOS \* DEPRECIACION\_ACTIVO)/365), 2) DEPRECIACION\_TRANSCURRIDA FROM

(SELECT ID\_ACTIVO, NOMBRE\_ACTIVO, VALOR\_ACTIVO, FECHA\_COMPRA, DEPRECIACION\_ACTIVO, ROUND(SYSDATE - FECHA\_COMPRA, 0) DIAS\_TRANSCURRIDOS FROM ACTIVO NATURAL JOIN COMPRA) SQ) S

***Implementación de PL/SQL***

A continuación se presenta una serie de procedimientos, funciones y disparadores agregados a la base de datos como implementación del lenguaje PL/SQL.

*Procedimientos*

El siguiente procedimiento imprime una facture que contiene el detalle de todos los activos comprados con una misma facture cuyo ID\_COMPRA se evalúa como parámetro del procedimiento.

create or replace PROCEDURE generarFactura(ID\_COM IN NUMBER)

IS

CURSOR activos IS SELECT ID\_ACTIVO, REFERENCIA\_ACTIVO, NOMBRE\_ACTIVO, MARCA\_ACTIVO, VALOR\_ACTIVO FROM ACTIVO WHERE ID\_COMPRA = ID\_COM ORDER BY NOMBRE\_aCTIVO, ID\_ACTIVO;

total NUMBER;

suma NUMBER;

reg activos%ROWTYPE;

fecha DATE;

BEGIN

SELECT VALOR\_COMPRA INTO total FROM COMPRA WHERE ID\_COMPRA = ID\_COM;

SELECT SUM(VALOR\_ACTIVO) INTO suma FROM ACTIVO WHERE ID\_COMPRA = ID\_COM;

SELECT FECHA\_COMPRA INTO fecha FROM COMPRA WHERE ID\_COMPRA = ID\_COM;

dbms\_output.put\_line('');

dbms\_output.put\_line('FECHA DE COMPRA: ' || fecha);

dbms\_output.put\_line('');

dbms\_output.put\_line('ID FACTURA: ' || ID\_COM);

dbms\_output.put\_line('-----------------------------------------------------------------------------------------------------------');

dbms\_output.put\_line(RPAD('ID Activo', 15) || RPAD('REFERENCIA', 25) || RPAD('NOMBRE', 30) || RPAD('MARCA', 17) || 'VALOR');

dbms\_output.put\_line('-----------------------------------------------------------------------------------------------------------');

OPEN activos;

LOOP

FETCH activos INTO reg;

EXIT WHEN activos%NOTFOUND;

dbms\_output.put\_line(RPAD(TO\_CHAR(reg.ID\_ACTIVO), 15) || RPAD(TO\_CHAR(reg.REFERENCIA\_ACTIVO), 25) || RPAD(TO\_CHAR(reg.NOMBRE\_ACTIVO), 30) ||

RPAD(TO\_CHAR(reg.MARCA\_ACTIVO), 17) || TO\_CHAR(reg.VALOR\_ACTIVO, '$999,999,999.00') || ' COP');

END LOOP;

dbms\_output.put\_line('-----------------------------------------------------------------------------------------------------------');

dbms\_output.put\_line('Suma: ' || TO\_CHAR(suma, '$999,999,999.00') || ' COP');

dbms\_output.put\_line('......................................................');

dbms\_output.put\_line('Valor Registrado en Factura: ' || TO\_CHAR(total, '$999,999,999.00') || ' COP');

dbms\_output.put\_line('......................................................');

dbms\_output.put\_line('Desfase: ' || TO\_CHAR(total-suma, '$999,999,999.00') || ' COP');

CLOSE activos;

END;

*Funciones*

El procedimiento implementado para generar una factura, se implementó también en su versión de función de PL/SQL. A continuación se presenta.

create or replace FUNCTION genFactura(ID\_COM IN NUMBER)

RETURN LONG

IS

CURSOR activos IS SELECT ID\_ACTIVO, REFERENCIA\_ACTIVO, NOMBRE\_ACTIVO, MARCA\_ACTIVO, VALOR\_ACTIVO FROM ACTIVO WHERE ID\_COMPRA = ID\_COM ORDER BY NOMBRE\_aCTIVO, ID\_ACTIVO;

total NUMBER;

suma NUMBER;

reg activos%ROWTYPE;

fecha DATE;

result LONG;

BEGIN

SELECT VALOR\_COMPRA INTO total FROM COMPRA WHERE ID\_COMPRA = ID\_COM;

SELECT SUM(VALOR\_ACTIVO) INTO suma FROM ACTIVO WHERE ID\_COMPRA = ID\_COM;

SELECT FECHA\_COMPRA INTO fecha FROM COMPRA WHERE ID\_COMPRA = ID\_COM;

result := 'FECHA DE COMPRA: ' || fecha;

result := result || chr(13)|| chr(10);

result := result || 'ID FACTURA: ' || ID\_COM;

result := result || chr(13)||chr(10);

result := result || '-----------------------------------------------------------------------------------------------------------';

result := result || chr(13)||chr(10);

result := result || RPAD('ID Activo', 15) || RPAD('REFERENCIA', 25) || RPAD('NOMBRE', 30) || RPAD('MARCA', 17) || 'VALOR';

result := result || '-----------------------------------------------------------------------------------------------------------';

result := result || chr(13)||chr(10);

OPEN activos;

LOOP

FETCH activos INTO reg;

EXIT WHEN activos%NOTFOUND;

result := result ||RPAD(TO\_CHAR(reg.ID\_ACTIVO), 15) || RPAD(TO\_CHAR(reg.REFERENCIA\_ACTIVO), 25) || RPAD(TO\_CHAR(reg.NOMBRE\_ACTIVO), 30) ||

RPAD(TO\_CHAR(reg.MARCA\_ACTIVO), 17) || TO\_CHAR(reg.VALOR\_ACTIVO, '$999,999,999.00') || ' COP';

result := result || chr(13)||chr(10);

END LOOP;

result := result ||'-----------------------------------------------------------------------------------------------------------';

result := result || chr(13)||chr(10);

result := result || 'Suma: ' || TO\_CHAR(suma, '$999,999,999.00') || ' COP';

result := result || chr(13)||chr(10);

result := result || '......................................................';

result := result || chr(13)||chr(10);

result := result || 'Valor Registrado en Factura: ' || TO\_CHAR(total, '$999,999,999.00') || ' COP';

result := result || chr(13)||chr(10);

result := result || '......................................................';

result := result || chr(13)||chr(10);

result := result || 'Desfase: ' || TO\_CHAR(total-suma, '$999,999,999.00') || ' COP';

result := result || chr(13)||chr(10);

CLOSE activos;

RETURN result;

END;

Se implementó una función para obtener la oficina en la que trabajaría un funcionario evaluando como parámetro el ID\_FUNCIONARIO.

create or replace FUNCTION getOficinaByFuncionario(id\_fun IN NUMBER)

RETURN NUMBER

IS

id\_loc NUMBER;

BEGIN

SELECT MIN(id\_locacion) INTO id\_loc FROM (SELECT id\_funcionario, id\_area FROM funcionario)

NATURAL JOIN AREA NATURAL JOIN LOCACION WHERE id\_fun = id\_funcionario;

RETURN id\_loc;

END getOficinaByFuncionario;

La función anterior no es útil por si sola, porque la llave primaria de LOCACION es compuesta. Puede haber oficinas de distintos edificios con el mismo identificador. Por eso se requirió implementar la siguiente función.

create or replace FUNCTION getEdificioByOficina(id\_ofi IN NUMBER)

RETURN NUMBER

IS

id\_edi NUMBER;

BEGIN

SELECT id\_edificio INTO id\_edi FROM LOCACION WHERE id\_ofi = id\_locacion;

RETURN id\_edi;

END getEdificioByOficina;

De esta manera, para implementar esta función, se debe hacer la siguiente composición:

getEdificioByOficina(getOficinaByFuncionario(ID\_FUNCIONARIO));

Así, al ejecutar ambas funciones, es posible obtener una llave primaria apropiada del edificio y la oficina en la cual trabaja un funcionario administrativo de la universidad.

Adicional a estas dos funciones, se generaron tres más que son útiles para el equipo de Contabilidad de la universidad. Estas funciones permiten obtener la información contable de cada activo.

La primera función, calcula los días transcurridos hasta la fecha, contando desde la fecha en que se compró el activo:

create or replace FUNCTION dias\_desde\_compra(fecha\_compra IN DATE)

RETURN NUMBER

IS

dias NUMBER;

BEGIN

dias := ROUND(SYSDATE - fecha\_compra, 0);

RETURN dias;

END dias\_desde\_compra;

La segunda función, calcula el porcentaje de depreciación transcurrido para un activo según la fecha en que se compró. Usa la función anterior para hacer el cálculo.

create or replace FUNCTION deprec\_trans(fecha\_compra IN DATE, deprec\_total IN NUMBER)

RETURN NUMBER

IS

deprec NUMBER;

BEGIN

deprec := ROUND((dias\_desde\_compra(fecha\_compra) \* deprec\_total)/365, 2);

RETURN deprec;

END deprec\_trans;

La última función calcula el valor actual del activo según el índice de depreciación transcurrido hasta la fecha. Usa la función anterior para generar el resultado.

create or replace FUNCTION valor\_activo\_hoy(valor\_activo IN NUMBER, deprec\_total IN NUMBER, fecha\_compra IN DATE)

RETURN NUMBER

IS

valor\_hoy NUMBER;

BEGIN

valor\_hoy := ROUND(valor\_activo-(valor\_activo\*deprec\_trans(fecha\_compra, deprec\_total))/100, 0) ;

RETURN valor\_hoy;

END valor\_activo\_hoy;

Estas funciones permiten ejecutar la consulta 18 mucho con mayor facilidad, al encapsular la complejidad de dichas operaciones. A continuación se muestra un ejemplo:

CREATE VIEW depreciacion AS

SELECT ID\_ACTIVO, NOMBRE\_ACTIVO, VALOR\_ACTIVO, FECHA\_COMPRA, DEPRECIACION\_ACTIVO,

DIAS\_DESDE\_COMPRA(FECHA\_COMPRA) DIAS\_TRANSCURRIDOS,

DEPREC\_TRANS(FECHA\_COMPRA, DEPRECIACION\_ACTIVO) DEPRECIACION\_ACUMULADA,

VALOR\_ACTIVO\_HOY(VALOR\_ACTIVO, DEPRECIACION\_ACTIVO, FECHA\_COMPRA) VALOR\_ACTIVO\_HOY

FROM ACTIVO NATURAL JOIN COMPRA

*Disparadores*

Se generaron dos autonumeradores para las tablas PROCEDIMIENTO y ACTIVO, de manera que cada vez que se inserte un registro para una de estas dos tablas, no importa cuál valor se escribe en el campo de la llave primaria, el valor que se almacena en la base de datos para este atributo es el siguiente número de una secuencia, de manera que el usuario no tenga que recordar cual es el último ID registrado para esa tabla.

Para hacer esto en la versión de ORACLE 11g se necesita crear una secuencia por cada autonumerador.

CREATE SEQUENCE procedimiento\_secuencia;

CREATE SEQUENCE activo\_secuencia START WITH 2793;

Para la secuencia de activos, se necesita especificar el valor inicial de la secuencia, pues ya habían 2793 activos registrados en la base de datos según el código de inserción generado.

Habiendo realizado esto, se procede a crear los disparadores.

