UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

SISTEMA DE INFORMACIÓN GEOGRÁFICA CON ORACLE SPATIAL

Autora: Vanessa Malta Donoso

Tutor: Manuel Velasco de Diego

Índice

AGRADECIMIENTOS

A mi familia, por ser el pilar de mi vida y brindarme en cada momento el apoyo, en forma de cariño y palabras, que siempre necesitamos en los acontecimientos importantes de la vida.

A Ali, por aparecer en mi vida de esa forma tan bonita y contribuir a ser quien soy.

A mi tutor, por confiar en mí y darme la oportunidad de realizar este trabajo.

A todas las personas que he conocido a lo largo de mi periodo universitario, profesores y alumnos que, de un modo u otro, me han ayudado a crecer personal y profesionalmente.

1. INTRODUCCIÓN

1.1 Descripción y objetivos generales del problema

El presente trabajo tiene como objetivo emular una aplicación que gestione el urbanismo de una ciudad, mediante su representación a través de figuras geométricas en un determinado instante de tiempo.

Estas figuras poseerán características tales como ubicación espacial en un sistema de referencia y un datum determinado, así como otros datos de interés, como áreas, distancias y volúmenes (no posibles en este caso). A su vez, estas geometrías pueden estar relacionadas con otras de distinta naturaleza y con información no espacial que enriquece al sistema.

Para conseguir los objetivos propuestos en el dominio de nuestra aplicación, se ha optado por representar la información en una base de datos espacial, mediante el SGBD de Oracle 10g, haciendo uso de su módulo Oracle Spatial, previamente digitalizada por un programa destinado a dicho fin, como es ArcView 3.3 Desktop, perteneciente al conjunto de productos software ArcGIS, donde se agrupan varias aplicaciones para la captura, edición, análisis, tratamiento, diseño, publicación e impresión de información geográfica.

La explotación de esta información es transformada en conocimiento útil en un momento determinado, dando respuesta a múltiples incógnitas que se puedan plantear y permitiendo, entre otras cosas, la toma de decisiones en base a estos resultados.

2. ESTADO DEL ARTE

En este apartado se abordarán conceptos como el de base de datos relacional (introducción, concepto, diseño, reglas de normalización, modelos de representación, tipos) y sistema de información geográfica (introducción,

concepto, objetivos, componentes y aplicaciones comunes) que han servido para la resolución de este trabajo.

2.1 Bases de Datos Relacionales

Una base de datos relacional es un conjunto de dos o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente de le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional.

Estrictamente hablando el término se refiere a una colección específica de datos, pero a menudo se le usa, en forma errónea, como sinónimo del software usado para gestionar esa colección de datos. Ese software se conoce como sistema gestor de base de datos relacional o RDBMS (Relational Database Management System).

El modelo relacional es el más utilizado para modelar problemas reales y administrar datos dinámicamente. En 1970 fueron postuladas sus bases por Edgar Frank Codd, de los laboratorios IBM en San José (California).

Las reglas de Codd son las siguientes:

0. Todo sistema relacional debe manejar sus datos a través de sus capacidades relacionales exclusivamente.
1. Representación de la información: toda información en una base de datos relacional debe representarse explícitamente a nivel lógico, y de manera única, por medio de valores en tablas.
2. Acceso garantizado: todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave y el nombre de una columna.

3. Tratamiento sistemático de valores nulos: los valores nulos, información desconocida o inaplicable, han de ser tratados sistemáticamente por el sistema, el cual ha de ofrecer las facilidades necesarias para su tratamiento.

4. Catálogo activo en línea basado en el modelo relacional: la representación de la meta información (descripción de la base de datos) debe ser igual a la de los otros datos, y su acceso debe poder realizarse por medio del mismo lenguaje relacional que el utilizado para los demás datos; es decir, el modelo de datos para la meta información debe ser el relacional.

5. Sublenguaje de datos completo: debe existir un lenguaje que permita un completo manejo de la base de datos:

    - definición de datos
    - definición de vistas
    - manipulación de datos
    - restricciones de integridad
    - autorizaciones
    - gestión de transacciones

6. Actualización de vistas: toda vista teóricamente actualizable debe poder ser actualizada por el sistema.

7. Inserciones, modificaciones y eliminaciones de alto nivel: todas las operaciones de manipulación de datos (consulta, inserción, modificación y borrado) deben operar sobre conjuntos de filas (lenguaje no navegable).

8. Independencia física de los datos: el acceso lógico a los datos debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento.

9. Independencia lógica de los datos: los programas de aplicación no deben verse afectados por cambios en las tablas que estén permitidos teóricamente y preserven la información.

10. Independencia de la integridad: Las reglas de integridad de una base de datos deben ser definibles por medio del sublenguaje de datos relacional y han de almacenarse en el catálogo de la base de datos (metabase), no en los programas de aplicación.

11. Independencia de la distribución: Debe existir un sublenguaje de datos que pueda soportar bases de datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente.

12. Regla de la no subversión: Si un SGBD soporta un lenguaje de bajo nivel que permite el acceso fila a fila, éste no puede utilizarse para saltarse las reglas de integridad expresadas por medio del lenguaje de más alto nivel.

La idea fundamental de Codd fue el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas.

Pese a esta teoría de las bases de datos relacionales, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es, pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Además, las bases de datos relacionales pasan por un proceso al que se le conoce como normalización, la cual es entendida como el proceso necesario para que una base de datos sea utilizada de manera óptima.

Una relación se encuentra en uno u otro grado de normalización si cumple una serie de propiedades (restricciones).

Las tres primeras formas normales las definió Codd en 1970 y se basan en las dependencias funcionales[1] que existen entre los atributos de la relación.

Más tarde, en 1974 y debido a que todavía existían anomalías, redefinió la tercera forma normal que definió como forma normal de Boyce-Codd (FNBC).

Posteriormente, y basándose en las dependencias multivaluadas y en combinación, se definieron dos niveles más de normalización, la cuarta y quinta forma normal.

