ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"COMPATIBILIDAD DE LAS ESTRUCTURAS LÓGICAS Y FÍSICAS ENTRE BASES DE DATOS PARA EL SISTEMA ACADÉMICO DE LA UE RIOBAMBA"

TRABAJO DE TITULACIÓN

TIPO: TÉCNICO – INVESTIGATIVO

Presentado para optar al grado académico de:

INGENIERO EN SISTEMAS INFORMÁTICOS

AUTOR: JORGE WILFRIDO CALUQUÍ CHASIGUANO

TUTOR: ING. JORGE HUILCA PALACIOS

Riobamba – Ecuador

2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA INGENIERÍA EN SISTEMAS

El Tribunal de Trabajos de Titulación certifica que: El trabajo de investigación: Tipo: Técnico – Investigativo "COMPATIBILIDAD DE LAS ESTRUCTURAS LÓGICAS Y FÍSICAS ENTRE BASES DE DATOS PARA EL SISTEMA ACADÉMICO DE LA UE RIOBAMBA", de responsabilidad del señor Jorge Wilfrido Caluquí Chasiguano, ha sido minuciosamente revisado por los Miembros del Tribunal de Trabajo de Titulación, quedando autorizada su presentación.

	FIRMA	FECHA
Dr. Julio Santillán **VICEDECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	……	……
Ing. MSc. Patricio Moreno C. **DIRECTOR ESCUELA DE INGENIERÍA EN SISTEMAS**	……	……
Ing. Jorge Huilca Palacios. **DIRECTOR DE TRABAJO DE TITULACIÓN**	……	……
Dr. Julio Santillán **MIEMBRO DEL TRIBUNAL DEL TRABAJO DE TITULACIÓN**	……	……

“Yo, **Jorge Wilfrido Caluquí Chasiguano** soy responsable de las ideas, doctrinas y resultados expuestos en este trabajo de titulación; y el patrimonio intelectual del trabajo de titulación pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**”**

Jorge Wilfrido Caluquí Chasiguano

DEDICATORIA

Este trabajo está dedicado a mis hermanas Irma y Eli a mi hermano Paquito a mis cuñados José y Jaime a Mi Cuñada Luz América, a todos mis sobrinos y sobrinas, a los cuales siempre les tengo en mi corazón y en especial a mis padres lolita Y Jorgito, que nunca dejaron de apoyarme y creer en mí, que siempre están presentes con su cariño, sus concejos, su cariño y su incondicional confianza me han llevado a culminar una etapa más de mi vida

Jorge Caluquí

AGRADECIMIENTO

Agradezco primeramente a Dios por darme la sabiduría para poder culminar una etapa más en mi vida, por guiar mis pasos en las decisiones del día a día y poder superar todos los obstáculos que se me presenten.

Agradezco a los docentes de la Escuela Superior Politécnica del Chimborazo, por la entrega de sus valiosos conocimientos que fortalecieron mi formación estudiantil, y de esta manera culminar una etapa más de formación académica y personal.

Agradezco a mis padres y hermanos quienes han sido un apoyo incondicional en esta etapa de mi carrera profesional, la cual me ha permitido alcanzar un desarrollo profesional, así como personal.

Jorge Caluquí

TABLA DE CONTENIDO

ÍNDICE DE TABLAS ix

ÍNDICE DE FIGURAS x

INDICE DE GRÁFICOS xii

ÍNDICE DE ABREVIATURAS xiii

RESUMEN xiv

SUMMARY xv

INTRODUCCIÓN 1

CAPITULO I

1. MARCO TEÓRICO 5

1.1. Base de datos 5

1.1.1. *Definición* 5

1.1.2. *Arquitectura de una base de datos* 6

1.2. Sistemas gestores de Base de Datos (SGDB) 8

1.2.1. *Componentes de un SGBD* 9

1.2.2. *Tipos de SGDB* 10

1.1.1.1 *Modelo Lógico* 10

1.1.1.2 *Número de usuarios* 15

1.1.1.3 *Número de sitios* 15

1.2.3. *SGDB en el Mercado* 16

1.2.4. *Oracle* 17

1.2.4.1 *¿Qué es Oracle?* 17

1.2.4.2 *Historia* 18

1.2.4.3 *Arquitectura* 19

1.2.5. *PostgreSQL* 21

1.2.5.1 *¿Qué es PostgreSQL?* 21

1.2.5.2 *Historia* 22

1.2.5.3 *Arquitectura* 23

1.2.6. *Oracle vs. PostgreSQL* 24

1.2.6.1 *Semejanzas* 24

1.2.6.2 *Ventajas y desventajas* 25

1.2.7. *Pl/SQL vs. PL/pgSQL* 27

1.1.1.4 *PL/SQL* 27

1.1.1.5 *PL/pgSQL* 28

CAPÍTULO II

2. MARCO METODOLÓGICO 29

2.1. Generalidades de la Institución 29

2.2. Planificación del proyecto 30

2.3. Análisis de requisitos 30

2.4. Diseño 31

2.4.1. *Creación de ambiente en Oracle* 31

2.4.1.1. *Requerimientos de hardware* 31

2.4.1.2. *Instalación de Oracle 11g XE* *32*

2.4.1.3. *Creación de base de datos* 36

2.4.2. *Creación de ambiente en PostgreSQL* 39

2.4.2.1. *Requerimientos de hardware* 39

2.4.2.2. *Instalación de PostgreSQL* 39

2.4.2.3. *Creación de esquema de base de datos* 45

2.5. Implementación 46

2.6. Comparativo de los SGDB de OracleDataBase XE y PostgreSQL 55

CAPÍTULO III

3. MARCO DE RESULTADOS, ANÁLISIS Y DISCUSIÓN 62

3.1. Diseño de la Investigación 62

3.2. Métodos 62

3.3. Técnicas 62

3.4. Procesamiento de la información 63

3.5. Población y Muestra 64

3.6. Análisis e interpretación de resultados 65

3.6.1. *Adaptabilidad* 65

3.6.2. *Instalabilidad* 66

3.6.3. *Coexistencia* 67

3.6.4. *Capacidad para reemplazar* 68

3.6.5. *Cumplimiento de la portabilidad* 69

3.7. Resultados Generales 70

CONCLUSIONES 73

RECOMENDACIONES 74

GLOSARIO DE TÉRMINOS

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE TABLAS

Tabla 1-2: Requerimiento de migración 31
Tabla 2-2: Funciones agregadas más comunes en Oracle 11g XE 31
Tabla 3-2: Requerimientos de hardware PostgreSQL 39
Tabla 4-2: Migración tipos de datos 46
Tabla 5-2: Migración de instrucciones generales 47
Tabla 6-2: Migración de instrucciones para creación de tablas. 47
Tabla 7-2: Migración de instrucciones para actualización de tablas 48
Tabla 8-2: Migración de instrucciones para creación de vistas. 49
Tabla 9-2: Migración de instrucciones para funciones 49
Tabla 10-2: Migración de instrucciones para inserción de datos 50
Tabla 11-2: Migración de instrucciones para eliminación de datos 50
Tabla 12-2: Migración de instrucciones para consultas 50
Tabla 13-2: Migración de instrucciones para predicados 52
Tabla 14-2: Migración de funciones numéricas 52
Tabla 15-2: Migración de funciones caracter 52
Tabla 16-2: Migración de funciones fecha y hora 53
Tabla 17-2: Migración de funciones relacionadas con null 54
Tabla 18-2: Migración de funciones agrupadas 54
Tabla 19-2: Comparativa de creación de tabla 55
Tabla 20-2: Comparativa de estructura de inserción de datos 56
Tabla 21-2: Comparativa de estructura de actualización de datos 57
Tabla 22-2: Comparativa de estructura de eliminación de datos 57
Tabla 23-2: Comparativa de una función. 58
Tabla 24-2: Comparativa de una vista. 58
Tabla 25-3: Indicador adaptabilidad 65
Tabla 26-3: Indicador instalabilidad 66
Tabla 27-3: Indicador coexistencia 67
Tabla 28-3: Indicador capacidad para reemplazar 68
Tabla 29-3: Indicador cumplimiento de la portabilidad 69
Tabla 30-3: Resumen parámetro compatibilidad 70

ÍNDICE DE FIGURAS

Figura 1-1: Arquitectura de base de datos 6
Figura 2-1: Funciones de los actores de base de datos 7
Figura 3-1: Base de datos simplificada 8
Figura 4-1: Funciones de un SGBD 9
Figura 5-1: Funciones de un SGBD 9
Figura 6-1: Tipos de SGBD 10
Figura 7-1: Modelo de base de datos Jerárquico 11
Figura 8-1: Modelo de base de datos Jerárquico 12
Figura 9-1: Estructura Relacional 13
Figura 10-1: Modelo lógico orientado a objetos 14
Figura 11-1: Bases de datos en el mercado 17
Figura 12-1: Historia de Oracle 18
Figura 13-1: Estructuras físicas y lógicas de Oracle 11g XE 19
Figura 14-1: Estructuras físicas y lógicas de PostgreSQL 20
Figura 15-1: Historia de PostgreSQL 22
Figura 16-1: Estructuras físicas y lógicas de PostgreSQL 23
Figura 17-1: Semejanzas Oracle y PostgreSQL 25
Figura 18-1: Ventajas de PostgreSQL sobre Oracle 26
Figura 19-1: Ventajas de Oracle sobre PostgreSQL 26
Figura 20-2: Planificación de migración de Oracle 11g XE a PostgreSQL 30
Figura 21-2: Pantalla de preparación de instalación Oracle 11g XE 32
Figura 22-2: Pantalla de preparación de bienvenida Oracle 11g XE 32
Figura 23-2: Pantalla términos de la licencia Oracle 11g XE 33
Figura 24-2: Pantalla términos de la licencia Oracle 11g XE 33
Figura 25-2: Pantalla especificación de contraseñas Oracle 11g XE 34
Figura 26-2: Pantalla resumen Oracle 11g XE 34
Figura 27-2: Pantalla resumen Oracle 11g XE 35
Figura 28-2: Pantalla finalización de instalación Oracle 11g XE 35
Figura 29-2: Pantalla promt Oracle 11g XE 36
Figura 30-2: Pantalla login Oracle 11g XE 36
Figura 31-2: Pantalla creación Oracle 11g XE 36
Figura 32-2: Pantalla asignación de privilegios en Oracle 11g XE 36
Figura 33-2: Pantalla resumen de creación de esquema en Oracle 11g XE 37
Figura 34-2: Pantalla SQL Developer para Oracle 11g XE 37
Figura 35-2: Pantalla conexión de base de datos Oracle 11g XE 38
Figura 36-2: Pantalla jerarquía estructural de base de datos Oracle 11g XE 38
Figura 37-2: Pantalla de inicio de instalación PostgreSQL 39
Figura 38-2: Pantalla de especificación de directorio PostgreSQL 40
Figura 39-2: Pantalla de ingreso de contraseña PostgreSQL 40
Figura 40-2: Pantalla de selección de puerto PostgreSQL 41
Figura 41-2: Pantalla de configuración regional PostgreSQL 41
Figura 42-2: Pantalla de resumen de instalación PostgreSQL 42
Figura 43-2: Pantalla de progreso de instalación PostgreSQL 42
Figura 44-2: Pantalla de fin de instalación PostgreSQL 43

Figura 45-2: Pantalla de inicio de aplicación PostgreSQL 43
Figura 46-2: Pantalla de login PostgreSQL 44
Figura 47-2: Pantalla de inicio de aplicación PostgreSQL 44
Figura 48-2: Pantalla de jerarquía estructural de base de datos PostgreSQL 45
Figura 49-2: Pantalla de creación de esquema PostgreSQL 45

INDICE DE GRÁFICOS

Gráfico 1-3: Indicador adaptabilidad 65
Gráfico 2-3: Indicador instalabilidad 66
Gráfico 3-3: Indicador coexistencia 67
Gráfico 4-3: Indicador capacidad para reemplazar 69
Gráfico 5-3: Indicador cumplimiento de la portabilidad 70
Gráfico 6-3: Resumen parámetro compatibilidad 71

ÍNDICE DE ABREVIATURAS

ACID	Atomicidad, Consistencia, Aislamiento y Durabilidad
ANSI-SPARC	American National Standard Institute - Standards Planning and Requirements Committee
DBA	Administrador de Base de Datos
IMS	Information Management System - Sistema de Administración de Información
ISO	Organización Internacional de Normalización
PL/pgSQL	Procedural Language / PostgreSQL Structured Query Language
POO	Programación orientada a objetos
SDK	Software development kit
SGBD	Sistema Gestor de Bases de Datos
SGBO	Sistemas de Gestión de Bases de Objetos
SQL	Structured Query Language - Lenguaje de consulta estructurada

RESUMEN

El gran volumen de información ha provocado una búsqueda de tecnologías, que faciliten el manejo de dicha información y minimicen el impacto económico. Es así que, para el sistema académico VENUSPRO10 de Unidad Educativa Riobamba se proyectó un crecimiento acelerado de la información recopilada, el tamaño de esta información sobrepasaría los límites de almacenamiento del motor de base de datos Oracle 11g XE; lo que generó la necesidad de una migración de estructuras físicas y lógicas hacia el motor de base de datos PostgreSQL; para ello fue necesario evaluar la compatibilidad de las estructuras entre ambos sistemas de gestión de bases datos (SGDB), esto determinó la complejidad en la migración tanto de estructuras como datos; permitió el establecimiento de un estándar que facilita el traslado de la base de datos del sistema, además permitió la detección de afectaciones que pueda sufrir el sistema. Esta investigación fue de tipo técnico – investigativo lo que permitió establecer una metodología adecuada para la migración; estableciendo un estudio de compatibilidad para ofrecer un rango de afectación mínima tanto en los datos, como en el aplicativo. Para ello se determinaron parámetros de comparación, según el numeral 6.6 de la norma ISO/IEC 9126-1. Los resultados obtenidos fueron: adaptabilidad: 42,95%, portabilidad: 100%, coexistencia: 100%, capacidad para reemplazar: 63,08%, instalabilidad de las estructuras: 90%. Es así que la migración ha alcanzado un promedio de 79,21% de compatibilidad; la dificultad en la migración se presenta en los tipos de datos, como en las estructuras de la base de datos del sistema, mientras que los datos migrados alcanzaron niveles satisfactorios, los datos trasladados no necesitaron de ningún artífice, reafirmando la equivalencia entre los tipos de datos de Oracle 11g XE y PostgreSQL.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <INFORMÁTICA>, <GESTORES DE BASES DE DATOS>, <COMPATIBILIDAD>, <ORACLE 11G (SOFTWARE)>, <POSTGRESQL (SOFTWARE)>, <ORACLE XE (SOFTWARE)>, <SISTEMAS DE GESTIÓN DE BASES DATOS (SGBD)>.

SUMMARY

Having an enormous amount of information has led to search technology. It facilitates the management of that information and minimizes the economic impact. So, for the VENUSPRO10 academic system of the Unidad Educativa Riobamba, an accelerated grown of the information collected was predicted. The size of this information would exceed the data base engine storage Oracle 11g XE. It generated the need of moving the physical and logical structures towards the database engine PostgreSQL. For this, it was necessary to evaluate the compatibility of the structures between the two database management systems (SGDB). This determined the complexity in the migration of both structures and data, it allowed the establishment of a standard that facilitates the transfer of the database of the system and besides, it allowed the detection of damages that the system could suffer. The current research was of a technical - investigative nature, which allowed establishing an adequate methodology for migration and thereby, establishing a compatibility study to offer a range of minimum impact on both the data and the application. For this, parameters of comparison were determined, according to number 6.6 of ISO/IEC 9126-1. The results obtained were: adaptability: 42.95%, portability: 100%, coexistence: 100%, capacity to replace: 63.08% instability of structures: 90%. Thereby, the migration has reached an average of 79.21% compatibility; the difficulty in migration is presented in the types of data and in the structures of the databases of the systems as well. While the migrated data reached satisfactory levels, the transferred data did not need any artificer; reaffirming the equivalence between Oracle 11g XE and PostgreSQL data types.

KEY WORDS: <TECHNOLOGY AND ENGINEERING SCIENCES>, <COMPUTING> <DATABASE MANAGERS>, < COMPATIBILITY>, <ORACLE 11G (SOFTWARE)> <POSTGRESQL (SOFTWARE)>, <ORACLE XE (SOFTWARE)>, <DATABASE MANAGEMENT SYSTEMS (SGBD)>

INTRODUCCIÓN

El constante crecimiento de las actividades sociales, conjuntamente con la necesidad de archivar información que ayude a corroborar dichas actividades, han provocado una notable evolución en los sistemas de almacenamiento.

Las instituciones se han visto obligadas a incorporar soluciones tecnológicas que faciliten el manejo de información relevante, a través de soluciones informáticas que incorporan en su arquitectura motores de bases de datos, los cuales pueden ser libres o comerciales.

La Unidad Educativa Riobamba posee un sistema de académico propio enlazado a un SGBD Oracle 11g XE, este gestor de base de datos está a punto de alcanzar los límites de almacenamiento permitido para la versión Express Edition, orillando a esta institución a realizar una migración al motor de base de datos PostgreSQL.

La parte introductoria de la investigación comprende: Formulación general del trabajo de titulación, antecedentes, justificación de la investigación, objetivos generales y específicos

El Capítulo I Marco Teórico, comprende el estudio de los conceptos de base de datos, definición de arquitecturas, y tipos de sistemas gestores de bases de datos, un análisis de los SGDB en el mercado, que nos permita evaluar si la selección del motor de base de destino, es la más adecuada, y las especificaciones de las bases de datos seleccionadas para la migración.

En el Capítulo II se hace una referencia a la manera en cómo se procedió para el establecimiento de los ambientes para la migración, la instalación de los motores de base de datos Oracle y PostgreSQL, así como las reglas que se requirieron para realizar la migración de tablas, tipos de datos, vistas, procedimientos, funciones y datos.

El Capítulo III está orientado a la evaluación de la migración de las estructuras verificando la compatibilidad de las mismas, para ello se realizó la evaluación de todas las estructuras a ser migradas de los módulos Kernel, Registration y Score del Sistema VenusPRO10.

FORMULACIÓN GENERAL DEL TRABAJO DE TITULACIÓN

Antecedentes

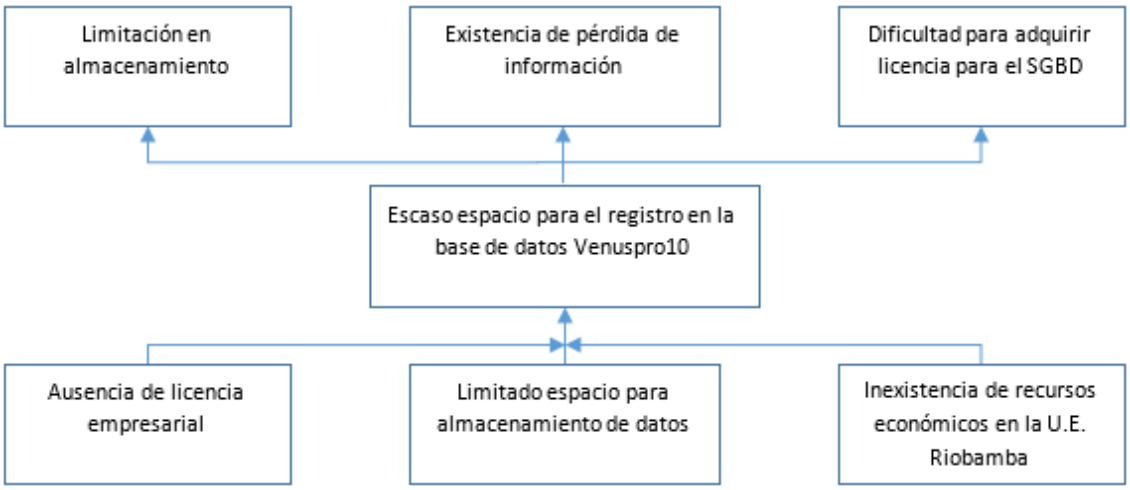
El avance tecnológico ha permitido el crecimiento de la sociedad, tanto en ámbito social como económico, es por ello que hoy en día las empresas, en el ámbito público o privado requieren

almacenar la información de sus actividades para obtener un manejo eficiente de sus datos y transacciones, facilitando los procesos y consultas para dar un servicio de calidad a sus clientes; estas experiencias ayudan a la toma decisiones en base a datos.

Los Sistemas Gestores de Base de Datos (SGBD), permiten el almacenamiento, modificación, eliminación y presentación de los datos en una forma rápida y eficiente. Existe una gran variedad de SGBD en el mercado siendo SGBD libres: PostgreSQL Licencia BSD, Firebird PUBLIC LICENSE Versión 1.0., SQLite Licencia Dominio Público, DB2 Express-C, Apache Derby; SGBD comerciales: Advantage Database, Microsoft SQL Server, Nexus BD, Oracle, Sybase ASE.

La Unidad Educativa Riobamba posee un sistema de académico propio enlazado a un SGBD Oracle XE; la proyección de los datos manejados por este sistema alcanzara los límites de la versión gratuita, por lo cual es necesario un estudio que permita establecer la complejidad en la migración de los datos al SGBD libre PostgreSQL; y con ello determinar el grado de seguridad y el nivel de afectación de las estructuras lógicas y físicas, sus datos y su aplicativo.

Formulación del problema



Realizado por: Caluquí Jorge, 2018

La Unidad Educativa Riobamba (Riobamba) administra los datos de estudiantes mediante el sistema académico VenusPro10 el cual cuenta con acceso a un SGBD Oracle con versión XE; se proyecta que el incremento de sus datos sobrepasara los niveles permitidos por la versión Express, lo que genera un escaso espacio para el registro en la base de datos, la inexistencia de recursos económicos, así como la ausencia de una licencia empresarial.

El limitado espacio, generaran perdida de información, limitación en el almacenamiento y dificultad en la adquisición de la licencia para el SGBD, por lo que se ha generado la necesidad

de migrar a un SGBD libre; dicha migración debe proveer las garantías necesarias sobre la información para la no afectación de la misma.

Sistematización del problema

Analizando la necesidad de un nuevo SGBD libre que pueda brindar estabilidad, confiabilidad, integridad referencial, cumpla con el conjunto de características ACID acrónimo de Atomicidad, Consistencia, Aislamiento y Durabilidad para los SGBD; se ha considerado al SGBD PostgreSQL para la migración.

Mediante la evaluación de compatibilidad de las estructuras físicas y lógicas de los SGBD PostgreSQL y OracleDataBase se determinarán porcentajes de afectación y seguridad en la migración de los datos.

Justificación del trabajo de grado

Justificación teórica

SGBD: Permite mantener controlado el acceso a los datos asegurar la integridad, acceso concurrente nos permiten recuperar los datos tras un fallo del sistema además nos da la posibilidad de realizar copias de seguridad. Tanto las bases de datos como los sistemas deben ser correctamente gestionados ya que son la esencia de las instituciones o empresas ya sea públicas o privadas. (Sanchez, 2012)

Oracle XE: Proporciona una gestión eficiente, confiable y segura de datos a nivel empresarial, así como aplicaciones transaccionales de carácter crítico, almacenamiento de datos mediante consultas intensivas, cargas de trabajo masivas, además todas las transacciones de Oracle cumplen con las propiedades básicas de una transacción de base de datos, conocida como propiedades ACID. (Oracle, 2015)

PostgreSQL: Es un potente sistema de base de datos, de código abierto. Cuenta con más de 15 años de desarrollo activo y posee una arquitectura comprobada obteniendo una reputación solida debido a su fiabilidad, integridad de datos y corrección. Se ejecuta en varios sistemas operativos, asi como los sitemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows.

Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, combinaciones, vistas y procedimientos almacenados, posee herencia entre tablas, copias de seguridad en caliente, múltiples métodos de autenticación, alta concurrencia.

Al realizar el estudio de compatibilidad entre los SGDB de Oracle y Postgres se podrá determinar un rango de afectación a cada uno de los objetos físicos y lógicos de los SGDB, el impacto a sus datos, así como al aplicativo, los cuales nos ayudaran a decidir si es prudente realizar dicha migración sin tener mayores consecuencias.

Justificación aplicativa

El sistema VenusPro10 cuenta con acceso OracleDatBase XE, el crecimiento de sus datos ha generado necesidades de migración a un SGBD libre, para ello se utilizará PostgreSQL, el cual posee características funcionales similares a OracleDatBase XE en lo que respecta al manejo y almacenamiento de los datos.

La migración se la realizará mediante la construcción de objetos físicos, los cuales son: Tablas, vistas, procedimientos, funciones. Los objetos lógicos son: Estructuras de almacenamiento, usuarios, permisos, y datos que serán realizados en los SGBD de estudio.

Esta evaluación determinará la complejidad en la migración de estructuras físicas y lógicas, así como sus datos, lo cual permitirá establecer un estándar que facilite el traslado desde OracleDatBase XE a PostgreSQL, además nos permitirá detectar las posibles afectaciones que pueda sufrir el sistema académico y su impacto en su utilización.

Objetivos

Objetivos Generales

Evaluar la compatibilidad de las estructuras lógicas y físicas entre SGBD OracleDataBase XE y PostgreSQL para el Sistema Académico de la UE Riobamba

Objetivos específicos

Realizar un análisis comparativo entre los SGDB de OracleDataBase XE y PostgreSQL

Analizar las estructuras físicas y lógicas de los SGDB de OracleDataBase XE y PostgreSQL

Construir las Estructuras Lógicas y Físicas de los SGDB de PostgreSQL a partir de la base de datos del Sistema Académico (Oracle XE), estableciendo parámetros y herramientas de comparación.

Evaluar un rango de afectación a las estructuras Físicas y lógicas de los SGBD.

Evaluar la afectación en el Sistema Académico al interactuar con el SGDB PostgreSQL.

CAPITULO I

1. MARCO TEÓRICO

1.1. Base de datos

La creciente necesidad de almacenamiento, interacción, explotación y compartición de la información ha generado una capacidad de acceso gigantesca, esto gracias a la diversidad de base de datos que van desde los tradicionales SGBDs (Sistemas de gestión de base de datos) hasta las más actualizadas que se relacionan con la WEB.

El termino Base de Datos fue empleado por primera vez en un simposio celebrado en California en los años 60.

1.1.1. Definición

Una base de datos es una colección de datos relacionados. Por datos, nos referimos a hechos conocidos que pueden ser grabados y que tienen un significado implícito. Por ejemplo, considere los nombres, teléfono números y direcciones de las personas que conoce. Es posible los datos fueran grabados en una libreta de direcciones indexada o puede haberla almacenado en un disco duro, computadora personal y software como Microsoft Access o Excel. Esta colección de datos relacionados con un significado implícito es una base de datos. (Ramez & Shamkant, 2007)

Podemos considerar las siguientes propiedades implícitas para una base de datos

- Representa aspectos de la realidad
- Identifica datos de una colección de forma lógica y coherente
- Establece un propósito específico
- Posee usuarios y aplicaciones con un interés común

Bajo las directrices informáticas podemos decir que una base de datos es un “almacén” el cual nos permite archivar de forma organizada cantidades de considerables de información en discos que permiten el acceso de forma directa o mediante un conjunto de aplicaciones, para beneficio de los sistemas de información de las diferentes empresas.

1.1.2. Arquitectura de una base de datos

En el año de 1975 la ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) aprobó una arquitectura de base de datos basados en tres niveles, lo cual establece una independencia considerable, esta independencia permite modificar el esquema de uno de sus niveles sin la necesidad de modificar el nivel superior.

Esto se logra manteniendo la separación entre las aplicaciones y los datos que se almacenan; permite el uso de múltiples vistas con las que puede interactuar el usuario y el empleo de una estructura a la cual se denomina catálogo, esta estructura permite el almacenamiento de la descripción o esquema de la base de datos.

Para tener una visión más clara de lo que implican estos tres niveles, hay que mirar los datos almacenados en cualquier tipo de dispositivo como un todo, donde los distintos niveles que presentamos no son más que una descripción abstracta dichos datos; estos elementos son manejados exclusivamente por su grupo de usuarios.

La arquitectura propuesta por la ANSI-SPARC, no ha sido establecida como un estándar, sin embargo, varios de los SGBD comerciales la emplean; esta estructura se encuentra conformada como se muestra en la Figura. Donde se establecen claramente los niveles: interno, conceptual y externo.

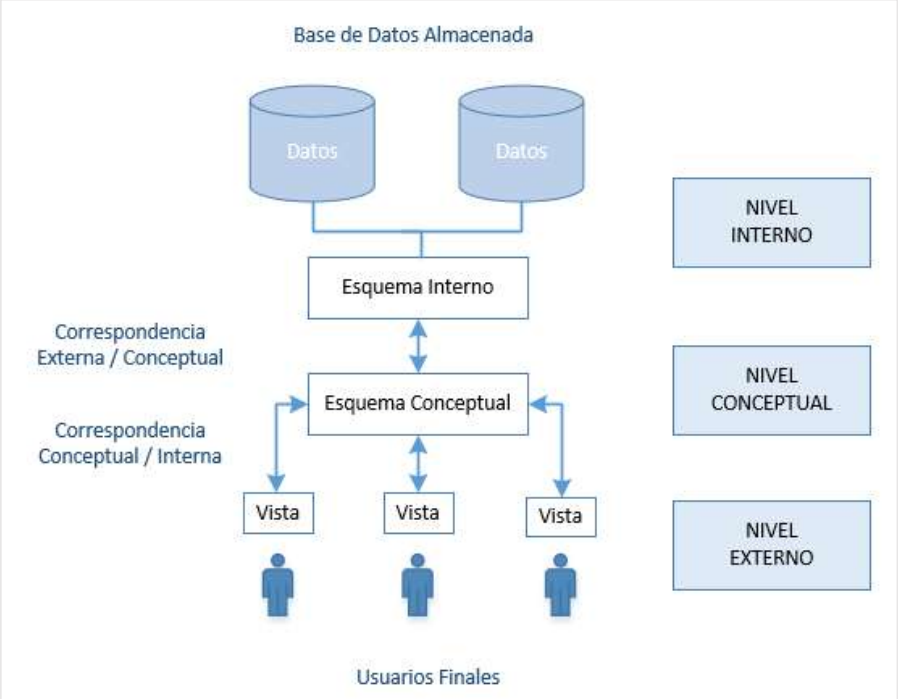


Figura 1-1: Arquitectura de base de datos
Realizado por: Caluquí Jorge, 2018

Nivel interno: también denominado como nivel físico ya que especifica mediante esquemas internos la estructura de la base de datos; describe detalladamente la forma de almacenamiento de los datos, entre las que se destacan: métodos de acceso, organización, tipificación de registros, tamaños de campos, etc.

Nivel conceptual: Este esquema es expresado mediante un esquema conceptual que incorpora una visión global de la base de datos, ocultando todo lo referente a estructuras que almacenan los datos. Esta distribución se encarga de representar entidades con sus tipos de datos, especificar relaciones existentes y operaciones de los usuarios.

Nivel externo: Este nivel está enfocado a la visión, el usuario es quien interactúa directamente con este esquema, es así que se oculta la estructura de la base de datos, así como la forma de almacenamiento; para expresar este nivel se emplean modelos lógicos.

Para el esquema planteado es necesario establecer una organización; agrupar los usuarios según las funciones que estos cumplan con respecto a los esquemas que la arquitectura de tres niveles describe; es así que los actores de bases de datos se separan según el esquema donde se desarrollan los usuarios externos, los diseñadores de base de datos y si la organización posee de tamaño considerable la administración de los recursos es fundamental, esto se realiza a través de un DBA (Administrador de Base de Datos).