Estos son dos disparadores del tipo BEFORE INSERT a nivel de fila.

create or replace TRIGGER autonum\_procedimiento

BEFORE INSERT ON procedimiento

FOR EACH ROW

BEGIN

SELECT procedimiento\_secuencia.nextval

INTO :new.id\_proce

FROM dual;

END;

create or replace TRIGGER autonum\_activo

BEFORE INSERT ON activo

FOR EACH ROW

BEGIN

SELECT activo\_secuencia.nextval

INTO :new.id\_activo

FROM dual;

END;

Se generarán dos disparadores que garantizan la actualización de la tabla ACTIVO cuando se hace una entrega de un activo como puesto de trabajo a un funcionario. El primero asigna el ID\_FUNCIONARIO al activo en cuestión, formalizando la entrega del activo del puesto de trabajo.

create or replace TRIGGER update\_funcionario\_on\_activo

AFTER INSERT ON procedimiento

FOR EACH ROW

BEGIN

IF (:NEW.tipo\_procedimiento = 'E') THEN

UPDATE ACTIVO

SET id\_funcionario = :NEW.funcionario\_entrega

WHERE id\_activo = :NEW.id\_activo;

END IF;

END;

El segundo disparador se asegura de actualizar la ubicación física del activo haciendo uso de las dos funciones creadas para encontrar la oficina y el edificio en el que trabajaría un funcionario.

create or replace TRIGGER update\_locacion\_on\_activo

AFTER INSERT ON procedimiento

FOR EACH ROW

BEGIN

IF (:NEW.tipo\_procedimiento = 'E') THEN

UPDATE ACTIVO

SET id\_locacion = getOficinaByFuncionario(:NEW.funcionario\_entrega),

id\_edificio = getEdificioByOficina(getOficinaByFuncionario(:NEW.funcionario\_entrega))

WHERE id\_activo = :NEW.id\_activo;

END IF;

END;

Otro disparador útil para la implementación de la base de datos es el siguiente, en el que se retira el valor ID\_FUNCIONARIO de un activo que es devuelto por un funcionario al retirarse de la compañía.

create or replace TRIGGER remove\_funcionario\_on\_activo

AFTER INSERT ON procedimiento

FOR EACH ROW

BEGIN

IF (:NEW.tipo\_procedimiento = 'P') THEN

UPDATE ACTIVO

SET id\_funcionario = NULL

WHERE id\_activo = :NEW.id\_activo;

END IF;

END;

Un último disparador busca cambiar el estado de revisión de un activo de la tabla PROCEDIMIENTO cuando ya ha sido reparado por un auxiliar de mantenimiento, de manera que se cambie dicho estado ‘M’ a NULL que de acuerdo con el diseño de dicha estructura, significaría que el activo está en buenas condiciones.

create or replace TRIGGER regReparacion

BEFORE INSERT ON procedimiento

FOR EACH ROW

BEGIN

IF (:NEW.tipo\_procedimiento = 'M') THEN

UPDATE PROCEDIMIENTO

SET ESTADO\_REVISION = NULL

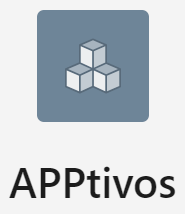
WHERE id\_activo = :NEW.id\_activo;

END IF;

END;

**DESARROLLO DE LA APLICACIÓN**

El nombre de la aplicación es APPtivos.



La aplicación se compone de 12 páginas.

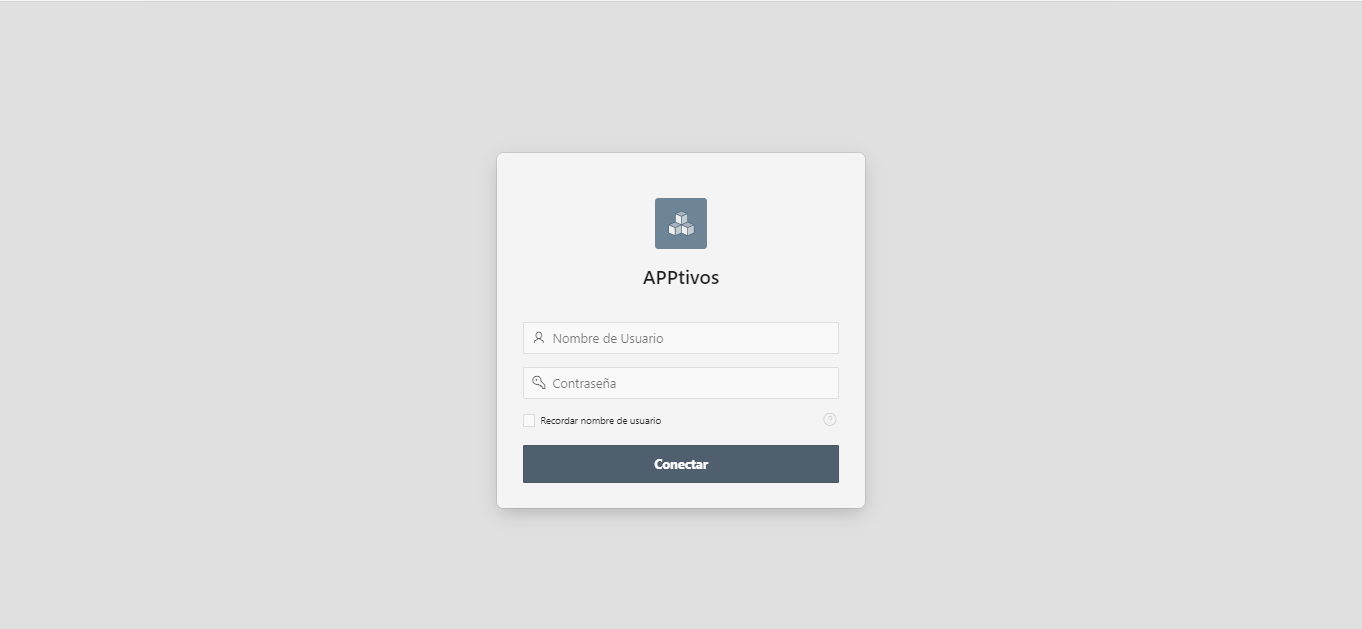
Cada página está diseña para una entidad interesada específica.

*Es posible garantizar accesos específicos para cada tipo de usuario, de manera que cada uno solo vea las páginas a las que tiene acceso. Esto se hace creando roles o perfiles de usuario y especificando en la sección de Seguridad de la página el tipo de usuario que puede acceder a ella. Para esta versión, no se implementó, con el fin de recorrer las páginas con mayor facilidad sin tener que desconectarse y conectarse cada vez a la aplicación, cada vez que se quiera ver un cambio entre las tablas.*

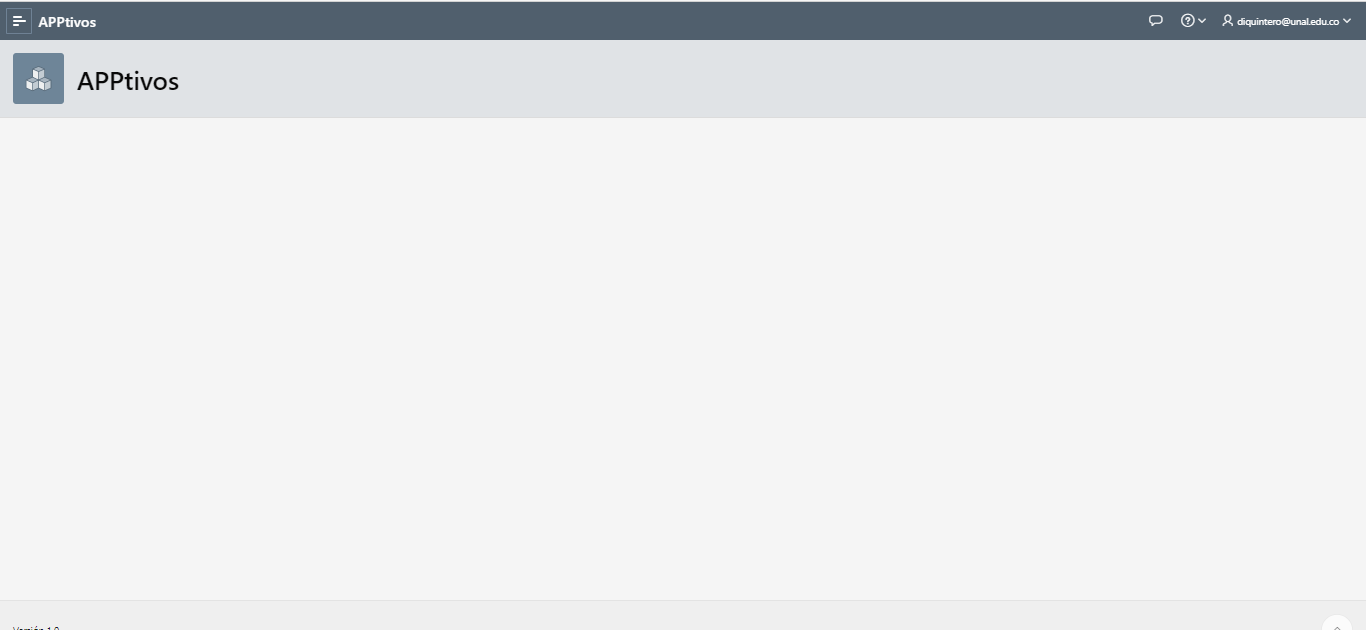
Los tipos de usuarios de la aplicación son:

* Auxiliares de Inventarios y personal de Compras y Almacén
* Auxiliares de Mantenimiento
* Personal administrativo de Logística, especialmente del área de Servicios Generales
* Personal de Contabilidad y Finanzas
* Personal de Gestión Humana y Desarrollo Organizacional

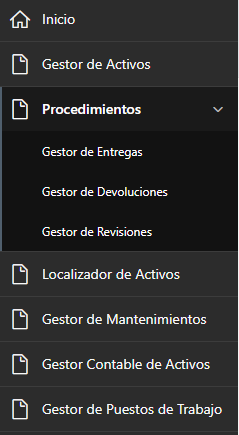
Los usuarios deben autenticarse en la siguiente página de Conexión, una vez abran la aplicación:



Una vez ingresen, la primera página de la aplicación es la de Inicio:



En la parte izquierda de la aplicación hay un menú de navegación por la aplicación que incluye una lista de enlaces a las páginas principales de la misma:



A continuación se indican los tipos de usuario que podrían acceder a cada una de estas páginas:

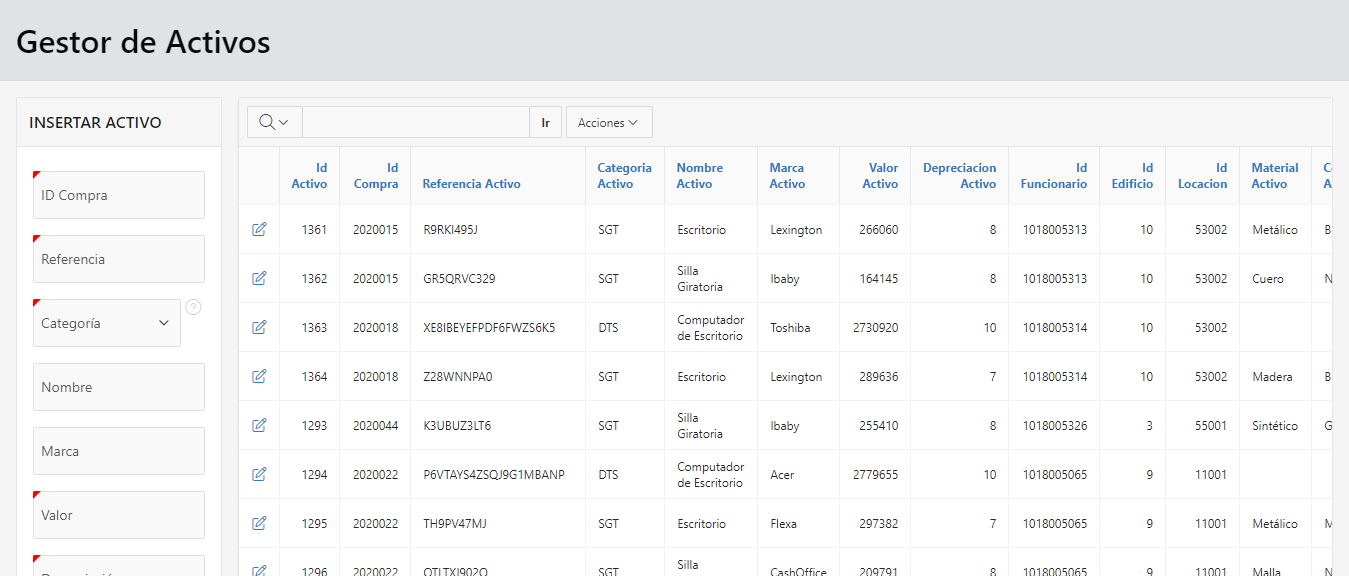
|  |  |
| --- | --- |
| Página | Tipo de Usuario |
| Gestor de Activos | Auxiliares de Inventarios y personal de Compras y Almacén |
| Procedimientos |
| Gestor de Entregas |
| Gestor de Devoluciones |
| Gestor de Revisiones |
| Localizador de Activos (Gestor Logístico de Activos) | Personal administrativo de Logística, especialmente del área de Servicios Generales |
| Gestor de Mantenimientos | Auxiliares de Mantenimiento |
| Gestor Contable de Activos | Personal de Contabilidad y Finanzas |
| Gestor de Puestos de Trabajo | Personal de Gestión Humana y Desarrollo Organizacional |

Ahora, se describirá cada una de estas páginas junto con algunos detalles de su diseño.