De esta forma, las relaciones en primera forma normal tienen más redundancia de datos que los niveles superiores, y por tanto más anomalías de actualización de datos.

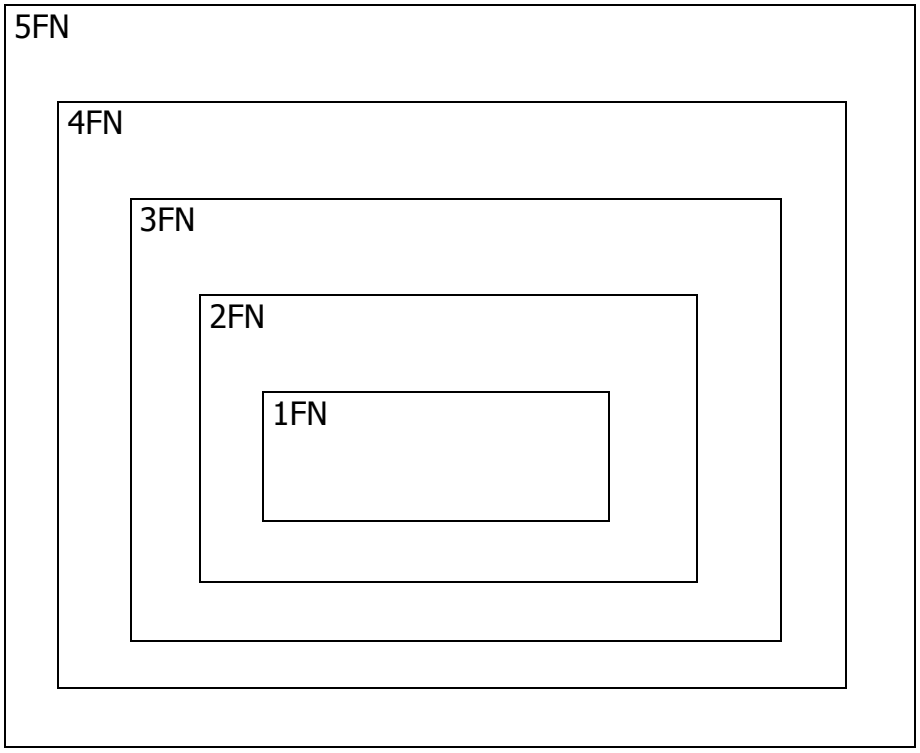


Figura 1. Jerarquía de las formas normales

1. Sean x e y atributos (o conjunto de atributos) de una relación R. Se dice que y depende funcionalmente de x (se denota por $x \longrightarrow y$) si cada valor de x tiene asociado un solo valor de y.

2.1.1 Diseño de una base de datos

En el diseño de una base de datos se debe realizar un modelo de datos que ayude a entender el significado de la misma y que facilite la comunicación en cuanto a los requisitos de información. La primera etapa es el diseño conceptual, en donde se construye un esquema de la información que maneja la empresa, independientemente de todas las consideraciones físicas. Después viene el diseño lógico, en el que el esquema anterior se transforma según el modelo de base de datos que se vaya a utilizar para implementar el sistema. Por último, en la etapa del diseño físico, se produce una descripción de la implementación de la base de datos en memoria secundaria.

El diseño de las aplicaciones, una fase que se debe llevar a cabo en paralelo con el diseño de la base de datos, está compuesta por dos actividades: el diseño de las transacciones y el diseño de las interfaces de usuario de informes y formularios.

2.1.1.1 El modelo conceptual

Un modelo conceptual es un conjunto de conceptos que permiten describir la realidad mediante representaciones lingüísticas y gráficas. Los modelos conceptuales deben poseer una serie de propiedades:

- Expresividad
- Simplicidad
- Minimalidad
- Formalidad

El modelo conceptual más utilizado es el modelo entidad-relación (E/R), que posee los siguientes conceptos:

- Entidades
- Relaciones
- Atributos
- Dominios de atributos
- Identificadores
- Jerarquías de generalización

2.1.1.2 El diseño lógico

Las dos fases de que consta el diseño lógico son la construcción y validación de los esquemas lógicos locales para cada vista de usuario, y la construcción y validación de un esquema lógico global. Cada una de estas fases consta de una serie de pasos.

Un paso importante es la conversión del esquema conceptual a un esquema lógico adecuado al modelo relacional. Para ello, se deben hacer algunas

transformaciones: eliminar las relaciones de muchos a muchos, eliminar las relaciones complejas, eliminar las relaciones recursivas, eliminar las relaciones con atributos, eliminar los atributos multivaluados, reconsiderar las relaciones de uno a uno y eliminar las relaciones redundantes.

Los esquemas lógicos se pueden validar mediante la normalización y frente a las transacciones de los usuarios.

Las restricciones de integridad son las restricciones que se imponen para que la base de datos nunca llegue a un estado inconsistente. Hay cinco tipos de restricciones de integridad: datos requeridos, restricciones de dominio, integridad de entidades, integridad referencial y reglas de negocio.

Para garantizar la integridad referencial se debe especificar el comportamiento de las claves ajenas: si aceptan nulos y qué hacer cuando se borra la tupla a la que se hace referencia, o cuando se modifica el valor de su clave primaria.

2.1.1.3 El diseño físico

El diseño físico es el proceso de producir una descripción de la implementación de la base de datos en memoria secundaria. Describe las relaciones base y las estructuras de almacenamiento y métodos de acceso que se utilizarán para acceder a los datos de modo eficiente. El diseño de las relaciones base sólo se puede realizar cuando el diseñador conoce perfectamente toda la funcionalidad que presenta el SGBD que se vaya a utilizar.

El primer paso consiste en traducir el esquema lógico global de modo que pueda ser fácilmente implementado por el SGBD específico. A continuación, se escogen las organizaciones de ficheros más apropiadas para almacenar las relaciones base, y los métodos de acceso, basándose en el análisis de las transacciones que se van a ejecutar sobre la base de datos.