A continuación, una descripción de las funciones de cada uno de los actores que intervienen en una base de datos:

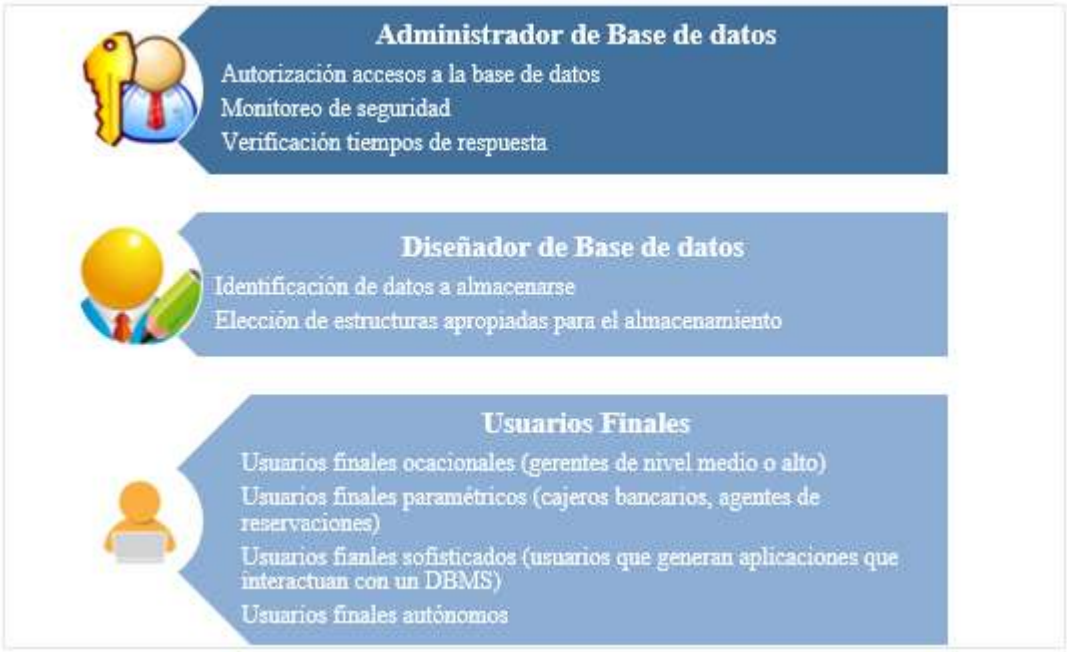


Figura 2-*1*: Funciones de los actores de base de datos
Realizado por: Caluqui Jorge

1.2. Sistemas gestores de Base de Datos (SGDB)

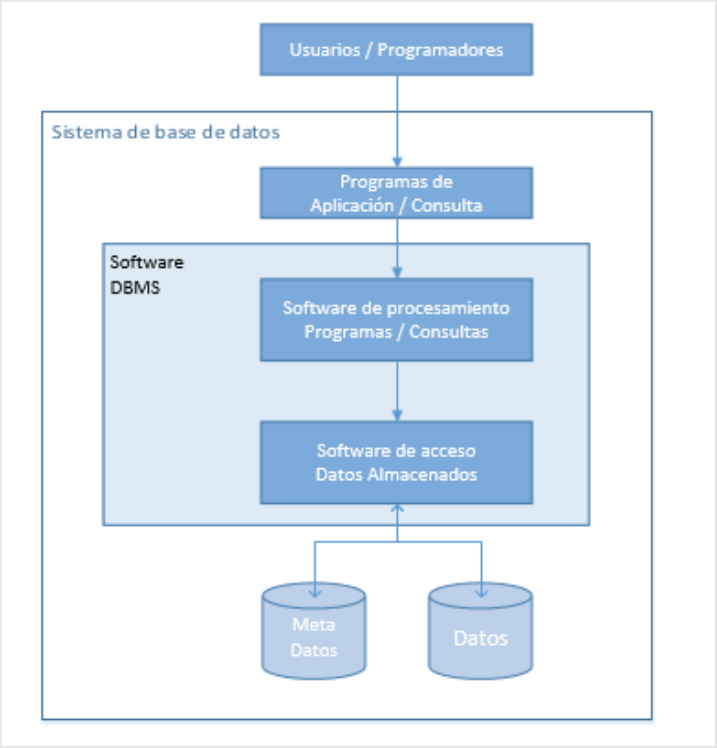


Figura 3-1: Base de datos simplificada
Fuente: http://iips.icci.edu.iq/images/exam/databases-ramaz.pdf

Los Sistemas Gestores de Bases de Datos, también conocidos como Sistemas Manejadores de Bases de Datos o DataBase Management System por sus siglas en inglés (DBMS) se pueden definir como un conjunto de aplicaciones que operan todo el acceso a la base de datos, de manera que sirven de interfaz entre los datos almacenados y el usuario. Esta integración permite la gestión: acceso, almacenamiento y modificación de la información contenida en ésta; mediante el uso de herramientas de análisis y generación de consultas e informes

El gestor de base de datos se encarga del control de las operaciones a ser ejecutadas por los usuario frente a la base de datos, esto cara al anterior sistema de gestión de archivos, un acumulado de programas que especificaban y trabajaban sus datos, el paso a los datos es autónomo de los programas que los gestionan, lo que origina una ventaja de para tratar grandes volúmenes de información (PowerData, 2016)

Entre las funciones que cumple un Sistema Gestor de Bases de Datos encontramos el siguiente gráfico:

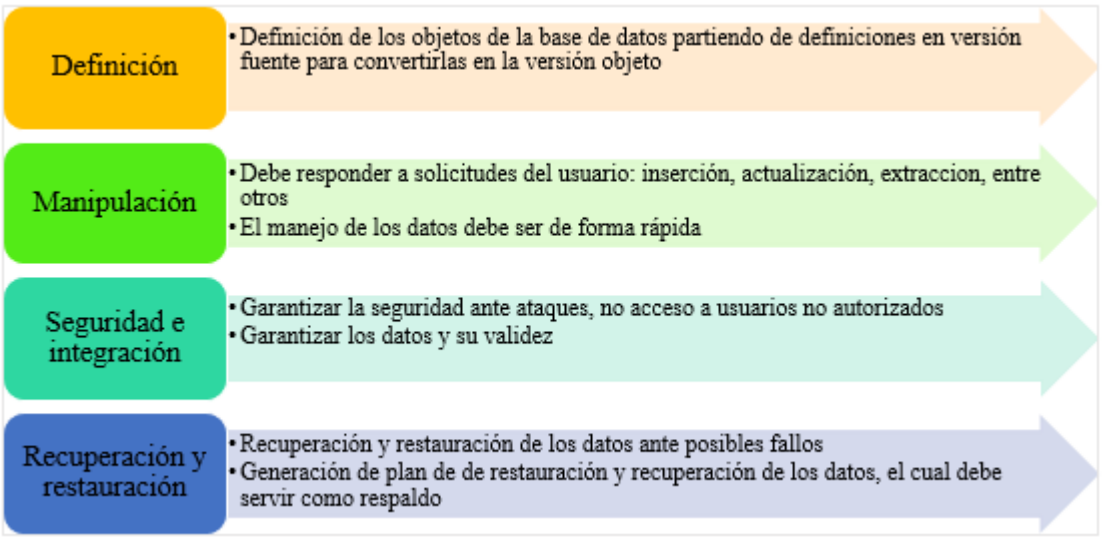


Figura 4-1: Funciones de un SGBD
Realizado por: Caluquí Jorge, 2018

1.2.1. Componentes de un SGBD

La arquitectura de SGBD describe sus componentes y sus interfaces, las cuales tratan de forma individual las responsabilidades del sistema general, mediante el siguiente gráfico:

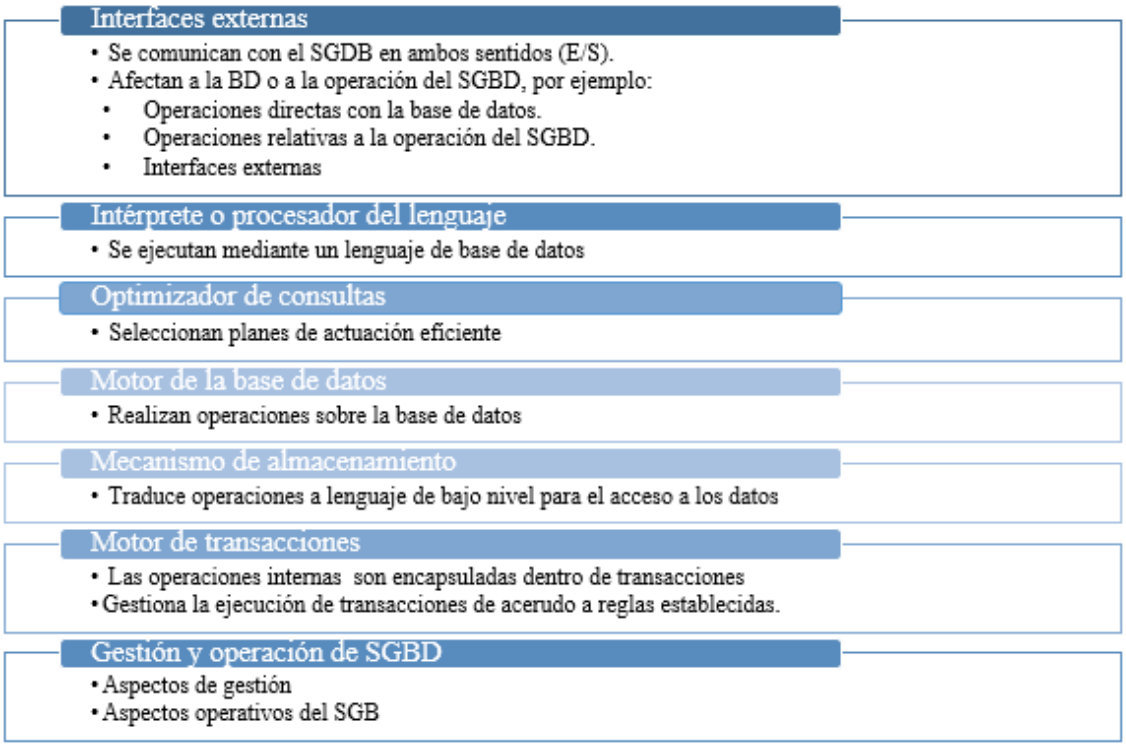


Figura 5-1: Funciones de un SGBD
Realizado por: Caluquí Jorge, 2018

1.2.2. Tipos de SGDB

Los SGDB se encuentran agrupados de diversas maneras, en función del modelo de datos que esperan para el acceso, de acuerdo al número de usuarios o de sitios suele ser lo más habitual, entre otros.

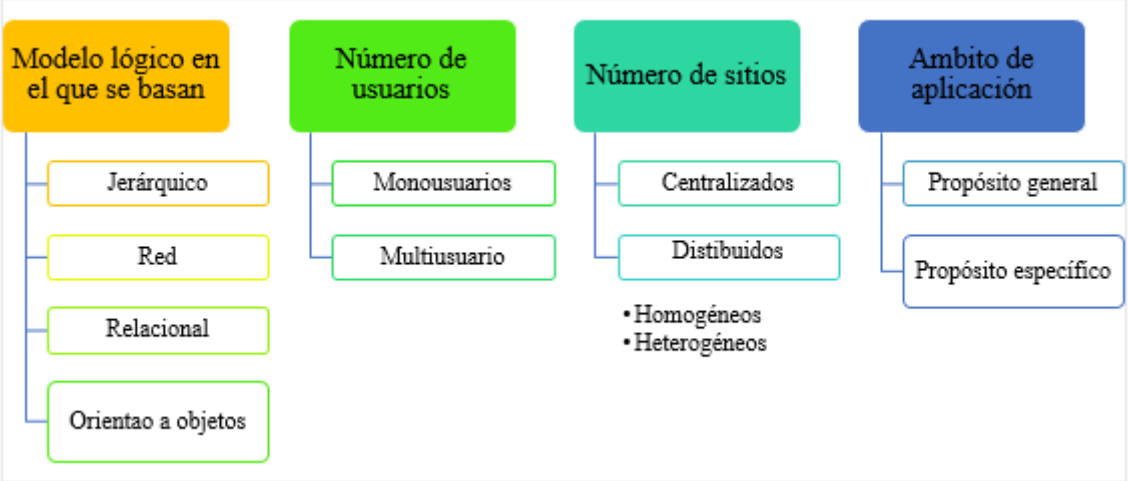


Figura 6-1: Tipos de SGBD
Realizado por: Caluquí Jorge, 2018

1.1.1.1. Modelo Lógico

Para la representación de los tres niveles de la arquitectura de un SGBD es necesario el establecimiento de modelos lógicos, dichos modelos son una agrupación de herramientas conceptuales apoyan la representación de las estructuras de las bases de datos. Los modelos lógicos conforman tres grupos: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos.

- Modelos lógicos basados en objetos. Poseen una estructuración flexible que puede especificar restricciones en los datos. Los modelos enmarcados dentro de este son los de entidad-relación y el orientado a objetos.
- Modelos lógicos basados en registros. Este modelo se basa en la estructura de un registro, donde la base está representada por registros de formato fijo de varios tipos. Entre los modelos pertenecientes a esta categoría tenemos: el modelo relacional, de red y jerárquico.
- Modelos físicos de datos. Representado por la forma en que son almacenados los datos. Representan este modelo: el modelo de memoria de elementos y el modelo unificador.

a) Jerárquico.

El modelo jerárquico es similar al modelo de red. Los datos y las relaciones se representan mediante registros y enlaces. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles. El modelo jerárquico se sirve de árboles para la representación lógica de los datos, su implementación se lleva a cabo mediante árboles y punteros (Ramos , et al., 2015).

Usado por IBM desde 1970 en su IMS (Information Management System, Sistema de Administración de Información)

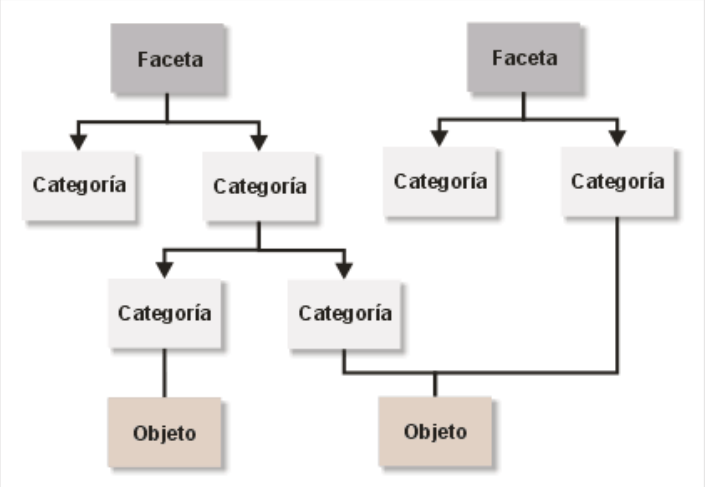


Figura 7-1: Modelo de base de datos Jerárquico
Fuente: **http://www.nosolousabilidad.com/articulos/img/class-f.gif**

Características

- Organización del árbol con un conjunto de niveles.
- Nivel alto conformado por un nodo raíz, el nivel 0.
- Los arcos representan asociaciones jerárquicas, no tienen nombre.
- Un nodo padre puede tener varios hijos, pero un hijo solo un padre.
- Los nodos sin descendencias toman el nombre de Hojas.
- Altura es el número de niveles

Conceptos básicos

Los segmentos, en función de su situación en el árbol, se denominan (Ramos , et al., 2015):

- Segmento padre: es el que tiene descendientes, todos ellos localizados en el mismo nivel.
- Segmento hijo: es el que depende de un segmento de nivel superior. Los hijos de un mismo padre están en el mismo nivel.

- Segmento raíz: es el padre que no tiene padre. Ocupa el nivel superior del árbol. El segmento raíz es único.

b) Modelo de Red

Este modelo utiliza estructuras de datos en red, también conocidas como estructuras plex, a menudo, limitan el cambio que el crecimiento de la Base de datos; exige, hasta tal punto que las representaciones lógicas de los datos pueden variar afectando a los programas de aplicación que usan esos datos (Ramos , et al., 2015)

Publicado en 1969 por CODASYL. Un modelo popular a inicios de los 70. se conforma mediante registros o nodos y enlaces entre ellos. Contrariamente al modelo jerárquico, un nodo puede tener varios padres.

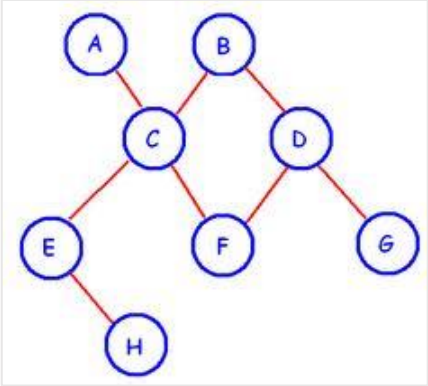


Figura 8-1: Modelo de base de datos Jerárquico
Fuente: **http://ejemplosde.org/img/basesdedatos2.jpg**

Características

- Admite relaciones de un registro con varios, relaciones N:N
- Permite representar de modo flexible los objetos y sus relaciones
- Una cualidad que lo distingue es que este esquema no tiene ningún tipo de restricción

Conceptos básicos

- Elemento: el conjunto de datos.
- Agregados de datos: el conjunto de datos identificados por nombres.
- Tipos de registro: representación de los nodos que contienen elementos
- Conjunto: conjunto de tipos de registros, relacionados entre sí. Se establece la relación de varios a varios.

- Ciclo: cuando se registra una relación con un miembro de descendientes y a la vez con sus antepasados.
- Bucle, lazo o loop: mientras se mantiene el mismo ciclo ente registros propietarios y miembros.

c) Estructura Relacional

El modelo de datos entidad-interrelación (E-R), también llamado entidad-relación, fue propuesto por Peter Chen en 1976 para la representación conceptual de los problemas del mundo real. En 1988, el ANSI lo seleccionó como modelo estándar para los sistemas de diccionarios de recursos de información. Es un modelo muy extendido y potente para la representación de los datos. Se simboliza haciendo uso de grafos y de tablas. Propone el uso de tablas bidimensionales para la representación de los datos y sus relaciones (Ramos , et al., 2015).

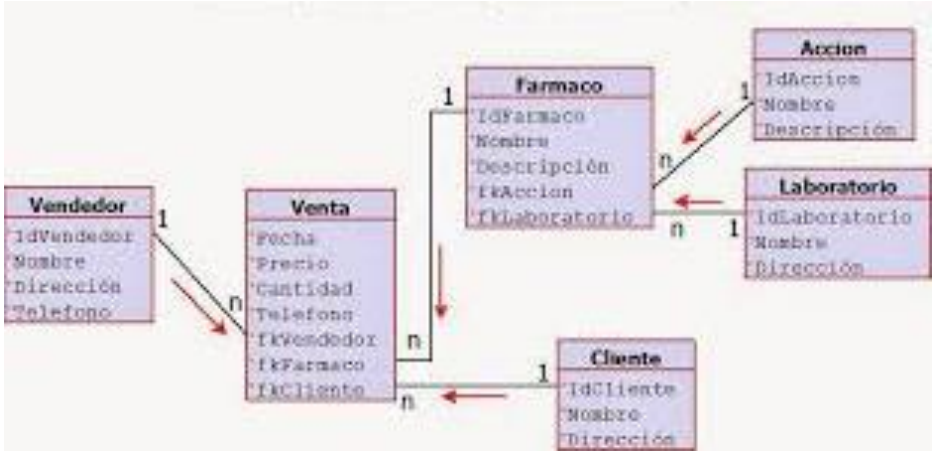


Figura 9-1: Estructura Relacional
Fuente:https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSlVA-ELt-rsSBFkRvi5C5xg-F7bYa2WDqf-70TMPSnjsgVJDpy

Características

- Las interrelaciones con correspondencias ente las entidades
- La forma en que se almacenan los datos se constituyen en filas o tuplas y columnas o atributos
- Este tipo de modelos no resulta fácil de comprender y usar por parte del usuario debido a su complejidad de combinaciones.

Conceptos básicos

- Entidad: son los objetos

- Tabla: los datos de la entidad son almacenados en filas y columnas.
- Atributo: representan los campos de la tabla en columnas
- Tupla: representada por los registros de la tabla, o filas.
- Dominio: son los calores de un atributo.
- Clave primaria: identifica cada registro como único.
- Relación entre tablas: si los datos se encuentran en una sola tabla las redundancias comenzarían

d) Orientado a objetos

Este modelo está basado en el paradigma de la programación orientada a objetos (POO). Los SGBO (Sistemas de Gestión de Bases de Objetos) o SDBDOO (Sistemas de Gestión de Bases de Datos Orientados a Objetos) gestionan objetos en los cuales están encapsulados los datos y las operaciones que interactúan con ellos. Además, proporcionan un modelo único de datos. No hay diferencia entre el modelo conceptual (el modelo E-R) y el modelo lógico (el relacional), y las aplicaciones pueden acceder directamente al modelo (Ramos , et al., 2015).

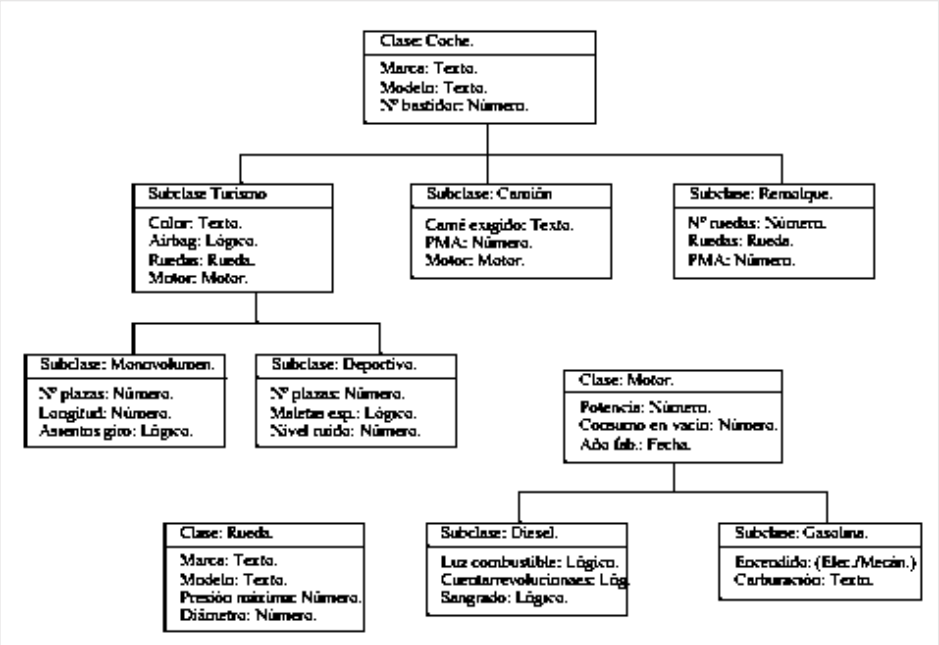


Figura 10-1: Modelo lógico orientado a objetos
Fuente: http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf

1.1.1.2. Número de usuarios

a) Monousuario

Derivado de mono que significa uno, son aquellas bases de datos que soportan no más de un usuario, sin importar el número de procesos que el usuario realice o que puedan ejecutarse en un mismo instante de tiempo.

b) Multiusuario

Las bases de datos multiusuarios, como su nombre lo indica pueden proveer servicios y procesamiento a múltiples usuarios de forma simultánea.

1.1.1.3. Número de sitios

a) Centralizados:

Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informática sin interaccionar con ninguna otra computadora. Tales sistemas comprenden el rango desde los sistemas de base de datos monousuario ejecutándose en computadoras personales hasta los sistemas de base de datos de alto rendimiento ejecutándose en grandes sistemas (Merino Rivera, 2012).

b) Distribuidos:

Una Base de Datos Distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, los cuales tienen la capacidad de procesamiento autónomo lo cual indica que puede realizar operaciones locales o distribuidas. Un sistema de Bases de Datos Distribuida (SBDD) es un sistema en el cual múltiples sitios de bases de datos están ligados por un sistema de comunicaciones de tal forma que, un usuario en cualquier sitio puede acceder los datos en cualquier parte de la red exactamente como si los datos estuvieran. (GUTIÉRREZ DÍAZ, 2016)

En un sistema distribuido de bases de datos se almacenan en varias computadoras. Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios.

1.2.3. SGDB en el Mercado

Un SGBD debe permitir especificar tipos y estructuras, permite la manipulación de los datos mediante consultas y la actualización de datos de manera sencilla; para ello que es necesario conocer los diferentes gestores de bases de datos, pero entre los más utilizados se destacan los mencionados en el siguiente cuadro:

MySQL	Es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Por un lado se ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso	
	Características	Velocidad al realizar las operaciones
		Bajo costo
		Facilidad de configura
Oracle	Es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation	
	Características	Soporte de transacciones
		Estabilidad
		Escalabilidad
		Multiplataforma
PostgreSQL	Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD, dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales.	
	Características	Alta concurrencia
		Varidad de tipos nativos
		Mejores costos de operación
		Estabilidad y confiabilidad
Microsoft SQL Server	Es un sistema de gestión de bases de datos relacionales basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea	
	Características	Soporte de transacciones
		Escalabilidad estabilidad y seguridad
		Soporta procedimientos almacenados

MariaDB	Es un sistema de gestión de base de datos con licencia GPL la cual se deriva de MySQL. Que introduce dos motores de almacenamiento nuevos: uno llamado Aria -que reemplaza con ventajas a MyISAM-, y otro llamado XtraDB -en sustitución de InnoDB. Fue desarrollado por Michael Widenius, fundador de MySQL, y la comunidad de desarrolladores de software libre.	
	Características	Cuenta con seguridad rápida y transparete
		Alto rendimiento
		Soporte de Oracle
		Compatibilidad y fácil migración
SQLite	Sistema de gestión de bases de datos relacional, que es compatible acon ACID; SQLite no es independiente del programa, esta biblioteca forma parte integral del mismo; el cual es llamda a través de llamadas simples a subrutinas y funciones	
	Características	Soporte de tablas, índices y vistas
		Operaiones más rápidas comparadas con MySQL y PostgreSQL
		Sin dependencias externas
		Soporte funciones SQL definidas por el usuario.
		Amplia documentación
MongoDB	Es un sistema de base de datos NoSQL multiplataforma, orientado a documentos desarrollado bajo la filosofía de software libre, los datos son guardados en la base datos en estructuras de datos similar a JSON de JavaScript e incluso tiene la capacidad de realizar consultas utilizando JavaScrip	
	Características	Almacenamiento orientado a documentos
		Desarrollado en C++
		No soporta transacciones ni relaciones Joins
		Consultas dinámicas
		Soporte comercia y consultoria

Figura 11-1: Bases de datos en el mercado
Realizado por: Caluquí Jorge, 2018

1.2.4. Oracle

1.2.4.1. ¿Qué es Oracle?

Oracle es la Primera Base de Datos Diseñada para Grid Computing, es un sistema de gestión de base de datos relacional fabricado por Oracle Corporation. es fundamentalmente una herramienta cliente/servidor para la gestión de base de datos, la gran fortaleza que tiene Oracle Corporation: la hacen una de las mayores compañías de software del mundo. Sus servicios van desde bases de datos Oracle hasta sistemas de gestión (Barreto Contreras , 2013).

1.2.4.2. Historia

La historia de Oracle, se resumen en el gráfico presentado a continuación

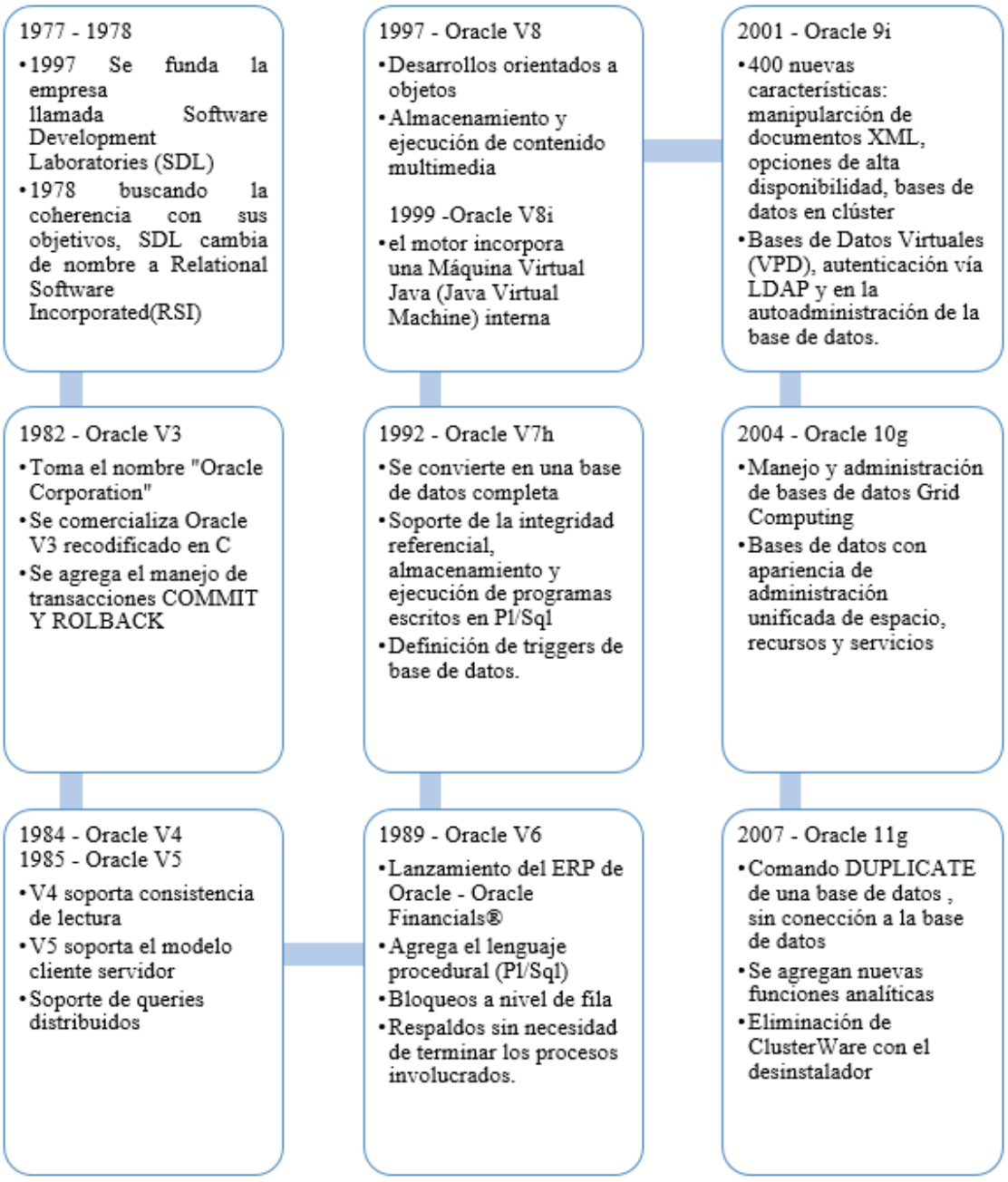


Figura 12-1: Historia de Oracle
Realizado por: Caluquí Jorge, 2018

Oracle a partir de la versión 10g Release 2, cuenta con las ediciones:

- Oracle Database Enterprise Edition(EE).
- Oracle Database Standard Edition (SE).
- Oracle Database Standard Edition One (SE1).
- Oracle Database Express Edition (XE).
- Oracle Database Personal Edition (PE).
- Oracle Database Lite Edition (LE).

1.2.4.3. Arquitectura

Oracle posee una estructura física y una estructura lógica que se conforman de manera separada:

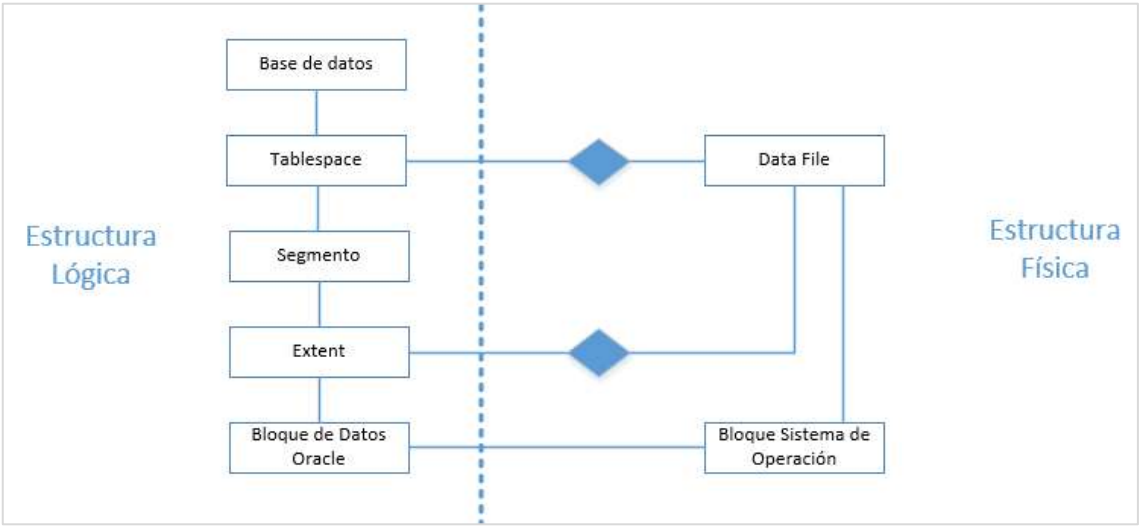


Figura 13-1: Estructuras físicas y lógicas de Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

Estructura lógica

Se dividen en unidades de almacenamiento lógicas que incluyen como mínimo un tablespaces que contiene segmentos, cada segmento está formado por extensiones, a su vez cada extensión está conformado por bloques lógicos, donde cada bloque lógico es la unidad más pequeña en las cuales se efectúan las operaciones de lectura y escritura.

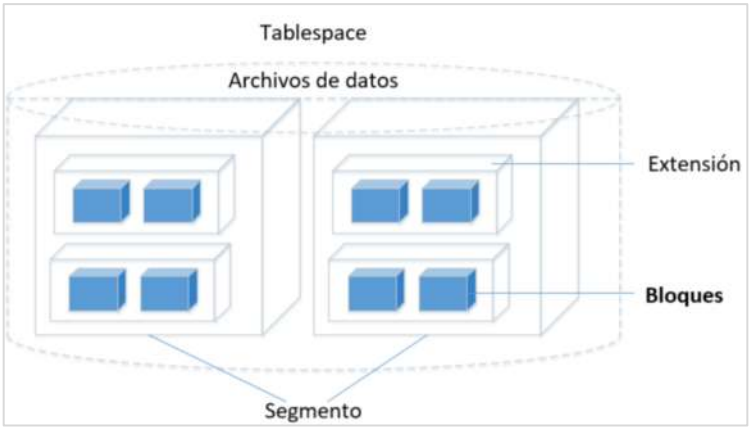


Figura 14-1: Estructuras físicas y lógicas de PostgreSQL
Realizado por: Caluquí Jorge, 2018

a) Esquemas y objetos del esquema:

Un esquema es una colección de objetos de la base de datos. Los objetos del esquema son estructuras lógicas que hacen referencia directa a datos de la base de datos (tablas, vistas, secuencias, procedimientos almacenados, sinónimos, índices, clusters y enlaces con otras bases de datos).

b) Data Base:

Es un conjunto de datos que tienen un representan una información captada del mundo real, con ellos se puede realizar diversos procesos.

c) Tablespace:

Una base de datos está formada por una o varias unidades lógicas llamadas tablespaces. Un tablespace es la unidad de almacenamiento lógico. Además, cada una de estos tablespaces está formada por uno o varios ficheros físicos que son los datafiles. Un datafile solamente puede pertenecer a un tablespace. Por lo tanto, los datafiles de una base de datos son todos los datafiles que forman parte de todos los tablespaces de la base.

d) Segment:

Un segmento almacena la información de una estructura lógica de Oracle dentro de un Tablespace. Está formado por una o más extensiones y, a medida que va creciendo el segmento se van asignando nuevas extensiones al mismo. Hay cuatro tipos de segmentos: de datos, de índices, temporales y de rollback.

e) Extent:

Una extensión es una unidad lógica de almacenamiento que está formada por un número determinado de bloques de datos contiguos. La agrupación de una o varias extensiones forman un segmento que puede ser una tabla, un índice, un segmento de rollback o un segmento temporal.

f) Data Block:

Un bloque es la unidad mínima de almacenamiento de información de Oracle. A los bloques también se les conoce como "bloques de datos", "bloques lógicos" o "bloques oracle". Cada uno de estos bloques está formado por un número determinado de bloques del sistema operativo.

Estructuras físicas

Este tipo de estructuras incluyen archivos de control, archivos redo log online y archivos de datos; una base de datos posee uno o más ficheros de datos de tamaño fijo

a) Data File:

Estos ficheros físicos son los que almacenan objetos que conforman un tablespace. Un datafile pertenece solamente a un tablespace y a una instancia de base de datos. Un tablespace puede estar formado por uno o varios datafiles. Cuando se crea un datafile, se debe indicar su nombre, su ubicación o directorio, el tamaño que va a tener y el tablespace al que va a pertenecer.

b) Os Block:

Conocidos como Disk Block, estos mapean a la data blocks. A la hora de crear una nueva base de datos se debe indicar cuántos bloques de sistema operativo formarán un bloque de datos o bloque oracle. (Alarcón P.P, 2013)

1.2.5. PostgreSQL

1.2.5.1. ¿Qué es PostgreSQL?

PostgreSQL es un sistema de gestión de bases de datos relacionales de objetos (ORDBMS). Su herencia se remonta al proyecto POSTGRESQL en el Departamento de Ciencias de la Computación de la Universidad de California en Berkeley. POSTGRES fue pionero en muchos

conceptos que solo estuvieron disponibles en otros sistemas de bases de datos mucho más tarde (The PostgrSQL Global Development Group, 2010)

1.2.5.2. Historia

Este SGBD ahora conocido como PostgreSQL se deriva del paquete POSTGRES escrito en la universidad de California en Berkeley; ahora con más de dos décadas de desarrollo; Post-Postgree es ahora la base de datos de código abierto más avanzada disponible a nivel mundial.