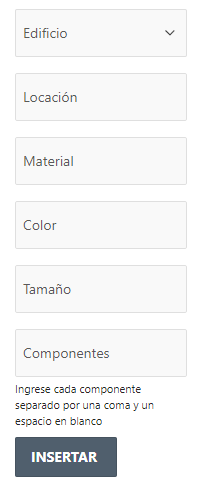
*Gestor de Activos*

Esta página tiene como objetivo mostrar todo el esquema de la tabla ACTIVO, de manera que pueda ser consultado por los auxiliares de inventarios y el personal de compras y almacén.

La página se compone de dos regiones. Un informe basado en un formulario que permite consultar la tabla ACTIVO y hacer modificaciones de registros en dicha tabla.



La segunda región es un formulario creado con una serie de campos de texto, campos de autocompletado, listas de selección y un botón para insertar un registro nuevo en la estructura. La siguiente imagen muestra algunos de estos campos y el botón de INSERTAR al final.



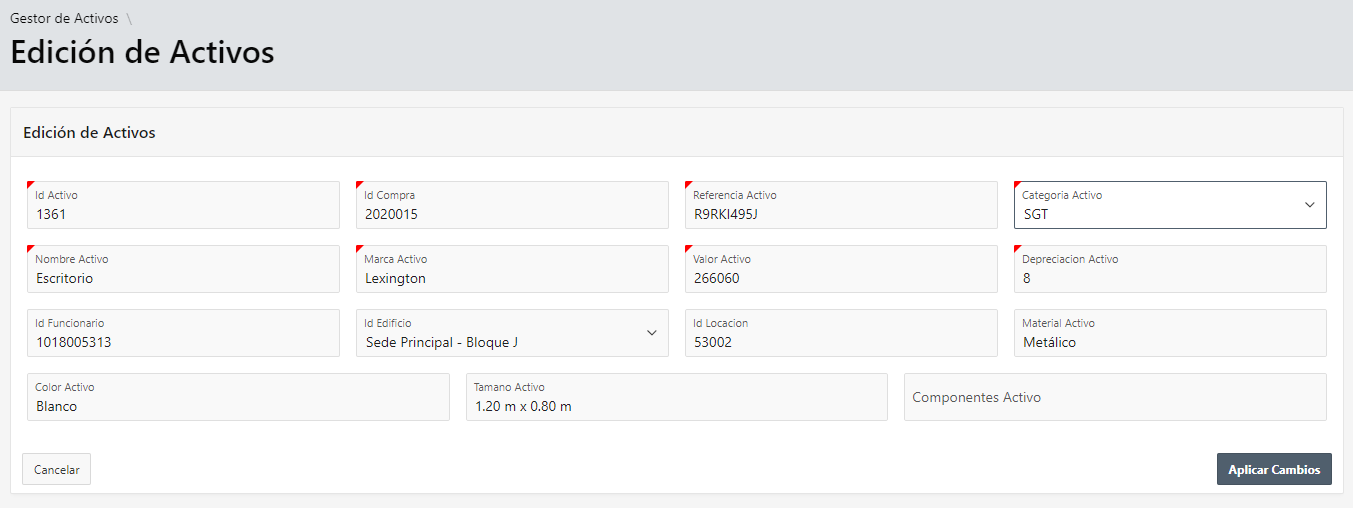
Algunos de estos campos del formulario cuentan con una consulta SQL asociada que permite desplegar opciones al usuario. A continuación se presentan estas consultas:

|  |  |  |
| --- | --- | --- |
| ATRIBUTO | CAMPO DE TEXTO | CONSULTA SQL |
| CATEGORIA\_ACTIVO | AC\_CAT | SELECT DISTINCT CATEGORIA\_ACTIVO AS display\_value, CATEGORIA\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |
| NOMBRE\_ACTIVO | AC\_NOM | SELECT DISTINCT NOMBRE\_ACTIVO as display\_value, NOMBRE\_ACTIVO as return\_value FROM ACTIVO ORDER BY display\_value |
| MARCA\_ACTIVO | AC\_MAR | SELECT DISTINCT MARCA\_ACTIVO AS display\_value, MARCA\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |
| ID\_EDIFICIO | AC\_EDI | SELECT DISTINCT (NOMBRE\_SEDE || ' - ' || NOMBRE\_EDIFICIO) AS display\_value, ID\_EDIFICIO AS return\_value FROM EDIFICIO NATURAL JOIN SEDE ORDER BY display\_value |
| ID\_LOCACION | AC\_LOC | SELECT DISTINCT NOMBRE\_LOCACION AS display\_value, ID\_LOCACION AS return\_value FROM LOCACION ORDER BY display\_value |
| MATERIAL\_ACTIVO | AC\_MAT | SELECT DISTINCT MATERIAL\_ACTIVO AS display\_value, MATERIAL\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |
| COLOR\_ACTIVO | AC\_COL | SELECT DISTINCT COLOR\_ACTIVO AS display\_value, COLOR\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |
| TAMANO\_ACTIVO | AC\_TAM | SELECT DISTINCT TAMANO\_ACTIVO AS display\_value, TAMANO\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |
| COMPONENTES\_ACTIVO | AC\_COM | SELECT DISTINCT COMPONENTES\_ACTIVO AS display\_value, COMPONENTES\_ACTIVO AS return\_value FROM ACTIVO ORDER BY display\_value |

El botón de inserción para este formulario tiene la siguiente sentencia SQL asociada:

INSERT INTO ACTIVO VALUES (9999, TO\_NUMBER(:AC\_COMPR), :AC\_REF, :AC\_CAT, :AC\_NOM, :AC\_MAR, TO\_NUMBER(:AC\_VAL), TO\_NUMBER(:AC\_DEP), TO\_NUMBER(:AC\_FUN), :AC\_EDI, TO\_NUMBER(:AC\_LOC), :AC\_MAT, :AC\_COL, :AC\_TAM, :AC\_COMPO);

*Edición de Activos*



La tercera página de la aplicación es el formulario que se genera cuando un usuario hace clic sobre el botón de editar registro en la tabla de ACTIVO.

Esta página se genera de forma automática por el creador de páginas de APEX. Simplemente se configuraron los campos de texto según las consultas listadas anteriormente.

También, se eliminó el botón de BORRAR para evitar que el personal de esta área elimine activos, lo cual solo le corresponde al personal de Contabilidad.

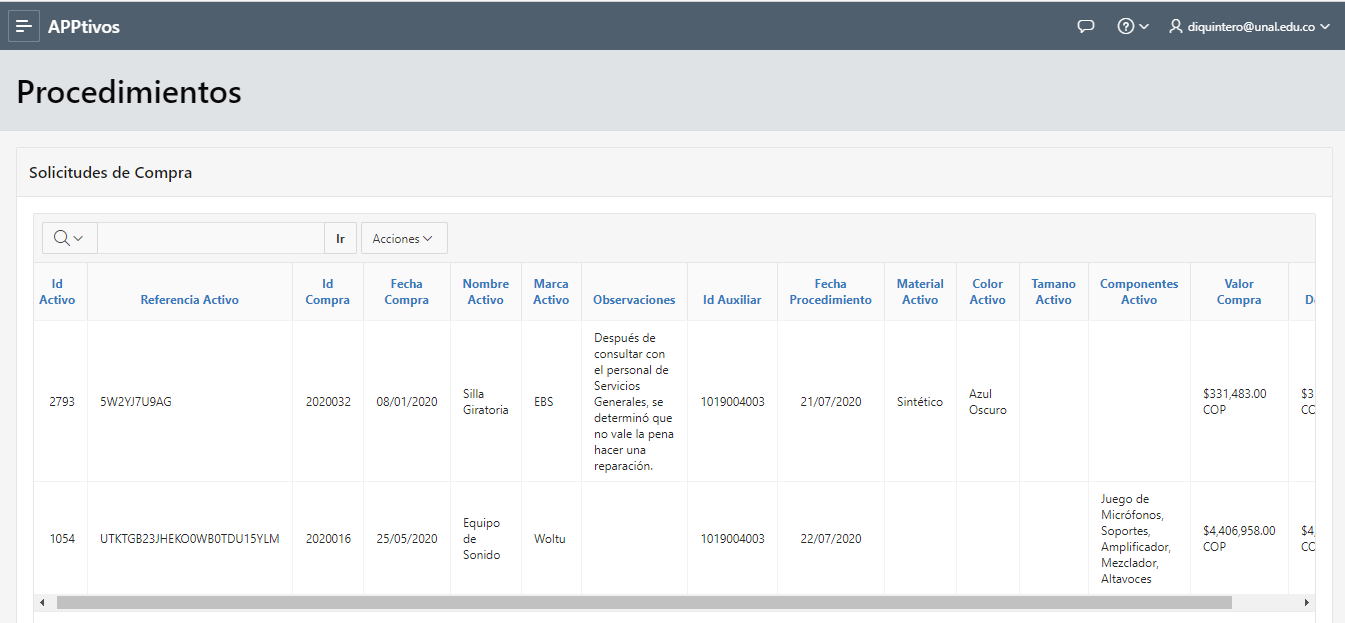
Asimismo se eliminó el botón de CREAR, puesto que ya se había creado el formulario de inserción en la página anterior Gestor de Activos.

*Procedimientos*

El objetivo de esta página es brindar información relevante al área de Compras y Almacén.

Se compone de varias regiones basadas en informes interactivos generados por consultas y una gráfica basada en una tabla.

La primera región muestra una tabla con los activos que después de haber sido revisados por los auxiliares de inventarios y/o de mantenimiento, necesitan ser reemplazados por unos nuevos, convirtiéndose así en solicitudes de compra.



La tabla se generó a partir de la siguiente consulta SQL

SELECT ID\_ACTIVO, REFERENCIA\_ACTIVO, ID\_COMPRA, FECHA\_COMPRA, NOMBRE\_ACTIVO, MARCA\_ACTIVO,

(TO\_CHAR(VALOR\_ACTIVO,'$999,999,999.99') || ' COP') VALOR\_COMPRA,

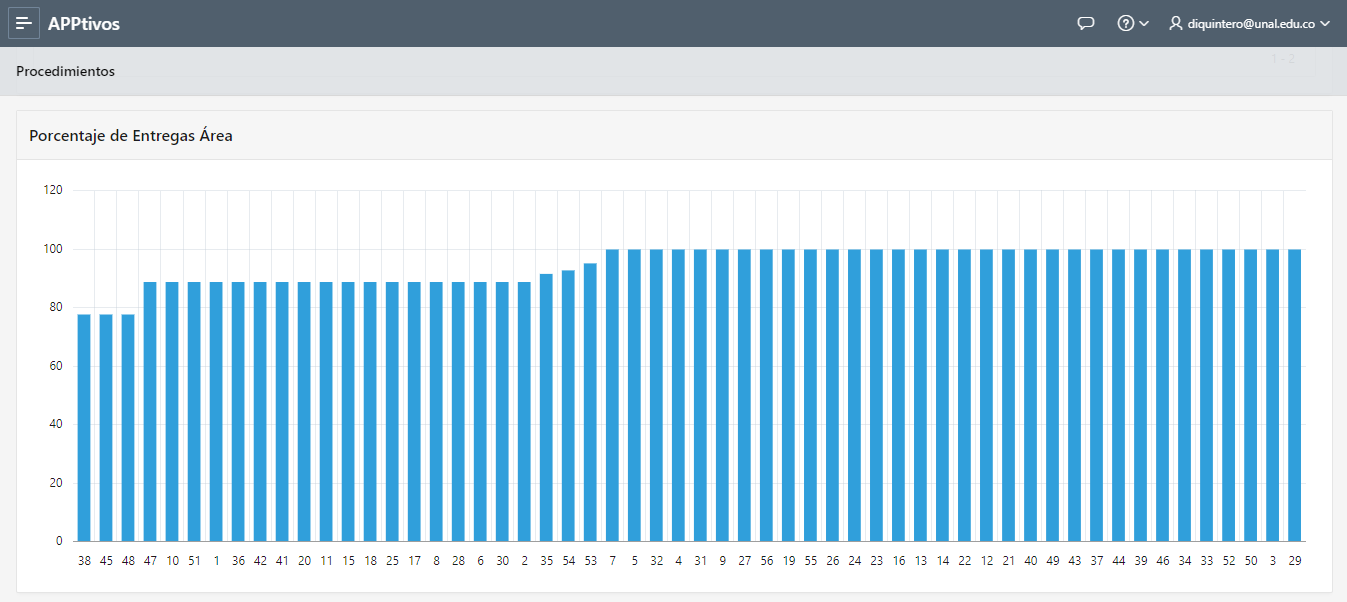
(TO\_CHAR(VALOR\_ACTIVO\_HOY(VALOR\_ACTIVO, DEPRECIACION\_ACTIVO, FECHA\_COMPRA), '$999,999,999.99') || ' COP') VALOR\_DEPRECIADO,