Se puede considerar la introducción de redundancias controladas para mejorar las prestaciones. Otra tarea a realizar en este paso es estimar el espacio en disco.

La seguridad de la base de datos es fundamental, por lo que el siguiente paso consiste en diseñar las medidas de seguridad necesarias mediante la creación de vistas y el establecimiento de permisos para los usuarios.

El último paso del diseño físico consiste en monitorizar y afinar el sistema para obtener las mejores prestaciones y satisfacer los cambios que se puedan producir en los requisitos.

2.2. Bases de datos espaciales

2.2.1 Introducción

Una Base de Datos Espacial (Spatial Database) es un sistema administrador de bases de datos que maneja datos existentes en un espacio o datos espaciales. El espacio establece un marco de referencia para definir la localización y relación entre objetos. El que normalmente se utiliza es el espacio físico que es un dominio manipulable, perceptible y que sirve de referencia. La construcción de una base de datos geográfica implica un proceso de abstracción para pasar de la complejidad del mundo real a una representación simplificada que pueda ser procesada por el lenguaje de las computadoras actuales. Este proceso de abstracción tiene diversos niveles y normalmente comienza con la concepción de la estructura de la base de datos, generalmente en capas; en esta fase, y dependiendo de la utilidad que se vaya a dar a la información a compilar, se seleccionan las capas temáticas a incluir.

La estructuración de la información espacial procedente del mundo real en capas conlleva cierto nivel de dificultad. En primer lugar, la necesidad de abstracción que requieren los computadores implica trabajar con primitivas básicas de dibujo, de tal forma que toda la complejidad de la realidad ha de ser reducida a puntos, líneas o polígonos. En segundo lugar, existen relaciones espaciales entre los objetos geográficos que el sistema no puede obviar; la topología, que en realidad es el método matemático-lógico usado para definir

las relaciones espaciales entre los objetos geográficos puede llegar a ser muy compleja, ya que son muchos los elementos que interactúan sobre cada aspecto de la realidad.

Un modelo de datos geográfico es una abstracción del mundo real que emplea un conjunto de objetos dato, para soportar el despliegue de mapas, consultas, edición y análisis. Los datos geográficos, presentan la información en representaciones subjetivas a través de mapas y símbolos, que representan la geografía como formas geométricas, redes, superficies, ubicaciones e imágenes, a los cuales se les asignan sus respectivos atributos que los definen y describen.

2.2.2 Tipos de datos espaciales

Un dato espacial es una variable asociada a una localización del espacio. Normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales.

Los datos espaciales se dividen en:

- Puntos: Un punto representa un objeto del cual sólo interesa conocer su localización en el espacio. Su forma de representación más habitual es mediante dos coordenadas (x, y) en el espacio. Se encuentran determinados por las coordenadas terrestres medidas por latitud y longitud. Por ejemplo, ciudades, accidentes geográficos puntuales, hitos.

- Líneas: Una línea es comprendida como una secuencia de puntos y por tanto como una secuencia de coordenadas (x, y), normalmente representa conexiones en el espacio. Debido a la forma esférica de la tierra también se las consideran como arcos. Líneas telefónicas, carreteras, ríos y vías de trenes son ejemplos de líneas geográficas.

- Polígonos o Regiones: Figuras planas conectadas por distintas líneas u objetos cerrados que cubren un área determinada, como por ejemplo

países, regiones o lagos. En la definición de las regiones, puede contemplarse la posibilidad de tener agujeros en su interior, lo cual se conoce como región cóncava.

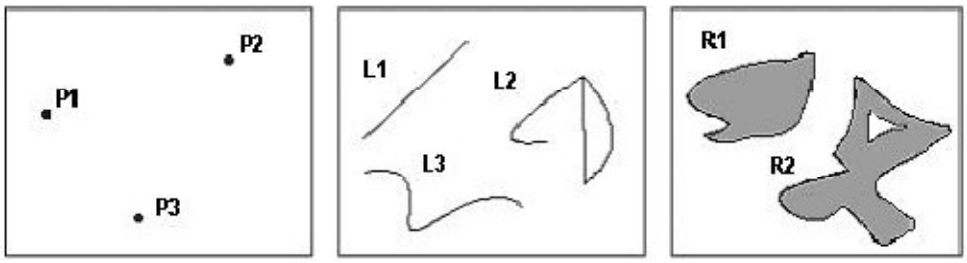


Figura 2: Entidades geométricas: Puntos, Líneas y Regiones

De esta forma la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas (x, y). La ubicación de una característica puntual, pueden describirse con un sólo punto (x, y). Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas (x, y). Las características poligonales, pueden almacenarse como un circuito cerrado de coordenadas. La otra forma de expresar datos espaciales es mediante rasterización, la cual, a través de una malla permite asociar datos a una imagen; es decir, se pueden relacionar paquetes de información a los píxeles de una imagen digitalizada.

Los datos espaciales además se caracterizan por su naturaleza georreferenciada y multidireccional. La primera se refiere a que la posición relativa o absoluta de cualquier elemento sobre el espacio contiene información valiosa, pues la localización debe considerarse explícitamente en cualquier análisis. Por multidireccional se entiende a que existen relaciones complejas no lineales, es decir que un elemento cualquiera se relaciona con su vecino y además con regiones lejanas, por lo que la relación entre todos los elementos no es unidireccional. Es decir, todos los elementos se relacionan entre sí, pero existe una relación más profunda entre los elementos más cercanos.

Los datos espaciales poseen algunas características que hacen inapropiada la utilización de los Sistemas Manejadores de Bases de Datos (DBMS)

tradicionales, motivando la aparición de las bases de datos espaciales con objeto de permitir su gestión de una forma eficiente. Algunas de esas características son:

- La complejidad de las estructuras de datos necesarias para su almacenamiento, como es el caso de un polígono, hace inviable la posibilidad de emplear los tipos de datos tradicionales, dado que se precisa registrar un conjunto de puntos que lo delimiten, puntos que pueden ser multidimensionales.