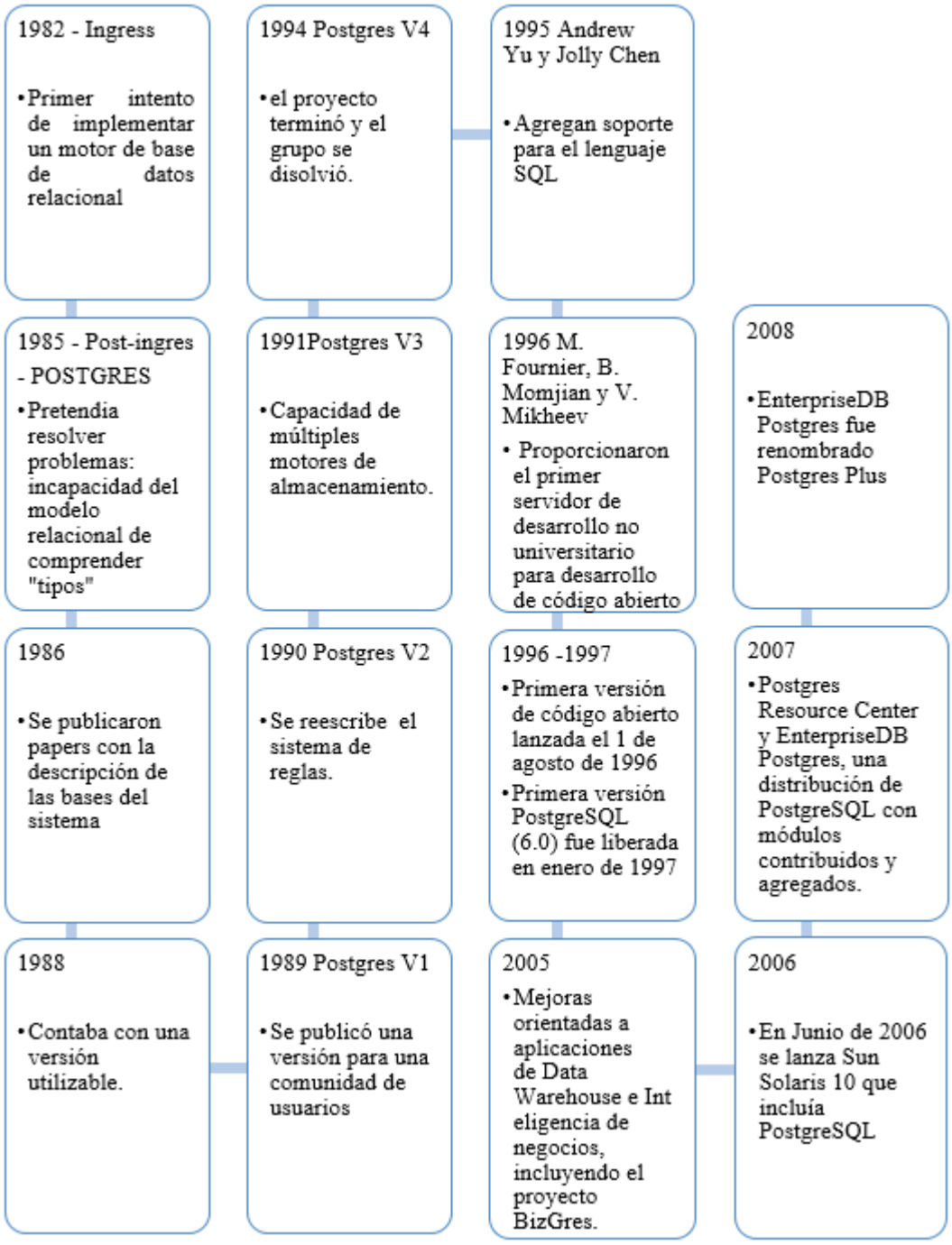


Figura 15-1: Historia de PostgreSQL
Realizado por: Caluquí Jorge, 2018

1.2.5.3. Arquitectura

En un servidor se nacen uno o varios clusters de bases de datos. La estructura física del cluster se crea con el programa initdb, con este programa se fija la ubicación física y el juego de caracteres. El cluster se crea en un directorio "data" dentro del directorio donde se ha instalado postgres. Normalmente, se define una variable de entorno, PGDATA que apunte al directorio donde se crea el cluster. (Alarcón Medina, 2017)

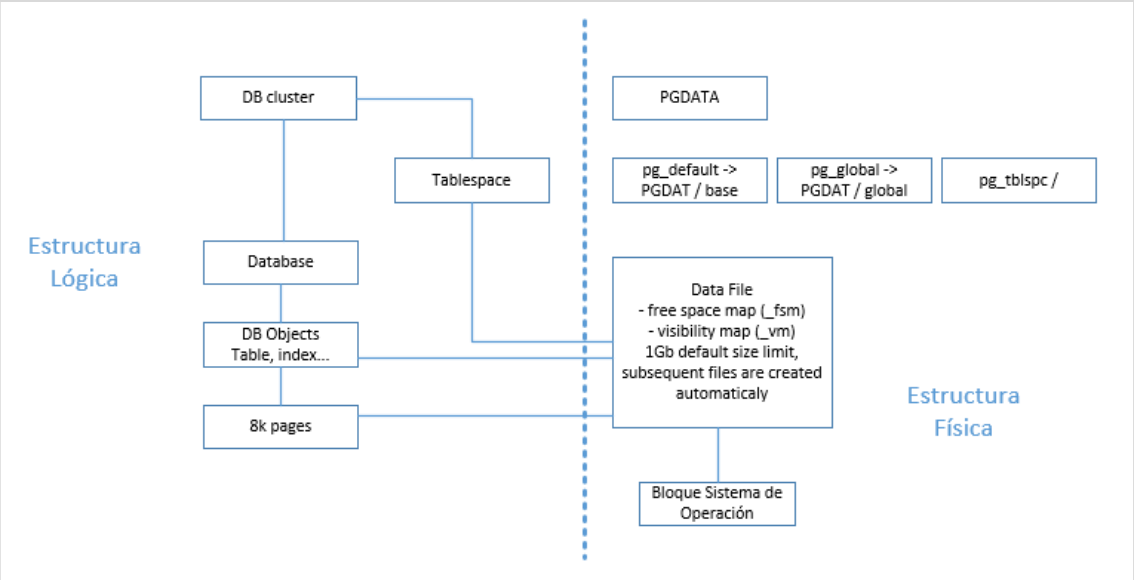


Figura 16-1: Estructuras físicas y lógicas de PostgreSQL
Realizado por: Caluquí Jorge, 2018

Estructura lógica

a) Cluster:

Este repositorio abarca un conjunto de bases de datos, que contienen objetos que se pueden almacenar en distintos tablespaces y un conjunto de usuarios que se conectan al cluster, una base de datos abarca un conjunto de esquemas, los cuales tienen un usuario propietario. En los esquemas es donde se crean los objetos (tablas, índices, procedimientos, vistas, etc.) y una sesión se conecta contra una base de datos (Alarcón Medina, 2017)

b) Bases de datos

Agrupaciones de esquemas. Por defecto, siempre hay tres bases de datos creadas, template0, template1 y postgres. (Alarcón Medina, 2017)

c) Tablespaces

Son las ubicaciones alternativas a la que por defecto tiene el cluster. Por defecto no se crea ninguno. (Alarcón Medina, 2017)

d) Roles

Engloba el concepto de usuarios (roles de login) y grupos de permisos (roles de grupo), estos últimos son lo mismo que los roles de Oracle. Por defecto, si al crear el cluster no se ha indicado otro usuario, se crea el usuario postgres como superusuario. (Alarcón Medina, 2017)

1.2.6. Oracle vs. PostgreSQL

1.2.6.1. Semejanzas

En la época de los 70, la corporación IBM fue la primera en publicar los documentos que darían comienzo a System R que más tarde originaría a DB2 escritos por el Sr. Edgar Frank Codd, tanto Oracle como PostgreSQL tuvieron sus inicios siguiendo estas publicaciones, por lo cual son similares entre sí; las semejanzas entre ellos se establecen debido a la visión de una base de datos robusta.

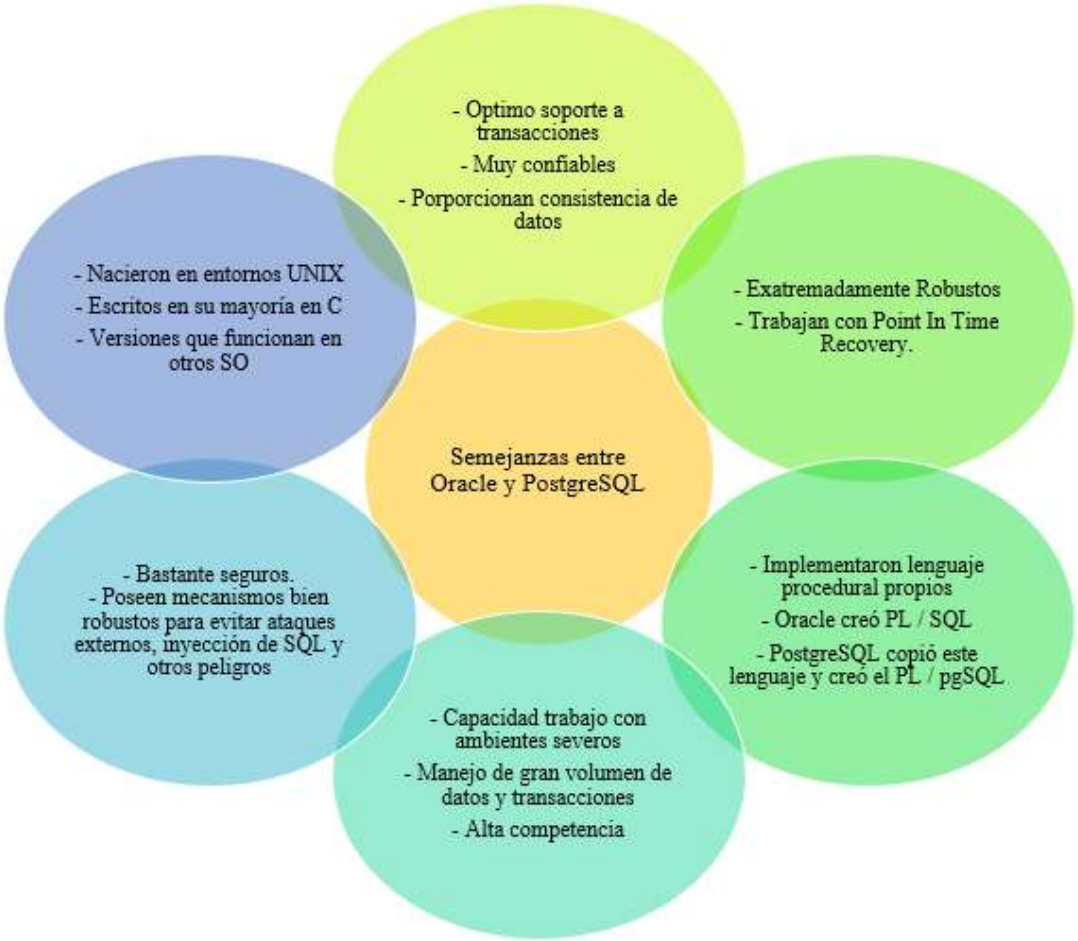


Figura 17-1: Semejanzas Oracle y PostgreSQL
Realizado por: Caluquí Jorge, 2018

1.2.6.2. Ventajas y desventajas

Entre las principales bondades que oferta PostgreSQL se encuentran: la alta concurrencia, la amplia variedad de tipos nativos, y diversas funciones más específicas; esto permite hacer transacciones eventualmente consistentes, brindando grandes ventajas en el rendimiento.

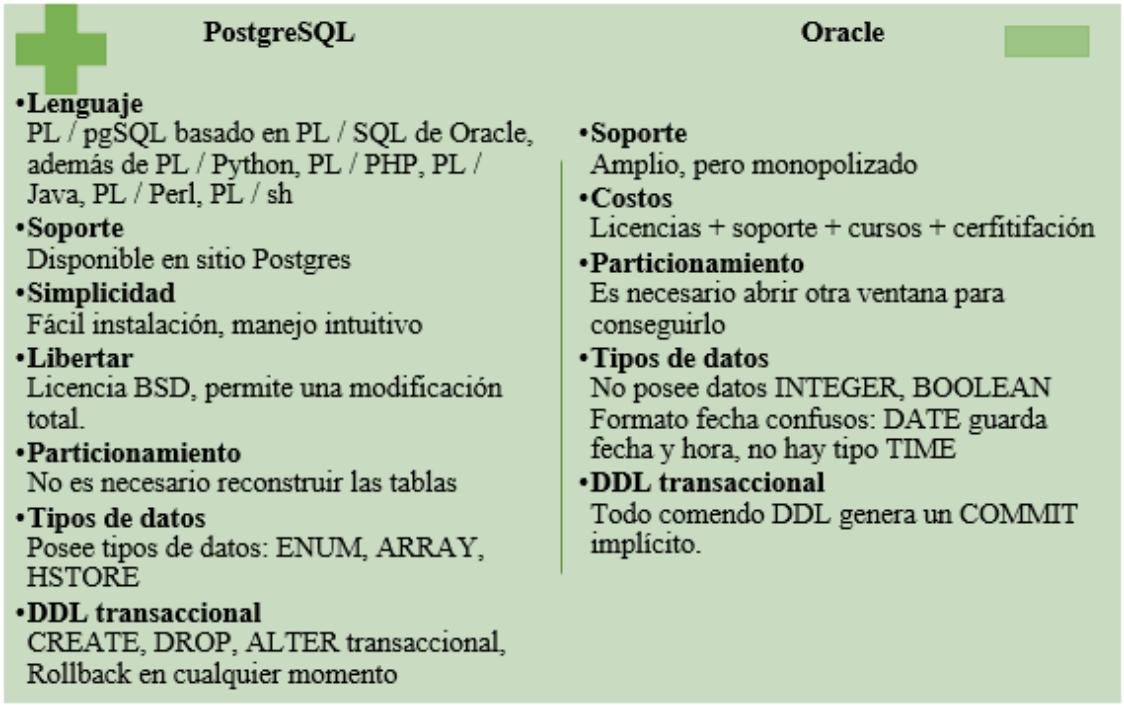


Figura 18-1: Ventajas de PostgreSQL sobre Oracle
Realizado por: Caluquí Jorge, 2018

Si bien existe una diferencia marcada entre Oracle y Postgres con respecto a los costos de soporte, instalación, capacitación, etc., es el soporte el que pone a Oracle en la cúspide, ya que este ofrece un experto en casi cada país, el cual ayuda a resolver de manera sencilla y rápida cualquier inconveniente que el cliente pueda tener. Este punto a favor es la clave para que las empresas elijan a Oracle como su gestor de base de datos; conjuntamente con la potencia y robustez que ofrece.

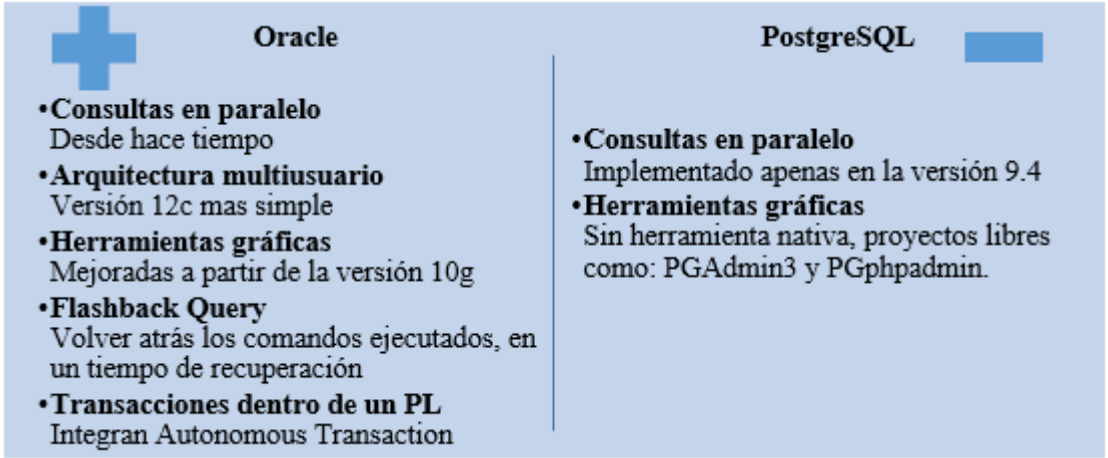


Figura 19-1: Ventajas de Oracle sobre PostgreSQL
Realizado por: Caluquí Jorge, 2018

1.2.7. Pl/SQL vs. PL/pgSQL

El lenguaje SQL por sus siglas en inglés Structured Query Language (lenguaje de consulta estructurada), establecido como estándar por el Instituto Nacional Estadounidense de Estándares (ANSI) en 1986 y por la Organización Internacional de Normalización (ISO) en 1987. Este lenguaje es portátil y de fácil utilización y aprendizaje; el utiliza el álgebra y calculo relacional para la ejecución de consultas y modificaciones a la base de datos.

SQL es considerados solo un lenguaje de consulta, que no cuenta con las características de los lenguajes de programación, siendo así no incluye el uso de: variables, estructuras de control de flujo, bucles, etc.; debido a esto una de sus mayores desventajas es la necesidad de ejecutar cada declaración SQL de forma individual en el servidor, es decir que la aplicación envía una por una las consultas a la base, espera que sea procesada, recibe los datos para su procesamiento y nuevamente envía una consulta, es por ello que tanto Oracle como PostgreSQL han implementado sus propios lenguajes, estos basados en SQL, de aquí nacen PL/SQL y PL/pgSQL.

1.1.1.4. PL/SQL

Por sus siglas en inglés Procedural Language/Structured Query Language es un lenguaje de programación que combina el lenguaje SQL con las características de procedimiento de los lenguajes de programación. Fue implementado a principios de los 90 por Oracle Corporation con el objetivo de optimizar las capacidades de SQL. Pero PL/SQL es solo uno de los tres lenguajes que Oracle integra en su Gestor de Bases de datos, conjuntamente se encuentra SQL y Java;

Con el lenguaje PL/SQL Oracle solo pretende extender el SQL estándar agregando otro tipo de instrucciones y elementos propios que son propios de los lenguajes de programación, entre las unidades programables se observan: procedimientos almacenados, funciones, triggers, scripts. Entre las ventajas de usar este tipo de procedimientos tenemos:

- Facilitar la gestión y la seguridad.
- Mejorar rendimiento, ya que están compilados y almacenados en la base de datos.
- Disminuir el uso de la memoria.
- Incrementar la productividad y efectividad.

1.1.1.5. PL/pgSQL

Sus siglas en inglés Procedural Language / PostgreSQL Structured Query Language, es considerado como un lenguaje imperativo creado para el gestor de base de datos PostgreSQL. Algunas de sus funcionalidades según (The PostgreSQL Global Development Group, 2017) son:

- Es fácil de usar.
- Se puede usar para crear funciones y activar procedimientos.
- Agrega estructuras de control al lenguaje SQL.
- Puede realizar cálculos complejos.
- Hereda todos los tipos, funciones y operadores definidos por el usuario.
- Se puede definir para que sea confiable para el servidor.

CAPÍTULO II

2. MARCO METODOLÓGICO

2.1. Generalidades de la Institución

Nombre de la Institución

Unidad Educativa Riobamba

Ubicación

Avenida Lizarzaburu S/N Avenida La Prensa

Misión y Visión

Misión

La Unidad Educativa Riobamba, es una institución dedicada a potenciar el espíritu humanístico, científico y tecnológico de los estudiantes consciente de sí mismo, responsables, comprometidos con los cambios sociales, con las habilidades, destrezas, competencias, con criterio de desempeño, conocimientos significativos y la práctica de valores éticos, estéticos morales para continuar sus estudios superiores y su formación personal, contribuyendo al desarrollo sustentable del país y a mejorar la calidad de vida de la comunidad.

Visión

La Unidad Educativa Riobamba, para el año 2018 habrá afianzado su proyecto educativo con características que respondan a normas Nacionales e Internacionales de calidad. Alcanzara un alto reconocimiento en el centro del país por el liderazgo y competencias de desempeño logradas mediante la investigación científica, técnica y humana en incidencia de impacto sostenido en la comunidad y con apertura a los cambios que la época lo exija

Análisis de la situación actual

La Unidad Educativa Riobamba (Riobamba) administra los datos de estudiantes mediante el sistema académico VenusPro10 el cual cuenta con acceso a un SGBD Oracle con versión XE; se proyecta que el incremento de sus datos sobrepasaran los niveles permitidos por la versión Express, lo que genera un escaso espacio para el registro en la base de datos, la inexistencia de recursos económicos así como la ausencia de una licencia empresarial, y el limitado espacio,

generaran perdida de información, limitación en el almacenamiento y dificultad en la adquisición de la licencia para el SGBD, por lo que se ha generado la necesidad de migrar a un SGBD libre; dicha migración debe proveer las garantías necesarias sobre la información para la no afectación de la misma.

Analizando la necesidad de un nuevo SGBD libre que pueda brindar estabilidad, confiabilidad, integridad referencial, cumpla con el conjunto de características ACID acrónimo de Atomicidad, Consistencia, Aislamiento y Durabilidad para los SGBD; se ha considerado al SGBD PostgreSQL para la migración.

Mediante la evaluación de compatibilidad de las estructuras físicas y lógicas de los SGBD PostgreSQL y OracleDataBase se determinarán porcentajes de afectación y seguridad en la migración de los datos.

2.2. Planificación del proyecto

El siguiente proyecto de migración específica los pasos a seguir desde la instalación de las bases de datos Oracle 11g XE, hacia una base de datos PosgreSQL; la cual comprende la creación de las estructuras en base a parametrizaciones establecidas mediante el análisis de los tipos de datos, instrucciones de creación, modificación y eliminación de tablas y columnas, así como el paso de los datos entre las tablas de las diferentes estructuras de bases de datos y finalmente la migración de procedimientos, vistas y funciones que permitan la interacción con el aplicativo VenusPro10.

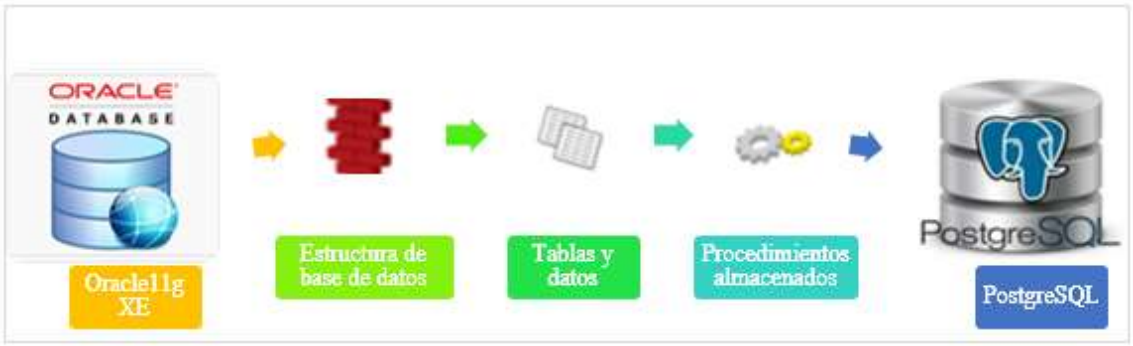


Figura 20-2: Planificación de migración de Oracle 11g XE a PostgreSQL
Realizado por: Caluquí Jorge, 2018

2.3. Análisis de requisitos

El sistema VenusPro10 cuenta con acceso OracleDatBase XE, el crecimiento de sus datos ha generado necesidades de migración a un SGBD, PostgreSQL; el cual posee características

funcionales similares a OracleDatBase XE en lo que respecta al manejo y almacenamiento de los datos.

Para la migración de las estructuras, datos, vistas, procedimientos, funciones se han considerado actividades puntuales, como las que se describen en la siguiente tabla.

Tabla 1-2: Requerimiento de migración

Identificación	Requerimientos
Req. 01	Análisis de tipos de datos, instrucciones de creación, modificación y eliminación de tablas, procedimientos, funciones y vistas en Oracle 11g XE y PostgreSQL
Req. 02	Migración de los tipos de datos de las tablas pertenecientes a los módulo Kernel, Registration y Score desde Oracle 11g XE hacia PostgreSQL.
Req. 03	Migración de los scripts de mantenimiento de tablas pertenecientes a los módulo Kernel, Registration y Score desde Oracle 11g XE hacia PostgreSQL.
Req. 04	Migración de los scripts de mantenimiento de vistas pertenecientes a los módulo Kernel, Registration y Score desde Oracle 11g XE hacia PostgreSQL.
Req. 05	Migración de los scripts de mantenimiento de procedimientos y funciones pertenecientes a los módulo Kernel, Registration y Score desde Oracle 11g XE hacia PostgreSQL.
Req. 06	Adaptación del módulo Kernel del aplicativo Venus Pro10 desde Oracle 11g XE hacia PostgreSQL.

Realizado por: Caluquí Jorge, 2018

2.4. Diseño

Guía de migración de Oracle a PostgreSQL para el sistema VenusPro10

Para la migración de las estructuras de datos empleados para la Unidad Educativa Riobamba en su sistema académico VenusPro10 la cual posee un SGBD Oracle 11g XE, hacia el SGBD ProstgreSQL, es necesario el establecimiento de ciertos parámetros que permitan realizar las conversiones tanto de tipos de datos, estructura de tablas, columnas, así como los procedimientos y funciones que permiten el desarrollo normal de las actividades de este sistema

2.4.1. Creación de ambiente en Oracle

2.4.1.1. Requerimientos de hardware

Para la instalación de una base de datos Oracle versión 11g XE es necesario tener en cuenta lo siguiente:

Tabla 2-2: Funciones agregadas más comunes en Oracle 11g XE

HARDWARE	REQUERIMIENTOS MINIMOS
Memoria física (RAM)	256 MB mínimo. Se recomienda 512MB
Memoria virtual	El doble de la RAM
Espacio de disco	Aproximadamente 2 GB dependiendo del tipo de instalación y de las opciones establecidas
Adaptador de video	256 colores

Procesador	550 MHz mínimo

Realizado por: Caluquí Jorge, 2018

2.4.1.2. Instalación de Oracle 11g XE

1. Al iniciar la instalación se presenta la pantalla de carga de archivos del instalador de Oracle

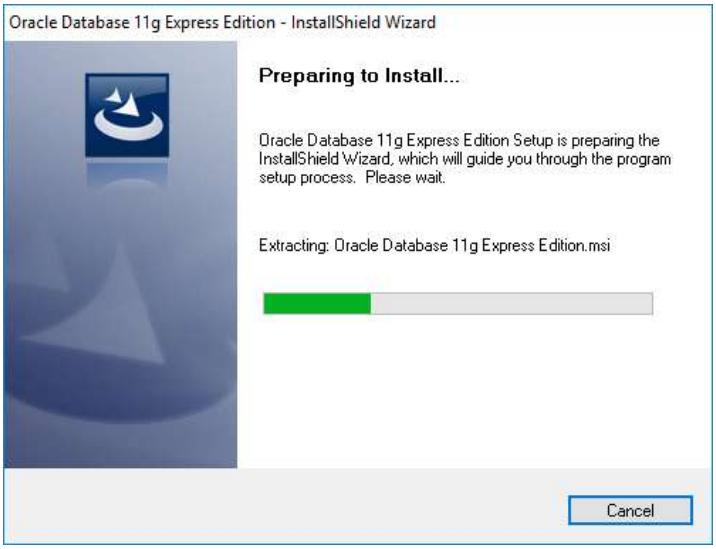


Figura 21-2: Pantalla de preparación de instalación Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

2. La pantalla de bienvenida de la instalación se muestra a continuación; presione el botón siguiente.

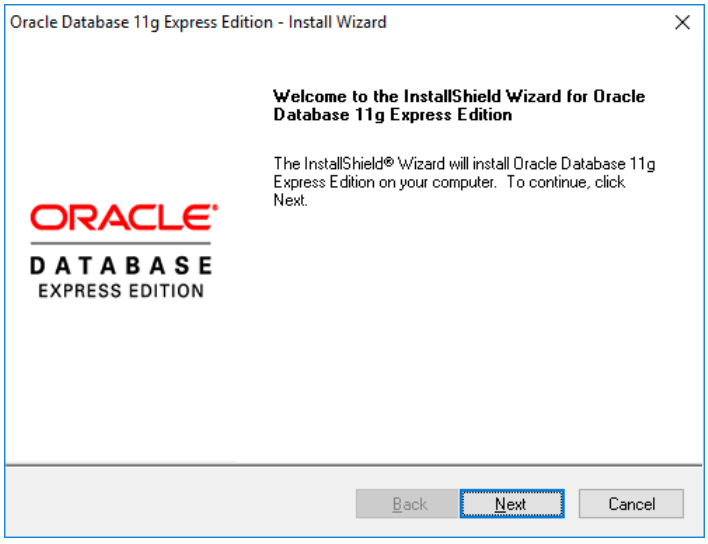


Figura 22-2: Pantalla de preparación de bienvenida Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

3. Se presentan los términos de la licencia de Oracle, acepte la licencia y presione el botón Next.

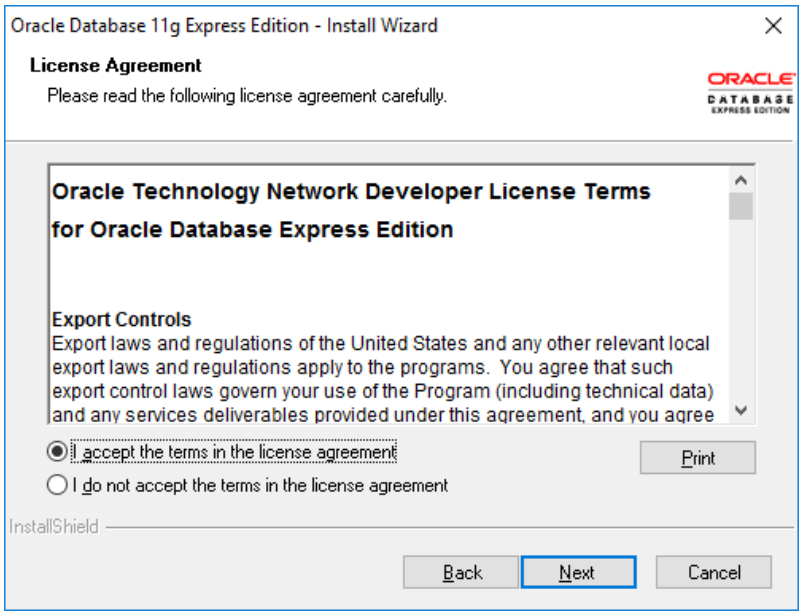


Figura 23-2: Pantalla términos de la licencia Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

4. Seleccione el tipo de edición de Oracle DataBase a instalar; la versión que se descarga por defecto es Oracle Database 11g Express Edition.

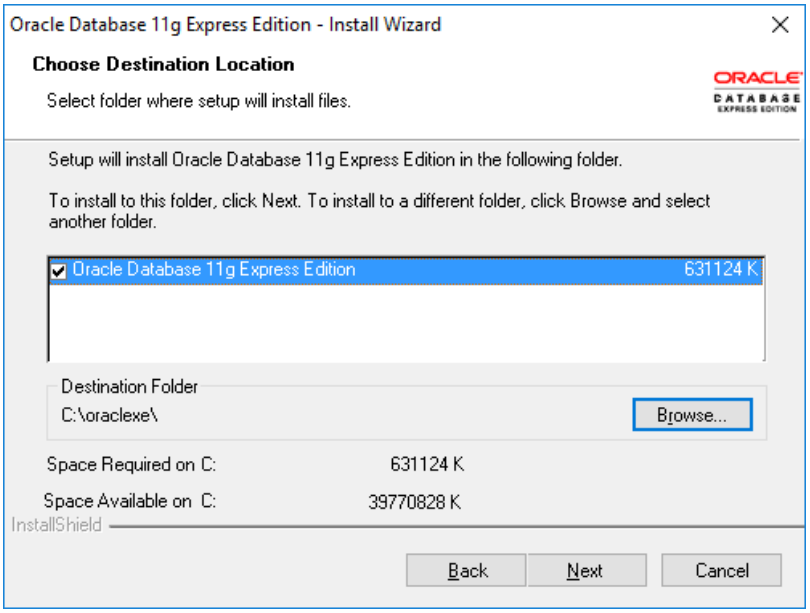


Figura 24-2: Pantalla términos de la licencia Oracle 11g XE
Realizado por: *Caluquí Jorge, 2018*

5. Introduzca la contraseña y confírmela; está será empleada para las cuentas propias de Oracle.

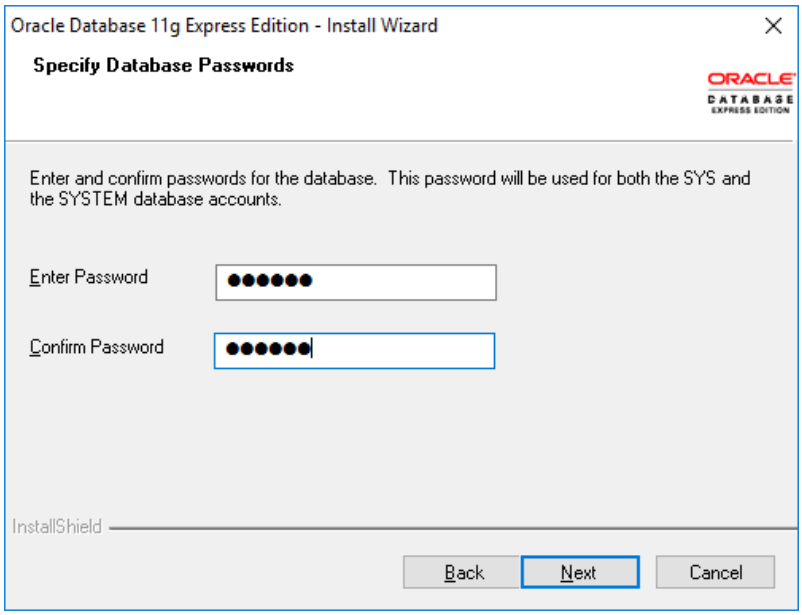


Figura 25-2: Pantalla especificación de contraseñas Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

6. La pantalla Sumary, muestra el resumen de instalación, los parámetros del producto y su relación con el sistema; presione el botón Install.

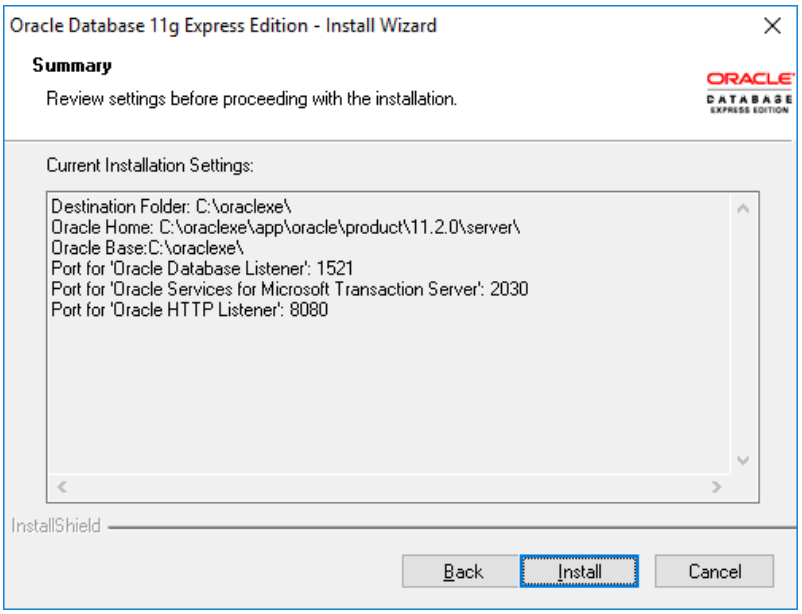


Figura 26-2: Pantalla resumen Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

7. La instalación se inicia, y en la pantalla se muestra su progreso; espere a que la base termine de instalarse

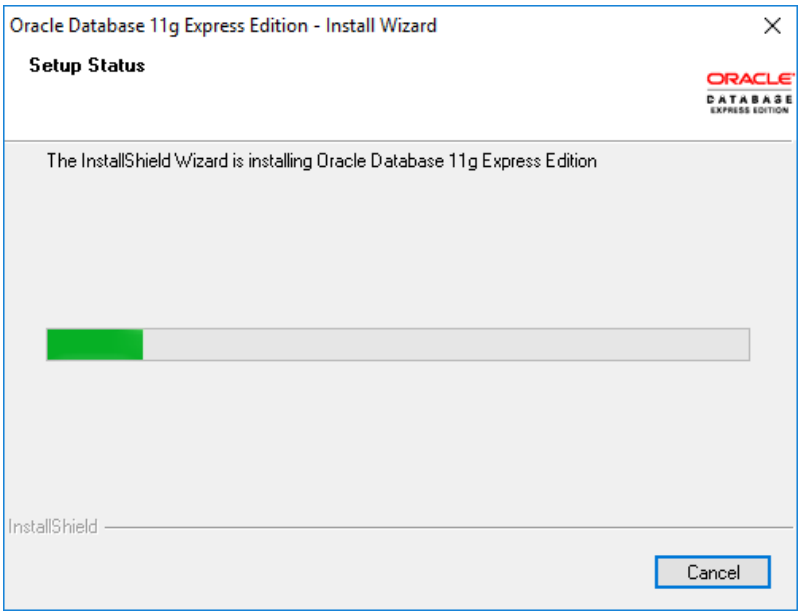


Figura 27-2: Pantalla resumen Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

8. Una vez concluida la instalación, se muestra la siguiente pantalla.

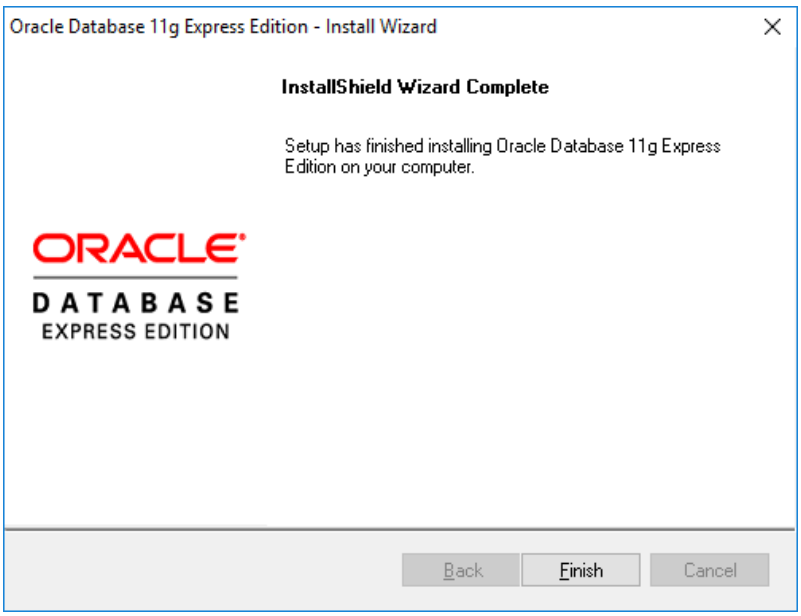


Figura 28-2: Pantalla finalización de instalación Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

2.4.1.3. Creación de base de datos

1. Para la creación de la base de datos, ingrese promt de Oracle, aparecerá la pantalla que se muestra a continuación

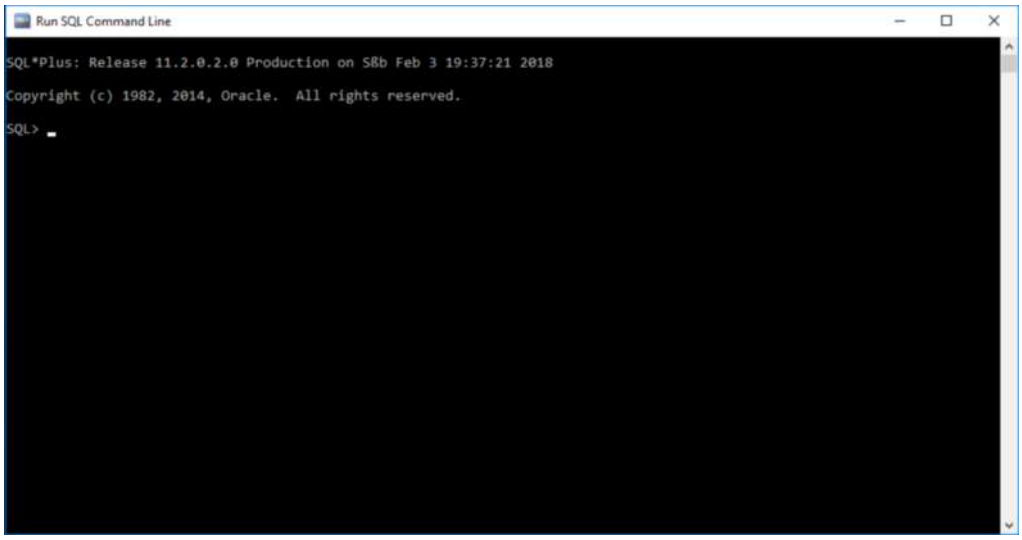


Figura 29-2: Pantalla promt Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

2. Conéctese a la base principal, para ello ingrese las credenciales proporcionadas en la instalación, como se muestra en la imagen.