OBSERVACIONES, ID\_AUXILIAR, FECHA\_PROCEDIMIENTO, MATERIAL\_ACTIVO, COLOR\_ACTIVO, TAMANO\_ACTIVO, COMPONENTES\_ACTIVO

FROM ACTIVO NATURAL JOIN COMPRA NATURAL JOIN PROCEDIMIENTO

WHERE ESTADO\_REVISION = 'R'

Luego se presenta una gráfica basada en una tabla que también se encuentra en esta página de la aplicación:



Busca mostrar el porcentaje de puestos asignados según la cantidad de funcionarios que trabajan en un área específica. La universidad tiene 56 áreas, y por ende hay 56 columnas.

Precisamente, la tabla que se relaciona con esta gráfica, es la siguiente:



Ambos objetos se generaron a partir de la siguiente consulta SQL:

WITH puestos\_asignados AS

(SELECT ID\_AREA, COUNT(ID\_AREA) PUESTOS\_ASIGNADOS

FROM (SELECT ID\_FUNCIONARIO FROM ACTIVO WHERE ID\_FUNCIONARIO IS NOT NULL GROUP BY ID\_FUNCIONARIO)

NATURAL JOIN FUNCIONARIO WHERE ID\_AREA IS NOT NULL GROUP BY ID\_AREA),

empleados\_por\_cargo AS

(SELECT ID\_AREA, COUNT(\*) EMPLEADOS\_EXISTENTES

FROM FUNCIONARIO WHERE ID\_AREA IS NOT NULL GROUP BY ID\_AREA)

SELECT puestos\_asignados.ID\_AREA,

AREA.NOMBRE\_AREA,

PUESTOS\_ASIGNADOS,

EMPLEADOS\_EXISTENTES,

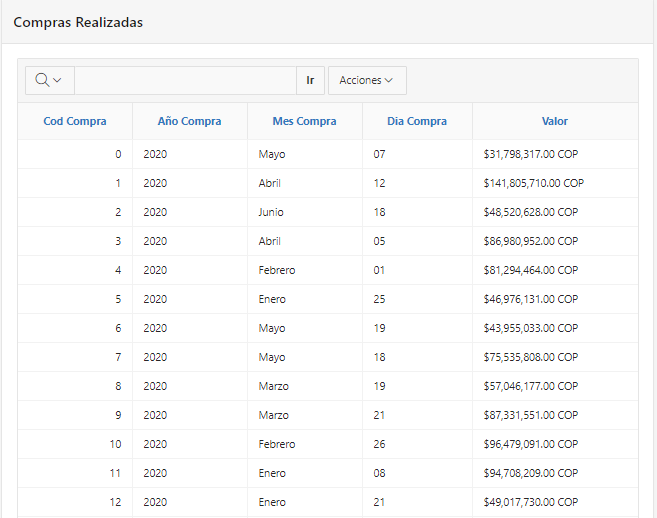
ROUND((PUESTOS\_ASIGNADOS\*100)/EMPLEADOS\_EXISTENTES, 2) PORCENTAJE\_ASIGNADO

FROM (puestos\_asignados INNER JOIN empleados\_por\_cargo

ON puestos\_asignados.ID\_AREA = empleados\_por\_cargo.ID\_AREA) INNER JOIN AREA ON puestos\_asignados.ID\_AREA = AREA.ID\_AREA

ORDER BY PORCENTAJE\_ASIGNADO

La página también cuenta con una tabla que muestra las compras realizadas hasta el momento y que han sido registradas en la base de datos. Por medio de una proyección generalizada, se especifican ciertos formatos para los resultados:



La consulta SQL que genera este informe interactivo, es la siguiente:

SELECT TO\_NUMBER(SUBSTR(TO\_CHAR(id\_compra),5, 3)) cod\_compra,

TO\_CHAR(fecha\_compra, 'yyyy') año\_compra,

TO\_CHAR(fecha\_compra, 'Month') mes\_compra,

TO\_CHAR(fecha\_compra, 'dd') dia\_compra,

(TO\_CHAR(VALOR\_COMPRA,'$999,999,999.99') || ' COP') VALOR

FROM COMPRA

Otro informe interactivo que se decidió incluir es el de una tabla en la que se muestren las compras cuyos montos no coinciden con la suma total de los activos asociados a dicha compra.



La consulta asociada a este objetos es:

WITH sumas AS (SELECT ID\_COMPRA, SUM(VALOR\_ACTIVO) SUMA FROM ACTIVO GROUP BY ID\_COMPRA)

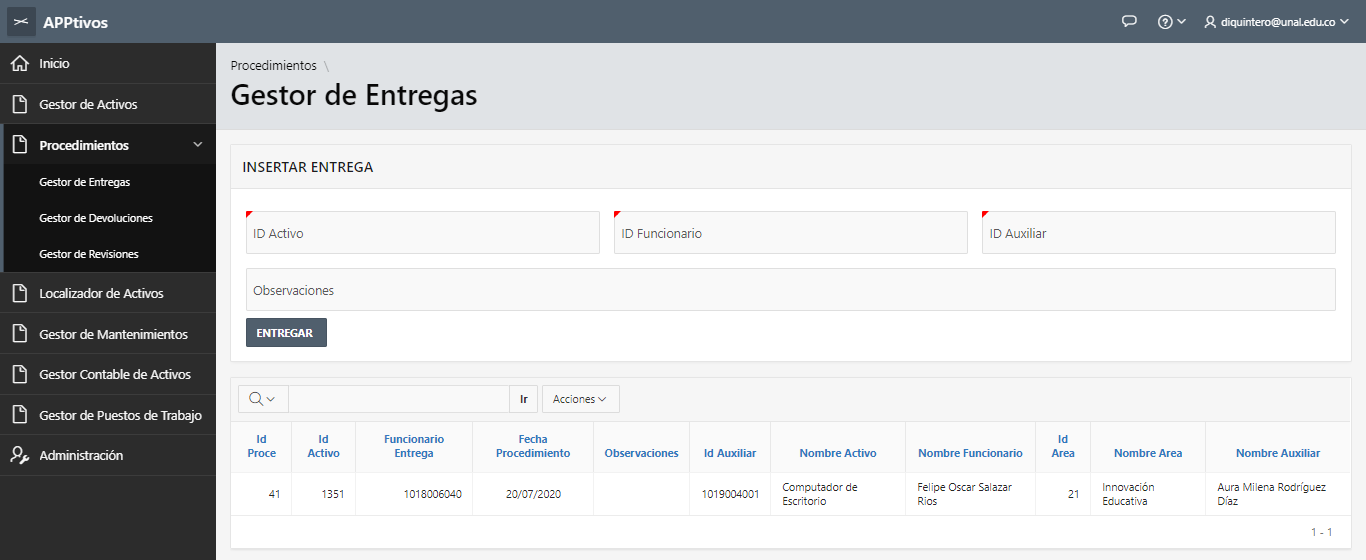
SELECT sumas.ID\_COMPRA, (TO\_CHAR(SUMA, '$999,999,999.00') || ' COP') SUMA\_ACTIVOS, (TO\_CHAR(COM.VALOR\_COMPRA, '$999,999,999.00') || ' COP') VALOR\_COMPRA, (TO\_CHAR(((SELECT VALOR\_COMPRA FROM COMPRA C WHERE C.ID\_COMPRA = sumas.ID\_COMPRA)-SUMA), '$999,999,999.00') || ' COP') DESFASE

FROM sumas INNER JOIN COMPRA COM ON sumas.ID\_COMPRA = COM.ID\_COMPRA

WHERE SUMA < (SELECT VALOR\_COMPRA FROM COMPRA WHERE ID\_COMPRA = sumas.ID\_COMPRA) OR SUMA > (SELECT VALOR\_COMPRA FROM COMPRA WHERE ID\_COMPRA = sumas.ID\_COMPRA)

Las siguientes páginas son específicas para los procedimientos que deben cumplir los auxiliares de inventarios. Se basan en la tabla PROCEDIMIENTO de la base de datos y su raíz en la aplicación es la página PROCEDIMIENTOS, según se puede ver en el menú desplegable.

*Gestor de Entregas*



Se generó un formulario de cuatro campos que deben ingresar los auxiliares cuando van a realizar un proceso de entrega de puesto de trabajo a un funcionario.

|  |  |  |
| --- | --- | --- |
| CAMPO | OBJETO | CONSULTA ASOCIADA |
| ID\_AUXILIAR | PRO\_AUX | SELECT DISTINCT (NOMBRE\_FUNCIONARIO || ' - ID: ' || ID\_FUNCIONARIO) display\_value, ID\_FUNCIONARIO return\_value FROM FUNCIONARIO WHERE TIPO\_AUXILIAR = 'INV' |

El botón ENTREGAR tiene asociada la siguiente sentencia SQL:

INSERT INTO PROCEDIMIENTO VALUES (9999, TO\_NUMBER(:PRO\_ACT), TO\_NUMBER(:PRO\_AUX), 'E', SYSDATE, null, :PRO\_OBS, TO\_NUMBER(:PRO\_FUN));

El informe interactivo tiene asociada la siguiente consulta SQL:

WITH R AS (SELECT ID\_PROCE, ID\_ACTIVO, NOMBRE\_ACTIVO, FUNCIONARIO\_ENTREGA, NOMBRE\_FUNCIONARIO, ID\_AREA, NOMBRE\_AREA, FECHA\_PROCEDIMIENTO, OBSERVACIONES FROM

(SELECT ID\_PROCE, ID\_ACTIVO, FUNCIONARIO\_ENTREGA, OBSERVACIONES, FECHA\_PROCEDIMIENTO, TIPO\_PROCEDIMIENTO FROM PROCEDIMIENTO WHERE TIPO\_PROCEDIMIENTO = 'E') NATURAL JOIN