  Esto hace las DBMS tradicionales, como pueden ser las relacionales con un tamaño de tupla fijo, inapropiadas en este dominio. Además la necesidad de contemplar toda una serie de operaciones, como la intersección o la unión que posteriormente necesitan una implementación eficiente en la base de datos, hace inevitable la extensión o definición de un modelo que les dé soporte.

- La necesidad de ofrecer estructuras de datos que permitan tanto la inserción, como el borrado y actualización de una forma robusta y con un rendimiento elevado, sin que el coste de procesamiento sea demasiado elevado.

- El elevado volumen de datos con el que se suele trabajar en aplicaciones que tratan con información geográfica, pone de manifiesto la necesidad de emplear sistemas de almacenamiento secundario que permitan su almacenamiento y recuperación de forma eficiente.

- La carencia de un álgebra espacial estándar, implica la imposibilidad de utilización de un conjunto de operaciones cuya semántica esté claramente definida. En su lugar, se encuentran diversas propuestas en este campo que en muchas ocasiones dependen demasiado del dominio de aplicación.

La dimensionalidad de los datos espaciales supone a su vez otro problema, debido a la inexistencia de un orden total que permita aplicar los predicados existentes en las bases de datos tradicionales. Este orden se refiere a la imposibilidad de, por ejemplo, realizar una consulta según los objetos espaciales que sean mayores que uno dado. No existe una definición del concepto mayor sobre los datos espaciales.

El problema de cierre de las operaciones espaciales aparece con demasiada frecuencia, debido a los diferentes tipos sobre los que se pueden aplicar. Tomando como ejemplo el caso de la intersección entre dos objetos espaciales, el resultado podría ser tanto un polígono, como una línea o un punto, o una serie de estos objetos. En muchas aproximaciones se desprecia parte del conjunto de posibles soluciones debido a deficiencias en el modelo en el que se basa.

Independientemente del modelo de DBMS que se emplee, sea orientado a objetos, relacional u objeto-relacional, se trata de realizar una extensión de su arquitectura de forma que soporte la representación de estos objetos geométricos. Esta extensión significa, en primer lugar, la modificación del modelo de datos para permitir la inclusión de nuevos tipos abstractos de datos espaciales, tales como puntos, líneas o regiones, que permitan recoger la geometría de las entidades que queremos almacenar y, en segundo lugar, el conjunto de operaciones que definen su comportamiento.

Junto a la inclusión de estos tipos de datos es imprescindible la extensión de los lenguajes tanto de consulta como de definición, que permitan manipular datos espaciales, así como un conjunto de técnicas de indexación adaptadas al tipo de datos con el que se esté trabajando, que permitan obtener un alto rendimiento del sistema.

2.2.3 Modelos de datos espaciales

El propósito de cualquier modelo de datos, es proporcionar un medio formal con el que representar un determinado tipo de información, así como un conjunto de operaciones que permitan su manipulación.

Esa representación trata de abstraer un determinado aspecto de la realidad. En este caso, el espacio n-dimensional que es por naturaleza infinito y más concretamente, el espacio Euclídeo en la mayoría de las aproximaciones.

Desafortunadamente, los modelos de datos que se presentarán seguidamente, dan una representación finita de esos infinitos e incluso innumerables conjuntos de puntos que componen los objetos espaciales registrados en las bases de datos, llevando en ocasiones a una pérdida inevitable de información.

Un modelo de datos es considerado generalmente como una abstracción, que incorpora solamente aquellas propiedades pensadas como relevantes a la aplicación o aplicaciones con las que se está tratando.

Usualmente una conceptualización humana de la realidad sin connotaciones tecnológicas en su mayor nivel de abstracción, es decir, sin considerar donde será implementado. Pero, la implementación del mismo conlleva generalmente a tomar decisiones en las que prima el rendimiento frente a la precisión en la representación. Esta decisión impacta directamente en las características de almacenamiento, manipulación y recuperación de las estructuras tanto física como de los datos en sí mismos.

Se hace necesario examinar esos temas utilizando un conjunto de criterios basados en el uso, de forma que la calidad total o la idoneidad de un modelo de datos específico pueda ser evaluada dentro de un contexto particular.

Los criterios generales son:

- Completitud, que puede plantearse en términos de la proporción de todas las entidades y relaciones de un fenómeno particular, existentes en la realidad y que son representadas en el modelo.

- Robustez, es el grado con que el modelo de datos puede acomodarse a circunstancias especiales o poco usuales, tales como un polígono con un agujero en su interior.

- Versatilidad, permitiendo contemplar situaciones no recogidas inicialmente en el modelo.

- Eficiencia, incluye tanto la compactación, es decir, eficiencia de almacenamiento, como la velocidad de uso entendida como eficiencia temporal.

- Facilidad de generación, es la cantidad de esfuerzo necesario para convertir datos de un formato a algún otro requerido por el modelo de datos.

El grado de variación de cada uno de estos factores ha de tomarse en consideración para cualquier campo de aplicación. La importancia relativa de cada factor es una función de los tipos de datos particulares a ser empleados y de los requisitos operacionales totales del sistema. Por ejemplo, si la base de datos a ser generada tiene un volumen elevado y tiene que ofrecer un rendimiento en un contexto interactivo, sería adecuado un compromiso entre los tres primeros factores, dado que la eficiencia total y la facilidad de generación predominarían.

Es posible medir cuantitativamente el rendimiento de varios de estos factores como la velocidad y la eficiencia en espacio para un modelo de datos particular.

No sin embargo, proporcionar medidas cuantitativas para los factores más abstractos como completitud, robustez o versatilidad.

Este hecho combinado con el escaso conocimiento sobre las características de rendimiento de un amplio rango de algoritmos de procesamiento espacial así como su interacción con otros algoritmos y modelos de datos, indica que el proceso de modelado de datos es mucho más un arte que una ciencia. Para ello, la experiencia y la intuición se mantendrán como factores clave en la interpretación de las especificaciones de requisitos de sistemas poco definidos y la construcción de modelos de datos satisfactorios, particularmente para sistemas de información geográfica integrados y completos.