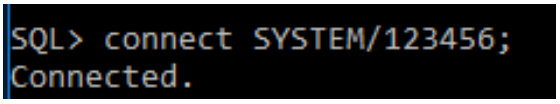


Figura 30-2: Pantalla login Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

3. Para crear la base de datos ingrese el comando CREATE USER seguido del nombre que quiera asignarle, conjuntamente con su contraseña, como se muestra a continuación.

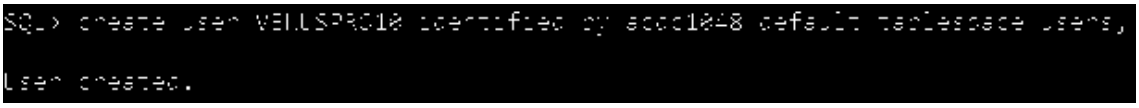


Figura 31-2: Pantalla creación Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

4. Una vez creada la base de datos, es necesario es necesario asignar privilegios de conexión a la base que ha sido creada.

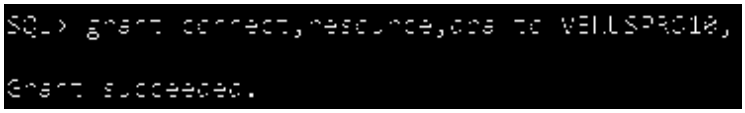


Figura 32-2: Pantalla asignación de privilegios en Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

5. Los comandos generados se ven como a continuación se presenta en la imagen

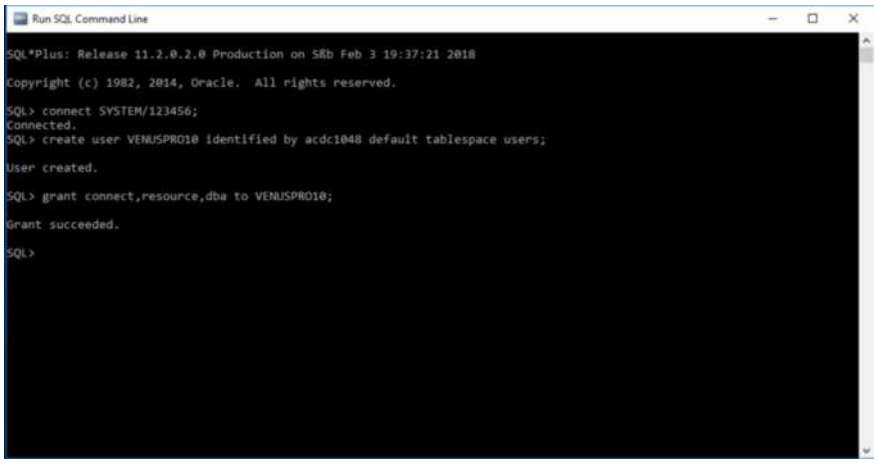


Figura 33-2: Pantalla resumen de creación de esquema en Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

6. Con la base creada, ingresamos al administrador de Oracle, para esto usamos SQL Developer, su pantalla principal se muestra a continuación

Figura 34-2: Pantalla SQL Developer para Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

7. Se procede a conectar la base de datos, con las credenciales proporcionadas en la creación de la base, probamos y guardamos la conexión.

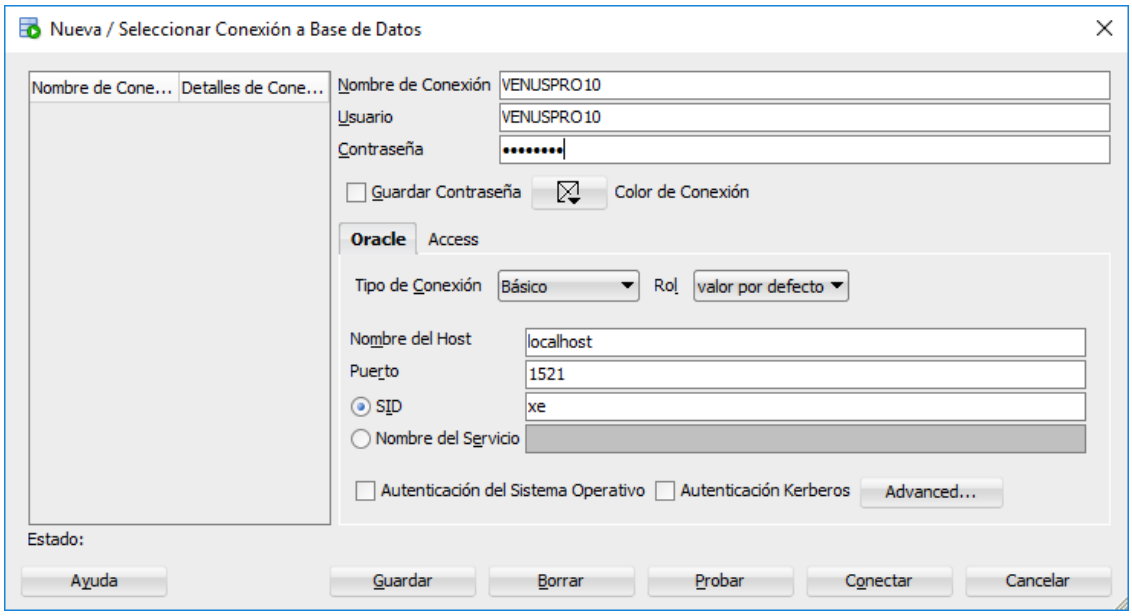


Figura 35-2: Pantalla conexión de base de datos Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

8. Una vez conectada la base de datos, se presenta a continuación la hoja de trabajo, sobre la cual se ejecutan los scripts.

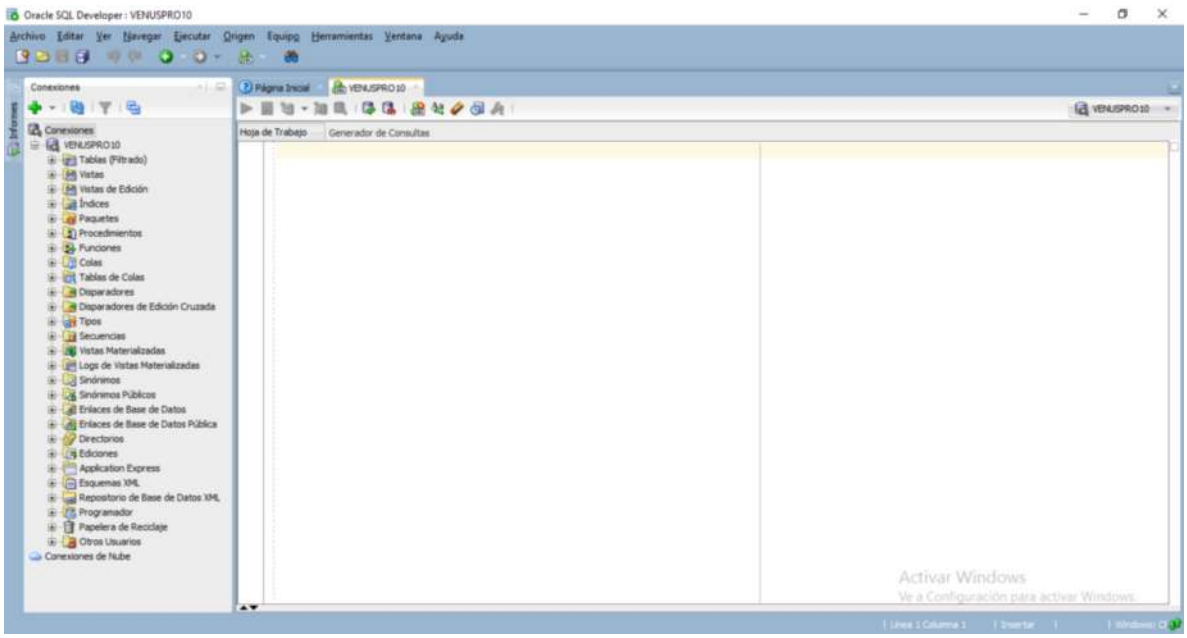


Figura 36-2: Pantalla jerarquía estructural de base de datos Oracle 11g XE
Realizado por: Caluquí Jorge, 2018

2.4.2. Creación de ambiente en PostgreSQL

2.4.2.1. Requerimientos de hardware

Para la instalación de una base de datos PostgreSQL es necesario tener en cuenta lo siguiente:

Tabla 3-2: Requerimientos de hardware PostgreSQL

HARDWARE	REQUERIMIENTOS MINIMOS
Memoria física (RAM)	Se recomienda 512MB
Espacio de disco	Aproximadamente 90MB dependiendo del tipo de instalación y de las opciones establecidas
Procesador	1.7 MHz

Realizado por: Caluquí Jorge, 2018

2.4.2.2. Instalación de PostgreSQL

1. Pantalla principal de instalación de PostgreSQL. Presione el botón siguiente

Figura 37-2: Pantalla de inicio de instalación PostgreSQL
Realizado por: Caluquí Jorge, 2018

2. Pantalla de selección de directorio de instalación de PostgreSQL. Se presenta la dirección de instalación por defecto. Presione el botón siguiente

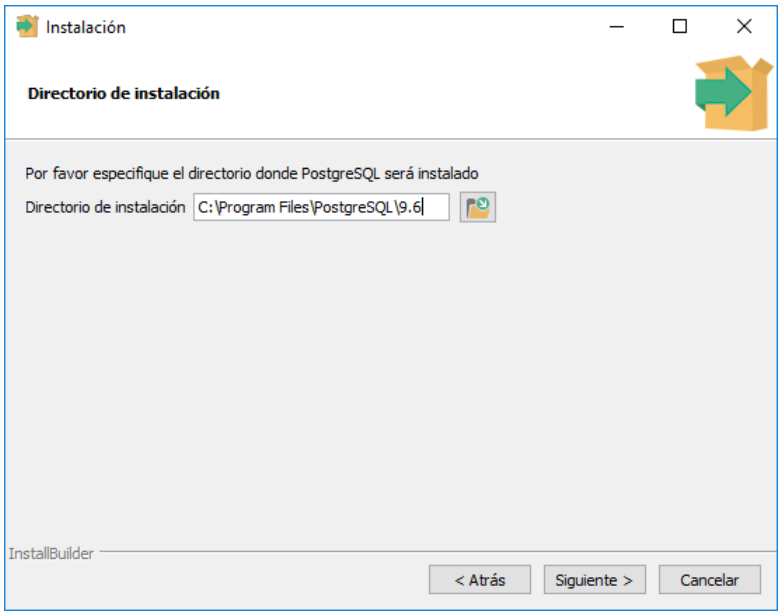


Figura 38-2: Pantalla de especificación de directorio PostgreSQL
Realizado por: Caluquí Jorge, 2018

3. Pantalla de ingreso de contraseña de la base de datos PostgreSQL. Ingrese la contraseña. Presione el botón siguiente

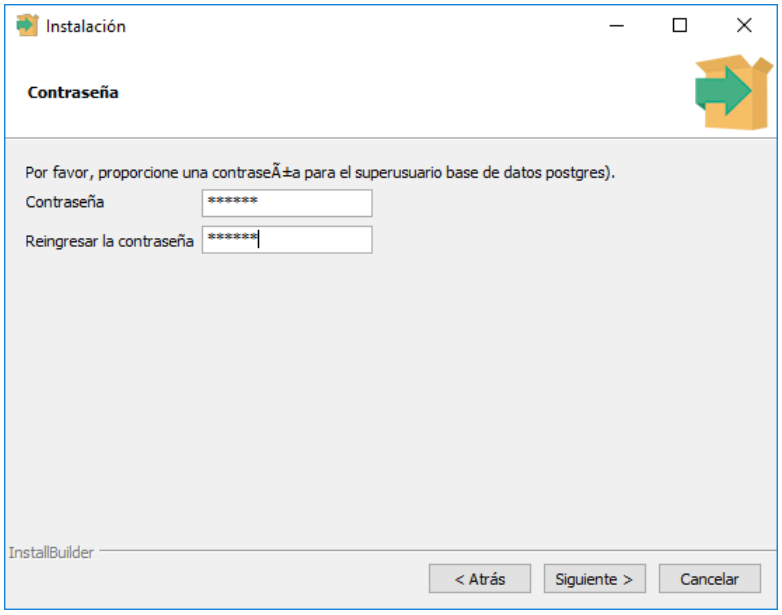


Figura 39-2: Pantalla de ingreso de contraseña PostgreSQL
Realizado por: Caluquí Jorge, 2018

4. Pantalla de ingreso asignación de puerto. Cámbielo si se requiere. Presione el botón siguiente

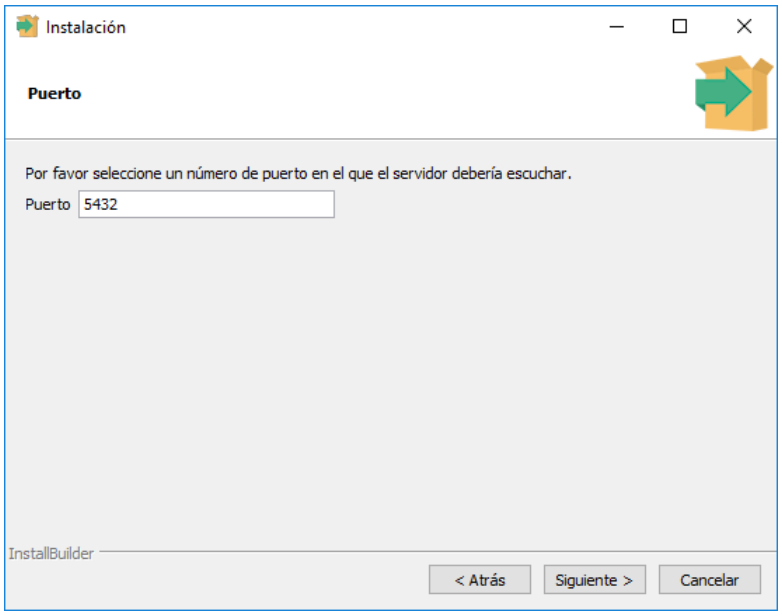


Figura 40-2: Pantalla de selección de puerto PostgreSQL
Realizado por: Caluquí Jorge, 2018

5. Pantalla de selección de la configuración regional. Cámbielo si se requiere. Presione el botón siguiente

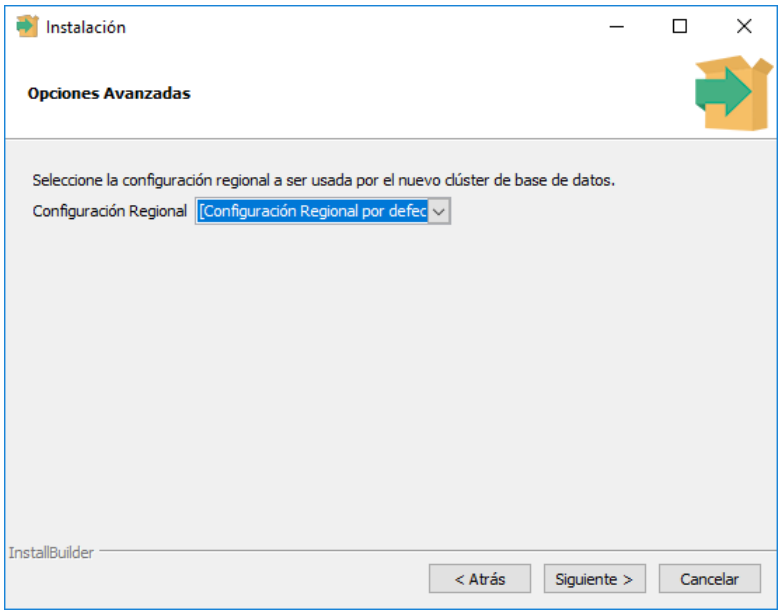


Figura 41-2: Pantalla de configuración regional PostgreSQL
Realizado por: Caluquí Jorge, 2018

6. Pantalla de confirmación de instalación, si está seguro de la instalación presione el botón siguiente

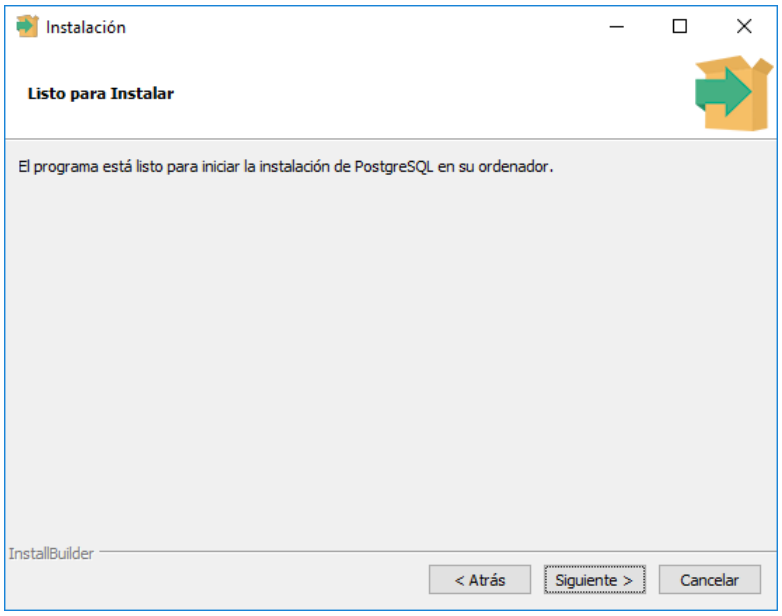


Figura 42-2: Pantalla de resumen de instalación PostgreSQL
Realizado por: Caluquí Jorge, 2018

7. Mientras se descargan los paquetes a su ordenador aparecerá la pantalla de progreso de instalación.

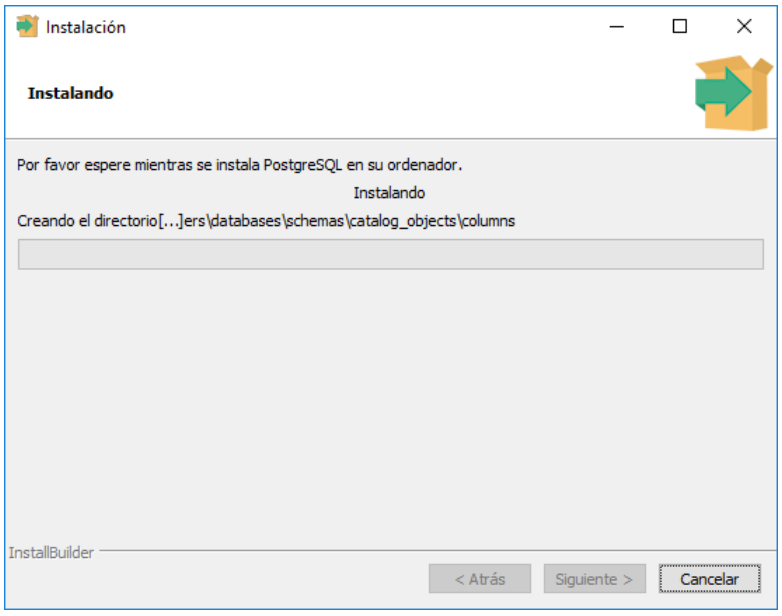


Figura 43-2: Pantalla de progreso de instalación PostgreSQL
Realizado por: Caluquí Jorge, 2018

8. Pantalla de confirmación de finalización de instalación. Presione el botón terminar.

Figura 44-2: Pantalla de fin de instalación PostgreSQL
Realizado por: Caluquí Jorge, 2018

9. Al ingresar a la aplicación se presenta una pantalla como la que se muestra a continuación.

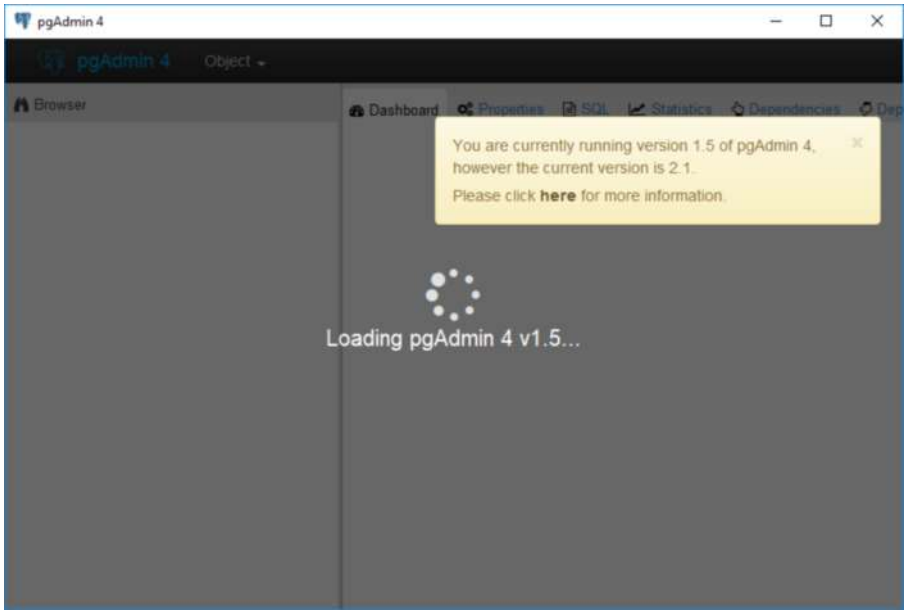


Figura 45-2: Pantalla de inicio de aplicación PostgreSQL
Realizado por: Caluquí Jorge, 2018

10. Para acceder a la base de datos, es necesario el ingreso de la contraseña registrada al momento de la instalación, si no posee una solicítela al administrador.

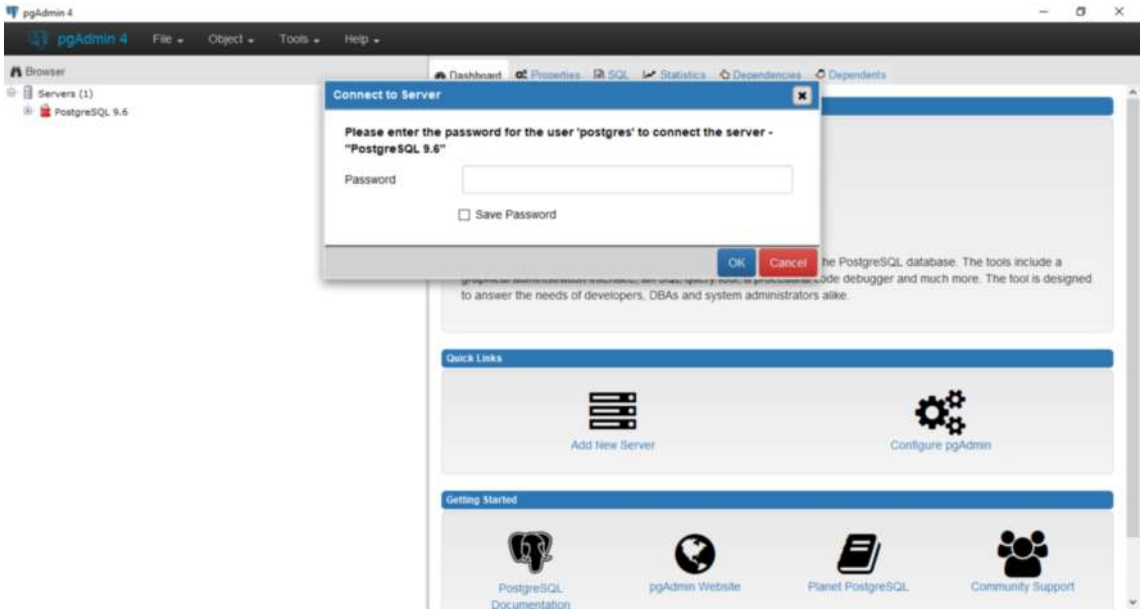


Figura 46-2: Pantalla de login PostgreSQL
Realizado por: Caluquí Jorge, 2018

11. Una vez abierta la conexión se presenta la pantalla, con dashboard que representan el rendimiento y productividad de la base de datos.

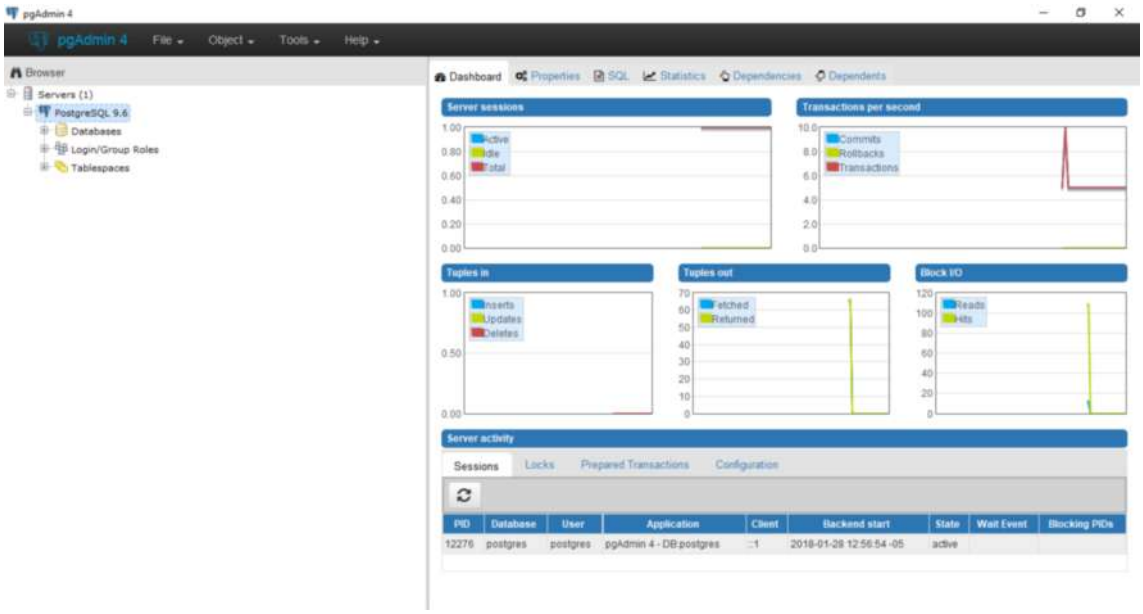


Figura 47-2: Pantalla de inicio de aplicación PostgreSQL
Realizado por: Caluquí Jorge, 2018

12. La pantalla que se presenta a continuación permite la ejecución de scripts que creen, actualicen, eliminen objetos en la base de datos.

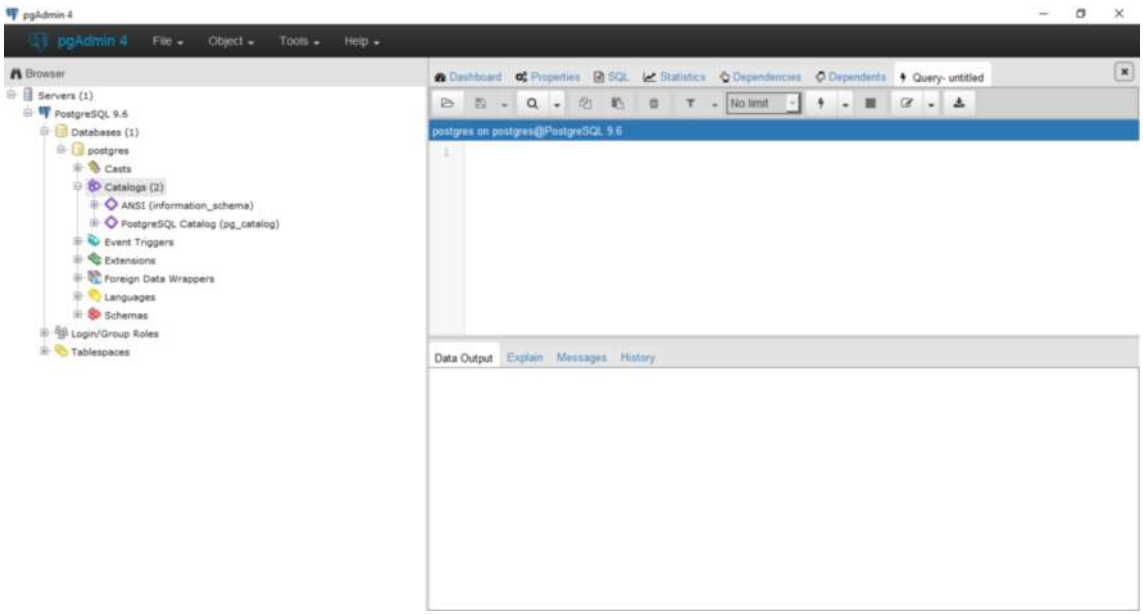


Figura 48-2: Pantalla de jerarquía estructural de base de datos PostgreSQL
Realizado por: Caluquí Jorge, 2018

2.4.2.3. Creación de esquema de base de datos

1. Creación de esquema de base de datos

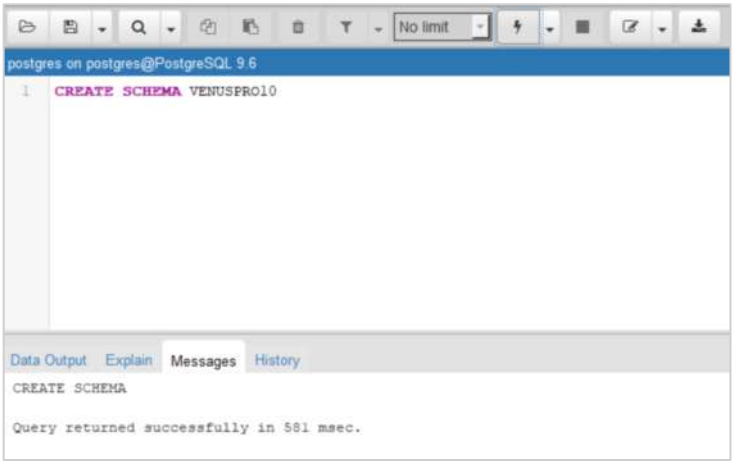


Figura 49-2: Pantalla de creación de esquema PostgreSQL
Realizado por: Caluquí Jorge, 2018

2.5. Implementación

Instrucciones de migración desde Oracle 11g XE a PostgreSQL

Tipo de datos

La definición de tipos de datos es fundamental para la creación de columnas de una tabla, esta especifica el tipo de información (carácter, número o fecha) que la columna albergará.

- **Tipos de datos Numérico**

Este tipo de dato entero es el más empleado, ya que ofrece un mejor equilibrio entre: rango, tamaño de almacenamiento y rendimiento; los tipos smallint, integer y bigint almacenan el número entero, número sin componentes fraccionarios, de varios rangos.

- **Tipos de datos carácter**

Los tipos de datos carácter almacenan data alfanumérica como palabras y texto sin formato, esta data es almacenada en cadenas de valores bytes correspondientes a un juego de caracteres, tal como ASCII ó EBCDIC, que es especificado al momento de crear la base de datos.

- **Tipos de datos fechas, hora e intervalo**

Este tipo de datos almacenan la fecha y hora, dependiendo de su funcionalidad.

Para la migración de los tipos de datos desde Oracle hacia Postgres es importante tener en cuenta la siguiente tabla como referencia para el cambio.

Tabla 4-2: Migración tipos de datos

ORACLE	POSTGRESQL
NUMBER (p)	SMALLINT – 2 bytes
	INTEGER – 4 bytes
	BIGINT – 8 bytes
NUMBER (p , s)	NUMERIC (p , s)
	REAL – 4 bytes, 6 decimales
	DOUBLE PRECISION – 8 bytes, 15 decimales
VARCHAR2 (size) tamaño máximo 4000	CHARACTER VARYING (n) – tamaño máximo 1 GB VARCHAR (n) como un alias
CHAR (size) tamaño máximo 2000	CHARACTER (n) – tamaño máximo 1 GB CHAR (n)
LONG – caracter variable tamaño mayor 2 GB	TEXT – longitud variable mayor a 1GB
DATE – Fecha y hora	TIMESTAMP

TIMESTAMP	
No hay coincidencia	DATE
No hay coincidencia	TIME
No hay coincidencia	BIT (n) Cadena de longitu de fija de 0 y 1

Realizado por: Caluquí Jorge, 2018

Instrucciones generales

Tabla 5-2: Migración de instrucciones generales

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
CREATE DATABASE	Si	
CREATE TABLE	No	
CREATE [OR REPLACE] VIEW	No	
CREATE [OR REPLACE] FUNCTION	Si	Reemplaza a los procedimientos
ALTER TABLE	No	
DROP TABLE	No	

Realizado por: Caluquí Jorge, 2018

Nota

- Cambiar la cláusula DEFAULT de las columnas de 'fecha' de defualt (sysdate) a defualt (CURRENT_TIMESTAMP)

a) CREATE TABLE

CREATE TABLE crea automáticamente un tipo de datos que representa el tipo de composición correspondiente a una fila, por lo tanto, las tablas no pueden tener el mismo nombre que cualquier tipo de datos existente en el mismo esquema.

Tabla 6-2: Migración de instrucciones para creación de tablas.

ORACLE	POSTGRESQL
CREATE TABLE	
CREATE TABLE nombre_tabla (nombre_columna1 tipo_dato1 [null | not null] , nombre_columna2 tipo_dato2 [null | not null] , nombre_columna3 tipo_dato3 [null | not null] , . . .)	CREATE nombre_tabla ([{ nombre_columna tipo_dato [DEFAULT expr_defecto] }])
ALTER TABLE nombre_tabla ADD CONSTRAINT PK_nombre_tabla PRIMARY KEY (columnaX, columnaY, …) ALTER TABLE nombre_tabla ADD CONSTRAINT U_nombre_tabla_nombre_columna UNIQUE (columnaX, columnaY, …)	CREATE nombre_tabla ([{ nombre_columna tipo_dato [DEFAULT expr_defecto] [columna_restriccion [...]] | table_resctriccion | LIKE table_padre [like_option ...] } [, ...]])

ALTER TABLE nombre_tabla ADD CONSTRAINT FK_nombre_tabla_tabla_referencial FOREING KEY (columnaX, columnaY, …) REFERENCE tabla_referenciada ALTER TABLE nombre_tabla ADD CONSTRAINT CK_ nombre_tabla_nombre_columna CHECK (condición) ALTER TABLE nombre_tabla MODIFY (columnaX DEFAULT valor_predeterminado)	Donde: columna_restriccion; [CONSTRAINT nombre_restriccion] { NOT NULL | NULL | CHECK (expresión) | DEFAULT default_expr | UNIQUE index_parameters | PRIMARY KEY index_parameters | REFERENCES reftable [(refcolumn)] [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] [ON DELETE action] [ON UPDATE action] }

Realizado por: Caluquí Jorge, 2018

b) ALTER TABLE

ALTER TABLE permite que las tablas puedan ser modificadas; es posible añadir o eliminar columnas, así como modificar las propiedades de la misma; también se pueden añadir, modificar o eliminar restricciones.