(SELECT ID\_ACTIVO, NOMBRE\_ACTIVO, ID\_FUNCIONARIO FROM ACTIVO) NATURAL JOIN

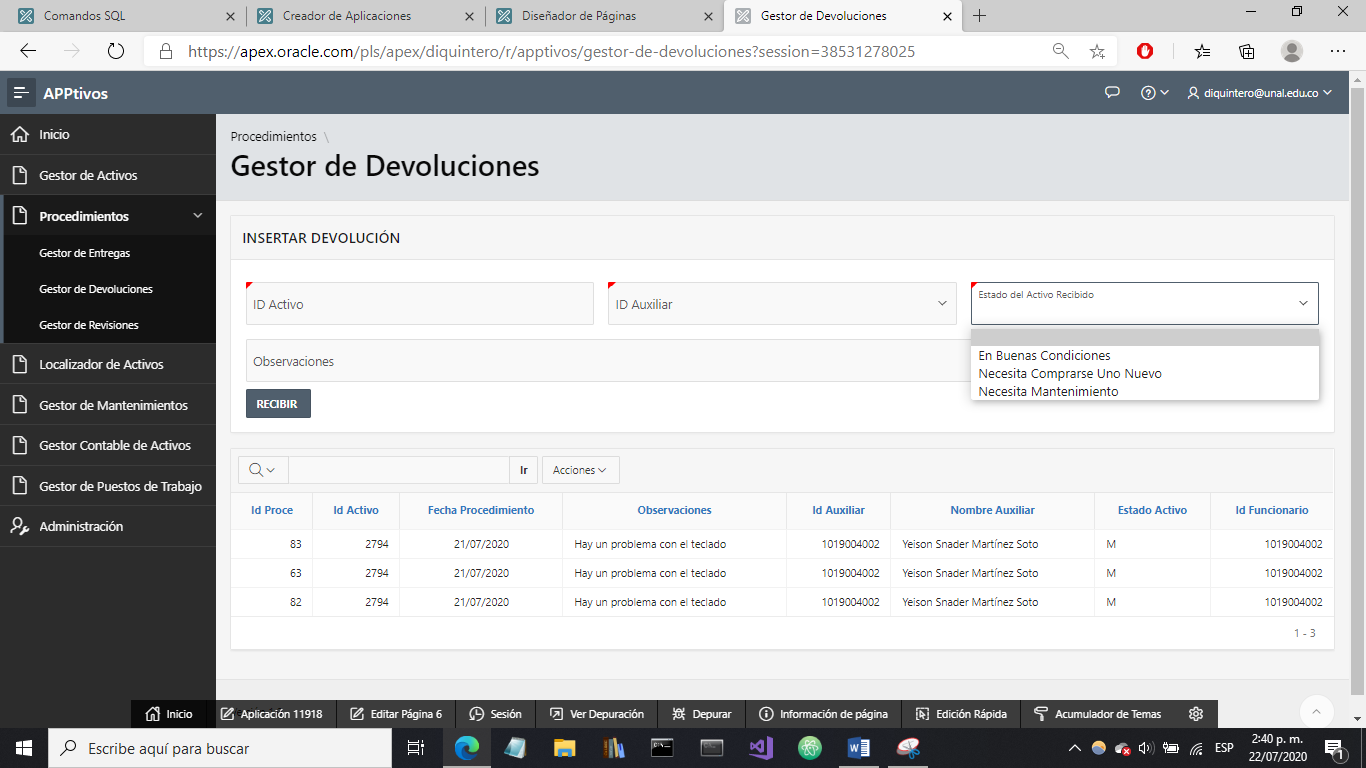
(SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, ID\_AREA FROM FUNCIONARIO) NATURAL JOIN

(SELECT ID\_AREA, NOMBRE\_AREA FROM AREA))

SELECT R.\*, ID\_AUXILIAR, FUNCIONARIO.NOMBRE\_FUNCIONARIO NOMBRE\_AUXILIAR FROM (R INNER JOIN PROCEDIMIENTO ON R.ID\_PROCE = PROCEDIMIENTO.ID\_PROCE) INNER JOIN FUNCIONARIO ON ID\_AUXILIAR = ID\_FUNCIONARIO

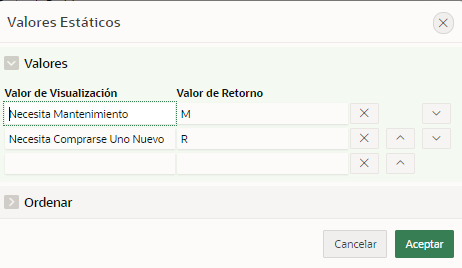
*Gestor de Devoluciones*

La página sirve como registro de las devoluciones de puestos de trabajo que han hecho los funcionarios al retirarse de la compañía.

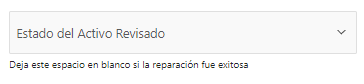


La estructura es muy similar a la página anterior. El campo ID\_AUXILIAR asociado a la lista de selección DEV\_AUX cuenta con la misma consulta ya mostrada para el objeto PRO\_AUX.

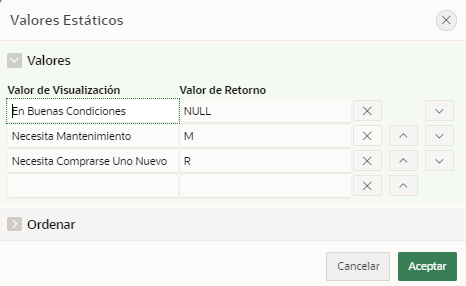
El campo ESTADO\_REVISION se configuró así:



Y la siguiente especificación:



El objeto asociado a ESTADO\_REVISION cuenta con la siguiente configuración:



La sentencia asociada al botón RECIBIR es:

INSERT INTO PROCEDIMIENTO VALUES (9999, TO\_NUMBER(:DEV\_ACT), TO\_NUMBER(:DEV\_AUX), 'P', SYSDATE, :DEV\_EST, :DEV\_OBS, null);

El informe interactivo se genera a partir de la siguiente consulta:

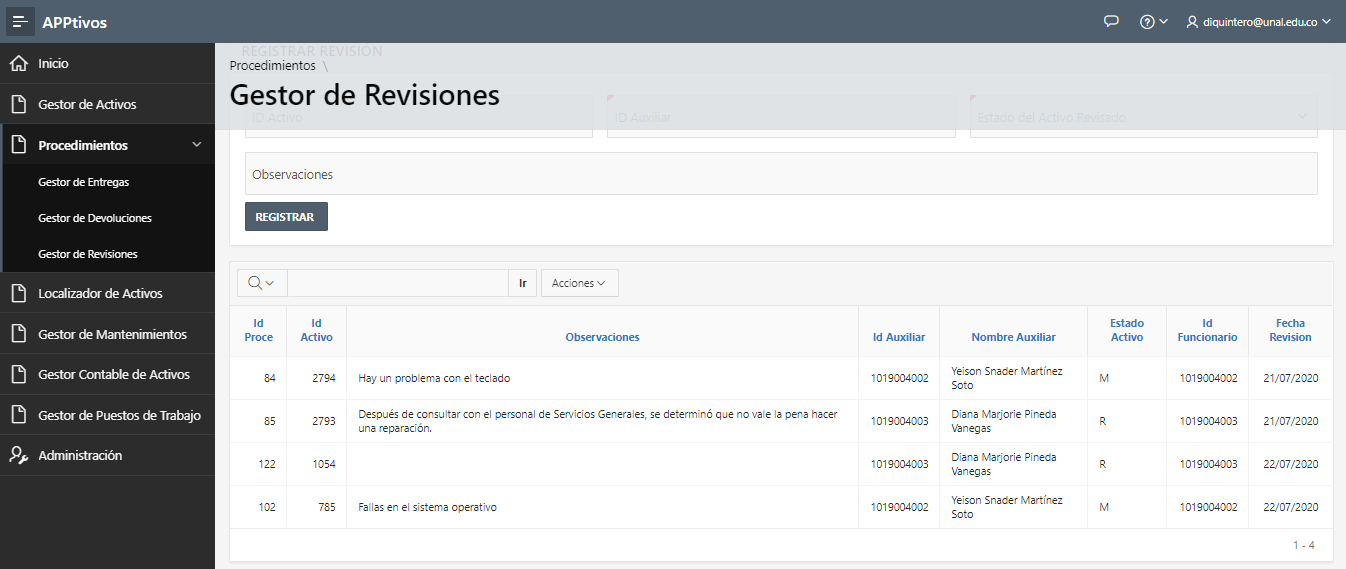
SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_PROCEDIMIENTO, ESTADO\_ACTIVO, OBSERVACIONES, ID\_AUXILIAR, ID\_FUNCIONARIO, NOMBRE\_AUXILIAR

FROM (SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_PROCEDIMIENTO, ESTADO\_REVISION ESTADO\_ACTIVO, OBSERVACIONES, ID\_AUXILIAR, TIPO\_PROCEDIMIENTO FROM PROCEDIMIENTO WHERE TIPO\_PROCEDIMIENTO = 'P') INNER JOIN

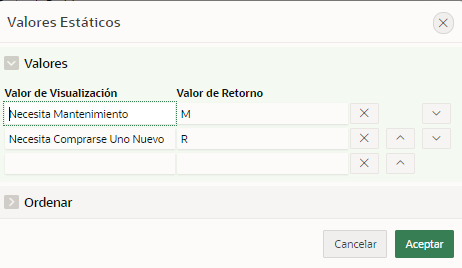
(SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO NOMBRE\_AUXILIAR FROM FUNCIONARIO) ON ID\_AUXILIAR = ID\_FUNCIONARIO

*Gestor de Revisiones*

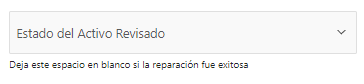
La séptima página en la aplicación guarda la misma estructura que la anterior salvo por el hecho de que las consultas asociadas al botón REGISTRAR y al informe interactivo son ligeramente distintas:



El campo ESTADO\_REVISION se configuró así:



Y la siguiente especificación:



Para el botón REGISTRAR la sentencia es:

INSERT INTO PROCEDIMIENTO VALUES (9999, TO\_NUMBER(:REV\_ACT), TO\_NUMBER(:REV\_AUX), 'R', SYSDATE, :REV\_EST, :REV\_OBS, null);

Para el informe interactivo, la consulta es:

SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_REVISION, ESTADO\_ACTIVO, OBSERVACIONES, ID\_AUXILIAR, ID\_FUNCIONARIO, NOMBRE\_AUXILIAR

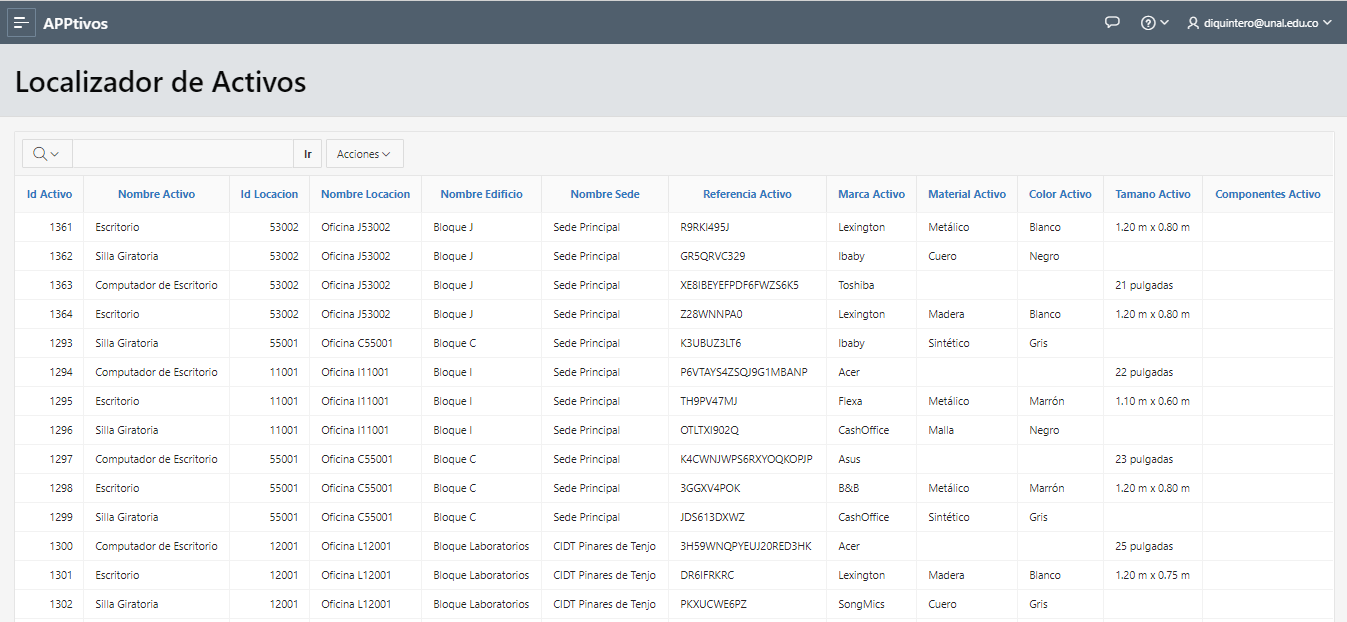
FROM (SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_PROCEDIMIENTO FECHA\_REVISION, ESTADO\_REVISION ESTADO\_ACTIVO, OBSERVACIONES, ID\_AUXILIAR, TIPO\_PROCEDIMIENTO FROM PROCEDIMIENTO WHERE TIPO\_PROCEDIMIENTO = 'R') INNER JOIN

(SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO NOMBRE\_AUXILIAR FROM FUNCIONARIO) ON ID\_AUXILIAR = ID\_FUNCIONARIO

*Gestor Logístico de Activo (Localizador de Activos)*

Se trata de una página con una serie de informes interactivos interesantes para el equipo administrativo de logística, especialmente para el personal de Servicios Generales, con el fin de coordinar traslados de activos cuando se necesite y determinar la disponibilidad de ciertos espacios en caso de eventos o actividades que se tengan que realizar en la universidad:

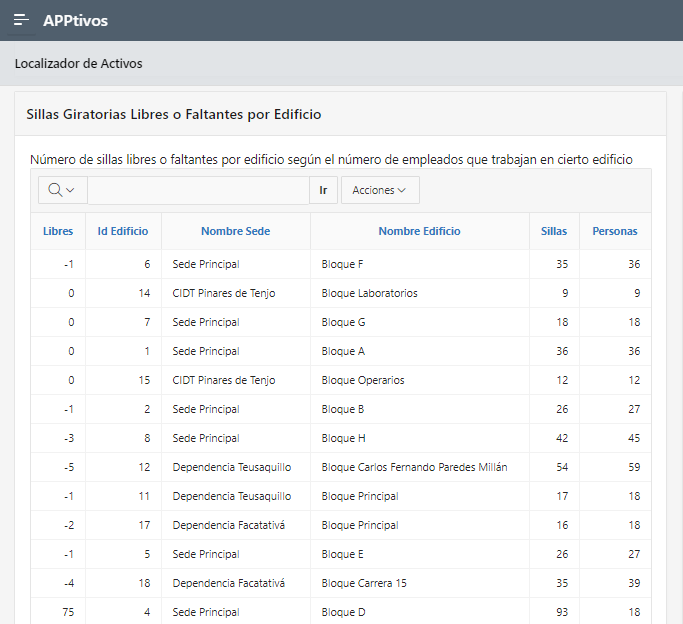
El primer informe es el Localizador de Activos, la cual es una tabla que enfatiza en la ubicación física que tiene cada uno de los activos, apoyándose en operaciones de JOIN para mostrar valores diferentes a las llaves foráneas.



La consulta asociada a este objeto es la siguiente:

SELECT ID\_ACTIVO, NOMBRE\_ACTIVO, ID\_LOCACION, NOMBRE\_LOCACION, NOMBRE\_EDIFICIO, NOMBRE\_SEDE, REFERENCIA\_ACTIVO, MARCA\_ACTIVO, MATERIAL\_ACTIVO, COLOR\_ACTIVO, TAMANO\_ACTIVO, COMPONENTES\_ACTIVO FROM ACTIVO NATURAL JOIN LOCACION NATURAL JOIN EDIFICIO NATURAL JOIN SEDE

Un siguiente informe, presenta el número de sillas libres o faltantes según cuántas haya por edificio en relación a cuántos funcionarios trabajan en dicho edificio. Esto permite dar una idea al área de logística, a cuál edificio deberían dirigirse en caso de necesitar sillas para un evento:



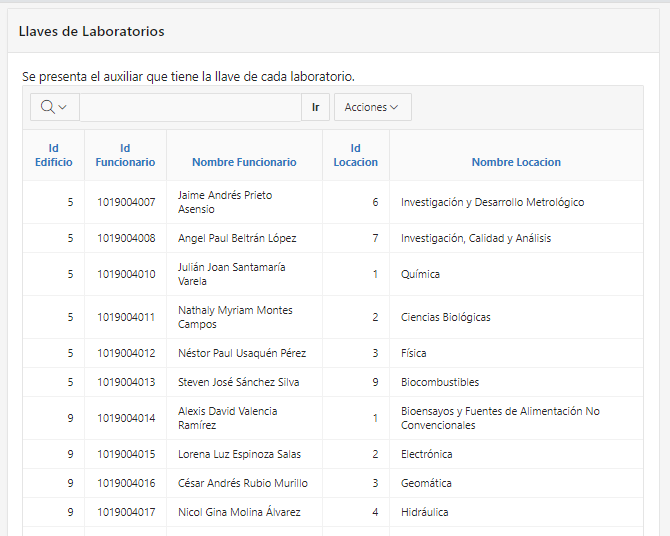
La consulta asociada a este objeto es:

WITH sillas\_por\_edificio AS (SELECT ID\_EDIFICIO, NOMBRE\_SEDE, NOMBRE\_EDIFICIO, COUNT(\*) SILLAS FROM ACTIVO NATURAL JOIN LOCACION NATURAL JOIN EDIFICIO NATURAL JOIN SEDE WHERE NOMBRE\_ACTIVO = 'Silla Giratoria' GROUP BY ID\_EDIFICIO, NOMBRE\_SEDE, NOMBRE\_EDIFICIO),

personas\_por\_edificio AS (SELECT getEdificioByOficina(getOficinaByFuncionario(ID\_FUNCIONARIO)) EDIFICIO, COUNT(\*) PERSONAS FROM FUNCIONARIO WHERE ID\_AREA IS NOT NULL GROUP BY getEdificioByOficina(getOficinaByFuncionario(ID\_FUNCIONARIO)))

SELECT ID\_EDIFICIO, NOMBRE\_SEDE, NOMBRE\_EDIFICIO, SILLAS, PERSONAS, (SILLAS-PERSONAS) LIBRES FROM sillas\_por\_edificio SPE INNER JOIN personas\_por\_edificio PPE ON SPE.ID\_EDIFICIO = PPE.EDIFICIO

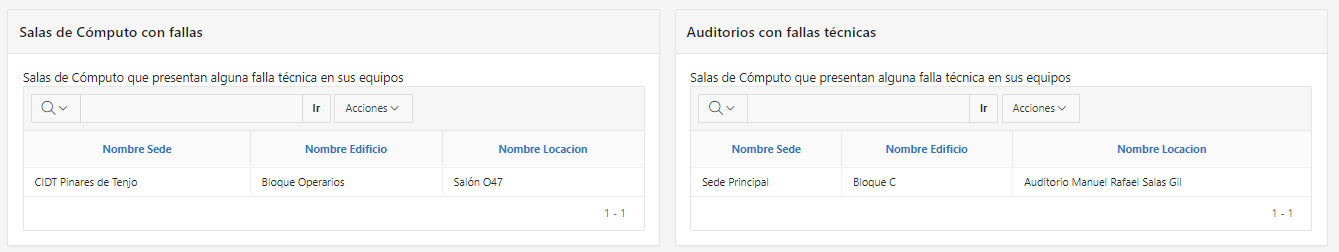
También se decidió incluir la siguiente relación en la que se muestran los auxiliares de laboratorio asociados a cada laboratorio, lo cual podría ser de interés para el área de logística. Esto es, conocer cuál funcionario tiene la llave de cierto laboratorio por ejemplo:



La consulta asociada a este informe es:

SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, ID\_EDIFICIO, ID\_LOCACION, NOMBRE\_LOCACION FROM FUNCIONARIO F INNER JOIN LOCACION L ON F.ID\_FUNCIONARIO = L.ID\_AUXILIAR

Los siguientes dos informes que se generaron y que funcionan prácticamente igual, son dos tablas que muestran las salas de cómputo y los auditorios en los cuales hay computadores o equipos de sonido con fallas, según hayan sido revisados por auxiliares de inventarios. Esto permitirá al equipo de Logística saber cuáles espacios no están las mejores condiciones para albergar un evento dado:



Las consultas asociadas a cada uno de estos dos informes son respectivamente:

WITH locaciones AS (SELECT \* FROM LOCACION NATURAL JOIN EDIFICIO NATURAL JOIN SEDE)

SELECT locaciones.NOMBRE\_SEDE, locaciones.NOMBRE\_EDIFICIO, locaciones.NOMBRE\_LOCACION

FROM (SELECT \* FROM ACTIVO NATURAL JOIN PROCEDIMIENTO NATURAL JOIN (SELECT ID\_EDIFICIO, ID\_LOCACION, NOMBRE\_LOCACION, TIPO\_SALON FROM LOCACION) WHERE ESTADO\_REVISION IN ('M', 'R') AND TIPO\_SALON = 'COM') R

INNER JOIN locaciones ON R.ID\_EDIFICIO = locaciones.ID\_EDIFICIO AND R.ID\_LOCACION = locaciones.ID\_LOCACION

WITH locaciones AS (SELECT \* FROM LOCACION NATURAL JOIN EDIFICIO NATURAL JOIN SEDE)

SELECT locaciones.NOMBRE\_SEDE, locaciones.NOMBRE\_EDIFICIO, locaciones.NOMBRE\_LOCACION

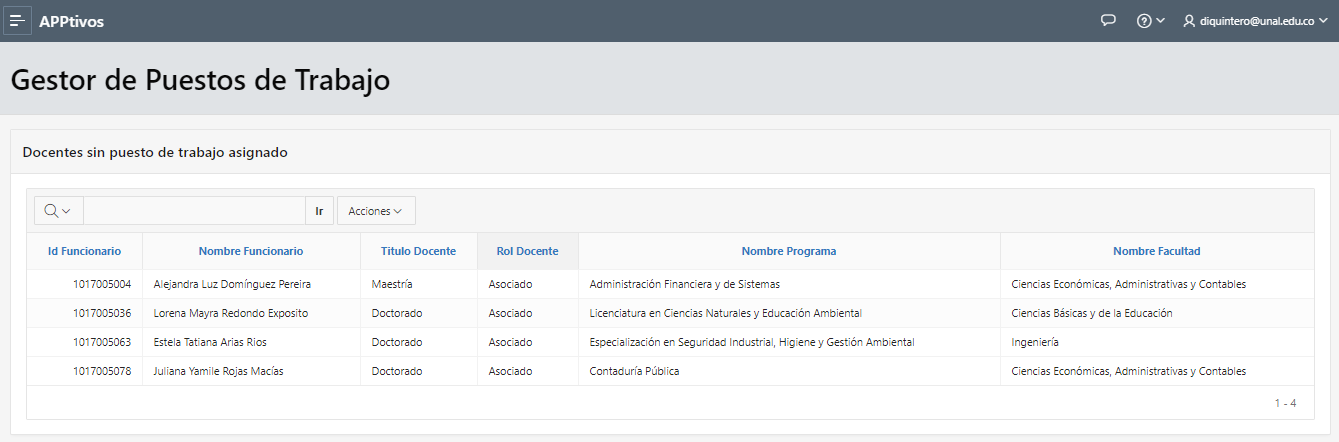
FROM (SELECT \* FROM ACTIVO NATURAL JOIN PROCEDIMIENTO NATURAL JOIN (SELECT ID\_EDIFICIO, ID\_LOCACION, NOMBRE\_LOCACION, TIPO\_SALON FROM LOCACION) WHERE ESTADO\_REVISION IN ('M', 'R') AND TIPO\_SALON = 'AUD') R