En la definición de estos modelos de datos encontramos dos alternativas claramente diferenciadas.

La primera de ellas realiza una definición abstracta de los tipos de datos de forma que permita su posterior integración en el DBMS con independencia de si éste es relacional, objetual o de cualquier otro tipo. En cambio, la segunda alternativa hace uso del modelo utilizado por el sistema gestor que pretende extender para realizar su definición.

Existe un punto en común a todas las alternativas que es la semántica asociada a los tipos de datos que se pretenden incorporar.

2.2.3.1 Modelo de datos raster

En este tipo de modelo la información geométrica es intencionalmente descrita por un número finito de puntos raster. La semántica en este modelo es que el número infinito de puntos en el entorno de un punto raster tienen sus mismas propiedades.

Los puntos raster están uniformemente distribuidos sobre el objeto que está siendo representado. Aunque el modelo es bastante intuitivo, presenta algunos problemas relacionados con la distribución, en ocasiones heterogénea, de las propiedades relevantes, por ejemplo una recta que no se ajusta a los puntos

que conforman el raster, dando lugar a problemas con el cierre de las operaciones.

Una de las características de este modelo es la extensibilidad de la arquitectura del sistema, dada la posibilidad de poder definir nuevos tipos de acuerdo a las necesidades del dominio concreto donde se vayan a aplicar.

Incluye también diferentes operaciones que permiten la conversión entre diferentes tipos, así como completitud en cuanto al grado de objetos espaciales que son representables mediante este modelo. Así mismo, es eficiente tanto en términos de utilización de almacenamiento, debido a la escasa duplicación de información, como temporal, gracias al cálculo de las intersecciones en el momento de inserción de los objetos espaciales.

El modelo de datos raster se utiliza en los siguientes casos:

- Rápida y correcta superposición de mapas y, en general, para el análisis espacial.
- Modelado y análisis de superficies.

2.2.3.2 Modelo de datos Polinomial

El modelo de datos con restricciones introducido por Kanellalis en 1995 trata de mostrar un paradigma para la representación de todos los tipos de datos desde un marco unificado. Las restricciones lineales sobre los números racionales han demostrado, mediante la representación modo vector, su idoneidad para la representación de datos espaciales.

Este formato de representación se enfrenta con una representación en la que se emplean los bordes de un objeto para representar los infinitos puntos que lo componen, y que conlleva la inexistencia de un estándar que permita aportar una solución única a este problema.

La idea básica de este modelo es representar los objetos espaciales como colecciones infinitas de puntos en el espacio de los racionales (Q2) que

satisfacen una formula de primer orden. Por ejemplo, un polígono convexo, que es la intersección de un conjunto de medios planos, es definido mediante la conjunción de las desigualdades que definen cada uno de ellos. Un polígono no convexo es definido mediante la unión de un conjunto de polígonos convexos.

Aunque no restringe el tipo de objetos que pueden ser almacenados en la base de datos, sí en cambio obliga a una descomposición del espacio en componentes convexos en el momento de la carga de la información. Aunque esto conlleva un mayor coste de procesamiento inicial, simplifica enormemente cualquier consulta posterior. Este tipo de representación se conoce como DNF (Disjunctive Normal Form, Figura 3). En cambio, ese procesamiento puede ser evitado si se permite la definición de polígonos con agujeros, es decir, la representación conocida como CHNF (Convex with Holes Normal Form, Figura 3).

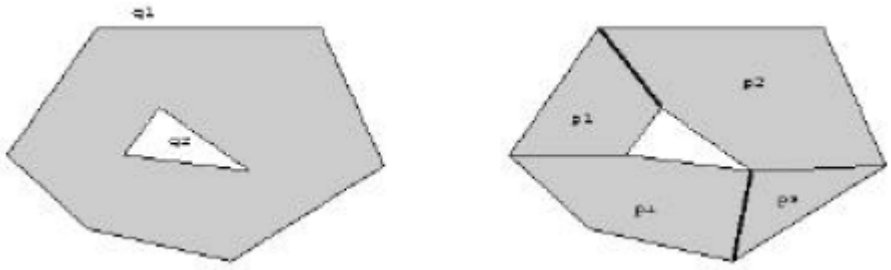


Figura 3: Representación DNF y CHNF de un polígono con agujeros

2.2.3.3 Modelo de datos Topológico

El interés principal de este tipo de modelo es ofrecer una base forma para aquellas consultas en la base de datos con referencia a las relaciones topológicas de los objetos, tales como adyacencia o solapamiento.

Una característica inherente de las bases de datos que soportan este tipo de relaciones es la equivalencia entre las bases de datos que son generadas mediante deformaciones topológicas, es decir, son equivalentes topológicamente.

Este tipo de modelo se aplica generalmente en dominios enfocados a modelar relaciones entre los objetos. Un caso de un dominio de aplicación es un sistema de control de tráfico donde, una de sus necesidades básicas es registrar los enlaces entre diferentes carreteras. Estas relaciones permiten, por ejemplo, determinar las posibles rutas alternativas.

2.3 Bases de datos temporales

Tradicionalmente, las bases de datos tratan de representar una visión de la información referente al mundo real. Dicha información tiene una validez determinada en un instante determinado, por lo que, si esta realidad se ve alterada en algún momento, el sistema gestor (DBMS) ha de reflejar este cambio mediante actualizaciones o borrados en la base de datos, según sea la naturaleza del cambio.

Esta manera de tratar las modificaciones conlleva que el estado previo de la información se pierda o se desprecie. Este tipo de bases de datos se conoce como base de datos snapshot, debido a que ofrecen una fotografía de la realidad en un momento concreto.

La motivación de gestionar así los cambios en la información proviene de la capacidad, siempre limitada, de almacenamiento o de carencias en el modelo de datos que permitan capturar esa evolución de la información.

La aparición de mejoras en la tecnología ha permitido utilizar mayores y más rápidos sistemas para almacenamiento y procesamiento de dicha información.