Tabla 7-2: Migración de instrucciones para actualización de tablas

ORACLE	POSTGRESQL	CONVERSION
ALTER TABLE [ONLY] nombre_tabla [*] acción [, ...]		No
ADD	ADD [COLUMN] nombre_columna tipo_dato [columna_restriccion [...]]	No
DROP	DROP [COLUMN] [IF EXISTS] nombre_columna [RESTRICT | CASCADE]	No
ALTER	ALTER [COLUMN] nombre_columna [SET DATA] TYPE tipo_dato ALTER [COLUMN] nombre_columna SET DEFAULT expresión ALTER [COLUMN] nombre_columna DROP DEFAULT ALTER [COLUMN] nombre_columna { SET | DROP } NOT NULL	No
DROP CONSTRAINT	DROP CONSTRAINT [IF EXISTS] nombre_restriccion [RESTRICT | CASCADE]	No

Realizado por: Caluquí Jorge, 2018

c) CREATE [OR REPLACE] VIEW

Tabla 8-2: Migración de instrucciones para creación de vistas.

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
FORCE	No	

Realizado por: Caluquí Jorge, 2018

Nota

- La sintaxis OUTER JOIN de Oracle Style debe cambiarse a ANSI: {LEFT|RIGHT} [OUTER] JOIN Tenga cuidado cuando hay muchas tablas en la cláusula FROM.

d) CREATE [OR REPLACE] FUNCTION

Tabla 9-2: Migración de instrucciones para funciones

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
RETURNS void AS $$	Si	No hay coincidencia
DECLARE	Si	No hay coincidencia
$$ LANGUAGE plpgsql;	Si	No hay coincidencia

POSTGRESQL		ORACLE	
CREATE OR REPLACE FUNCTION		CREATE OR REPLACE FUNCTION | PROCEDURE	
param datatype(length)	Definición de parámetros de entrada	param datatype	No especifica tamaño
RETURNS **datatype**	Retorna un tipo de dato	RETURN **datatype**	
RETURNS void	No retorna valores	Converted to CREATE PROCEDURE	
$$ or $**body**$	Inicio del cuerpo	Removed	
DECLARE keyword	Declaración de variables	Removed	
SELECT **exp** INTO **var**	Asigna un valor	SELECT **exp** INTO **var** FROM dual	
TRUNCATE TABLE **name**	Trunca una tabla	EXECUTE IMMEDIATE 'TRUNCATE TABLE **name**'	
$$ LANGUAGE plpgsql	Fin del cuerpo	/	
VOLATILE		Removed	
COST **num**		Removed	

Realizado por: Caluquí Jorge, 2018

Operaciones de mantenimiento de datos

a) INSERT

Tabla 10-2: Migración de instrucciones para inserción de datos

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
INSERT [INTO] nombre_tabla [(lista_de_columnas)] VALUES (lista_de_valores)		
INSERT de cláusula "SUBQUERY"	No	
INSERT de cláusula "VALUES(...)"	No	
INSERT INTO TABLE	No	
IF THEN ELSE	No	

Realizado por: Caluquí Jorge, 2018

b) DELETE

Tabla 11-2: Migración de instrucciones para eliminación de datos

CLÁUSULA	CONVERSIÓN	DETALLES
DELETE [FROM] nombre_tabla [WHERE condición_de_las_filas_a_eliminar]		
DELETE FROM TABLE	No	

Realizado por: Caluquí Jorge, 2018

c) SELECT

Tabla 12-2: Migración de instrucciones para consultas

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
AS	Si	No hay coincidencia
CASE	Si	Reemplaza a la instrucción DECODE
COALESCE	Si	Reemplaza a la instrucción NVL
GROUP BY	No	
HAVING	No	
INTO	No	
ORDER BY	No	
Pseudocolumna ROWNUM	No	
SELECT ALL	No	
SELECT DISTINCT	No	
WHERE(field_name1,…,fieldnameN) IN (subquery_with_multiple_fields)	No	

WHERE comparison_condition (=;<>;!=;<;>;>=;<=;ANY;SOME;ALL)	No	
WHERE field_name BETWEEN arg1 AND arg2	No	
WHERE field_name IN (const1,...,constN)	No	
WHERE field_name IN (subconsulta)	No	
WHERE like_condition (operador LIKE)	No	

POSTGRESQL		**ORACLE**	
FROM table AS alias	Es opcional la palabra AS dentro FROM	FROM table alias	AS No es permitida
SELECT a, CASE WHEN a=1 THEN 'one' WHEN a=2 THEN 'two' ELSE 'other' END FROM test		SELECT producto_id, DECODE (almacen_id, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle', 'Non-domestic') quantity_on_hand FROM inventarios	
SELECT empleado_id, COALESCE(fecha_contrato, 'now'::datetime) FROM empleado WHERE empleado_id = 10;		SELECT empleado_id, NVL(fecha_contrato, sysdate) FROM empleado WHERE empleado_id = 10	

Realizado por: Caluquí Jorge, 2018

Predicados

Tabla 13-2: Migración de instrucciones para predicados

CLÁUSULA	CONVERSIÓN
Condiciones booleanas	No
Condiciones de comparación: '=, <>, <, <=, >, >=, ANY, SOME, ALL	No
Condiciones Exists	No
Condiciones In	No
Condiciones Null	No
Condiciones de coincidencia de patrones	No
Condiciones Range	No

Realizado por: Caluquí Jorge, 2018

Funciones

a) FUNCIONES NUMÉRICAS

Tabla 14-2: Migración de funciones numéricas

CLÁUSULA	CONVERSIÓN	DETALLES
ABS(número)	No	
EXP (n)	No	
FLOOR(número)	No	
MOD(dividendo, divisor)	No	
NVL(n2, n1)	Si	PostgreSQL no admite la función
POWER(valor, n)	No	
ROUND (número, entero)	No	
SQRT(número)	No	
TRUNC (número)	No	

Realizado por: Caluquí Jorge, 2018

b) FUNCIONES DE CARACTERES

Tabla 15-2: Migración de funciones caracter

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
CHR(número)	No	
CONCAT(carácter1, carácter2)	No	
INITCAP(cadena)	No	
LOWER(cadena)	No	

LTRIM(cadena)	No	
REPLACE(cadena, búsqueda)	No	
REPLACE(cadena, búsqueda, reemplazar)	No	
RTRIM(cadena)	No	
SUBSTR(cadena, pos, lon)	No	
LENGTH(cadena)	No	
TRANSLATE(cadena, de, a)	No	
TRIM([type trim FROM] cadena)	No	
UPPER(cadena)	No	

Realizado por: Caluquí Jorge, 2018

c) FUNCIONES DE FECHA Y HORA

Tabla 16-2: Migración de funciones fecha y hora

CLÁUSULA	CONVERSIÓN	DETALLES
ADD_MONTHS(fecha, número)	No	
EXTRACT(DAY FROM fecha)	No	
EXTRACT(HOUR FROM fecha)	No	
EXTRACT(MINUTE FROM hora)	No	
EXTRACT(MONTH FROM fecha)	No	
EXTRACT(SECOND FROM hora)	No	
EXTRACT(YEAR FROM fecha)	No	
LAST_DAY(date)	No	
ROUND (fecha)	Si	PostgreSQL no admite la función
SYSDATE	Si	CURRENT_DATE
SYSTIMESTAMP	Si	CURRENT_TIME/ CURRENT_TIMESTAMP
TO_CHAR (datetime, formato)	No	
TRUNC (datetime)	Si	PostgreSQL no admite la función

Realizado por: Caluquí Jorge, 2018

d) FUNCIONES RELACIONADAS CON NULL

Tabla 17-2: Migración de funciones relacionadas con null

CLÁUSULA	CONVERSIÓN	DETALLES
COALESCE(exp1, exp2, …)	Sí	
NVL	No	PostgreSQL no admite la función
NULLIF(exp1, exp2)	Sí	
NVL(exp, reemplazo)	Sí	
NVL2(exp1, exp2, exp3)	Sí	

Realizado por: Caluquí Jorge, 2018

e) FUNCIONES AGRUPADAS

Tabla 18-2: Migración de funciones agrupadas

CLÁUSULA	CONVERSIÓN	OBSERVACIÓN
AVG	Sí	
COUNT	Sí	
MAX	Sí	
MIN	Sí	
ROW_NUMBER	Sí	
SUM	Sí	

Realizado por: Caluquí Jorge, 2018

2.6. Comparativo de los SGDB de OracleDataBase XE y PostgreSQL

Para la realización de la comparativa entre los SGDB de OracleDataBase XE y PostgreSQL vamos a crear un esquema de cada estructura de la base de datos Oracle con su correspondiente en Postgres estableciendo la comparativa de las afectaciones tanto en instrucciones como en tipos de datos, para este proceso se ha utilizado una columna denominada estado donde se establece la veracidad del cambio de una estructura a otra.

a) CREACIÓN DE TABLA: La tabla T_AREAACADEMICA_K se encarga del almacenamiento de las diferentes áreas que posee una malla curricular para el Sistema.

Tabla 19-2: Comparativa de creación de tabla

ORACLE	POSTGRESQL	ESTADO
T_AREAACADEMICA_K	T_AREAACADEMICA_K	
CREATE TABLE "T_AREAACADEMICA_K" (	CREATE TABLE "T_AREAACADEMICA_K" (	VERDADERO
"AREACODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_AREACODIGO" CHECK (AREACODIGO>0) ENABLE,	"AREACODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_AREACODIGO" CHECK ("AREACODIGO">0),	FALSO
"AREANOMBRE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"AREANOMBRE" VARCHAR(80) NOT NULL,	FALSO
"AREADETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"AREADETALLE" VARCHAR(80) NOT NULL,	FALSO
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO
"AREAORDEN" NUMBER(2,0) NOT NULL ENABLE,	"AREAORDEN" SMALLINT NOT NULL,	FALSO
PRIMARY KEY ("ANICODIGO", "AREACODIGO"));	PRIMARY KEY ("ANICODIGO", "AREACODIGO"),	VERDADERO
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO
ALTER TABLE "T_AREAACADEMICA_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO

Realizado por: Caluquí Jorge, 2018

b) INSTRUCCIONES DE MANTENIMIENTO DE DATOS: Se determina las estructuras para la inserción, actualización y eliminación de datos almacenados en la base de datos.

Tabla 20-2: Comparativa de estructura de inserción de datos

ORACLE	POSTGRESQL	ESTADO
S_INSERT_AREAACADEMICA_K	S_INSERT_AREAACADEMICA_K	
CREATE OR REPLACE PROCEDURE "S_INSERT_AREAACADEMICA_K" (	CREATE OR REPLACE FUNCTION "S_INSERT_AREAACADEMICA_K" (	FALSO
NOMBRE IN VARCHAR2,	NOMBRE IN VARCHAR,	FALSO
DETALLE IN VARCHAR2,	DETALLE IN VARCHAR,	FALSO
ANIO IN NUMBER,	ANIO IN SMALLINT,	FALSO
ORDEN IN NUMBER)	ORDEN IN SMALLINT)	FALSO
	RETURNS void AS $$	FALSO
IS		FALSO
	DECLARE	FALSO
VCODIGO NUMBER(2,0);	VCODIGO SMALLINT;	FALSO
BEGIN	BEGIN	VERDADERO
SELECT NVL(MAX(AREACODIGO),0)	SELECT COALESCE(MAX("AREACODIGO"),0)	FALSO
INTO VCODIGO	INTO VCODIGO	VERDADERO
FROM "T_AREAACADEMICA_K"	FROM "T_AREAACADEMICA_K"	VERDADERO
WHERE "ANICODIGO" = ANIO;	WHERE "ANICODIGO" = ANIO;	VERDADERO
INSERT INTO "T_AREAACADEMICA_K"(	INSERT INTO "T_AREAACADEMICA_K"(	VERDADERO
"AREACODIGO",	"AREACODIGO",	VERDADERO
"AREANOMBRE",	"AREANOMBRE",	VERDADERO
"AREADETALLE",	"AREADETALLE",	VERDADERO
"ANICODIGO",	"ANICODIGO",	VERDADERO
AREAORDEN)	"AREAORDEN")	FALSO
VALUES(	VALUES(	VERDADERO
VCODIGO+1,	VCODIGO+1,	VERDADERO
NOMBRE,	NOMBRE,	VERDADERO
DETALLE,	DETALLE,	VERDADERO
ANIO,	ANIO,	VERDADERO
ORDEN);	ORDEN);	VERDADERO
END;	END;	VERDADERO
	$$ LANGUAGE plpgsql;	FALSO

Realizado por: Caluquí Jorge, 2018

Tabla 21-2: Comparativa de estructura de actualización de datos

ORACLE	POSTGRESQL	ESTADO
S_UPDATE_AREAACADEMICA_K	S_UPDATE_AREAACADEMICA_K	
CREATE OR REPLACE PROCEDURE "S_UPDATE_AREAACADEMICA_K" (	CREATE OR REPLACE FUNCTION "S_UPDATE_AREAACADEMICA_K" (	FALSO
CODIGO IN NUMBER,	CODIGO IN SMALLINT,	FALSO
NOMBRE IN VARCHAR2,	NOMBRE IN VARCHAR,	FALSO
DETALLE IN VARCHAR2,	DETALLE IN VARCHAR,	FALSO
ANIO IN NUMBER,	ANIO IN SMALLINT,	FALSO
ORDEN IN NUMBER)	ORDEN IN SMALLINT)	FALSO
	RETURNS void AS $$	FALSO
IS		FALSO
BEGIN	BEGIN	VERDADERO
UPDATE T_AREAACADEMICA_K	UPDATE "T_AREAACADEMICA_K"	FALSO
SET	SET	VERDADERO
"AREANOMBRE "= NOMBRE,	"AREANOMBRE "= NOMBRE,	VERDADERO
"AREADETALLE "= DETALLE,	"AREADETALLE "= DETALLE,	VERDADERO
"AREAORDEN "= ORDEN	"AREAORDEN "= ORDEN	VERDADERO
WHERE "ANICODIGO" = ANIO AND "AREACODIGO" = CODIGO;	WHERE "ANICODIGO" = ANIO AND "AREACODIGO" = CODIGO;	VERDADERO
END;	END;	VERDADERO
	$$ LANGUAGE plpgsql;	FALSO

Realizado por: Caluquí Jorge, 2018

Tabla 22-2: Comparativa de estructura de eliminación de datos

ORACLE	POSTGRESQL	ESTADO
S_DELETE_AREAACADEMICA_K	S_DELETE_AREAACADEMICA_K	
CREATE OR REPLACE PROCEDURE "S_DELETE_AREAACADEMICA_K" (	CREATE OR REPLACE FUNCTION "S_DELETE_AREAACADEMICA_K" (	FALSO
CODIGO IN NUMBER,	CODIGO IN SMALLINT,	FALSO
ANIO IN NUMBER)	ANIO IN SMALLINT)	FALSO
	RETURNS void AS $$	FALSO
IS		FALSO
BEGIN	BEGIN	VERDADERO
DELETE	DELETE	VERDADERO
FROM "T_AREAACADEMICA_K"	FROM "T_AREAACADEMICA_K"	VERDADERO
WHERE "ANICODIGO" = ANIO AND "AREACODIGO" = CODIGO;	WHERE "ANICODIGO" = ANIO AND "AREACODIGO" = CODIGO;	VERDADERO
END;	END;	VERDADERO
	$$ LANGUAGE plpgsql;	FALSO

Realizado por: Caluquí Jorge, 2018

c) CREACION DE FUNCIONES: Se muestra el formato de la estructura de cómo debe ser creada una función en la base de datos, en este caso la función me retornara el año lectivo activo.

Tabla 23-2: Comparativa de una función.

ORACLE	POSTGRESQL	ESTADO
F_ANIO_LECTIVO_W	F_ANIO_LECTIVO_W	
CREATE OR REPLACE FUNCTION public."F_ANIO_LECTIVO_W"()	create or replace FUNCTION "F_ANIO_LECTIVO_W"	FALSO
RETURNS smallint	RETURN NUMBER	FALSO
LANGUAGE 'plpgsql'		FALSO
COST 100		FALSO
VOLATILE		FALSO
AS $BODY$		FALSO
DECLARE	IS	FALSO
V_ANIO SMALLINT;	V_ANIO NUMBER(3, 0);	FALSO
BEGIN	BEGIN	VERDADERO
V_ANIO := 0;	V_ANIO := 0;	VERDADERO
SELECT "ANICODIGO" INTO V_ANIO	SELECT "ANICODIGO" INTO V_ANIO	VERDADERO
FROM "T_ANIOLECTIVO_K"	FROM "T_ANIOLECTIVO_K"	VERDADERO
WHERE "ANIESTADO" = 1;	WHERE "ANIESTADO" = 1;	VERDADERO
RETURN V_ANIO;	RETURN V_ANIO;	VERDADERO
END;	END;	VERDADERO
$BODY$;		FALSO

Realizado por: Caluquí Jorge, 2018

d) CREACION DE VISTAS: Se muestra el formato a utilizar en el momento de crear una vista para obtener datos de la base, esta vista obtiene un listado de cursos.

Tabla 24-2: Comparativa de una vista.

ORACLE	POSTGRESQL	ESTADO
V_LISTADO_CURSOS_K	V_LISTADO_CURSOS_K	
CREATE VIEW "V_LISTADO_CURSOS_K" AS	CREATE VIEW public."V_LISTADO_CURSOS_K" AS	FALSO
SELECT "C"."DOCCODIGO",	SELECT "C"."DOCCODIGO",	VERDADERO
"D"."DOCIDENTIFICACION",	"D"."DOCIDENTIFICACION",	VERDADERO
"D"."DOCNOMBRES",	"D"."DOCNOMBRES",	VERDADERO
"D"."DOCALIAS",	"D"."DOCALIAS",	VERDADERO
"D"."DOCDIRECCION",	"D"."DOCDIRECCION",	VERDADERO
"D"."DOCFONOMOVIL",	"D"."DOCFONOMOVIL",	VERDADERO
"D"."DOCEMAIL",	"D"."DOCEMAIL",	VERDADERO
"A"."ANIDETALLE",	"A"."ANIDETALLE",	VERDADERO

"A"."ANICODIGO",	"A"."ANICODIGO",	VERDADERO
"C"."CURCODIGO",	"C"."CURCODIGO",	VERDADERO
"C"."CURDETALLE",	"C"."CURDETALLE",	VERDADERO
"C"."CURPARALELO",	"C"."CURPARALELO",	VERDADERO
"C"."CURSECUENCIA",	"C"."CURSECUENCIA",	VERDADERO
"C"."CURSECCION",	"C"."CURSECCION",	VERDADERO
"C"."CURSUBNIVEL",	"C"."CURSUBNIVEL",	VERDADERO
"C"."CURTIPO"	"C"."CURTIPO"	VERDADERO
FROM "T_ANIOLECTIVO_K" "A"	FROM "T_ANIOLECTIVO_K" "A"	FALSO
INNER JOIN "T_CURSO_K" "C" ON "A"."ANICODIGO" = "C"."ANICODIGO"	JOIN "T_CURSO_K" "C" ON "A"."ANICODIGO" = "C"."ANICODIGO"	FALSO
INNER JOIN "T_DOCENTE_S" "D" ON "D"."DOCCODIGO" = "C"."DOCCODIGO";	JOIN "T_DOCENTE_S" "D" ON "D"."DOCCODIGO" = "C"."DOCCODIGO";	FALSO

Realizado por: Caluquí Jorge, 2018

OTRAS CONSIDERACIONES

Postgres no es CASE-SENSITIVE, es decir, no diferencia mayúsculas de minúsculas como otros lenguajes de programación como C o Java. Sin embargo, debemos recordar que *Oracle* es CASE-SENSITIVE en las búsquedas de texto

Palabra clave AS necesaria para los alias de columna en Postgres. La sugerencia es modificar incluso el código de Oracle para utilizar esta convención, ya que esta convención, que es estándar, es compatible con muchas bases de datos y causará menos dolor a cualquiera que trate de corregir un error en ambas versiones (Oracle y Postgres) de código.

No use palabras clave como identificadores. Nuevamente, esta es una sugerencia de la convención de codificación de que debemos abstenernos de usar palabras clave como 'fecha', 'hora', 'marca de tiempo' como columna o nombres de objetos.

Oracle trata {quote}{quote} (empty string) como nulo, pero Postgres no lo hace. Por lo tanto, haga que la aplicación lo advierta, o adjunte un trigger ANTES DE CADA FILA en cada tabla que tenga una columna char / varchar, y establezca 'if new.char_col = {quote}{quote}then new.char_col = null'.

Oracle permite ordenar por columnas que no están en la lista de selección, pero Postgres necesita tener a la columna ahí.

Tenga cuidado con las operaciones date + int en la aplicación; significan cosas diferentes en Oracle y Postgres.

La subconsulta en la cláusula SET del comando UPDATE que actualiza varias columnas no es compatible con Postgres.

En Postgres existe la posibilidad de hacer SELECT ... sin ninguna tabla. Existe la tabla DUAL falsa para que pueda hacer SELECT ... FROM DUAL en ambas plataformas.

Postgres no permite saltarse palabra clave FROM en DELETE FROM nombre_tabla

Manejo de tipo de datos

Se debe tener especial cuidado al mapear los tipos de datos. Se recomiendan los siguientes mapeos genéricos, pero es posible que necesitemos algunas iteraciones para lograr un rendimiento óptimo sin perder el comportamiento de la aplicación y la portabilidad.

- varchar2, nvarchar2, clob, long -> varchar (or text)
- nchar(p) -> char(n)
- number -> numeric (alternativas: bigint, int, smallint, float, double, real)
- date -> timestamp (porque 'date' en PG almacena el componente de tiempo como 00:00:00); Elija cuidadosamente, ¿qué necesita la aplicación? fecha, hora o fecha + hora?

Operadores

El conjunto de operadores y funciones SQL es muy similar, ambos DBMS tienen la concatenación operador "||", substr, upper, to_char y otras funciones con la misma sintaxis.

Migración de Secuencias

Las secuencias están disponibles tanto en Oracle como en Postgres, con pequeñas diferencias.

- En lugar de seq_name.nextval, en Postgres deberíamos hacer nextval ('seqname').
- La cláusula MAXVAL del comando CREATE SEQUENCE en Postgres es sensible al tipo de datos subyacente.

Migración de Procedimientos / Funciones / Trigger

- Valores predeterminados para los parámetros.
- COMMIT / ROLLBACK dentro de PL / PGSQL. Los límites de transacción dentro de PL / PGSQL no son compatibles.
- SAVEPOINTs en PL / PGSQL 'son' compatibles con la función de manejo EXCEPTION.

- El mecanismo RAISE de la excepción de rastreos debe migrarse para usar RAISE EXCEPTION.
- Declaraciones de variables FOR . LOOP implícitas. PL / SQL no requiere una declaración explícita de las variables utilizadas para LOOPing. Tenemos que declarar esas variables explícitamente en PL / PGSQL. El tipo de datos para CURSOR-FOR LOOP debe ser RECORD.
- Se admite el uso de% TYPE en los tipos de datos de parámetros.
- La función NVL no existe en PG, debe reemplazarlo con la función COALESCE que hace el mismo trabajo en ambas plataformas: coalesce(expr1, expr2, expr3,....), la cual devuelve la primera expresión no nula.
- La función TO_NUMBER (cadena) no se reconoce en PG. Tenemos que usar la versión TO_NUMBER (cadena, cadena) - con el segundo parámetro, que es el formato numérico. Esto es lo mismo que en Oracle.
- La función TO_DATE (cadena, cadena) devuelve la fecha (es decir, sin horas / minutos / segundos) en PG. Tenemos que usar TO_TIMESTAMP (cadena, cadena) que devuelve tanto la fecha como la hora del día tanto en Oracle como en PG.

CAPÍTULO III

3. MARCO DE RESULTADOS, ANÁLISIS Y DISCUSIÓN

3.1. Diseño de la Investigación

La investigación a ejecutarse es de tipo descriptivo – investigativo ya que pretende establecer la metodología necesaria para la migración del Sistema VenusPro10 desde el motor de base Oracle 11g XE, hacia el motor de base de datos PostgreSQL. Sin limitar la experiencia solo al enfoque de recolección de datos, sino también el establecimiento de un estudio de compatibilidad que garantice una alternativa de migración rápida ofreciendo un rango de afectación mínima tanto en los datos, así como al aplicativo.

La expresión de los datos es de tipo cualitativo que ayudarán a identificar los factores importantes que deben ser medidos.

3.2. Métodos

Se utilizará para este estudio los siguientes métodos de investigación:

Método Inductivo: debido que al observar particularmente las estructuras físicas y lógicas de los SGDB determinara el proceso adecuado en la migración.

Método Científico y de Observación: ya que se tendrá que estudiar los estándares en la creación de las estructuras físicas lógicas y tipos de datos, con el propósito de verificar la compatibilidad entre los SGDB

Método de Análisis: ya que se tendrá que analizar y estudiar dos SGDB en base a parámetros de evaluación definidos.

3.3. Técnicas

Se utilizarán ciertas técnicas para la recolección de información tales como:

- Observación la cual permitirá apreciar las distintas estructuras y objetos con las que se trabaja tanto en Oracle 11g XE como en PostgreSQL; así como sus scripts para la generación y mantenimiento de dichas estructuras.

- Comparación o Confrontación, que ayudará a identificar cada uno de los cambios que debe sufrir cada estructura u objeto al ser migrado de una base a otra
- Revisión Selectiva, la que permite seleccionar cada estructura a evaluarse, de acuerdo a la necesidad de migración que posee el proyecto
- Recopilación de información para el establecimiento de reglas que faciliten la migración de las estructuras, objetos y datos
- Pruebas, que establezcan niveles de compatibilidad entre estructuras, y la afectación sobre los datos existentes en la base

3.4. Procesamiento de la información

Para la realización del estudio de compatibilidad de las estructuras lógicas y físicas se ha tomado en cuenta que, según el diccionario de la Real Academia, la compatibilidad se la puede comprender como "Cualidad de compatible", y compatible se especifica en su segunda acepción como "Dicho de una persona o de una cosa: Que puede estar, funcionar o coexistir sin impedimento con otra". Esta definición también es aplicable al producto software. (Sánchez-Carrasco García, 2015)

La ISO/IEC 9126 define la compatibilidad como la capacidad de un producto software para ser transferido desde un entorno a otro. La compatibilidad es un factor importante a tener en cuenta a la hora de diseñar un producto software, ya que ahorra costes a la hora de transferir el producto, además de llegar a un mayor número de clientes potenciales. (Sánchez-Carrasco García, 2015).

En el numeral 6.6 de la norma ISO/IEC 9126-1, se detallan las características de la compatibilidad (Aquí definida como Portability). Define la portabilidad como "La capacidad del producto software para ser transferido de un entorno a otro". (Sánchez-Carrasco García, 2015)

- Adaptabilidad: La capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para ese propósito por el propio software considerado. (Sánchez-Carrasco García, 2015)

$$\frac{Número\ de\ componentes\ intactos}{Número\ total\ de\ componentes}$$

- Instalabilidad: La capacidad del producto software para ser instalado en un entorno especificado. (Sánchez-Carrasco García, 2015)

$$\frac{Numero\ de\ elementos\ para\ instalación\ o\ montaje}{Número\ total\ de\ componentes}$$

- Coexistencia: La capacidad del producto software para coexistir con otro software independiente, en un entorno común. (Sánchez-Carrasco García, 2015)

$$\frac{Numero\ de\ componentes\ comunes}{Número\ total\ de\ componentes}$$

- Capacidad para reemplazar: La capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito y en el mismo entorno. (Sánchez-Carrasco García, 2015)

$$\frac{Numero\ de\ componentes\ reemplazables}{Número\ total\ de\ componentes}$$

- Cumplimiento de la portabilidad: La capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad. (Sánchez-Carrasco García, 2015)

$$\frac{Numero\ de\ componentes\ portables}{Número\ total\ de\ componentes}$$

Para la incorporación de estos términos a esta evaluación, se tomará como el producto software a todas las estructuras de la base de datos (tablas, tipos de datos, procedimientos, funciones y los datos mismo). Una vez realizado la selección de los parámetros e indicadores de evaluación, se definen tablas comparativas, y tablas resumen por cada indicador planteado.

3.5. Población y Muestra

La población es el conjunto de elementos a ser evaluados, la cual constituye los objetos (tablas, vistas, procedimientos y funciones) a ser migrados desde el motor de base de datos Oracle 11g XE al motor de base de datos PostgreSQL; donde la muestra se encuentra formada por el universo entero de objetos de la base de datos del Sistema VenusPro10 de los módulos: Kernel, Registration y Score.

3.6. Análisis e interpretación de resultados

3.6.1. Adaptabilidad

Tabla 25-3: Indicador adaptabilidad

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Tipos de datos	0,01	0,04	0,01	**2,11**
Scripts de mantenimiento de tablas	0,28	0,56	0,28	**37,28**
Scripts de mantenimiento de vistas	0,53	0,62	0,63	**59,33**
Scripts de mantenimiento de procedimientos y funciones	0,48	0,57	0,58	**54,35**
Interfaz	0,60	0,68	0,57	**61,67**
TOTAL(%)	**38,00**	**49,33**	**41,52**	**42,95**

Realizado por: Caluquí Jorge, 2018

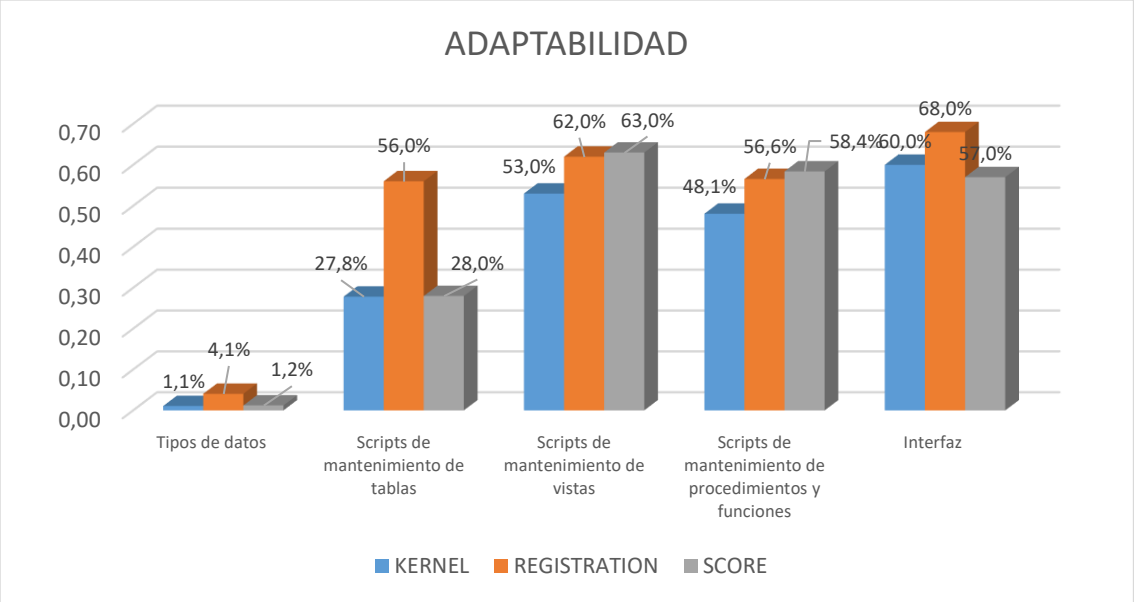


Gráfico 1-3: Indicador adaptabilidad
Realizado por: Caluquí Jorge, 2018

Interpretación

La adaptabilidad ha sido medida para los tres módulos del Sistema VenusPro10, en el gráfico anterior podemos notar de forma clara que los tipos de datos han sido los más afectados en esta migración alcanzando apenas un promedio de 2,11% en este indicador; mientras que la interfaz fue la menos afectada según este indicador, alcanzando un promedio de 61,7%, los scripts de mantenimiento de tablas alcanzaron un promedio de 37,28%, seguidos de los scripts de

mantenimiento de procesos con un promedio de 54,35%, y los scripts de mantenimiento de vistas con su promedio de 59,33%.

Para este indicador se alcanza un porcentaje del 42,95%; ya que las estructuras si sufrieron afectaciones en la migración, para lo cual se aplicaron acciones y mecanismos distintos de aquellos proporcionados por las herramientas de Oracle 11g XE, y propios de PostgreSQL.

3.6.2. Instalabilidad

Tabla 16-3: Indicador instalabilidad

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Tipos de datos	1,00	1,00	1,00	**100,00**
Scripts de mantenimiento de tablas	1,00	1,00	1,00	**100,00**
Scripts de mantenimiento de vistas	1,00	1,00	1,00	**100,00**
Scripts de mantenimiento de procedimientos y funciones	0,50	0,50	0,50	**50,00**
Interfaz	1,00	1,00	1,00	**100,00**
TOTAL(%)	**90,00**	**90,00**	**90,00**	**90,00**

Realizado por: Caluquí Jorge, 2018

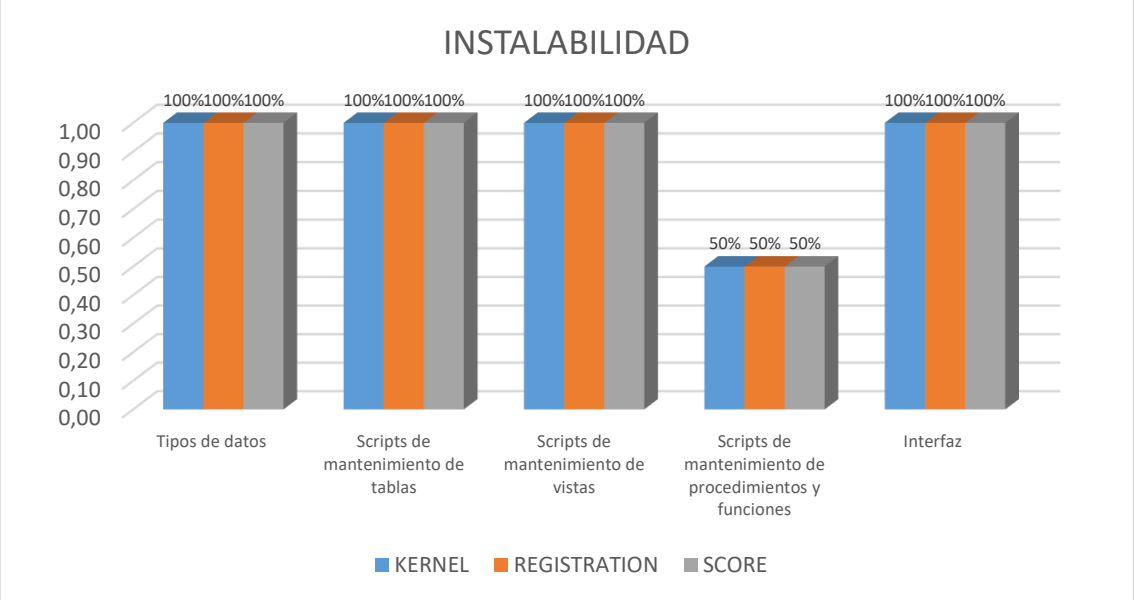


Gráfico 2-3: Indicador instalabilidad
Realizado por: Caluquí Jorge, 2018

Interpretación

Para el indicador de instalabilidad se consideró la posibilidad de migración de las estructuras, es decir, para los tipos de datos se alcanzó, un 100% debido a que todos fueron migrados, en algunos casos reemplazados por sus equivalentes, pero ninguna de las columnas fue retirada, en igual manera sucede con los scripts de mantenimiento de tablas y vistas.

Para los scripts de mantenimiento de procedimientos y funciones solo se obtuvo un 50% debido a que PostgreSQL no considera los procedimientos como tales, su estructura fue cambiada por una función, quitando el termino de procedimiento. A nivel de interfaz se alcanzó un 100% de instalabilidad.