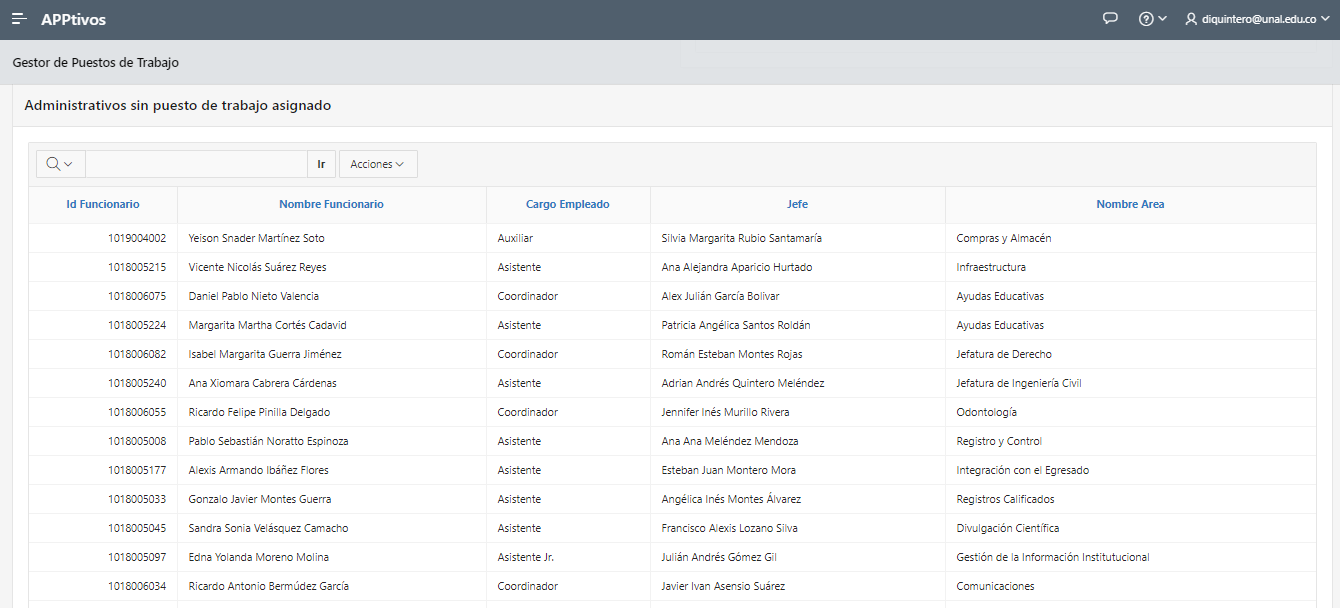
INNER JOIN locaciones ON R.ID\_EDIFICIO = locaciones.ID\_EDIFICIO AND R.ID\_LOCACION = locaciones.ID\_LOCACION

*Gestor de Puestos de Trabajo*

Esta página es de interés para el personal de Gestión Humana y Desarrollo Organizacional. Presenta una serie de informes centrados en saber qué funcionarios no cuentan con un puesto de trabajo formalmente asignado. Verificar si hay funcionarios con un puesto de trabajo que presenta fallas o necesita reemplazarse. Como un registro de posibles cargos disciplinarios si hay evidencia de un uso inadecuado del activo por parte del funcionario.

Dos informes similares presentan los funcionarios sin puestos de trabajo. Uno centrado en los docentes y el otro en el personal administrativo:





Las consultas asociadas a cada informe respectivamente son:

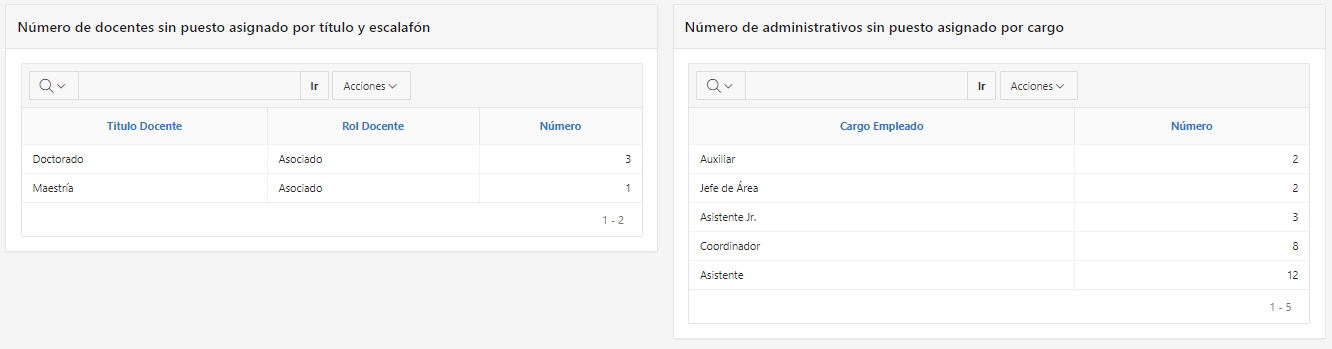
SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO, TITULO\_DOCENTE, ROL\_DOCENTE, NOMBRE\_PROGRAMA, NOMBRE\_FACULTAD FROM FUNCIONARIO NATURAL JOIN FACULTAD NATURAL JOIN PROGRAMA WHERE ID\_FUNCIONARIO NOT IN (SELECT ID\_FUNCIONARIO FROM FUNCIONARIO NATURAL JOIN ACTIVO) AND TIPO\_FUNCIONARIO = 'D'

SELECT R.ID\_FUNCIONARIO, R.NOMBRE\_FUNCIONARIO, R.CARGO\_EMPLEADO, R.JEFE, NOMBRE\_AREA FROM (SELECT F.ID\_FUNCIONARIO, F.NOMBRE\_FUNCIONARIO, F.TIPO\_FUNCIONARIO, F.CARGO\_EMPLEADO, J.NOMBRE\_FUNCIONARIO JEFE, F.ID\_AREA

FROM FUNCIONARIO F INNER JOIN FUNCIONARIO J ON F.ID\_JEFE = J.ID\_FUNCIONARIO) R INNER JOIN AREA A ON R.ID\_AREA = A.ID\_AREA

WHERE R.ID\_FUNCIONARIO NOT IN (SELECT ID\_FUNCIONARIO FROM FUNCIONARIO NATURAL JOIN ACTIVO) AND R.TIPO\_FUNCIONARIO = 'E'

Adicionalmente, se generaron dos informes que muestran la misma información anterior pero clasificada por el título y rol del docente como por el cargo de los administrativos:



Las consultas asociadas son respectivamente:

SELECT TITULO\_DOCENTE, ROL\_DOCENTE, COUNT(\*) NÚMERO FROM FUNCIONARIO

WHERE ID\_FUNCIONARIO NOT IN (SELECT ID\_FUNCIONARIO FROM ACTIVO NATURAL JOIN FUNCIONARIO) AND TIPO\_FUNCIONARIO ='D'

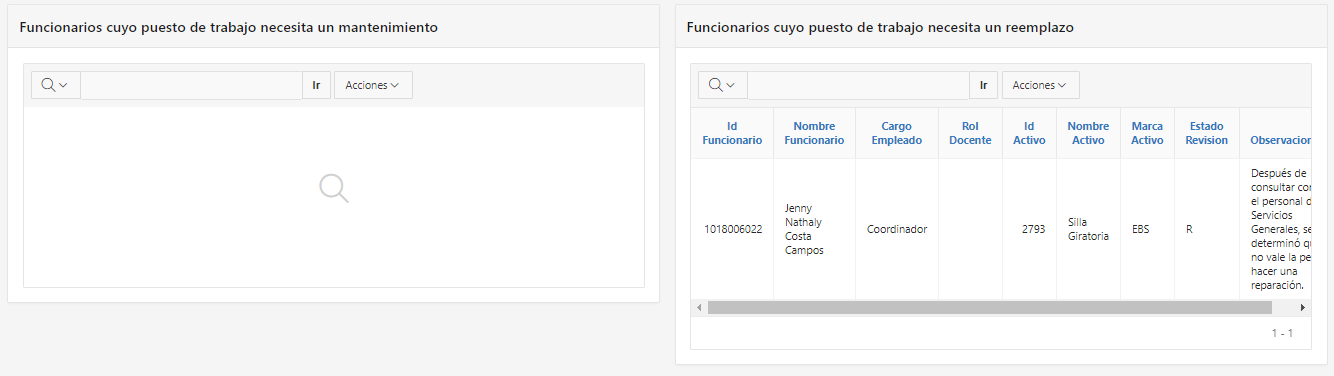
GROUP BY TITULO\_DOCENTE, ROL\_DOCENTE

SELECT CARGO\_EMPLEADO, COUNT(\*) NÚMERO FROM FUNCIONARIO

WHERE ID\_FUNCIONARIO NOT IN (SELECT ID\_FUNCIONARIO FROM ACTIVO NATURAL JOIN FUNCIONARIO) AND TIPO\_FUNCIONARIO ='E'

GROUP BY CARGO\_EMPLEADO ORDER BY NÚMERO

Otros dos informes similares entre sí, muestran los funcionarios cuyo puesto de trabajo presenta un activo que necesita una reparación o que necesita un cambio, es decir, comprarse uno nuevo.



Las consultas asociadas son:

WITH c AS (SELECT ID\_ACTIVO, ID\_FUNCIONARIO FROM ACTIVO NATURAL JOIN PROCEDIMIENTO WHERE ESTADO\_REVISION = 'M')

SELECT DISTINCT F.ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO FROM c INNER JOIN FUNCIONARIO F ON c.ID\_FUNCIONARIO = F.ID\_FUNCIONARIO

SELECT F.ID\_FUNCIONARIO, F.NOMBRE\_FUNCIONARIO, F.CARGO\_EMPLEADO, F.ROL\_DOCENTE, ID\_ACTIVO, NOMBRE\_ACTIVO, MARCA\_ACTIVO, ESTADO\_REVISION, OBSERVACIONES

FROM (SELECT \* FROM ACTIVO NATURAL JOIN PROCEDIMIENTO) R INNER JOIN FUNCIONARIO F ON R.ID\_FUNCIONARIO = F.ID\_FUNCIONARIO

WHERE ESTADO\_REVISION = 'M' OR ESTADO\_REVISION = 'R' AND OBSERVACIONES IS NOT NULL

Finalmente, se presenta un informe en el que se listan posibles casos de descargos, es decir, procesos disciplinarios que deben cumplirse cuando se evidencia un uso inadecuado por parte del usuario de uno de los activos de su puesto de trabajo. El personal de gestión humana encargado de estos procedimientos, inicia el proceso estipulado y determina si aplica o no la sanción.



La consulta asociada es la siguiente:

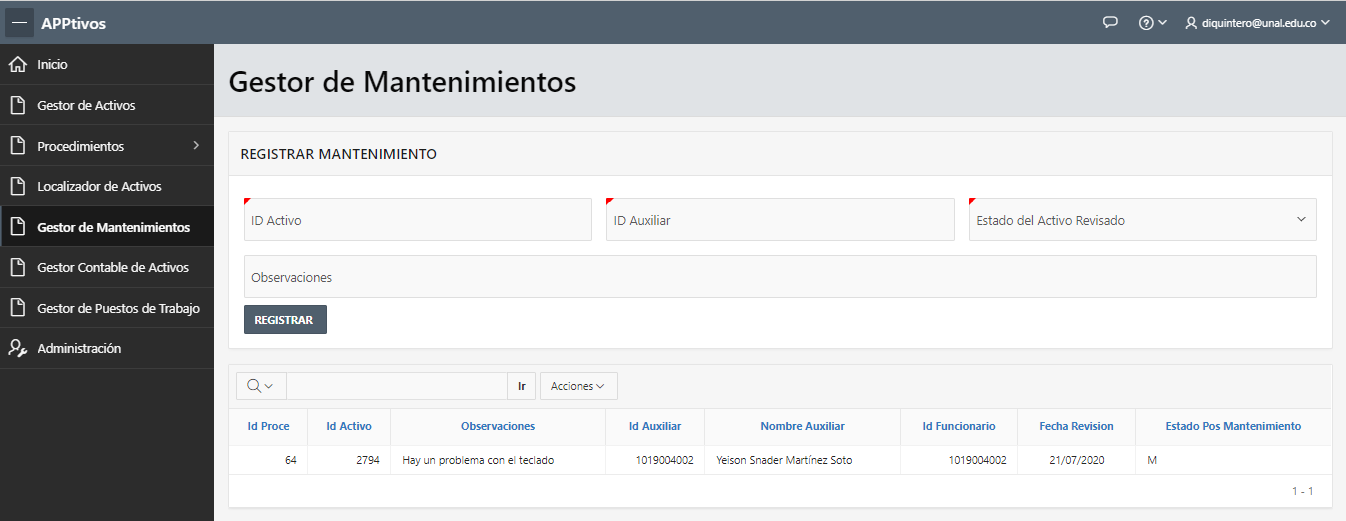
SELECT F.ID\_FUNCIONARIO, F.NOMBRE\_FUNCIONARIO, F.CARGO\_EMPLEADO, F.ROL\_DOCENTE, ID\_ACTIVO, NOMBRE\_ACTIVO, MARCA\_ACTIVO, ESTADO\_REVISION, OBSERVACIONES

FROM (SELECT \* FROM ACTIVO NATURAL JOIN PROCEDIMIENTO) R INNER JOIN FUNCIONARIO F ON R.ID\_FUNCIONARIO = F.ID\_FUNCIONARIO

WHERE (ESTADO\_REVISION = 'M' OR ESTADO\_REVISION = 'R') AND OBSERVACIONES IS NOT NULL AND (TIPO\_PROCEDIMIENTO IN ('R', 'P'))

*Gestor de Mantenimientos*

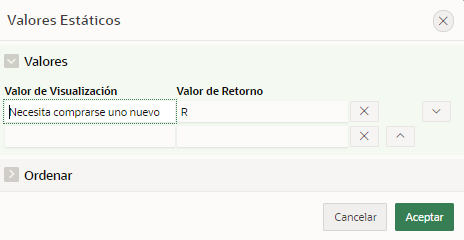
La décima página de la aplicación está dirigida únicamente a los auxiliares de mantenimiento. Se compone de un formulario muy similar al de los otros gestores de procedimientos y dos informes interactivos.