A estas mejoras en la tecnología se une la aparición de las bases de datos temporales, que se caracterizan, esencialmente, por incorporar la representación de aspectos temporales en el modelo de datos, con una semántica especial (que veremos en el siguiente apartado) junto con las facilidades en el sistema gestor necesarias para su manipulación.

Las bases de datos temporales han sido un área de investigación activa durante las últimas décadas, como podemos apreciar por la numerosa bibliografía que trata sobre ellas, tanto en temas relativos a los modelos de datos, lenguajes, indexación o implementación.

En el marco del modelo relacional, la aparición de TSQL2 es un reflejo del trabajo que se ha estado realizando por aunar esfuerzos y que trata de integrar las diferentes alternativas propuestas. Éste consiste en una extensión consistente del lenguaje estándar SQL-92, que permite introducir esa dimensión temporal, desarrollado con la intención de ser incluido en el estándar SQL3.

Los tipos de datos temporales, sus operaciones asociadas y las diferentes aproximaciones que existen para llevar a cabo su integración en el sistema gestor, han supuesto la aparición de diferentes conceptos que han de ser estudiados.

2.3.1 Conceptos básicos

Según *Date*, una base de datos temporal es aquella que contiene datos históricos en lugar de, o así como, datos actuales, es decir, situaciones en las que es necesario reflejar la validez temporal de la información, durante qué intervalo o instante de tiempo se consideró o fue considerada como válida.

Los datos tienen una referencia temporal, introducida mediante una fecha, por ejemplo "1 de Enero de 1999", que recibe, en la literatura inglesa, el nombre de timestamp, que marca en ambos casos un instante temporal. El timestamp nos permite incluir una marca temporal sobre un momento de validez de la información.

Podemos introducir dos conceptos claves en el campo de las bases de datos temporales: tiempo de validez (valid time) y tiempo de transacción (transaction time). El primero de ellos expresa el tiempo durante el cual una cierta proposición es cierta, mientras que el tiempo de transacción indica el tiempo en

que una proposición aparece reflejada en la base de datos como cierta, el momento en que incorporamos esa información en la base de datos.

El tiempo de validez o (valid time) puede ser actualizado por el usuario para reflejar un cambio en la proposición, en cambio, el segundo puede ser únicamente modificado por el sistema gestor, cualquier cambio que un usuario realice sobre la información quedaría reflejado mediante un nuevo tiempo de transacción que marcaría el momento en que hizo la modificación.

El tiempo de validez y de transacción no debe confundirse con el tiempo de usuario (user time). Éste representa el empleo, por parte del usuario de un tipo de datos estándar (date) de un modo similar al empleo de los enteros o los reales, para representar alguna información temporal.

Por ejemplo, en una base de datos de personal, al introducir una información como la fecha de alta de un usuario o su fecha de nacimiento se está reflejando una información temporal, no su período o instante de validez.

En este caso, no se ofrece un soporte especial del lenguaje de consulta como en el caso del tiempo de validez o de transacción.

De igual forma que el tiempo de transacción y de validez son aplicables sobre una información almacenada en la base de datos, también es posible aplicar estos tiempos a una relación. Es decir, permitir que pueda reflejarse cuando una relación ha sido incluida en el esquema o cuando se han efectuado modificaciones sobre ella en el primer caso, o delimitar, en el caso de tiempo de validez, en que instante o durante qué intervalo de tiempo la relación ha sido válida.

Las DBMS comerciales almacenan sólo un estado del mundo real, usualmente el estado más reciente. Aquellas bases de datos usualmente son llamadas "snapshot databases" cuyo contexto de tiempo valido y tiempo de transacción es mostrado en la siguiente figura:

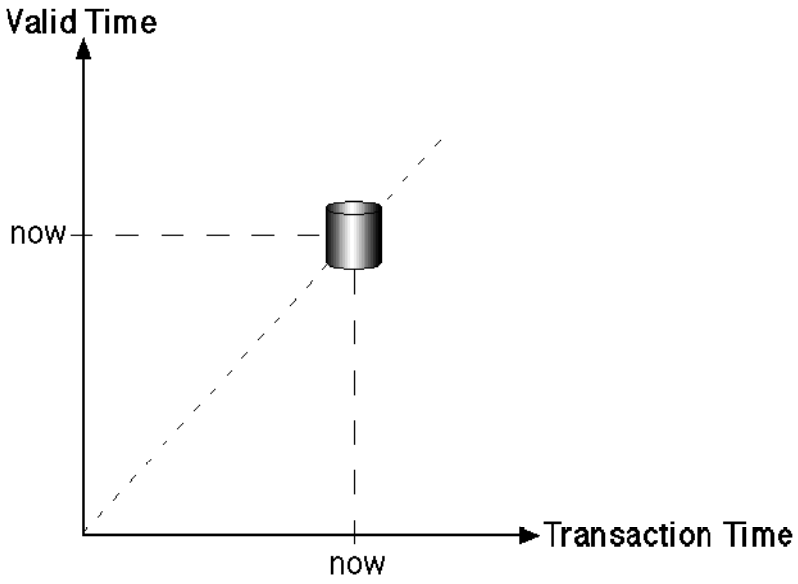


Figura 4: Snapshot Database

Snodgrass clasifica las bases de datos en estáticas, históricas, rollback o bitemporales, según como se contemplen, o no, los conceptos anteriores dentro del esquema. Las bases de datos históricas no contemplan ni el tiempo de validez ni de transacción, no almacenan ninguna de estas referencias temporales en el modelo que emplean.

Las bases de datos históricas contemplan únicamente el tiempo de validez, almacenan la información que conocemos como válida, tanto presente como pasada o futura, pero los cambios producidos en la información (como ha sido la evolución de las actualizaciones que se han realizado) no quedan almacenados.

Las bases de datos rollback, en cambio, contienen exclusivamente el tiempo de transacción, nos permiten devolver la base de datos a un estado anterior en caso de que sea necesario, pero no permiten conocer durante qué intervalo de tiempo ha permanecido como válida una determinada información.