El promedio de instabilidad es del 90%, ya que permite que la mayor parte de la estructura del Sistema VenusPro10

3.6.3. Coexistencia

Tabla 27-3: Indicador coexistencia

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Número de reportes	1,00	1,00	1,00	**100,00**
Carga de Data	1,00	1,00	1,00	**100,00**
Modificación de formato de Reporte	1,00	1,00	1,00	**100,00**
TOTAL(%)	**100,00**	**100,00**	**100,00**	**100,00**

Realizado por: Caluquí Jorge, 2018

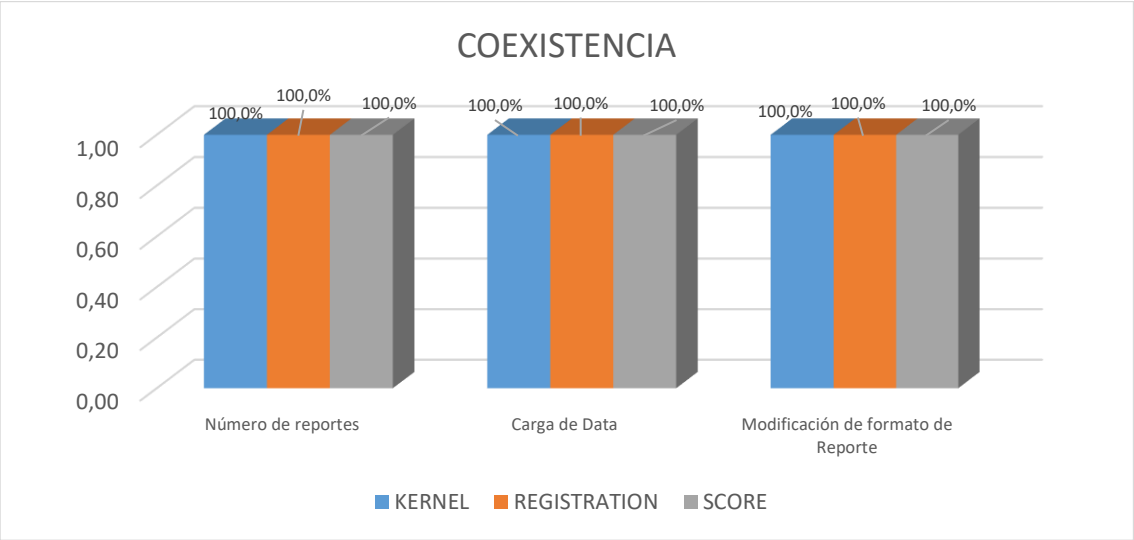


Gráfico 3-1: Indicador coexistencia
Realizado por: Caluquí Jorge, 2018

Interpretación

Para indicador de coexistencia se consideró la posibilidad de que PostgreSQL pueda ser funcional con Crystal Report para lo cual se tomó las siguientes variables de medición, la cantidad de reportes que presenta el sistema alcanzo un nivel del 100% debido a que se ejecutaron en el aplicativo conectado a Oracle y PostgreSQL sin ninguna diferencia, para la carga de data no se obtuvo ninguna diferencia al realizarlo con la nueva base de datos Postgres.

Por lo que su nivel alcanzo un 100%, y en el parámetro de formato de los reportes, los mismo no tuvieron alteración alguna dando como resultado una correcta funcionalidad al mostrar los datos obtenidos desde la nueva base de datos PostgreSQL obteniendo un nivel del 100%.

La coexistencia promedio de los tres módulos es del 100%, permitiendo que todos los reportes se puedan ejecutar y mostrar la información, sin ningún tipo de alteración al consultar en Oracle 11g XE como en PostgreSQL.

3.6.4. Capacidad para reemplazar

Tabla 28-3: Indicador capacidad para reemplazar

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Tipos de datos	0,96	0,96	0,92	**94,59**
Scripts de mantenimiento de tablas	0,68	0,67	0,60	**64,90**
Scripts de mantenimiento de vistas	0,35	0,32	0,35	**34,00**
Scripts de mantenimiento de procedimientos y funciones	0,30	0,27	0,30	**29,26**
Interfaz	0,90	0,95	0,93	**92,67**
TOTAL(%)	**63,79**	**63,45**	**62,00**	**63,08**

Realizado por: Caluquí Jorge, 2018

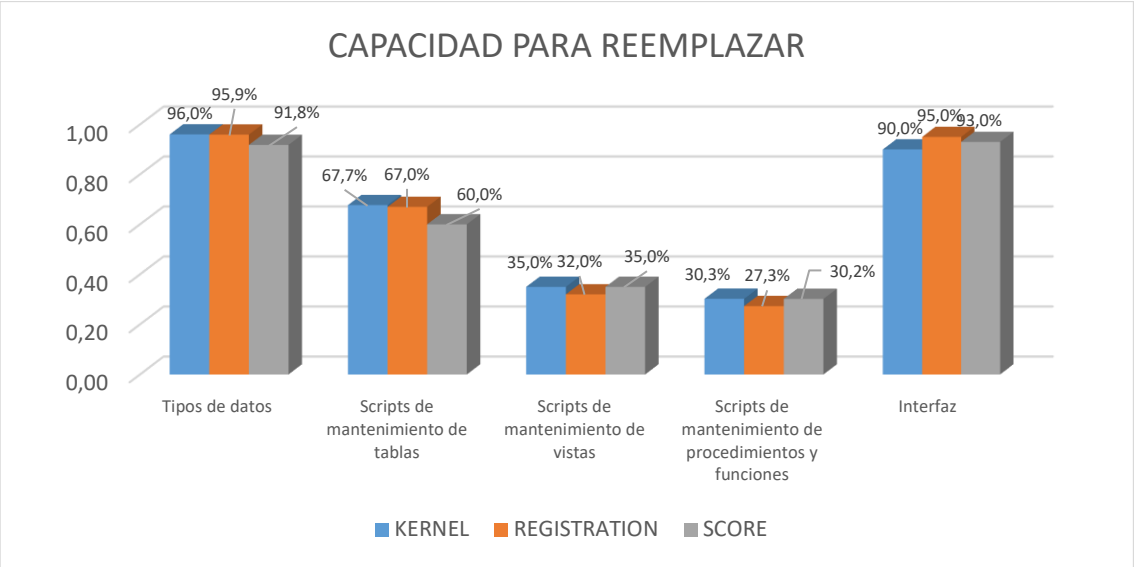


Gráfico 4-3: Indicador capacidad para reemplazar
Realizado por: Caluquí Jorge, 2018

Interpretación

Para el indicador de la capacidad para reemplazar, los scripts de mantenimiento de procedimientos y funciones han alcanzado el promedio más bajo, con un 29,26% debido a que no es necesario el reemplazar cada uno de estos scripts, mientras que en los tipos de datos y la interfaz la capacidad de reemplazar es mayor, esto se debe a que los tipos de datos no son iguales, pero si equivalentes, y pueden ser reemplazados con facilidad, obteniendo así un promedio de 94,69% y 92,67% respectivamente; mientras que para los scripts de mantenimiento de tablas se ha alcanzado un 64,9%.

La capacidad para reemplazar promedio es de 63.08%, ya que sus componentes Oracle fueron reemplazados por sus correspondientes en PostgreSQL, para cumplir con el mismo propósito y en un mismo entorno

3.6.5. Cumplimiento de la portabilidad

Tabla 29-3: Indicador cumplimiento de la portabilidad

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Datos migrados	1,00	1,00	1,00	**100,00**
TOTAL(%)	**100,00**	**100,00**	**100,00**	**100,00**

Realizado por: Caluquí Jorge, 2018

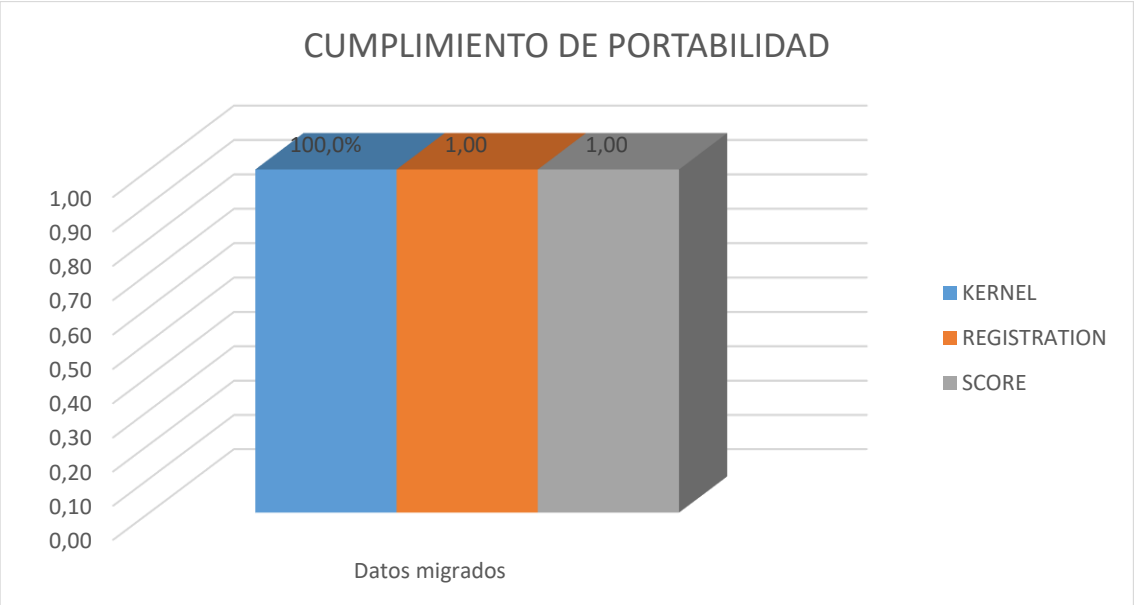


Gráfico 5-3: Indicador cumplimiento de la portabilidad
Realizado por: Caluquí Jorge, 2018

Interpretación

Para este indicador fueron considerados todos los datos de los módulos Kernel, Registration y Score, los cuales fueron migrados sin ninguna afectación, es por ello que el promedio de este indicador es del 100%.

3.7. Resultados Generales

Tabla 30-3: Resumen parámetro compatibilidad

	KERNEL	REGISTRATION	SCORE	TOTAL(%)
Adaptabilidad	0,38	0,49	0,42	**42,95**
Instalabilidad	0,90	0,90	0,90	**90,00**
Coexistencia	1,00	1,00	1,00	**100,00**
Capacidad para reemplazar	0,64	0,63	0,62	**63,08**
Cumplimiento de la portabilidad	1,00	1,00	1,00	**100,00**
TOTAL(%)	**78,36**	**80,56**	**78,71**	**79,21**

Realizado por: Caluquí Jorge, 2018

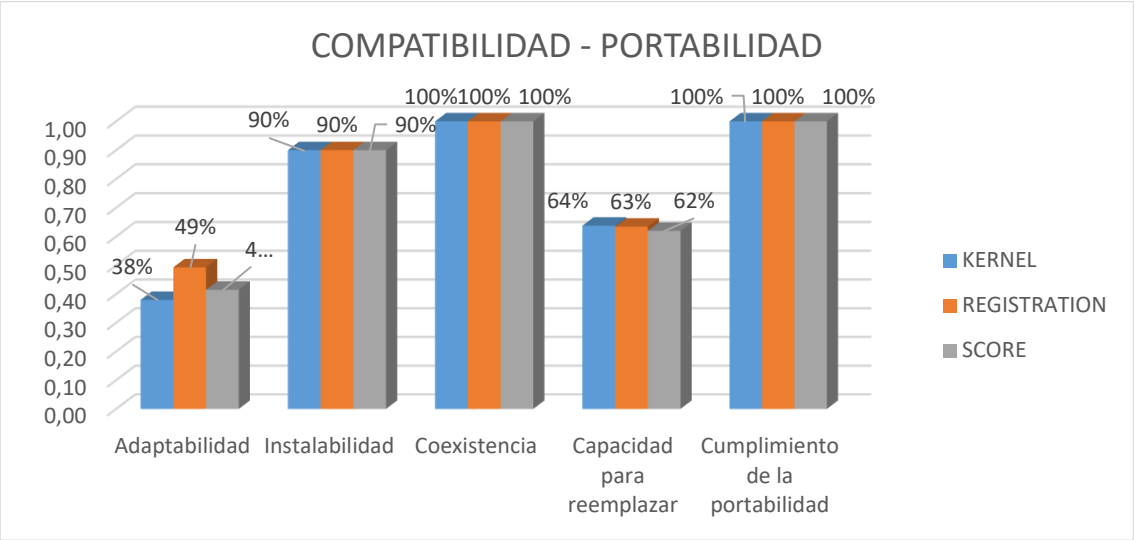


Gráfico 6-3: Resumen parámetro compatibilidad
Realizado por: Caluquí Jorge, 2018

Interpretación

La adaptabilidad del sistema, es una de los iniciadores con menor porcentaje, esto debido a que se necesitan acciones y mecanismos que permitan a la estructura adaptarse a la base de datos PostgreSQL y a su lenguaje propio; el módulo Kernel y Score, presentan la mayor parte de modificaciones debido a la diversidad de tipos de datos y funciones que emplea dentro de su estructura.

La portabilidad de los datos ha demostrado una migración satisfactoria, los datos trasladados no necesitaron de ningún artífice para su traslado a la estructura de PostgreSQL, demostrando de esta manera la equivalencia en los tipos de datos, estableciendo de esta manera un promedio del 100%.

La coexistencia logra valores altos, con un promedio del 100%, obtiene este porcentaje alto debido a que los reportes se conectaron en su totalidad, en carga de data no tuvo ningún tipo de complicación y los reportes mantienen su formato, lo que comprueba que PostgreSQL puede interactuar con un tercer software sin complicación alguna.

La capacidad para reemplazar también obtiene el promedio de 63,08%, indica que los scripts de mantenimiento de tablas, procedimientos, funciones y vistas, tienen la capacidad para reemplazar ya sea con funciones propias de cada lenguaje, no implica mayores complicaciones.

La instalabilidad de las estructuras posee un promedio del 90%, esto debido al traslado de los procedimientos (estructuras en Oracle), a funciones en PostgreSQL, lo que representa un incremento en el tiempo de migración.

Estos indicadores demuestran que la migración del Sistema VenusPro10 ha alcanzado un promedio de 79,21% de compatibilidad, dentro de este porcentaje encontramos que la dificultad en esta migración se presenta tanto en los tipos de datos, como en las estructuras que conforman la base de datos del sistema, presentándose la mayor complejidad en los módulos Kernel y Score.

CONCLUSIONES

- Los SGDB OracleDataBase XE y PostgreSQL son muy similares entre sí; las semejanzas entre ellos se establecen debido a la visión de una base de datos robusta, ya que ambas pretenden un manejo de grandes volúmenes de datos y transacciones; generando así una alta competencia entre ellas; por otro lado las limitaciones de OracleDataBase con respecto a su capacidad de almacenamiento en la versión exprés edición, lo cual genera la necesidad de adquirir las licencias de este SGDB, lo cual que conlleva costos de soporte, instalación, capacitación, etc.; siendo esto desfavorable para la Unidad Educativa Riobamba.
- Para verificar la compatibilidad de las estructuras para la migración se establecieron grupos de estructuras: tablas y tipos de datos; scripts de mantenimiento de tablas, scripts de mantenimiento de procedimientos y funciones; scripts de mantenimiento de vistas; y las llamadas que se realizan desde la aplicación; esta compatibilidad es la capacidad que tiene o posees un producto software con la finalidad de ser transportado desde un entorno asía otro, para la medición de este parámetro es necesario emplear sus indicadores: adaptabilidad, instalabilidad, coexistencia, capacidad para reemplazar, cumplimiento de la portabilidad.
- La compatibilidad alcanzada para el Sistema VenusPro10 según sus indicadores demuestran que la migración del Sistema VenusPro10 ha alcanzado un promedio del 79.21%, presentándose la mayor complejidad en los módulos Kernel y Score por su compleja estructura
- La evaluación de los datos migrados a la base de datos PostgreSQL ha alcanzado niveles satisfactorios, los datos trasladados no necesitaron de ningún artífice para su traslado, demostrando de esta manera la equivalencia en los tipos de datos.

RECOMENDACIONES

- Antes de proceder a realizar una migración de estructura y datos de una base a otra se debe tener un conocimiento de la lógica de negocio de la base a migrar, así como de sus scripts de cada estructura en forma ordenada y jerárquica, para de esta manera minimizar lo más posible el impacto de la migración.

- Se recomienda el uso de la guía metodológica implementada para la migración de tablas procedimientos, vistas, funciones, con la finalidad de garantizar una correcta migración de estructuras SQL, así como de los datos que posee la base de Oracle a Postgres.

- Se recomienda el uso de la librería Npgsql en la parte de interfaz, pues se ha establecido un impacto bajo en el aplicativo, siendo necesario cambios mínimos en las clases del aplicativo e imperceptibles dichos cambios para el usuario final.

- Este trabajo de titulación puede servir como base de futuras migraciones de bases de datos con similares características a las expuestas, así como poder ampliar el nivel de compatibilidad de los tipos de datos en su totalidad entre los SGBD expuestos.

GLOSARIO DE TÉRMINOS

SGBD o DBMS (Sistema Gestor de Bases de Datos): colección de datos estructurados, organizados y relacionados entre sí, y el conjunto de programas que acceden y gestionan esos datos

ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee): comité que propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal es el de separar los programas de aplicación de la BD física.

Java: Lenguaje de programación de propósito general, concurrente, orientado específicamente para tener pocas dependencias.

Open Source: Código abierto.

Data File: Almacenan objetos que conforman un tablespace. Un datafile pertenece solamente a un tablespace y a una instancia de base de datos.

Os Block: Disk Block, estos mapean a la data blocks. o

BIBLIOGRAFÍA

Alarcón Medina, J. M. *Sistemas de Gestión de Bases de Datos y Lenguajes de Bases de Datos* [En línea] *Universidad de Valencia.* 2017.
[Consulta: 20 12 2017]. Disponible en: https://www.uv.es/alarmedi/iti2007/curso-iti-2007.pdf

Alarcón P.P., J. G. *Particionamiento en Oracle* [En línea] *http://www.oei.eui.upm.es.* 2013.
[Consulta: 28 12 2016]. Disponible en:
http://www.oei.eui.upm.es/Asignaturas/BD/ABD/doc/temas/Tema3_ampliacionOracle.pdf

Barreto Contreras , J. F. *Oracle la Primera Base de Datos Diseñada para Grid Computing* [En línea] *Prezi.* 2013.
[Consulta: 15 11 2016]. Disponible en: https://prezi.com/r5kjengxl7y2/oracle-la-primera-base-de-datos-disenada-para-grid-computing/

Chambi, R. *7 sistemas gestores base de datos populares* [En línea] *Gitmedio.* 2016.
[Consulta: 24 03 2017]. Disponible en: http://www.gitmedio.com/gitmedio/7-sistemas-gestores-base-de-datos/

Czech and Slovak PostgreSQL Users Group, *PostgreSQL for Oracle DBA* [En línea] PostgreSQL. 2016.
[Consulta: 26 02 2017]. Disponible en: https://p2d2.cz/files/postgres-for-oracle-dbas.pdf

Grant , G. *PostgreSQL Porting Guide* [En línea] GitHub. 2016.
[Consulta: 26 04 2017]. Disponible en:
https://github.com/spacewalkproject/spacewalk/wiki/PostgreSQLPortingGuide

Gutiérrez Díaz, A. *Bases de Datos Distribuidas*
[En línea] Open Courses @Atlantic International University. 2016.
[Consulta: 15 04 2017]. Disponible en:
http://cursos.aiu.edu/base%20de%20datos%20distribuidas/pdf/tema%201.pdf

ISO/IEC FDIS 9126-1. *Information technology-Software product evaluation-Quality characteristics and guidelines for their use*

Merino Rivera, J. *Arquitectura Centralizada* [En línea] Blogspot. 2012.
[Consulta: 05 10 2017]. Disponible en: http://upcbase.blogspot.com/2012/11/5.html

Merson, P. Porting from Oracle to PostgreSQL [En línea] Carnegie Mellon Univerity School of Computer Science. 2002.
[Consulta: 20 01 2018]. Disponible en:
https://www.cs.cmu.edu/~pmerson/docs/OracleToPostgres.pdf

OGh Tech Experience. *Comparing PostgreSQL to Oracle* [En línea] *NL Oracle User Group.* 2016.
[Consulta: 03 02 2017]. Disponible en:
https://www.nloug.nl/downloads/ogh20170616_jan_karremans.pdf

Oracle, *SQL and PL/SQL* [En línea] *Database Documentation.* 2015.
[Consulta: 15 09 2016]. Disponible en: https://docs.oracle.com/en/database

Ortiz Torres, G. *Atributos de Calidad Portabilidad*
[En línea] *Pontificia Universidad Javeriana.* 2017.
[Consulta: 06 01 2018]. Disponible en:
https://sophia.javeriana.edu.co/~cbustaca/docencia/DEAS-2017-01/exposiciones/portabilidad.pdf

PowerData. *¿Qué es un gestor de datos y para qué sirve?* [En línea] *Power Data. Especialistas en gestión de datos.* 2016.
[Consulta: 23 12 2016]. Disponible en: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve

Ramez , E. & Shamkant, N. *Fundamentos de Sistemas de Bases de Datos.* Quinta ed. Madrid: Pearson Educación S.A. 2007.

Ramos , J., Ramos , A. & Montero, F., *Sistemas gestores de bases de datos* [En línea] *McGraw-Hill Interamericana de España, SL*. 2015. [Consulta: 29 12 2016] Disponible en: http://assets.mheducation.es/bcv/guide/capitulo/8448148797.pdf

San Juan , P. & Mezentsev, D. *Reglas de código PL-SQL para escribir código Oracle y Postgresql* [En línea] Openbravo wiki. 2011.
[Consulta: 05 12 2016]. Disponible en: http://wiki.openbravo.com/wiki/PL-SQL_code_rules_to_write_Oracle_and_Postgresql_code

Sánchez-Carrasco García, M. *Auditoría compatibilidad producto software* [En línea] Auditoría compatibilidad producto software. 2015.
[Consulta: 06 05 2017]. Disponible en:
http://eprints.ucm.es/33446/1/TFG%20Matias%20Sanchez-Carrasco%20Garcia.pdf

Sanchez, J. *Bases de Datos* [En línea] *JorgeSanchez.net.* 2012.
[Consulta: 15 02 2015]. Disponible en: http://www.jorgesanchez.net/index.html

SQLines. *PostgreSQL to Oracle Migration Tools and Services* [En línea] *Data analytics Platform Migration.* 2017.
[Consulta: 09 12 2017]. Disponible en: http://www.sqlines.com/postgresql-to-oracle

Telles Rodríguez, F. *Oracle X PostgreSQL* [En línea] *Savepoint Ideas not Commited yet.* 2015.
[Consulta: 25 11 2017]. Disponible en: http://www.savepoint.blog.br/2015/01/08/oracle-x-postgresql-parte-i-semelhancas/

The PostgreSQL Global Development Group *Data Definition* [En línea] *www.postgresql.org.* 2017.
[Consulta: 25 03 2017]. Disponible en: https://www.postgresql.org/docs/9.6/static/ddl-constraints.html

The PostgrSQL Global Development Group. *PostgreSQL Reference Manual*: Network Thery Ltd. 2010

Zevallos Palacios, V. *Comparacion de Gestores de Base de Datos* [En línea] *Slideshare.* 2012.
[Consulta: 24 11 2016]. Disponible en: https://es.slideshare.net/VictorZevallos/comparacion-de-gestores-de-base-de-datos

ANEXOS

Anexo A: Análisis de Tablas

TABLAS ORACLE	TABLAS POSTGRESQL	ADAPTABILIDAD				CAPACIDAD PARA SER REEMLAZADO			ADAPTABILIDAD	CAPACIDAD PARA SER REEMLAZADO
----------	----------	COMPARACION	IGUALES	CAMBIADOS	TOTAL	SI	NO	TOTAL		
--DDL for Table T_SISTEMA_K	--DDL for Table T_SISTEMA_K		7	3	10	2	7	9	0,70	0,22
----------	----------									
CREATE TABLE "T_SISTEMA_K" (	CREATE TABLE "T_SISTEMA_K" (	VERDADERO								
"SISCODIGO" CHAR(1 BYTE) NOT NULL ENABLE,	"SISCODIGO" CHAR NOT NULL,	FALSO				X				
"SISNOMBRE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"SISNOMBRE" VARCHAR(50) NOT NULL,	FALSO				X				
PRIMARY KEY ("SISCODIGO"));	PRIMARY KEY ("SISCODIGO"));	VERDADERO					X			
COMMENT ON COLUMN "T_SISTEMA_K"."SISCODIGO"	COMMENT ON COLUMN public."T_SISTEMA_K"."SISCODIGO"	FALSO					X			
IS 'K=KERNEL	IS 'K=KERNEL	VERDADERO					X			
R=REGISTRATION	R=REGISTRATION	VERDADERO					X			
S=SCORE	S=SCORE	VERDADERO					X			
D=DEGREE	D=DEGREE	VERDADERO					X			
I=INCOME';	I=INCOME';	VERDADERO					X			
----------	----------									
--DDL for Table T_PROCESO_K	--DDL for Table T_PROCESO_K		4	7	11	7	3	10	0,36	0,70
----------	----------									
CREATE TABLE "T_PROCESO_K"(	CREATE TABLE "T_PROCESO_K"(	VERDADERO								
"SISCODIGO" CHAR(1 BYTE) NOT NULL ENABLE,	"SISCODIGO" CHAR NOT NULL,	FALSO				X				
"PROCODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_PROCCODIGO" CHECK (PROCODIGO>=0) ENABLE,	"PROCODIGO" SMALLINT NOT NULL CHECK ("PROCODIGO" >= 0),	FALSO				X				
"PRODETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"PRODETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"PROESTADO" NUMBER(1,0) NOT NULL ENABLE,	"PROESTADO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("SISCODIGO", "PROCODIGO"));	PRIMARY KEY ("SISCODIGO", "PROCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("SISCODIGO") REFERENCES "T_SISTEMA_K"("SISCODIGO"));	FALSO					X			

COMMENT ON COLUMN "T_PROCESO_K"."PROESTADO"	COMMENT ON COLUMN public."T_PROCESO_K"."PROESTADO"	FALSO				X				
IS '1=ACTIVO	IS '1=ACTIVO	VERDADERO				X				
2=INACTIVO';	2=INACTIVO';	VERDADERO				X				
ALTER TABLE "T_PROCESO_K" ADD FOREIGN KEY ("SISCODIGO") REFERENCES "T_SISTEMA_K" ("SISCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_USUARIO_K	--DDL for Table T_USUARIO_K		2	5	7	5	1	6	0,29	0,83
----------	----------									
CREATE TABLE "T_PERMISO_K" (	CREATE TABLE "T_PERMISO_K" (	VERDADERO								
"USUCODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_USUCODIGO" CHECK ("USUCODIGO">0) ENABLE,	"USUCODIGO" SMALLINT NOT NULL CHECK ("USUCODIGO" > 0),	FALSO				X				
"USUNOMBRE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"USUNOMBRE" VARCHAR(80) NOT NULL,	FALSO				X				
"USUALIAS" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"USUALIAS" VARCHAR(20) NOT NULL,	FALSO				X				
"USUCLAVE" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"USUCLAVE" VARCHAR(20) NOT NULL,	FALSO				X				
"USUDEPARTAMENTO" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"USUDEPARTAMENTO" VARCHAR(80) NOT NULL,	FALSO				X				
PRIMARY KEY ("USUCODIGO"));	PRIMARY KEY ("USUCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_PERMISO_K	--DDL for Table T_PERMISO_K		2	9	11	6	5	11	0,18	0,55
----------	----------									
CREATE TABLE "T_PERMISO_K" (	CREATE TABLE "T_PERMISO_K" (	VERDADERO								
"SISCODIGO" CHAR(1 BYTE) NOT NULL ENABLE,	"SISCODIGO" CHAR NOT NULL,	FALSO				X				
"PROCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"PROCODIGO" SMALLINT NOT NULL,	FALSO				X				
"USUCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"USUCODIGO" SMALLINT NOT NULL,	FALSO				X				
"PERTIPO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_PERTIPO" CHECK (PERTIPO>=1 AND PERTIPO<=2) ENABLE,	"PERTIPO" SMALLINT NOT NULL CONSTRAINT "CHK_PERTIPO" CHECK ("PERTIPO">=1 AND "PERTIPO"<=2),	FALSO				X				
PRIMARY KEY ("SISCODIGO", "PROCODIGO", "USUCODIGO"));	PRIMARY KEY ("SISCODIGO", "PROCODIGO", "USUCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO"),	FALSO					X			
	FOREIGN KEY ("SISCODIGO", "PROCODIGO") REFERENCES "T_PROCESO_K" ("SISCODIGO", "PROCODIGO"));	FALSO					X			

COMMENT ON COLUMN "T_PERMISO_K"."PERTIPO"	COMMENT ON COLUMN public."T_PERMISO_K"."PERTIPO"	FALSO				X				
IS '1=ESCRITURA 2=LECTURA';	IS '1=ESCRITURA 2=LECTURA';					X				
ALTER TABLE "T_PERMISO_K" ADD FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_PERMISO_K" ADD FOREIGN KEY ("SISCODIGO", "PROCODIGO") REFERENCES "T_PROCESO_K" ("SISCODIGO", "PROCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_AUDITORIA_K	--DDL for Table T_AUDITORIA_K		2	10	12	4	5	9	0,17	0,44
----------	----------									
CREATE TABLE "T_PERMISO_K" (	CREATE TABLE "T_PERMISO_K" (	VERDADERO								
"AUDCODIGO" NUMBER(10,0) NOT NULL CONSTRAINT "CHK_AUDCODIGO" CHECK (AUDCODIGO>0) ENABLE,	"AUDCODIGO" INTEGER NOT NULL CONSTRAINT "CHK_AUDCODIGO" CHECK ("AUDCODIGO">0),	FALSO								
"AUDFECHAHORA" DATE NOT NULL ENABLE,	"AUDFECHAHORA" DATE NOT NULL,	FALSO								
"AUDEQUIPO" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"AUDEQUIPO" VARCHAR(20) NOT NULL,	FALSO				X				
"SISCODIGO" CHAR(1 BYTE) NOT NULL ENABLE,	"SISCODIGO" CHAR NOT NULL,	FALSO				X				
"PROCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"PROCODIGO" SMALLINT NOT NULL,	FALSO				X				
"USUCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"USUCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("AUDCODIGO"));	PRIMARY KEY ("AUDCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO"),	FALSO					X			
	FOREIGN KEY ("SISCODIGO", "PROCODIGO") REFERENCES "T_PROCESO_K" ("SISCODIGO", "PROCODIGO"));	FALSO					X			
ALTER TABLE "T_AUDITORIA_K" ADD FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_AUDITORIA_K" ADD FOREIGN KEY ("SISCODIGO", "PROCODIGO") REFERENCES "T_PROCESO_K" ("SISCODIGO", "PROCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_INSTITUCION_K	--DDL for Table T_INSTITUCION_K		2	16	18	16	1	17	0,11	0,94
----------	----------									
CREATE TABLE "T_INSTITUCION_K" (	CREATE TABLE "T_INSTITUCION_K" (	VERDADERO								

"INSCODIGO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_INSCODIGO" CHECK (INSCODIGO>0) ENABLE,	"INSCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_INSCODIGO" CHECK ("INSCODIGO">0),	FALSO				X				
"INSNOMBRE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSNOMBRE" VARCHAR(80) NOT NULL,	FALSO				X				
"INSDIRECCION" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"INSDIRECCION" VARCHAR(100) NOT NULL,	FALSO				X				
"INSTELEFONO" VARCHAR2(30 BYTE) NOT NULL ENABLE,	"INSTELEFONO" VARCHAR(30) NOT NULL,	FALSO				X				
"INSWEB" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"INSWEB" VARCHAR(50) NOT NULL,	FALSO				X				
"INSEMAIL" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"INSEMAIL" VARCHAR(50) NOT NULL,	FALSO				X				
"INSSLOGAN" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSSLOGAN" VARCHAR(80) NOT NULL,	FALSO				X				
"INSRUTALOGO" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"INSRUTALOGO" VARCHAR(200) NOT NULL,	FALSO				X				
"INSCODIGOSECUNDARIO" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"INSCODIGOSECUNDARIO" VARCHAR(10) NOT NULL,	FALSO				X				
"INSCOORDINACION" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSCOORDINACION" VARCHAR(80) NOT NULL,	FALSO				X				
"INSDISTRITO" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSDISTRITO" VARCHAR(80) NOT NULL,	FALSO				X				
"INSCIUDAD" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSCIUDAD" VARCHAR(80) NOT NULL,	FALSO				X				
"INSNOMBRESEC" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INSNOMBRESEC" VARCHAR(80) NOT NULL,	FALSO				X				
"INSASUNTO" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"INSASUNTO" VARCHAR(100) NOT NULL,	FALSO				X				
"INSEMAILENVIOS" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"INSEMAILENVIOS" VARCHAR(100) NOT NULL,	FALSO				X				
"INSCLAVE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"INSCLAVE" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("INSCODIGO"));	PRIMARY KEY ("INSCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_AUXILIATURA_K	--DDL for Table T_AUXILIATURA_K		2	3	5	3	1	4	0,40	0,75
----------	----------									
CREATE TABLE "T_AUXILIATURA_K" (	CREATE TABLE "T_AUXILIATURA_K" (	VERDADERO								
"AUXCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_AUXCODIGO" CHECK (AUXCODIGO>0) ENABLE,	"AUXCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_AUXCODIGO" CHECK ("AUXCODIGO">0),	FALSO				X				
"AUXDETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"AUXDETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"AUXTITULO" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"AUXTITULO" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("AUXCODIGO"));	PRIMARY KEY ("AUXCODIGO"));	VERDADERO					X			
----------	----------									

--DDL for Table T_ESPECIALIDAD_K	--DDL for Table T_ESPECIALIDAD_K		2	3	5	3	1	4	0,40	0,75
----------	----------									
CREATE TABLE "T_ESPECIALIDAD_K" (	CREATE TABLE "T_ESPECIALIDAD_K" (	VERDADERO								
"ESPCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_ESPCODIGO" CHECK (ESPCODIGO>0) ENABLE,	"ESPCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_ESPCODIGO" CHECK ("ESPCODIGO">0),	FALSO				X				
"ESPDETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"ESPDETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"ESPTITULO" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"ESPTITULO" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("ESPCODIGO"));	PRIMARY KEY ("ESPCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_CAJA_K	--DDL for Table T_CAJA_K		2	4	6	2	1	3	0,33	0,67
----------	----------									
CREATE TABLE "T_CAJA_K" (	CREATE TABLE "T_CAJA_K" (	VERDADERO								
"CAJCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_CAJCODIGO" CHECK (CAJCODIGO>0) ENABLE,	"CAJCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_CAJCODIGO" CHECK ("CAJCODIGO">0),	FALSO				X				
"CAJDETALLE" VARCHAR2(40 BYTE) NOT NULL ENABLE,	"CAJDETALLE" VARCHAR(40) NOT NULL,	FALSO				X				
PRIMARY KEY ("CAJCODIGO"));	PRIMARY KEY ("CAJCODIGO"),	VERDADERO					X			
	CONSTRAINT "UNI_CAJCODIGO" UNIQUE ("CAJCODIGO"));	FALSO								
CREATE UNIQUE INDEX "T_CAJA_K_PK" ON "VENUSPRO10"."T_CAJA_K" ("CAJCODIGO");		FALSO								
----------	----------									
--DDL for Table T_FORMAPAGO_K	--DDL for Table T_FORMAPAGO_K		14	14	28	26	1	27	0,50	0,96
----------	----------									
CREATE TABLE "T_FORMAPAGO_K" (	CREATE TABLE "T_FORMAPAGO_K" (	VERDADERO								
"FORCODIGO" NUMBER(5,0) NOT NULL CONSTRAINT "CHK_FORCODIGO" CHECK (FORCODIGO>0) ENABLE,	"FORCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_FORCODIGO" CHECK ("FORCODIGO">0),	FALSO				X				
"FORDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"FORDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"FORTIPO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_FORTIPO" CHECK (FORTIPO>=1 AND FORTIPO<=10) ENABLE,	"FORTIPO" SMALLINT NOT NULL CONSTRAINT "CHK_FORTIPO" CHECK ("FORTIPO">=1 AND "FORTIPO"<=10),	FALSO				X				
"FORCUOTA" NUMBER(5,0) NOT NULL CONSTRAINT "CHK_FORCUOTA" CHECK (FORCUOTA>=0) ENABLE,	"FORCUOTA" SMALLINT NOT NULL CONSTRAINT "CHK_FORCUOTA" CHECK ("FORCUOTA">=0),	FALSO				X				