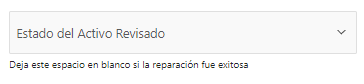
El campo ID\_AUXILIAR es nuevamente una lista de selección con nombre MAN\_AUX y con consulta SQL asociada:

SELECT DISTINCT (NOMBRE\_FUNCIONARIO || ' - ID: ' || ID\_FUNCIONARIO) display\_value, ID\_FUNCIONARIO return\_value FROM FUNCIONARIO WHERE TIPO\_AUXILIAR = 'MAN'

El campo ESTADO\_REVISION asociado al objeto MAN\_EST cuenta con la siguiente configuración:



Y agregando la siguiente especificación:



El botón REGISTRAR incluye la siguiente sentencia:

INSERT INTO PROCEDIMIENTO VALUES (9999, TO\_NUMBER(:MAN\_ACT), TO\_NUMBER(:MAN\_AUX), 'M', SYSDATE, :MAN\_EST, :MAN\_OBS, null);

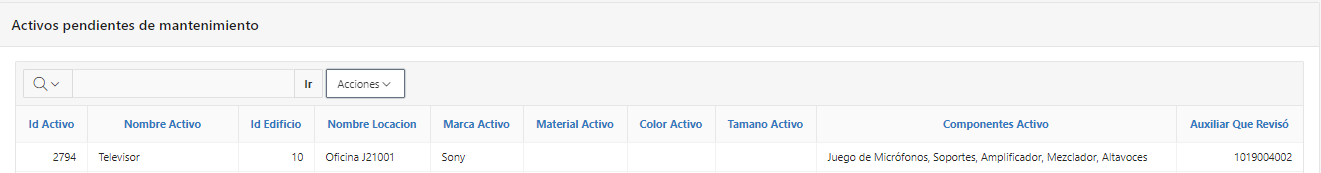
El informe interactivo que va almacenando los registros insertados se construye a partir de la siguiente consulta:

SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_REVISION, ESTADO\_ACTIVO ESTADO\_POS\_MANTENIMIENTO, OBSERVACIONES, ID\_AUXILIAR, ID\_FUNCIONARIO, NOMBRE\_AUXILIAR

FROM (SELECT ID\_PROCE, ID\_ACTIVO, FECHA\_PROCEDIMIENTO FECHA\_REVISION, ESTADO\_REVISION ESTADO\_ACTIVO, OBSERVACIONES, ID\_AUXILIAR, TIPO\_PROCEDIMIENTO FROM PROCEDIMIENTO WHERE TIPO\_PROCEDIMIENTO = 'M') INNER JOIN

(SELECT ID\_FUNCIONARIO, NOMBRE\_FUNCIONARIO NOMBRE\_AUXILIAR FROM FUNCIONARIO) ON ID\_AUXILIAR = ID\_FUNCIONARIO

También se incluye el siguiente informe en el que se muestran los activos pendientes de mantenimiento:



La consulta asociada a este objeto es:

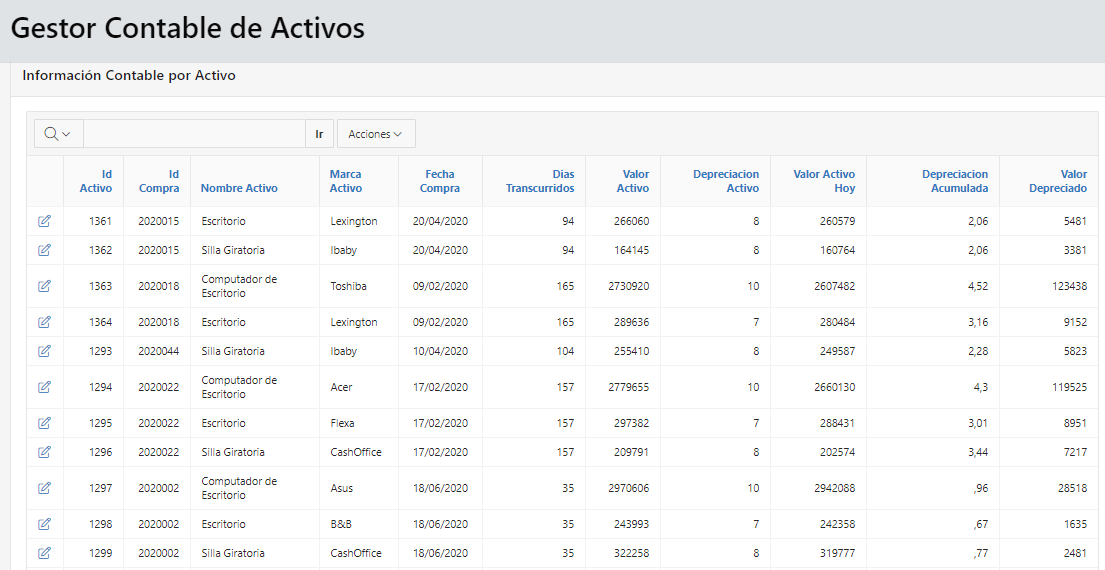
WITH c AS (SELECT A.ID\_ACTIVO, NOMBRE\_ACTIVO, ID\_EDIFICIO, ID\_LOCACION, MARCA\_ACTIVO, MATERIAL\_ACTIVO, COLOR\_ACTIVO, TAMANO\_ACTIVO, COMPONENTES\_ACTIVO, ID\_AUXILIAR FROM ACTIVO A INNER JOIN PROCEDIMIENTO P ON A.ID\_ACTIVO = P.ID\_ACTIVO WHERE ESTADO\_REVISION = 'M')

SELECT ID\_ACTIVO, NOMBRE\_ACTIVO, c.ID\_EDIFICIO, NOMBRE\_LOCACION, MARCA\_ACTIVO, MATERIAL\_ACTIVO, COLOR\_ACTIVO, TAMANO\_ACTIVO, COMPONENTES\_ACTIVO, c.ID\_AUXILIAR AUXILIAR\_QUE\_REVISÓ FROM c INNER JOIN LOCACION ON c.ID\_EDIFICIO = locacion.ID\_EDIFICIO AND c.ID\_LOCACION = LOCACION.ID\_LOCACION

*Gestor Contable de Activos (y Formulario de Edición y Supresión de Activos)*

Esta página está dirigida al personal de Contabilidad y Finanzas. Se compone de dos informes interactivos.

El primero presenta toda la información contable de los activos según las funciones implementadas con PL/SQL.



El informe se generó basado en un formulario para posibilitar la edición, pero se hicieron las siguientes modificaciones.

Se eliminó el botón de CREAR registro.

Se eliminaron todos los campos del formulario generado en la página (12) de edición salvo el campo DEPRECIACION\_ACTIVO, el cual es un atributo que puede ser cambiado por el equipo de este departamento si así lo consideran.



También se conservó el botón SUPRIMIR, en caso de que el personal de este departamento considere que un registro no es un activo de interés para la universidad.

La consulta asociada al informe es:

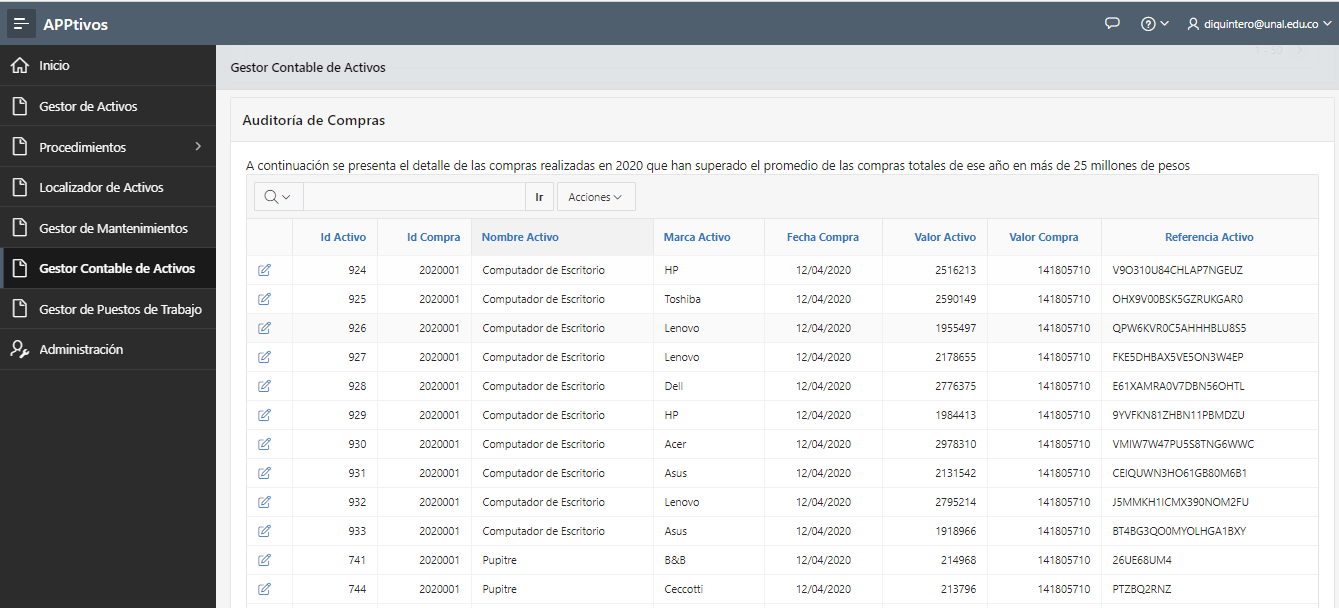
SELECT ID\_ACTIVO ID, ID\_ACTIVO, ID\_COMPRA, NOMBRE\_ACTIVO, MARCA\_ACTIVO, FECHA\_COMPRA, DIAS\_DESDE\_COMPRA(FECHA\_COMPRA) DIAS\_TRANSCURRIDOS, VALOR\_ACTIVO,

DEPRECIACION\_ACTIVO, VALOR\_ACTIVO\_HOY(VALOR\_ACTIVO, DEPRECIACION\_ACTIVO, FECHA\_COMPRA) VALOR\_ACTIVO\_HOY, DEPREC\_TRANS(FECHA\_COMPRA, DEPRECIACION\_ACTIVO) DEPRECIACION\_ACUMULADA,

(VALOR\_ACTIVO - VALOR\_ACTIVO\_HOY(VALOR\_ACTIVO, DEPRECIACION\_ACTIVO, FECHA\_COMPRA)) VALOR\_DEPRECIADO

FROM ACTIVO NATURAL JOIN COMPRA

También se incluyó un segundo informe que se llamó Auditoría de Compras, en la que el personal de Contabilidad y Finanzas pueden examinar en detalle compras que podrían considerar ‘sospechosas’ o por lo menos muy grandes comparadas al promedio del valor de las compras que se hacen regularmente.



La consulta asociada a este informe es:

SELECT ID\_COMPRA, VALOR\_COMPRA, FECHA\_COMPRA, ID\_ACTIVO, REFERENCIA\_ACTIVO, NOMBRE\_ACTIVO, MARCA\_ACTIVO, VALOR\_ACTIVO FROM COMPRA NATURAL JOIN ACTIVO WHERE (VALOR\_COMPRA - (SELECT AVG(VALOR\_COMPRA) FROM COMPRA)) >= 25000000 ORDER BY ID\_COMPRA