Por último, las bitemporales soportan tanto el tiempo de transacción como el de validez, por lo que nos dan una visión más precisa de la evolución que ha sufrido la información tanto en sus intervalos de validez como de sus

actualizaciones, al contemplar los diferentes estados por los que pasó la información, y permitir su recuperación.

Los estados almacenados en una Base de Datos bitemporal están representados en la siguiente figura:

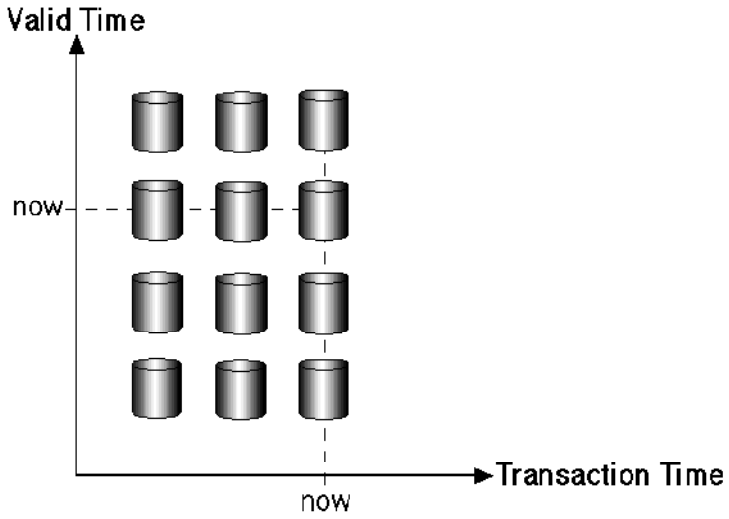


Figura 5: Base de Datos Bitemporal

Otro concepto, que afecta especialmente a la semántica de las operaciones y que ha de ser incluido es la definición de Crono o quantum temporal. Éste hace referencia a la duración de tiempo más corta, al instante temporal más breve, soportado por el DBMS. Un quantum determina un subintervalo de tiempo, con una duración fija a lo largo de la línea temporal.

2.4 Los sistemas de información geográfica

2.4.1 Introducción

El uso de los GIS ha aumentado enormemente en las décadas de los ochenta y noventa; ha pasado del total desconocimiento a la práctica cotidiana en el mundo de los negocios, en las universidades y en los organismos gubernamentales, usándose para resolver problemas diversos. Es lógico, por tanto, que hayan sido propuestas varias definiciones. Una definición precisa y completa podría ser:

"Un conjunto de equipos informáticos, de programas, de datos geográficos y técnicos organizados para recoger, almacenar, actualizar, manipular, analizar y presentar eficientemente todas las formas de información georreferenciada".

Otras definiciones de GIS:

"Un sistema para capturar, almacenar, comprobar, integrar, manipular, analizar y visualizar datos que están espacialmente referenciados a la tierra".

"Sistemas automatizados para la captura, almacenamiento, composición, análisis y visualización de datos espaciales".

"Un sistema de hardware, software y procedimientos diseñados para soportar la captura, gestión, manipulación, análisis, modelado y visualización de datos espacialmente referenciados para resolver problemas complejos de planeamiento y gestión".

Desde un punto de vista práctico, un Sistema de Información Geográfica es un sistema informático capaz de realizar una gestión completa de datos geográficos referenciados. Por referenciados se entiende que estos datos geográficos o mapas tienen unas coordenadas geográficas reales asociadas, las cuales nos permiten manejar y hacer análisis con datos reales como longitudes, perímetros o áreas.

Todos estos datos alfanuméricos asociados a los mapas, más los que queramos añadir, los gestiona una base de datos integrada con el GIS.

Entre sus múltiples utilidades, se encuentran tales como investigaciones científicas, la gestión de los recursos, gestión de activos, la arqueología, la evaluación del impacto ambiental, la planificación urbana, la cartografía, la sociología, la geografía histórica, el marketing, la logística, etc.

2.4.2 Historia y evolución de los GIS

En el siglo XIX, caracterizado por su avance tecnológico basado en el conocimiento científico de la tierra, se produjeron grandes volúmenes de información geomorfológica que se debía cartografiar. La orientación espacial de la información se conservó con la superposición de mapas temáticos especializados sobre un mapa topográfico base.

Recientemente la fotografía aérea y particularmente las imágenes de satélite han permitido la observación periódica de los fenómenos sobre la superficie de la corteza terrestre. La información producida por este tipo de sensores ha exigido el desarrollo de herramientas para lograr una representación cartográfica de este tipo de información.

El medio en el cual se desarrollaron estas herramientas tecnológicas correspondió a las ciencias de teledetección, análisis de imágenes, reconocimiento de patrones y procesamiento digital de información, en general estudiadas por físicos, matemáticos y científicos expertos en procesamiento espacial. Obviamente, éstos tenían un concepto diferente al de los cartógrafos, con respecto a la representación visual de la información.

Con el transcurso del tiempo se ha logrado desarrollar un trabajo multidisciplinar, y por esta razón ha sido posible pensar en utilizar la herramienta conocida como "Sistemas de Información Geográfica, SIG (GIS)"

En el año 1962, en Canadá, se diseñó el primer sistema "formal" de información geográfica a escala mundial de recursos naturales. En el Reino Unido se empezó a trabajar en la unidad de cartografía experimental.

Durante los años 60 y 70 se empezó a aplicar la tecnología del computador digital al desarrollo de tecnología automatizada. No fue hasta la época de los 80 cuando surgió la comercialización de los SIG. Excluyendo cambios estructurales en el manejo de la información, la mayoría de programas estuvieron dirigidos hacia la automatización del trabajo cartográfico; unos pocos exploraron nuevos métodos para el manejo de información espacial, y se siguieron básicamente dos tendencias:

- Producción automática de dibujos con un alto nivel de calidad pictórica.
- Producción de información basada en el análisis espacial pero con el costo de una baja calidad gráfica.