"FORENTRADA" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_FORENTRADA" CHECK (FORENTRADA>=1 AND FORENTRADA<=2) ENABLE,	"FORENTRADA" SMALLINT NOT NULL CONSTRAINT "CHK_FORENTRADA" CHECK ("FORENTRADA">=1 AND "FORENTRADA"<=2),	FALSO				X				
"FORTIPOCUOTA" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_FORTIPOCUOTA" CHECK (FORTIPOCUOTA>=1 AND FORTIPOCUOTA<=4) ENABLE,	"FORTIPOCUOTA" SMALLINT NOT NULL CONSTRAINT "CHK_FORTIPOCUOTA" CHECK ("FORTIPOCUOTA">=1 AND "FORTIPOCUOTA"<=4),	FALSO				X				
"FORINTERES" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_FORINTERES" CHECK (FORINTERES>=0) ENABLE,	"FORINTERES" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_FORINTERES" CHECK ("FORINTERES">=0),	FALSO				X				
"FORMORA" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_FORMORA" CHECK (FORMORA>=0) ENABLE,	"FORMORA" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_FORMORA" CHECK ("FORMORA">=0),	FALSO				X				
"FORAPLICACION" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_FORAPLICACION" CHECK (FORAPLICACION>=1 AND FORAPLICACION<=2) ENABLE,	"FORAPLICACION" SMALLINT NOT NULL CONSTRAINT "CHK_FORAPLICACION" CHECK ("FORAPLICACION">=1 AND "FORAPLICACION"<=2),	FALSO				X				
"FORDIAS" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_BONO" CHECK ("FORDIAS">=1 AND "FORDIAS"<=2) DISABLE,	"FORDIAS" SMALLINT NOT NULL CONSTRAINT "CHK_BONO" CHECK ("FORDIAS">=1 AND "FORDIAS"<=2),	FALSO				X				
PRIMARY KEY ("FORCODIGO"));	PRIMARY KEY ("FORCODIGO"));	VERDADERO					X			
COMMENT ON COLUMN "T_FORMAPAGO_K"."FORTIPO" IS '1=CONTADO	COMMENT ON COLUMN public."T_FORMAPAGO_K"."FORTIPO" IS '1=CONTADO	FALSO				X				
2=CREDITO DIRECTO	2=CREDITO DIRECTO	VERDADERO				X				
3=CREDITO CUOTAS	3=CREDITO CUOTAS	VERDADERO				X				
4=CHEQUES	4=CHEQUES	VERDADERO				X				
5=TARJETAS	5=TARJETAS	VERDADERO				X				
6=DEPOSITOS Y TRANSFERENCIAS	6=DEPOSITOS Y TRANSFERENCIAS	VERDADERO				X				
7=TARJETA PROMOCIONAL	7=TARJETA PROMOCIONAL	VERDADERO				X				
8=NOTA DE CREDITO	8=NOTA DE CREDITO	VERDADERO				X				
9=RETENCION	9=RETENCION	VERDADERO				X				
10=OTROS DESCUENTO';	10=OTROS DESCUENTO';	VERDADERO				X				
COMMENT ON COLUMN "T_FORMAPAGO_K"."FORENTRADA" IS '1= ACEPTA 2= NO ACEPTA';	COMMENT ON COLUMN public."T_FORMAPAGO_K"."FORENTRADA" IS '1= ACEPTA 2= NO ACEPTA';	FALSO				X				
COMMENT ON COLUMN "T_FORMAPAGO_K"."FORTIPOCUOTA" IS '1=DIARIA	COMMENT ON COLUMN public."T_FORMAPAGO_K"."FORTIPOCUOTA" IS '1=DIARIA	FALSO				X				
2=SEMANAL	2=SEMANAL	VERDADERO				X				
3=MENSUAL	3=MENSUAL	VERDADERO				X				

4=QUINCENAL';	4=QUINCENAL';	VERDADERO				X				
COMMENT ON COLUMN "T_FORMAPAGO_K"."FORAPLICACION" IS '1= FIJO 2= VARIABLE';	COMMENT ON COLUMN public."T_FORMAPAGO_K"."FORAPLICACION" IS '1= FIJO 2= VARIABLE';	FALSO				X				
----------	----------									
--DDL for Table T_INSFINANCIERAS_K	--DDL for Table T_INSFINANCIERAS_K		2	2	4	2	1	3	0,50	0,67
----------	----------									
CREATE TABLE "T_INSFINANCIERAS_K" (	CREATE TABLE "T_INSFINANCIERAS_K" (	VERDADERO								
"INFCODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_INFCODIGO" CHECK (INFCODIGO>0) ENABLE,	"INFCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_INFCODIGO" CHECK ("INFCODIGO">0),	FALSO				X				
"INFNOMBRE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"INFNOMBRE" VARCHAR(80) NOT NULL,	FALSO				X				
PRIMARY KEY ("INFCODIGO"));	PRIMARY KEY ("INFCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_NACIONALIDAD_K	--DDL for Table T_NACIONALIDAD_K		2	2	4	2	1	3	0,50	0,67
----------	----------									
CREATE TABLE "T_NACIONALIDAD_K" (	CREATE TABLE "T_NACIONALIDAD_K" (	VERDADERO								
"NACCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_NACCODIGO" CHECK (NACCODIGO>0) ENABLE,	"NACCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_NACCODIGO" CHECK ("NACCODIGO">0),	FALSO				X				
"NACDETALLE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"NACDETALLE" VARCHAR(50) NOT NULL,	FALSO				X				
PRIMARY KEY ("NACCODIGO"));	PRIMARY KEY ("NACCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_PROVINCIA_K	--DDL for Table T_PROVINCIA_K		2	2	4	2	1	3	0,50	0,67
----------	----------									
CREATE TABLE "T_PROVINCIA_K" (	CREATE TABLE "T_PROVINCIA_K" (	VERDADERO								
"PROCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_PROCODIGO" CHECK (PROCODIGO>0) ENABLE,	"PROCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_PROCODIGO" CHECK ("PROCODIGO">0),	FALSO				X				
"PRODETALLE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"PRODETALLE" VARCHAR(50) NOT NULL,	FALSO				X				
PRIMARY KEY ("PROCODIGO"));	PRIMARY KEY ("PROCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_CANTON_K	--DDL for Table T_CANTON_K		2	5	7	3	3	6	0,29	0,50
----------	----------									

CREATE TABLE "T_CANTON_K" (	CREATE TABLE "T_CANTON_K" (	VERDADERO								
"CANCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_CANCODIGO" CHECK (CANCODIGO>0) ENABLE,	"CANCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_CANCODIGO" CHECK ("CANCODIGO">0),	FALSO				X				
"CANDETALLE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"CANDETALLE" VARCHAR(50) NOT NULL,	FALSO				X				
"PROCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"PROCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("CANCODIGO", "PROCODIGO"));	PRIMARY KEY ("CANCODIGO", "PROCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("PROCODIGO") REFERENCES "T_PROVINCIA_K" ("PROCODIGO"));	FALSO					X			
ALTER TABLE "T_CANTON_K" ADD FOREIGN KEY ("PROCODIGO") REFERENCES "T_PROVINCIA_K" ("PROCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_PARROQUIA_K	--DDL for Table T_PARROQUIA_K		2	6	8	4	3	7	0,25	0,57
----------	----------									
CREATE TABLE "T_PARROQUIA_K" (	CREATE TABLE "T_PARROQUIA_K" (	VERDADERO								
"PARCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_PARCODIGO" CHECK (PARCODIGO>0) ENABLE,	"PARCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_PARCODIGO" CHECK ("PARCODIGO">0),	FALSO				X				
"PARDETALLE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"PARDETALLE" VARCHAR(50) NOT NULL,	FALSO				X				
"CANCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"CANCODIGO" SMALLINT NOT NULL,	FALSO				X				
"PROCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"PROCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("PARCODIGO", "CANCODIGO", "PROCODIGO"));	PRIMARY KEY ("PARCODIGO", "CANCODIGO", "PROCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("CANCODIGO", "PROCODIGO") REFERENCES "T_CANTON_K" ("CANCODIGO", "PROCODIGO"));	FALSO					X			
ALTER TABLE "T_PARROQUIA_K" ADD FOREIGN KEY ("CANCODIGO", "PROCODIGO") REFERENCES "T_CANTON_K" ("CANCODIGO", "PROCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_CONFIGURACION_K	--DDL for Table T_CONFIGURACION_K		3	15	18	13	3	16	0,17	0,81
----------	----------									
CREATE TABLE "T_CONFIGURACION_K" (	CREATE TABLE "T_CONFIGURACION_K" (	VERDADERO								
"CONCODFACTURA" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"CONCODFACTURA" VARCHAR(10) NOT NULL,	FALSO				X				
"INSCODIGO" NUMBER(1,0) NOT NULL ENABLE,	"INSCODIGO" SMALLINT NOT NULL,	FALSO				X				

"CONIVA" VARCHAR2(5 BYTE) NOT NULL ENABLE,	"CONIVA" VARCHAR(5) NOT NULL,	FALSO				X				
"CONCODRETENCION" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"CONCODRETENCION" VARCHAR(10) NOT NULL,	FALSO				X				
"CONNUMFACTURA" NUMBER(12,0) NOT NULL ENABLE,	"CONNUMFACTURA" INTEGER NOT NULL,	FALSO								
"CONOTROS" VARCHAR2(5 BYTE) NOT NULL ENABLE,	"CONOTROS" VARCHAR(5) NOT NULL,	FALSO				X				
"CONCUENTARESULTADO" NUMBER(1,0) NOT NULL ENABLE,	"CONCUENTARESULTADO" SMALLINT NOT NULL,	FALSO				X				
"CONRAZONSOCIAL" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"CONRAZONSOCIAL" VARCHAR(200) NOT NULL,	FALSO				X				
"CONRUC" VARCHAR2(13 BYTE) NOT NULL ENABLE,	"CONRUC" VARCHAR(13) NOT NULL,	FALSO				X				
"CONTIPOFACTURA" NUMBER(1,0) NOT NULL ENABLE,	"CONTIPOFACTURA" SMALLINT NOT NULL,	FALSO				X				
"CONESTABLECIMIENTO" VARCHAR2(3 BYTE) NOT NULL ENABLE,	"CONESTABLECIMIENTO" VARCHAR(3) NOT NULL,	FALSO				X				
"CONPUNTOEMISION" VARCHAR2(3 BYTE) NOT NULL ENABLE,	"CONPUNTOEMISION" VARCHAR(3) NOT NULL,	FALSO				X				
PRIMARY KEY ("INSCODIGO"));	PRIMARY KEY ("INSCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("INSCODIGO") REFERENCES "T_INSTITUCION_K" ("INSCODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_CONFIGURACION_K"."CONCUENTARESULTADO" IS '"1= UNA CUENTA (CUENTA RESULTADO)	COMMENT ON COLUMN public."T_CONFIGURACION_K"."CONCUENTA RESULTADO" IS '"1= UNA CUENTA (CUENTA RESULTADO)	FALSO				X				
2= DOS CUENTAS(CUENTA PERDIDA Y CUENTA UTILIDAD)"';	2= DOS CUENTAS(CUENTA PERDIDA Y CUENTA UTILIDAD)"';	VERDADERO				X				
ALTER TABLE "T_CONFIGURACION_K" ADD FOREIGN KEY ("INSCODIGO") REFERENCES "T_INSTITUCION_K" ("INSCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_DOCENTE_S	--DDL for Table T_DOCENTE_S		17	34	51	48	1	49	0,33	0,98
----------	----------									
CREATE TABLE "T_DOCENTE_S" (	CREATE TABLE "T_DOCENTE_S" (	VERDADERO								
"DOCCODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_DOCCODIGO" CHECK (DOCCODIGO>0) ENABLE,	"DOCCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_DOCCODIGO" CHECK ("DOCCODIGO">0),	FALSO				X				
"DOCNOMBRES" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"DOCNOMBRES" VARCHAR(80) NOT NULL,	FALSO				X				
"DOCIDENTIFICACION" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCIDENTIFICACION" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCTITULO" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"DOCTITULO" VARCHAR(50) NOT NULL,	FALSO				X				
"DOCDIRECCION" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"DOCDIRECCION" VARCHAR(100) NOT NULL,	FALSO				X				

"DOCFONOFIJO" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCFONOFIJO" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCFONOMOVIL" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCFONOMOVIL" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCEMAIL" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCEMAIL" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCALIAS" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"DOCALIAS" VARCHAR(50) NOT NULL,	FALSO				X				
"DOCFECHAINSTITUCION" DATE NOT NULL ENABLE,	"DOCFECHAINSTITUCION" DATE NOT NULL,	FALSO				X				
"DOCFECHAMAGISTERIO" DATE NOT NULL ENABLE,	"DOCFECHAMAGISTERIO" DATE NOT NULL,	FALSO				X				
"DOCESTADOCIVIL" NUMBER(1,0) NOT NULL ENABLE,	"DOCESTADOCIVIL" SMALLINT NOT NULL,	FALSO				X				
"DOCCATEGORIA" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"DOCCATEGORIA" VARCHAR(10) NOT NULL,	FALSO				X				
"DOCOBSERVACION" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"DOCOBSERVACION" VARCHAR(200) NOT NULL,	FALSO				X				
"DOCCLAVE" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCCLAVE" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCTIPO" NUMBER(1,0) NOT NULL ENABLE,	"DOCTIPO" SMALLINT NOT NULL,	FALSO				X				
"DOCTITULONUMERO" NUMBER(1,0) NOT NULL ENABLE,	"DOCTITULONUMERO" SMALLINT NOT NULL,	FALSO				X				
"DOCCARGO" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"DOCCARGO" VARCHAR(200) NOT NULL,	FALSO				X				
"DOCNACIMIENTO" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"DOCNACIMIENTO" VARCHAR(200) NOT NULL,	FALSO				X				
"DOCFECHANACIMIENTO" DATE NOT NULL ENABLE,	"DOCFECHANACIMIENTO" DATE NOT NULL,	FALSO				X				
"DOCCONYUGE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"DOCCONYUGE" VARCHAR(100) NOT NULL,	FALSO				X				
"DOCAFILIACIONIESS" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCAFILIACIONIESS" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCCEDULAMILITAR" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"DOCCEDULAMILITAR" VARCHAR(20) NOT NULL,	FALSO				X				
"DOCINSTRUCCION" VARCHAR2(150 BYTE) NOT NULL ENABLE,	"DOCINSTRUCCION" VARCHAR(150) NOT NULL,	FALSO				X				
"DOCSANCIONES" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"DOCSANCIONES" VARCHAR(200) NOT NULL,	FALSO				X				
"DOCCOMIENZOSERVICIO" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"DOCCOMIENZOSERVICIO" VARCHAR(200) NOT NULL,	FALSO				X				
"DOCFECHAGRADUACION" DATE NOT NULL ENABLE,	"DOCFECHAGRADUACION" DATE NOT NULL,	FALSO								
"DOCGENERO" NUMBER(1,0) NOT NULL ENABLE,	"DOCGENERO" SMALLINT NOT NULL,	FALSO				X				
"DOCTIPOID" NUMBER(1,0) NOT NULL ENABLE,	"DOCTIPOID" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("DOCCODIGO"));	PRIMARY KEY ("DOCCODIGO"));	VERDADERO					X			
COMMENT ON COLUMN "T_DOCENTE_S"."DOCESTADOCIVIL" IS '1=SOLTERO(A)	COMMENT ON COLUMN public."T_DOCENTE_S"."DOCESTADOCIVIL" IS '1=SOLTERO(A)	FALSO				X				

2=CASADO(A)	2=CASADO(A)	VERDADERO				X				
3=VIUDO(A)	3=VIUDO(A)	VERDADERO				X				
4=DIVORCIADO(A)	4=DIVORCIADO(A)	VERDADERO				X				
5=UNION LIBRE';	5=UNION LIBRE';	VERDADERO				X				
COMMENT ON COLUMN "T_DOCENTE_S"."DOCTIPO" IS '1 = DOCENTE	COMMENT ON COLUMN public."T_DOCENTE_S"."DOCTIPO" IS '1 = DOCENTE	FALSO				X				
2 = ADMINISTRATIVO	2 = ADMINISTRATIVO	VERDADERO				X				
3 = EMPLEADO';	3 = EMPLEADO';	VERDADERO				X				
COMMENT ON COLUMN "T_DOCENTE_S"."DOCTITULONUMERO" IS '1 = LICENCIADO	COMMENT ON COLUMN public."T_DOCENTE_S"."DOCTITULONUMERO" IS '1 = LICENCIADO	FALSO				X				
2 = DOCTOR	2 = DOCTOR	VERDADERO				X				
3 = MASTER	3 = MASTER	VERDADERO				X				
4 = INGENIERO	4 = INGENIERO	VERDADERO				X				
5 = TECNOLOGO	5 = TECNOLOGO	VERDADERO				X				
6 = TECNICO	6 = TECNICO	VERDADERO				X				
7 = OTRO';	7 = OTRO';	VERDADERO				X				
COMMENT ON COLUMN "T_DOCENTE_S"."DOCGENERO" IS '1 = MASCULINO 2 = FEMENINO';	COMMENT ON COLUMN public."T_DOCENTE_S"."DOCGENERO" IS '1 = MASCULINO 2 = FEMENINO';	FALSO				X				
COMMENT ON COLUMN "T_DOCENTE_S"."DOCTIPOID" IS '1 = CEDULA	COMMENT ON COLUMN public."T_DOCENTE_S"."DOCTIPOID" IS '1 = CEDULA	FALSO				X				
2 = RUC	2 = RUC	VERDADERO				X				
3 = PASAPORTE	3 = PASAPORTE	VERDADERO				X				
4 = OTROS';	4 = OTROS';	VERDADERO				X				
----------	----------									
--DDL for Table T_CONFIGURA_DOCENTE_K	--DDL for Table T_CONFIGURA_DOCENTE_K		2	35	37	33	3	36	0,05	0,92
----------	----------									
CREATE TABLE "T_CONFIGURA_DOCENTE_K" (	CREATE TABLE "T_CONFIGURA_DOCENTE_K" (	VERDADERO								
"DOCCODIGO" NUMBER(10,0) NOT NULL ENABLE,	"DOCCODIGO" INTEGER NOT NULL,	FALSO				X				
"CNDPARCIAL11" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL11" CHECK (CNDPARCIAL11 >= 1 AND CNDPARCIAL11 <= 2) ENABLE,	"CNDPARCIAL11" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL11" CHECK ("CNDPARCIAL11" >= 1 AND "CNDPARCIAL11" <= 2),	FALSO				X				

"CNDPARCIAL12" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL12" CHECK (CNDPARCIAL12 >= 1 AND CNDPARCIAL12 <= 2) ENABLE,	"CNDPARCIAL12" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL12" CHECK ("CNDPARCIAL12" >= 1 AND "CNDPARCIAL12" <= 2),	FALSO				X				
"CNDPARCIAL13" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL13" CHECK (CNDPARCIAL13 >= 1 AND CNDPARCIAL13 <= 2) ENABLE,	"CNDPARCIAL13" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL13" CHECK ("CNDPARCIAL13" >= 1 AND "CNDPARCIAL13" <= 2),	FALSO				X				
"CNDPARCIAL14" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL14" CHECK (CNDPARCIAL14 >= 1 AND CNDPARCIAL14 <= 2) ENABLE,	"CNDPARCIAL14" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL14" CHECK ("CNDPARCIAL14" >= 1 AND "CNDPARCIAL14" <= 2),	FALSO				X				
"CNDPARCIAL15" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL15" CHECK (CNDPARCIAL15 >= 1 AND CNDPARCIAL15 <= 2) ENABLE,	"CNDPARCIAL15" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL15" CHECK ("CNDPARCIAL15" >= 1 AND "CNDPARCIAL15" <= 2),	FALSO				X				
"CNDPARCIAL21" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL21" CHECK (CNDPARCIAL21 >= 1 AND CNDPARCIAL21 <= 2) ENABLE,	"CNDPARCIAL21" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL21" CHECK ("CNDPARCIAL21" >= 1 AND "CNDPARCIAL21" <= 2),	FALSO				X				
"CNDPARCIAL22" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL22" CHECK (CNDPARCIAL22 >= 1 AND CNDPARCIAL22 <= 2) ENABLE,	"CNDPARCIAL22" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL22" CHECK ("CNDPARCIAL22" >= 1 AND "CNDPARCIAL22" <= 2),	FALSO				X				
"CNDPARCIAL23" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL23" CHECK (CNDPARCIAL23 >= 1 AND CNDPARCIAL23 <= 2) ENABLE,	"CNDPARCIAL23" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL23" CHECK ("CNDPARCIAL23" >= 1 AND "CNDPARCIAL23" <= 2),	FALSO				X				
"CNDPARCIAL24" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL24" CHECK (CNDPARCIAL24 >= 1 AND CNDPARCIAL24 <= 2) ENABLE,	"CNDPARCIAL24" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL24" CHECK ("CNDPARCIAL24" >= 1 AND "CNDPARCIAL24" <= 2),	FALSO				X				
"CNDPARCIAL25" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDPARCIAL25" CHECK (CNDPARCIAL25 >= 1 AND CNDPARCIAL25 <= 2) ENABLE,	"CNDPARCIAL25" SMALLINT NOT NULL CONSTRAINT "CHK_CNDPARCIAL25" CHECK ("CNDPARCIAL25" >= 1 AND "CNDPARCIAL25" <= 2),	FALSO				X				
"CNDRECUPERACION" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDRECUPERACION" CHECK (CNDRECUPERACION >= 1 AND CNDRECUPERACION <= 2) ENABLE,	"CNDRECUPERACION" SMALLINT NOT NULL CONSTRAINT "CHK_CNDRECUPERACION" CHECK ("CNDRECUPERACION" >= 1 AND "CNDRECUPERACION" <= 2),	FALSO				X				
"CNDSUPLETORIO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDSUPLETORIO" CHECK (CNDSUPLETORIO >= 1 AND CNDSUPLETORIO <= 2) ENABLE,	"CNDSUPLETORIO" SMALLINT NOT NULL CONSTRAINT "CHK_CNDSUPLETORIO" CHECK ("CNDSUPLETORIO" >= 1 AND "CNDSUPLETORIO" <= 2),	FALSO				X				
"CNDREMEDIAL" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDREMEDIAL" CHECK (CNDREMEDIAL >= 1 AND CNDREMEDIAL <= 2) ENABLE,	"CNDREMEDIAL" SMALLINT NOT NULL CONSTRAINT "CHK_CNDREMEDIAL" CHECK ("CNDREMEDIAL" >= 1 AND "CNDREMEDIAL" <= 2),	FALSO				X				
"CNDGRACIA" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDGRACIA" CHECK (CNDGRACIA >= 1 AND CNDGRACIA <= 2) ENABLE,	"CNDGRACIA" SMALLINT NOT NULL CONSTRAINT "CHK_CNDGRACIA" CHECK ("CNDGRACIA" >= 1 AND "CNDGRACIA" <= 2),	FALSO				X				
"CNDQUIMESTRE1" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDQUIMESTRE1" CHECK	"CNDQUIMESTRE1" SMALLINT NOT NULL CONSTRAINT "CHK_CNDQUIMESTRE1"	FALSO				X				

(CNDQUIMESTRE1 >= 1 AND CNDQUIMESTRE1 <= 2) ENABLE,	CHECK ("CNDQUIMESTRE1" >= 1 AND "CNDQUIMESTRE1" <= 2),									
"CNDQUIMESTRE2" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CNDQUIMESTRE2" CHECK (CNDQUIMESTRE2 >= 1 AND CNDQUIMESTRE2 <= 2) ENABLE,	"CNDQUIMESTRE2" SMALLINT NOT NULL CONSTRAINT "CHK_CNDQUIMESTRE2" CHECK ("CNDQUIMESTRE2" >= 1 AND "CNDQUIMESTRE2" <= 2),	FALSO				X				
PRIMARY KEY ("DOCCODIGO"));	PRIMARY KEY ("DOCCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL11" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL11" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL12" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL12" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL13" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL13" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL14" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL14" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL15" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL15" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL21" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL21" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL22" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL22" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL23" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL23" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL24" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL24" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDPARCIAL25" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDPARCIAL25" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDRECUPERACION" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDRECUPERACION" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDSUPLETORIO" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDSUPLETORIO" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDREMEDIAL" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDREMEDIAL" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				

COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDGRACIA" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDGRACIA" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDQUIMESTRE1" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDQUIMESTRE1" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
COMMENT ON COLUMN "T_CONFIGURA_DOCENTE_K"."CNDQUIMESTRE2" IS '1=ACTIVO 2=INACTIVO';	COMMENT ON COLUMN public."T_CONFIGURA_DOCENTE_K"."CNDQUIMESTRE2" IS '1=ACTIVO 2=INACTIVO';	FALSO				X				
ALTER TABLE "T_CONFIGURA_DOCENTE_K" ADD CONSTRAINT "T_CONFIGURA_DOCENTE_K_T_D_FK1" FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_ANIOLECTIVO_K	--DDL for Table T_ANIOLECTIVO_K		2	20	22	20	1	21	0,09	0,95
----------	----------									
CREATE TABLE "T_ANIOLECTIVO_K" (	CREATE TABLE "T_ANIOLECTIVO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_ANICODIGO" CHECK (ANICODIGO>0) ENABLE,	"ANICODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_ANICODIGO" CHECK ("ANICODIGO">0),	FALSO				X				
"ANIDETALLE" VARCHAR2(30 BYTE) NOT NULL ENABLE,	"ANIDETALLE" VARCHAR(30) NOT NULL,	FALSO				X				
"ANIORDINICIO" DATE NOT NULL ENABLE,	"ANIORDINICIO" DATE NOT NULL,	FALSO				X				
"ANIORDFIN" DATE NOT NULL ENABLE,	"ANIORDFIN" DATE NOT NULL,	FALSO				X				
"ANIEXTINICIO" DATE NOT NULL ENABLE,	"ANIEXTINICIO" DATE NOT NULL,	FALSO				X				
"ANIEXTFIN" DATE NOT NULL ENABLE,	"ANIEXTFIN" DATE NOT NULL,	FALSO				X				
"ANIEXCINICIO" DATE NOT NULL ENABLE,	"ANIEXCINICIO" DATE NOT NULL,	FALSO				X				
"ANIEXCFIN" DATE NOT NULL ENABLE,	"ANIEXCFIN" DATE NOT NULL,	FALSO				X				
"ANIESTADO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_ANIESTADO" CHECK (ANIESTADO>=1 AND ANIESTADO<=2) ENABLE,	"ANIESTADO" SMALLINT NOT NULL CONSTRAINT "CHK_ANIESTADO" CHECK ("ANIESTADO">=1 AND "ANIESTADO"<=2),	FALSO				X				
"ANIPARCIALES" NUMBER(1,0) NOT NULL ENABLE,	"ANIPARCIALES" SMALLINT NOT NULL,	FALSO				X				
"ANIRETIRADO" NUMBER(1,0) NOT NULL ENABLE,	"ANIRETIRADO" SMALLINT NOT NULL,	FALSO				X				
"ANIREDONDEO" NUMBER(1,0) NOT NULL ENABLE,	"ANIREDONDEO" SMALLINT NOT NULL,	FALSO				X				
"ANIARCHIVO" NUMBER(1,0) NOT NULL ENABLE,	"ANIARCHIVO" SMALLINT NOT NULL,	FALSO				X				
"ANIDESTREZAS" NUMBER(1,0) NOT NULL ENABLE,	"ANIDESTREZAS" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO"));	PRIMARY KEY ("ANICODIGO"));	VERDADERO					X			

COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIESTADO" IS '1=ABIERTO 2=CERRADO';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIESTADO" IS '1=ABIERTO 2=CERRADO';	FALSO				X				
COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIPARCIALES" IS '1=COMPONENETS 2=PARCIALES';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIPARCIALES" IS '1=COMPONENETS 2=PARCIALES';	FALSO				X				
COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIRETIRADO" IS '1 = SI 2 = NO';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIRETIRADO" IS '1 = SI 2 = NO';	FALSO				X				
COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIREDONDEO" IS '1 = SI 2 = NO';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIREDONDEO" IS '1 = SI 2 = NO';	FALSO				X				
COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIARCHIVO" IS '1 = SI CARGA ARCHIVO 2 = NO CARGA ARCHIVO';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIARCHIVO" IS '1 = SI CARGA ARCHIVO 2 = NO CARGA ARCHIVO';	FALSO				X				
COMMENT ON COLUMN "T_ANIOLECTIVO_K"."ANIDESTREZAS" IS '1 = SI 2 = NO';	COMMENT ON COLUMN public."T_ANIOLECTIVO_K"."ANIDESTREZAS" IS '1 = SI 2 = NO';	FALSO				X				
----------	----------									
--DDL for Table T_EDUCACION_MEDIA_K	--DDL for Table T_EDUCACION_MEDIA_K		2	3	5	3	1	4	0,40	0,75
----------	----------									
CREATE TABLE "T_EDUCACION_MEDIA_K" (	CREATE TABLE "T_EDUCACION_MEDIA_K" (	VERDADERO								
"EDUCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_EDUCODIGO" CHECK (EDUCODIGO>0) ENABLE,	"EDUCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_EDUCODIGO" CHECK ("EDUCODIGO">0),	FALSO				X				
"EDUNOMBRE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"EDUNOMBRE" VARCHAR(100) NOT NULL,	FALSO				X				
"EDUDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"EDUDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("EDUCODIGO"));	PRIMARY KEY ("EDUCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_DOCENTEANIO_K	--DDL for Table T_DOCENTEANIO_K		2	8	10	4	5	9	0,20	0,44
----------	----------									
CREATE TABLE "T_DOCENTEANIO_K" (	CREATE TABLE "T_DOCENTEANIO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"DOCCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"DOCCODIGO" SMALLINT NOT NULL,	FALSO				X				
"DOCACTIVO" NUMBER(1,0) NOT NULL ENABLE,	"DOCACTIVO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "DOCCODIGO"));	PRIMARY KEY ("ANICODIGO", "DOCCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"),	FALSO					X			

	FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_DOCENTEANIO_K"."DOCACTIVO" IS '1=SI 2=NO';	COMMENT ON COLUMN public."T_DOCENTEANIO_K"."DOCACTIVO" IS '1=SI 2=NO';	FALSO				X				
ALTER TABLE "T_DOCENTEANIO_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_DOCENTEANIO_K" ADD FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_CONSEJOEJECUTIVO_K	--DDL for Table T_CONSEJOEJECUTIVO_K		2	36	38	12	25	37	0,05	0,32
----------	----------									
CREATE TABLE "T_CONSEJOEJECUTIVO_K" (	CREATE TABLE "T_CONSEJOEJECUTIVO_K" (	VERDADERO								
"CONRECTOR" NUMBER(3,0) NOT NULL ENABLE,	"CONRECTOR" SMALLINT NOT NULL,	FALSO				X				
"CONVICERECTOR" NUMBER(3,0) NOT NULL ENABLE,	"CONVICERECTOR" SMALLINT NOT NULL,	FALSO				X				
"CONSECRETARIA" NUMBER(3,0) NOT NULL ENABLE,	"CONSECRETARIA" SMALLINT NOT NULL,	FALSO				X				
"CONVOCAL1" NUMBER(3,0) NOT NULL ENABLE,	"CONVOCAL1" SMALLINT NOT NULL,	FALSO				X				
"CONVOCAL2" NUMBER(3,0) NOT NULL ENABLE,	"CONVOCAL2" SMALLINT NOT NULL,	FALSO				X				
"CONVOCAL3" NUMBER(3,0) NOT NULL ENABLE,	"CONVOCAL3" SMALLINT NOT NULL,	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CONINSPECTOR" NUMBER(2,0) NOT NULL ENABLE,	"CONINSPECTOR" SMALLINT NOT NULL,	FALSO				X				
"CONCOLECTURIA" NUMBER(2,0) NOT NULL ENABLE,	"CONCOLECTURIA" SMALLINT NOT NULL,	FALSO				X				
"CONPSICOLOGO" NUMBER(2,0) NOT NULL ENABLE,	"CONPSICOLOGO" SMALLINT NOT NULL,	FALSO				X				
"CONMEDICO" NUMBER(2,0) NOT NULL ENABLE,	"CONMEDICO" SMALLINT NOT NULL,	FALSO				X				
"CONDEPTECNICO" NUMBER(2,0) NOT NULL ENABLE,	"CONDEPTECNICO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO","CONRECTOR", "CONVICERECTOR", "CONSECRETARIA", "CONVOCAL1", "CONVOCAL2", "CONVOCAL3", "CONINSPECTOR", "CONCOLECTURIA", "CONPSICOLOGO", "CONMEDICO", "CONDEPTECNICO"));	PRIMARY KEY ("ANICODIGO","CONRECTOR", "CONVICERECTOR", "CONSECRETARIA", "CONVOCAL1", "CONVOCAL2", "CONVOCAL3", "CONINSPECTOR", "CONCOLECTURIA", "CONPSICOLOGO", "CONMEDICO", "CONDEPTECNICO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"),	FALSO					X			
	FOREIGN KEY ("CONRECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			

	FOREIGN KEY ("CONVICERECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONSECRETARIA") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONVOCAL1") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONVOCAL2") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONVOCAL3") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONINSPECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONCOLECTURIA") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONPSICOLOGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONMEDICO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"),	FALSO					X			
	FOREIGN KEY ("CONDEPTECNICO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"));	FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONRECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONVICERECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONSECRETARIA") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONVOCAL1") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONVOCAL2") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD FOREIGN KEY ("CONVOCAL3") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD CONSTRAINT "T_CONSEJOEJECUTIVO_K_FK1" FOREIGN KEY ("CONINSPECTOR") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			

ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD CONSTRAINT "T_CONSEJOEJECUTIVO_K_FK2" FOREIGN KEY ("CONCOLECTURIA") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD CONSTRAINT "T_CONSEJOEJECUTIVO_K_FK3" FOREIGN KEY ("CONPSICOLOGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD CONSTRAINT "T_CONSEJOEJECUTIVO_K_FK4" FOREIGN KEY ("CONMEDICO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CONSEJOEJECUTIVO_K" ADD CONSTRAINT "T_CONSEJOEJECUTIVO_K_FK5" FOREIGN KEY ("CONDEPTECNICO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_NOMENCLATURA_K	--DDL for Table T_NOMENCLATURA_K		2	2	4	2	1	3	0,50	0,67
----------	----------									
CREATE TABLE "T_NOMENCLATURA_K" (	CREATE TABLE "T_NOMENCLATURA_K" (	VERDADERO								
"NOMCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_NOMCODIGO" CHECK (NOMCODIGO>0) ENABLE,	"NOMCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_NOMCODIGO" CHECK ("NOMCODIGO">0),	FALSO				X				
"NOMDETALLE" VARCHAR2(50 BYTE) NOT NULL ENABLE,	"NOMDETALLE" VARCHAR(50) NOT NULL,	FALSO				X				
PRIMARY KEY ("NOMCODIGO"));	PRIMARY KEY ("NOMCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_ESCALA_K	--DDL for Table T_ESCALA_K		2	7	9	4	3	7	0,22	0,57
----------	----------									
CREATE TABLE "T_ESCALA_K" (	CREATE TABLE "T_ESCALA_K" (	VERDADERO								
"NOMCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"NOMCODIGO" SMALLINT NOT NULL,	FALSO				X				
"ESCCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_ESCCODIGO" CHECK (ESCCODIGO>0) ENABLE,	"ESCCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_ESCCODIGO" CHECK ("ESCCODIGO">0),	FALSO				X				
"ESCESCALA" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"ESCESCALA" VARCHAR(20) NOT NULL,	FALSO				X				
"ESCVALOR" NUMBER(4,2) NOT NULL CONSTRAINT "CHK_ESCVALOR" CHECK (ESCVALOR>=0) ENABLE,	"ESCVALOR" DECIMAL(4,2) NOT NULL CONSTRAINT "CHK_ESCVALOR" CHECK ("ESCVALOR">=0),	FALSO								
"ESCOBSERVACION" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"ESCOBSERVACION" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("NOMCODIGO", "ESCCODIGO"));	PRIMARY KEY ("NOMCODIGO", "ESCCODIGO"),	VERDADERO					X			

	FOREIGN KEY ("NOMCODIGO") REFERENCES "T_NOMENCLATURA_K" ("NOMCODIGO"));	FALSO					X			
ALTER TABLE "T_ESCALA_K" ADD FOREIGN KEY ("NOMCODIGO") REFERENCES "T_NOMENCLATURA_K" ("NOMCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_ESCALACOMPORTAMIENTO_K	--DDL for Table T_ESCALACOMPORTAMIENTO_K		2	7	9	4	3	7	0,22	0,57
----------	----------									
CREATE TABLE "T_ESCALACOMPORTAMIENTO_K" (	CREATE TABLE "T_ESCALACOMPORTAMIENTO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"ESCOMCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ESCOMCODIGO" SMALLINT NOT NULL,	FALSO				X				
"ESCOMESCALA" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"ESCOMESCALA" VARCHAR(20) NOT NULL,	FALSO				X				
"ESCOMVALOR" NUMBER(4,2) NOT NULL CONSTRAINT "CHK_ESCOMVALOR" CHECK (ESCOMVALOR >=0) ENABLE,	"ESCOMVALOR" DECIMAL(4,2) NOT NULL CONSTRAINT "CHK_ESCOMVALOR" CHECK ("ESCOMVALOR" >=0),	FALSO								
"ESCOMBSERVACION" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"ESCOMBSERVACION" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "ESCOMCODIGO"));	PRIMARY KEY ("ANICODIGO", "ESCOMCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
ALTER TABLE "T_ESCALACOMPORTAMIENTO_K" ADD CONSTRAINT "T_ESCALACOMPORTAMIENTO_K_FK1" FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_CURSO_K	--DDL for Table T_CURSO_K		23	24	47	37	9	46	0,49	0,80
----------	----------									
CREATE TABLE "T_CURSO_K" (	CREATE TABLE "T_CURSO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_CURCODIGO" CHECK (CURCODIGO>0) ENABLE,	"CURCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_CURCODIGO" CHECK ("CURCODIGO">0),	FALSO				X				
"CURALIAS" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"CURALIAS" VARCHAR(20) NOT NULL,	FALSO				X				
"CURDETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"CURDETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"CURPARALELO" VARCHAR2(4 BYTE) NOT NULL ENABLE,	"CURPARALELO" VARCHAR(4) NOT NULL,	FALSO				X				

"CURSUBNIVEL" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CURSUBNIVEL" CHECK (CURSUBNIVEL>=1 AND CURSUBNIVEL<=7) ENABLE,	"CURSUBNIVEL" SMALLINT NOT NULL CONSTRAINT "CHK_CURSUBNIVEL" CHECK ("CURSUBNIVEL">=1 AND "CURSUBNIVEL"<=7),	FALSO				X				
"CURSECUENCIA" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_CURSECUENCIA" CHECK (CURSECUENCIA>=0 AND CURSECUENCIA<=13) ENABLE,	"CURSECUENCIA" SMALLINT NOT NULL CONSTRAINT "CHK_CURSECUENCIA" CHECK ("CURSECUENCIA">=0 AND "CURSECUENCIA"<=13),	FALSO				X				
"CURSECCION" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_CURSECCION" CHECK (CURSECCION>=1 AND CURSECCION<=13) ENABLE,	"CURSECCION" SMALLINT NOT NULL CONSTRAINT "CHK_CURSECCION" CHECK ("CURSECCION">=1 AND "CURSECCION"<=13),	FALSO				X				
"ESPCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ESPCODIGO" SMALLINT NOT NULL,	FALSO				X				
"AUXCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"AUXCODIGO" SMALLINT NOT NULL,	FALSO				X				
"DOCCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"DOCCODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURTIPO" NUMBER(1,0) NOT NULL ENABLE,	"CURTIPO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"),	FALSO					X			
	FOREIGN KEY ("ESPCODIGO") REFERENCES "T_ESPECIALIDAD_K" ("ESPCODIGO"),	FALSO					X			
	FOREIGN KEY ("AUXCODIGO") REFERENCES "T_AUXILIATURA_K" ("AUXCODIGO"),	FALSO					X			
	FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_CURSO_K"."CURSUBNIVEL" IS '1=INICIAL 1	COMMENT ON COLUMN public."T_CURSO_K"."CURSUBNIVEL" IS '1=INICIAL 1	FALSO				X				
2=INICIAL 2	2=INICIAL 2	VERDADERO				X				
3=PREPARATORIA	3=PREPARATORIA	VERDADERO				X				
4=BASICA ELEMENTAL	4=BASICA ELEMENTAL	VERDADERO				X				
5=BASICA MEDIA	5=BASICA MEDIA	VERDADERO				X				
6=BASICA SUPERIOR	6=BASICA SUPERIOR	VERDADERO				X				
7=BACHILLERATO';	7=BACHILLERATO';	VERDADERO				X				
COMMENT ON COLUMN "T_CURSO_K"."CURSECUENCIA" IS '0=INICIAL 1 Y 2	COMMENT ON COLUMN public."T_CURSO_K"."CURSECUENCIA" IS '0=INICIAL 1 Y 2	FALSO				X				
1=PRIMERO	1=PRIMERO	VERDADERO				X				
2=SEGUNDO	2=SEGUNDO	VERDADERO				X				
3=TERCERO	3=TERCERO	VERDADERO				X				

4=CUARTO	4=CUARTO	VERDADERO				X				
5=QUINTO	5=QUINTO	VERDADERO				X				
6=SEXTO	6=SEXTO	VERDADERO				X				
7=SEPTIMO	7=SEPTIMO	VERDADERO				X				
8=OCTAVO	8=OCTAVO	VERDADERO				X				
9=NOVENO	9=NOVENO	VERDADERO				X				
10=DECIMO	10=DECIMO	VERDADERO				X				
11=PRIMERO BACHILLERATO	11=PRIMERO BACHILLERATO	VERDADERO				X				
12=SEGUNDO BACHILLERATO	12=SEGUNDO BACHILLERATO	VERDADERO				X				
13=TERCERO BACHILLERATO';	13=TERCERO BACHILLERATO';	VERDADERO				X				
COMMENT ON COLUMN "T_CURSO_K"."CURSECCION" IS '1=MATUTINA	COMMENT ON COLUMN public."T_CURSO_K"."CURSECCION" IS '1=MATUTINA	FALSO				X				
2=VESPERTINA	2=VESPERTINA	VERDADERO				X				
3=NOCTURNA';	3=NOCTURNA';	VERDADERO				X				
COMMENT ON COLUMN "T_CURSO_K"."CURTIPO" IS '1= MATRICULAS 2=INSCRICIONES';	COMMENT ON COLUMN public."T_CURSO_K"."CURTIPO" IS '1= MATRICULAS 2=INSCRICIONES';	FALSO				X				
ALTER TABLE "T_CURSO_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CURSO_K" ADD FOREIGN KEY ("ESPCODIGO") REFERENCES "T_ESPECIALIDAD_K" ("ESPCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CURSO_K" ADD FOREIGN KEY ("AUXCODIGO") REFERENCES "T_AUXILIATURA_K" ("AUXCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_CURSO_K" ADD FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_ACCESO_CURSO_K	--DDL for Table T_ACCESO_CURSO_K		2	7	9	3	5	8	0,22	0,38
----------	----------									
CREATE TABLE "T_ACCESO_CURSO_K" (	CREATE TABLE "T_ACCESO_CURSO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				

"USUCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"USUCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO", "USUCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO", "USUCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO"),	FALSO					X			
	FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO"));	FALSO					X			
ALTER TABLE "T_ACCESO_CURSO_K" ADD FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_ACCESO_CURSO_K" ADD FOREIGN KEY ("USUCODIGO") REFERENCES "T_USUARIO_K" ("USUCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_AREAACADEMICA_K	--DDL for Table T_AREAACADEMICA_K		2	7	9	5	3	8	0,22	0,63
----------	----------									
CREATE TABLE "T_AREAACADEMICA_K" (	CREATE TABLE "T_AREAACADEMICA_K" (	VERDADERO								
"AREACODIGO" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_AREACODIGO" CHECK (AREACODIGO>0) ENABLE,	"AREACODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_AREACODIGO" CHECK ("AREACODIGO">0),	FALSO				X				
"AREANOMBRE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"AREANOMBRE" VARCHAR(80) NOT NULL,	FALSO				X				
"AREADETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"AREADETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"AREAORDEN" NUMBER(2,0) NOT NULL ENABLE,	"AREAORDEN" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "AREACODIGO"));	PRIMARY KEY ("ANICODIGO", "AREACODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
ALTER TABLE "T_AREAACADEMICA_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_MATERIA_K	--DDL for Table T_MATERIA_K		2	26	28	20	7	27	0,07	0,74
----------	----------									
CREATE TABLE "T_MATERIA_K" (	CREATE TABLE "T_MATERIA_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				

"MATCODIGO" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_MATTCODIGO" CHECK (MATCODIGO>0) ENABLE,	"MATCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_MATTCODIGO" CHECK ("MATCODIGO">0),	FALSO				X				
"MATALIAS" VARCHAR2(20 BYTE) NOT NULL ENABLE,	"MATALIAS" VARCHAR(20) NOT NULL,	FALSO				X				
"MATDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"MATDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"MATORDEN" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_MATORDEN" CHECK (MATORDEN>=0) ENABLE,	"MATORDEN" SMALLINT NOT NULL CONSTRAINT "CHK_MATORDEN" CHECK ("MATORDEN">=0),	FALSO				X				
"MATHORAS" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_MATHORAS" CHECK (MATHORAS>=0) ENABLE,	"MATHORAS" SMALLINT NOT NULL CONSTRAINT "CHK_MATHORAS" CHECK ("MATHORAS">=0),	FALSO				X				
"MATITPO" NUMBER(1,0) NOT NULL ENABLE,	"MATITPO" SMALLINT NOT NULL,	FALSO				X				
"MATCLASE" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_MATCLASE" CHECK (MATCLASE>=1 AND MATCLASE<=2) ENABLE,	"MATCLASE" SMALLINT NOT NULL CONSTRAINT "CHK_MATCLASE" CHECK ("MATCLASE">=1 AND "MATCLASE"<=2),	FALSO				X				
"MATAUXILIATURA" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_MATAUXILIATURA" CHECK (MATAUXILIATURA>=1 AND MATAUXILIATURA<=2) ENABLE,	"MATAUXILIATURA" SMALLINT NOT NULL CONSTRAINT "CHK_MATAUXILIATURA" CHECK ("MATAUXILIATURA">=1 AND "MATAUXILIATURA"<=2),	FALSO				X				
"MATGRADO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_MATGRADO" CHECK (MATGRADO>=1 AND MATGRADO<=2) ENABLE,	"MATGRADO" SMALLINT NOT NULL CONSTRAINT "CHK_MATGRADO" CHECK ("MATGRADO">=1 AND "MATGRADO"<=2),	FALSO				X				
"MATOPTATIVA" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_MATOPTATIVA" CHECK (MATOPTATIVA>=1 AND MATOPTATIVA<=2) ENABLE,	"MATOPTATIVA" SMALLINT NOT NULL CONSTRAINT "CHK_MATOPTATIVA" CHECK ("MATOPTATIVA">=1 AND "MATOPTATIVA"<=2),	FALSO				X				
"NOMCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"NOMCODIGO" SMALLINT NOT NULL,	FALSO				X				
"DOCCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"DOCCODIGO" SMALLINT NOT NULL,	FALSO				X				
"AREACODIGO" NUMBER(3,0) NOT NULL ENABLE,	"AREACODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("NOMCODIGO") REFERENCES "T_NOMENCLATURA_K" ("NOMCODIGO"),	FALSO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO"),	FALSO					X			
	FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_MATERIA_K"."MATITPO" IS '1=PROMEDIADA 2=NO PROMEDIADA 3=OTROS';	COMMENT ON COLUMN public."T_MATERIA_K"."MATITPO" IS '1=PROMEDIADA 2=NO PROMEDIADA 3=OTROS';	FALSO				X				
COMMENT ON COLUMN "T_MATERIA_K"."MATCLASE" IS '1=NUMERICA 2=NO NUMERICA';	COMMENT ON COLUMN public."T_MATERIA_K"."MATCLASE" IS '1=NUMERICA 2=NO NUMERICA';	FALSO				X				
COMMENT ON COLUMN "T_MATERIA_K"."MATAUXILIATURA" IS '1=SI 2=NO';	COMMENT ON COLUMN public."T_MATERIA_K"."MATAUXILIATURA" IS '1=SI 2=NO';	FALSO				X				

COMMENT ON COLUMN "T_MATERIA_K"."MATGRADO" IS '1=SI 2=NO';	COMMENT ON COLUMN public."T_MATERIA_K"."MATGRADO" IS '1=SI 2=NO';	FALSO				X				
COMMENT ON COLUMN "T_MATERIA_K"."MATOPTATIVA" IS '1=SI 2=NO';	COMMENT ON COLUMN public."T_MATERIA_K"."MATOPTATIVA" IS '1=SI 2=NO';	FALSO				X				
ALTER TABLE "T_MATERIA_K" ADD FOREIGN KEY ("NOMCODIGO") REFERENCES "T_NOMENCLATURA_K" ("NOMCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_MATERIA_K" ADD FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_MATERIA_K" ADD FOREIGN KEY ("DOCCODIGO") REFERENCES "T_DOCENTE_S" ("DOCCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_MATERIA_AMBITO_K	--DDL for Table T_MATERIA_AMBITO_K		2	9	11	7	3	10	0,18	0,70
----------	----------									
CREATE TABLE "T_MATERIA_AMBITO_K" (	CREATE TABLE "T_MATERIA_AMBITO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				
"MATCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"MATCODIGO" SMALLINT NOT NULL,	FALSO				X				
"MATAMBCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"MATAMBCODIGO" SMALLINT NOT NULL,	FALSO				X				
"MATAMBDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"MATAMBDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"MATAMBALIAS" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"MATAMBALIAS" VARCHAR(100) NOT NULL,	FALSO				X				
"MATAMBORDEN" NUMBER(2,0) NOT NULL ENABLE,	MATAMBORDEN SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO", "MATAMBCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO", "MATAMBCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO") REFERENCES "T_MATERIA_K" ("ANICODIGO", "CURCODIGO", "MATCODIGO"));	FALSO					X			
ALTER TABLE "T_MATERIA_AMBITO_K" ADD FOREIGN KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO") REFERENCES "T_MATERIA_K" ("ANICODIGO", "CURCODIGO", "MATCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_ANIOCALIFICACION_K	--DDL for Table T_ANIOCALIFICACION_K		2	12	14	9	3	12	0,14	0,75
----------	----------									

CREATE TABLE "T_ANIOCALIFICACION_K" (	CREATE TABLE "T_ANIOCALIFICACION_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"ANINUMPARCIAL" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_ANINUMPARCIAL" CHECK (ANINUMPARCIAL>=1 AND ANINUMPARCIAL<=5) ENABLE,	"ANINUMPARCIAL" SMALLINT NOT NULL CONSTRAINT "CHK_ANINUMPARCIAL" CHECK ("ANINUMPARCIAL">=1 AND "ANINUMPARCIAL"<=5),	FALSO				X				
"ANIPONPARCIAL1" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL1" CHECK (ANIPONPARCIAL1>=0 AND ANIPONPARCIAL1<=100) ENABLE,	"ANIPONPARCIAL1" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL1" CHECK ("ANIPONPARCIAL1">=0 AND "ANIPONPARCIAL1"<=100),	FALSO				X				
"ANIPONPARCIAL2" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL2" CHECK (ANIPONPARCIAL2>=0 AND ANIPONPARCIAL2<=100) ENABLE,	"ANIPONPARCIAL2" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL2" CHECK ("ANIPONPARCIAL2">=0 AND "ANIPONPARCIAL2"<=100),	FALSO				X				
"ANIPONPARCIAL3" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL3" CHECK (ANIPONPARCIAL3>=0 AND ANIPONPARCIAL3<=100) ENABLE,	"ANIPONPARCIAL3" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL3" CHECK ("ANIPONPARCIAL3">=0 AND "ANIPONPARCIAL3"<=100),	FALSO				X				
"ANIPONPARCIAL4" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL4" CHECK (ANIPONPARCIAL4>=0 AND ANIPONPARCIAL4<=100) ENABLE;,	"ANIPONPARCIAL4" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL4" CHECK ("ANIPONPARCIAL4">=0 AND "ANIPONPARCIAL4"<=100),	FALSO				X				
"ANIPONPARCIAL5" NUMBER(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL5" CHECK (ANIPONPARCIAL5>=0 AND ANIPONPARCIAL5<=100) ENABLE,	"ANIPONPARCIAL5" DECIMAL(5,2) NOT NULL CONSTRAINT "CHK_ANIPONPARCIAL5" CHECK ("ANIPONPARCIAL5">=0 AND "ANIPONPARCIAL5"<=100),	FALSO				X				
"ANINOTAMIN" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_ANINOTAMIN" CHECK (ANINOTAMIN>=0) ENABLE,	"ANINOTAMIN" SMALLINT NOT NULL CONSTRAINT "CHK_ANINOTAMIN" CHECK ("ANINOTAMIN">=0),	FALSO				X				
"ANINOTAMAX" NUMBER(3,0) NOT NULL CONSTRAINT "CHK_ANINOTAMAX" CHECK (ANINOTAMAX>=0) ENABLE,	ANINOTAMAX SMALLINT NOT NULL CONSTRAINT "CHK_ANINOTAMAX" CHECK ("ANINOTAMAX">=0),	FALSO				X				
PRIMARY KEY ("ANICODIGO"));	PRIMARY KEY ("ANICODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_ANIOCALIFICACION_K"."ANINUMPARCIAL" IS 'ENTRE 1 Y 5';	COMMENT ON COLUMN public."T_ANIOCALIFICACION_K"."ANINUMPARCIAL" IS 'ENTRE 1 Y 5';	FALSO								
ALTER TABLE "T_ANIOCALIFICACION_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_TIPO_HORARIO_K	--DDL for Table T_TIPO_HORARIO_K		2	7	9	5	3	8	0,22	0,63
----------	----------									
CREATE TABLE "T_TIPO_HORARIO_K" (	CREATE TABLE "T_TIPO_HORARIO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				

"CURSECCION" NUMBER(3,0) NOT NULL ENABLE,	"CURSECCION" SMALLINT NOT NULL,	FALSO				X				
"TIPHORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"TIPHORCODIGO" SMALLINT NOT NULL,	FALSO				X				
"TIPHORDETALLE" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"TIPHORDETALLE" VARCHAR(200) NOT NULL,	FALSO				X				
"TIPHOROBSERVACION" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"TIPHOROBSERVACION" VARCHAR(200) NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "TIPHORCODIGO", "CURSECCION"));	PRIMARY KEY ("ANICODIGO", "TIPHORCODIGO", "CURSECCION"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
ALTER TABLE "T_TIPO_HORARIO_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_HORAS_K	--DDL for Table T_HORAS_K		1	13	14	10	3	13	0,07	0,77
----------	----------									
CREATE TABLE "T_HORAS_K" (	CREATE TABLE "T_HORAS_K" (	VERDADERO								
"HORCODIGO" NUMBER NOT NULL CONSTRAINT "CHK_HORCODIGO" CHECK (HORCODIGO>0) ENABLE,	"HORCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_HORCODIGO" CHECK ("HORCODIGO">0),	FALSO				X				
"HORORDEN" NUMBER NOT NULL CONSTRAINT "CHK_HORORDEN" CHECK (HORORDEN>=0) ENABLE,	"HORORDEN" SMALLINT NOT NULL CONSTRAINT "CHK_HORORDEN" CHECK ("HORORDEN">=0),	FALSO				X				
"HORHORAINICIO" NUMBER NOT NULL CONSTRAINT "CHK_HORHORAINICIO" CHECK (HORHORAINICIO>0) ENABLE,	"HORHORAINICIO" SMALLINT NOT NULL CONSTRAINT "CHK_HORHORAINICIO" CHECK ("HORHORAINICIO">0),	FALSO				X				
"HORHORAFIN" NUMBER NOT NULL CONSTRAINT "CHK_HORHORAFIN" CHECK (HORHORAFIN>0) ENABLE,	"HORHORAFIN" SMALLINT NOT NULL CONSTRAINT "CHK_HORHORAFIN" CHECK ("HORHORAFIN">0),	FALSO				X				
"HORMINUTOSINICIO" NUMBER NOT NULL CONSTRAINT "CHK_HORMINUTOSINICIO" CHECK (HORMINUTOSINICIO>=0) ENABLE,	"HORMINUTOSINICIO" SMALLINT NOT NULL CONSTRAINT "CHK_HORMINUTOSINICIO" CHECK ("HORMINUTOSINICIO">=0),	FALSO				X				
"HORMINUTOSFIN" NUMBER NOT NULL CONSTRAINT "CHK_HORMINUTOSFIN" CHECK (HORMINUTOSFIN>=0) ENABLE,	"HORMINUTOSFIN" SMALLINT NOT NULL CONSTRAINT "CHK_HORMINUTOSFIN" CHECK ("HORMINUTOSFIN">=0),	FALSO				X				
"HOROBSERVACION" VARCHAR2(150 BYTE) NOT NULL ENABLE,	"HOROBSERVACION" VARCHAR(150) NOT NULL,	FALSO				X				
"CURSECCION" NUMBER(3,0) NOT NULL ENABLE,	"CURSECCION" SMALLINT NOT NULL,	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"TIPHORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	TIPHORCODIGO SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY("HORCODIGO", "TIPHORCODIGO", "ANICODIGO", "CURSECCION"));	PRIMARY KEY("HORCODIGO", "TIPHORCODIGO", "ANICODIGO", "CURSECCION"),	FALSO					X			

	FOREIGN KEY ("ANICODIGO", "TIPHORCODIGO", "CURSECCION") REFERENCES "T_TIPO_HORARIO_K" ("ANICODIGO", "TIPHORCODIGO", "CURSECCION"));	FALSO					X			
ALTER TABLE "T_HORAS_K" ADD CONSTRAINT "T_HORAS_K_FK1" FOREIGN KEY ("ANICODIGO", "TIPHORCODIGO", "CURSECCION") REFERENCES "T_TIPO_HORARIO_K" ("ANICODIGO", "TIPHORCODIGO", "CURSECCION") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_HORARIO_K	--DDL for Table T_HORARIO_K		2	12	14	8	5	13	0,14	0,62
----------	----------									
CREATE TABLE "T_HORARIO_K" (	CREATE TABLE "T_HORARIO_K" (	VERDADERO								
"HRADIA" NUMBER(2,0) NOT NULL CONSTRAINT "CHK_HRADIA" CHECK (HRADIA>0) ENABLE,	"HRADIA" SMALLINT NOT NULL CONSTRAINT "CHK_HRADIA" CHECK ("HRADIA">0),	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURSECCION" NUMBER(3,0) NOT NULL ENABLE,	"CURSECCION" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				
"MATCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"MATCODIGO" SMALLINT NOT NULL,	FALSO				X				
"HORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"HORCODIGO" SMALLINT NOT NULL,	FALSO				X				
"HRADOCENTE" NUMBER(10,0) NOT NULL ENABLE,	"HRADOCENTE" SMALLINT NOT NULL,	FALSO				X				
"TIPHORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"TIPHORCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO", "HORCODIGO", "HRADIA", "CURSECCION"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO", "HORCODIGO", "HRADIA", "CURSECCION"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO"),	FALSO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO") REFERENCES "T_MATERIA_K" ("ANICODIGO", "CURCODIGO", "MATCODIGO"));	FALSO					X			
ALTER TABLE "T_HORARIO_K" ADD CONSTRAINT "T_HORARIO_K_FK2" FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_HORARIO_K" ADD CONSTRAINT "T_HORARIO_K_T_MATERIA_K_FK1" FOREIGN KEY ("ANICODIGO", "CURCODIGO", "MATCODIGO") REFERENCES "T_MATERIA_K" ("ANICODIGO", "CURCODIGO", "MATCODIGO") ENABLE;		FALSO					X			

----------	----------									
--DDL for Table T_ASIGNA_HORARIO_K	--DDL for Table T_ASIGNA_HORARIO_K		2	8	10	4	5	9	0,20	0,44
----------	----------									
CREATE TABLE "T_ASIGNA_HORARIO_K" (	CREATE TABLE "T_ASIGNA_HORARIO_K" (	VERDADERO								
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURSECCION" NUMBER(3,0) NOT NULL ENABLE,	"CURSECCION" SMALLINT NOT NULL,	FALSO				X				
"TIPHORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"TIPHORCODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "CURCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"),	FALSO					X			
	FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO"));	FALSO					X			
ALTER TABLE "T_ASIGNA_HORARIO_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_ASIGNA_HORARIO_K" ADD FOREIGN KEY ("ANICODIGO", "CURCODIGO") REFERENCES "T_CURSO_K" ("ANICODIGO", "CURCODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_TMPHORARIO_K	--DDL for Table T_TMPHORARIO_K		2	8	10	8	1	9	0,20	0,89
----------	----------									
CREATE TABLE "T_TMPHORARIO_K" (	CREATE TABLE "T_TMPHORARIO_K" (	VERDADERO								
"HRADIA" NUMBER(2,0) NOT NULL ENABLE,	"HRADIA" SMALLINT NOT NULL,	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"CURSECCION" NUMBER(3,0) NOT NULL ENABLE,	"CURSECCION" SMALLINT NOT NULL,	FALSO				X				
"CURCODIGO" NUMBER(3,0) NOT NULL ENABLE,	"CURCODIGO" SMALLINT NOT NULL,	FALSO				X				
"MATCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"MATCODIGO" SMALLINT NOT NULL,	FALSO				X				
"HORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	HORCODIGO SMALLINT NOT NULL,	FALSO				X				
"HRADOCENTE" NUMBER(10,0) NOT NULL ENABLE,	"HRADOCENTE" SMALLINT NOT NULL,	FALSO				X				
"TIPHORCODIGO" NUMBER(2,0) NOT NULL ENABLE,	"TIPHORCODIGO" SMALLINT NOT NULL,	FALSO				X				

PRIMARY KEY ("ANICODIGO", "CURCODIGO", "HORCODIGO", "HRADIA", "CURSECCION", "TIPHORCODIGO"));	PRIMARY KEY ("ANICODIGO", "CURCODIGO", "HORCODIGO", "HRADIA", "CURSECCION", "TIPHORCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_TIPOCUENTAS_K	--DDL for Table T_TIPOCUENTAS_K		2	2	4	2	1	3	0,50	0,67
----------	----------									
CREATE TABLE "T_TIPOCUENTAS_K" (	CREATE TABLE "T_TIPOCUENTAS_K" (	VERDADERO								
"TCUCODIGO" NUMBER(10,0) NOT NULL CONSTRAINT "CHK_TCUCODIGO" CHECK (TCUCODIGO>0) ENABLE,	"TCUCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_TCUCODIGO" CHECK ("TCUCODIGO">0),	FALSO				X				
"TCUDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"TCUDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
PRIMARY KEY ("TCUCODIGO"));	PRIMARY KEY ("TCUCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_TIPORETIVA_K	--DDL for Table T_TIPORETIVA_K		2	4	6	4	1	5	0,33	0,80
----------	----------									
CREATE TABLE "T_TIPORETIVA_K" (	CREATE TABLE "T_TIPORETIVA_K" (	VERDADERO								
"TRICODIGO" NUMBER(10,0) NOT NULL CONSTRAINT "CHK_TRICODIGO" CHECK (TRICODIGO>0) ENABLE,	"TRICODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_TRICODIGO" CHECK ("TRICODIGO">0),	FALSO				X				
"TRIDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"TRIDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"TRIFACTORMULTI" NUMBER(12,4) NOT NULL CONSTRAINT "CHK_TRIFACTORMULTI" CHECK (TRIFACTORMULTI>=0) ENABLE,	"TRIFACTORMULTI" DECIMAL(12,4) NOT NULL CONSTRAINT "CHK_TRIFACTORMULTI" CHECK ("TRIFACTORMULTI">=0),	FALSO				X				
"TRICODIGOSECUNDARIO" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"TRICODIGOSECUNDARIO" VARCHAR(10) NOT NULL,	FALSO				X				
PRIMARY KEY ("TRICODIGO"));	PRIMARY KEY ("TRICODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_TIPORETRENTA_K	--DDL for Table T_TIPORETRENTA_K		2	4	6	4	1	5	0,33	0,80
----------	----------									
CREATE TABLE "T_TIPORETRENTA_K" (	CREATE TABLE "T_TIPORETRENTA_K" (	VERDADERO								
"TRRCODIGO" NUMBER(10,0) NOT NULL CONSTRAINT "CHK_TRRCODIGO" CHECK (TRRCODIGO>0) ENABLE,	"TRRCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_TRRCODIGO" CHECK ("TRRCODIGO">0),	FALSO				X				
"TRRDETALLE" VARCHAR2(100 BYTE) NOT NULL ENABLE,	"TRRDETALLE" VARCHAR(100) NOT NULL,	FALSO				X				
"TRRFACTORMULTI" NUMBER(10,2) NOT NULL CONSTRAINT "CHK_TRRFACTORMULTI" CHECK (TRRFACTORMULTI>=0) ENABLE,	"TRRFACTORMULTI" DECIMAL(10,2) NOT NULL CONSTRAINT "CHK_TRRFACTORMULTI" CHECK ("TRRFACTORMULTI">=0),	FALSO				X				

"TRRCODIGOSECUNDARIO" VARCHAR2(10 BYTE) NOT NULL ENABLE,	"TRRCODIGOSECUNDARIO" VARCHAR(10) NOT NULL,	FALSO				X				
PRIMARY KEY ("TRRCODIGO"));	PRIMARY KEY ("TRRCODIGO"));	VERDADERO					X			
----------	----------									
--DDL for Table T_TIPO_RUBROS_K	--DDL for Table T_TIPO_RUBROS_K		2	6	8	4	3	7	0,25	0,57
----------	----------									
CREATE TABLE "T_TIPO_RUBROS_K" (	CREATE TABLE "T_TIPO_RUBROS_K" (	VERDADERO								
"TIRUBCODIGO" NUMBER(4,0) NOT NULL ENABLE,	"TIRUBCODIGO" SMALLINT NOT NULL,	FALSO				X				
"TIRUBDETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"TIRUBDETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"TIRUBOBSERVACION" VARCHAR2(200 BYTE) NOT NULL ENABLE,	"TIRUBOBSERVACION" VARCHAR(200) NOT NULL,	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("TIRUBCODIGO", "ANICODIGO"));	PRIMARY KEY ("TIRUBCODIGO", "ANICODIGO"),	VERDADERO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
ALTER TABLE "T_TIPO_RUBROS_K" ADD FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			
----------	----------									
--DDL for Table T_RUBROS_K	--DDL for Table T_RUBROS_K		2	16	18	12	5	17	0,11	0,71
----------	----------									
CREATE TABLE "T_RUBROS_K" (	CREATE TABLE "T_RUBROS_K" (	VERDADERO								
"RUBCODIGO" NUMBER(4,0) NOT NULL CONSTRAINT "CHK_RUBCODIGO" CHECK (RUBCODIGO>0) ENABLE,	"RUBCODIGO" SMALLINT NOT NULL CONSTRAINT "CHK_RUBCODIGO" CHECK ("RUBCODIGO">0),	FALSO				X				
"RUBDETALLE" VARCHAR2(80 BYTE) NOT NULL ENABLE,	"RUBDETALLE" VARCHAR(80) NOT NULL,	FALSO				X				
"RUBVALOR" NUMBER(8,2) NOT NULL CONSTRAINT "CHK_RUBVALOR" CHECK (RUBVALOR>0) ENABLE,	"RUBVALOR" DECIMAL(8,2) NOT NULL CONSTRAINT "CHK_RUBVALOR" CHECK ("RUBVALOR">0),	FALSO				X				
"RUBTIPO" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_RUBTIPO" CHECK (RUBTIPO>=0) ENABLE,	"RUBTIPO" SMALLINT NOT NULL CONSTRAINT "CHK_RUBTIPO" CHECK ("RUBTIPO">=0),	FALSO				X				
"RUBAPLICACION" NUMBER(1,0) NOT NULL CONSTRAINT "CHK_RUBAPLICACION" CHECK (RUBAPLICACION>=0) ENABLE,	"RUBAPLICACION" SMALLINT NOT NULL CONSTRAINT "CHK_RUBAPLICACION" CHECK ("RUBAPLICACION">=0),	FALSO				X				
"ANICODIGO" NUMBER(2,0) NOT NULL ENABLE,	"ANICODIGO" SMALLINT NOT NULL,	FALSO				X				
"RUBMES" NUMBER(2,0) NOT NULL ENABLE,	"RUBMES" SMALLINT NOT NULL,	FALSO				X				
"RUBIVA" NUMBER(1,0) NOT NULL ENABLE,	"RUBIVA" SMALLINT NOT NULL,	FALSO				X				

"TIRUBCODIGO" NUMBER(4,0) NOT NULL ENABLE,	"TIRUBCODIGO" SMALLINT NOT NULL,	FALSO				X				
PRIMARY KEY ("ANICODIGO", "TIRUBCODIGO", "RUBCODIGO"));	PRIMARY KEY ("ANICODIGO", "TIRUBCODIGO", "RUBCODIGO"),	VERDADERO					X			
	FOREIGN KEY ("TIRUBCODIGO", "ANICODIGO") REFERENCES "T_TIPO_RUBROS_K" ("TIRUBCODIGO", "ANICODIGO"),	FALSO					X			
	FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO"));	FALSO					X			
COMMENT ON COLUMN "T_RUBROS_K"."RUBTIPO" IS '1= MATRICULA 2=PENSION 3=OTROS';	COMMENT ON COLUMN public."T_RUBROS_K"."RUBTIPO" IS '1= MATRICULA 2=PENSION 3=OTROS';	FALSO				X				
COMMENT ON COLUMN "T_RUBROS_K"."RUBAPLICACION" IS '1=FIJO 2=VARIABLE';	COMMENT ON COLUMN public."T_RUBROS_K"."RUBAPLICACION" IS '1=FIJO 2=VARIABLE';	FALSO				X				
COMMENT ON COLUMN "T_RUBROS_K"."RUBIVA" IS '1 = SI MARCA IVA 2 = NO MARCA IVA';	COMMENT ON COLUMN public."T_RUBROS_K"."RUBIVA" IS '1 = SI MARCA IVA 2 = NO MARCA IVA';	FALSO				X				
ALTER TABLE "T_RUBROS_K" ADD CONSTRAINT "T_RUBROS_K_FK1" FOREIGN KEY ("TIRUBCODIGO", "ANICODIGO") REFERENCES "T_TIPO_RUBROS_K" ("TIRUBCODIGO", "ANICODIGO") ENABLE;		FALSO					X			
ALTER TABLE "T_RUBROS_K" ADD CONSTRAINT "T_RUBROS_K__FK1" FOREIGN KEY ("ANICODIGO") REFERENCES "T_ANIOLECTIVO_K" ("ANICODIGO") ENABLE;		FALSO					X			

Anexo B: Resumen de análisis de tablas

TABLAS	ADAPTABILIDAD	CAPACIDAD PARA SER REEMLAZADO
KERNEL	**0.28**	**0.68**
--DDL for Table T_SISTEMA_K	0.70	0.22
--DDL for Table T_PROCESO_K	0.36	0.70
--DDL for Table T_USUARIO_K	0.29	0.83
--DDL for Table T_PERMISO_K	0.18	0.55
--DDL for Table T_AUDITORIA_K	0.17	0.44
--DDL for Table T_INSTITUCION_K	0.11	0.94
--DDL for Table T_AUXILIATURA_K	0.40	0.75
--DDL for Table T_ESPECIALIDAD_K	0.40	0.75
--DDL for Table T_CAJA_K	0.33	0.67
--DDL for Table T_FORMAPAGO_K	0.50	0.96
--DDL for Table T_INSFINANCIERAS_K	0.50	0.67
--DDL for Table T_NACIONALIDAD_K	0.50	0.67
--DDL for Table T_PROVINCIA_K	0.50	0.67
--DDL for Table T_CANTON_K	0.29	0.50
--DDL for Table T_PARROQUIA_K	0.25	0.57
--DDL for Table T_CONFIGURACION_K	0.17	0.81
--DDL for Table T_DOCENTE_S	0.33	0.98
--DDL for Table T_CONFIGURA_DOCENTE_K	0.05	0.92
--DDL for Table T_ANIOLECTIVO_K	0.09	0.95
--DDL for Table T_EDUCACION_MEDIA_K	0.40	0.75
--DDL for Table T_DOCENTEANIO_K	0.20	0.44
--DDL for Table T_CONSEJOEJECUTIVO_K	0.05	0.32
--DDL for Table T_NOMENCLATURA_K	0.50	0.67
--DDL for Table T_ESCALA_K	0.22	0.57
--DDL for Table T_ESCALACOMPORTAMIENTO_K	0.22	0.57
--DDL for Table T_CURSO_K	0.49	0.80
--DDL for Table T_ACCESO_CURSO_K	0.22	0.38
--DDL for Table T_AREAACADEMICA_K	0.22	0.63
--DDL for Table T_MATERIA_K	0.07	0.74
--DDL for Table T_MATERIA_AMBITO_K	0.18	0.70
--DDL for Table T_ANIOCALIFICACION_K	0.14	0.75
--DDL for Table T_TIPO_HORARIO_K	0.22	0.63
--DDL for Table T_HORAS_K	0.07	0.77
--DDL for Table T_HORARIO_K	0.14	0.62
--DDL for Table T_ASIGNA_HORARIO_K	0.20	0.44
--DDL for Table T_TMPHORARIO_K	0.20	0.89
--DDL for Table T_TIPOCUENTAS_K	0.50	0.67
--DDL for Table T_TIPORETIVA_K	0.33	0.80
--DDL for Table T_TIPORETRENTA_K	0.33	0.80
--DDL for Table T_TIPO_RUBROS_K	0.25	0.57
--DDL for Table T_RUBROS_K	0.11	0.71