La producción automática de dibujos se basó en la tecnología de diseño asistido por computador (CAD). El CAD se utilizó en la cartografía para aumentar la productividad en la generación y actualización de mapas. El modelo de base de datos de CAD maneja la información espacial como dibujos electrónicos compuestos por entidades gráficas organizadas en planos de visualización o capas.

Cada capa contiene la información de los puntos en la pantalla (o píxeles) que debe encender para la representación por pantalla. Estos conjuntos de puntos organizados por planos de visualización se guardan en un formato vectorial.

Las bases de datos utilizadas en esta disciplina incluyen funciones gráficas primitivas que se emplean para construir nuevos conjuntos de puntos o líneas en nuevas capas y definir un símbolo imaginado por el usuario. Por ejemplo una capa que contenga una línea vertical se puede sumar lógicamente a una capa que contenga un área circular para generar el símbolo de un palo de golf o una nota musical, definido una nueva capa.

Posteriormente, a la simbología se le añadió una variable "inteligente" al incorporar el texto.

El desarrollo de la tecnología CAD se aplicó para la manipulación de mapas y dibujos y para la optimización del manejo gerencial de información cartográfica. De ahí surgió la tecnología AM/FM (Automated Mapping / Facilities Management)

El desarrollo paralelo de las disciplinas que incluyen la captura, el análisis y la presentación de datos en un contexto de áreas afines como catastro, cartografía, topografía, ingeniería civil, geografía, planeación urbana y rural, servicios públicos, entre otros, ha implicado duplicidad de esfuerzos. Hoy en día, se ha logrado reunir el trabajo en el área de sistemas de información geográfica multipropósito, en la medida en que se superan los problemas técnicos y conceptuales inherentes al proceso.

En los años ochenta se hizo latente la expansión del uso de los SIG, facilitado por la comercialización simultánea de un gran número de herramientas de dibujo y diseño asistido por ordenador (en ingles CAD), así como la generalización del uso de microordenadores y estaciones de trabajo en la industria y la aparición y consolidación de las Bases de Datos relacionales (SGBD), junto a las primeras modelizaciones de las relaciones espaciales o topología.

En este sentido la aparición de productos como ArcInfo en el ámbito del SIG, o IGDS[1] en el ámbito del CAD fue determinante para lanzar un nuevo mercado con una rapidísima expansión.

La aparición de la Orientación a Objetos (OO) en los SIG como el Tigris de Intergraph, inicialmente aplicado en el ámbito militar (Defense Map Agency – DMA) permite nuevas concepciones de los SIG donde se integra todo lo referido a cada entidad (p. e. una parcela) simbología, geometría, topología, atribución.

[1] Interactive Graphics Design System - sistema de diseño gráfico interactivo.

Pronto los SIG se comienzan a utilizar en cualquier disciplina que necesite la combinación de planos cartográficos y bases de datos como diseño de carreteras, presas y embalses, estudios medioambientales, estudios socioeconómicos y demográficos, planificación de líneas de comunicación, ordenación del territorio, estudios geológicos y geofísicos. Prospección y explotación de minas, entre otros.

Los años noventa se caracterizan por la madurez en el uso de estas tecnologías en los ámbitos tradicionales mencionados y por su expansión a nuevos campos (como el de los negocios) todo esto propiciado por el uso de los ordenadores de gran potencia y sin embargo muy asequibles.

La enorme expansión de las comunicaciones da lugar a la aparición de los sistemas distribuidos con tecnologías como: Distributed COM (DCOM) o Common Object Request Broker Architecture (CORBA), y la fuerte tendencia a la unificación de formatos de intercambio de datos geográficos, la aparición de una oferta proveedora (Open Gis) que suministra datos a un enorme mercado de usuario final.

El incremento de la popularidad de las tendencias de programación distribuida y la expansión y beneficios de la máquina virtual de Java (JVM), permiten la creación de nuevas formas de programación de sistemas distribuidos, de esta manera aparecen los agentes móviles que tratan de solucionar el tráfico excesivo que hoy en día se encuentra en Internet.

Los agentes móviles utilizan la invocación de métodos remotos y la serialización de objetos de Java para lograr transportar la computación y los datos. Naciendo así un nuevo paradigma para el acceso a consultas y recopilación de datos en los sistemas de información geográfica, cuyos mayores beneficios se esperan obtener en los siguientes años.

El Mapa del Futuro es una Imagen Inteligente. A partir de 1998 se empezaron a colocar en distintas órbitas una serie de familias de satélites que nos traen a los

computadores personales fotografías digitales de la superficie de la tierra con resoluciones que oscilan entre 10 metros y 50 centímetros.

2.4.3 Objetivos de un GIS

Los GIS son sistemas computarizados diseñados para soportar la captura, procesamiento y recuperación de datos referenciados espacialmente (en un ámbito geográfico) a fin de resolver problemas de planeamiento y administración.

Un GIS debe tratar de responder: al qué, quién, cuándo, dónde, por qué y cómo, por lo que se pueden generalizar en estas cinco preguntas:

a. Localización: ¿Qué hay en este lugar?

Identificar que es lo que se encuentra en una determinada localización que puede describirse por su topónimo, por un código clasificado, o por referencias geográficas como latitud, longitud y altura. Esta información puede ser digitalizada o referenciada gráficamente (Por ejemplo indicar en un plano la ubicación de una empresa de manera que se pueda tener en pantalla su información registrada).

b. Condición: ¿Dónde se encuentra?

Se busca un determinado lugar que reúna ciertas condiciones y que requiera de un análisis espacial de búsqueda (por ejemplo, tener seleccionados en pantalla todos los locales industriales registrados que tengan un área mayor de 2000 metros cuadrados).

c. Tendencia: ¿Qué ha cambiado desde?

Permite conocer la variación de algunas características a través de un determinado periodo (por ejemplo, en el transcurso de un año, cuántos y cuáles son los nuevos locales industriales que se han registrado).

d. Distribución: ¿Qué patrones de distribución espacial existen?

Busca determinar en una zona específica, las relaciones que pudieran existir entre dos o más variables. (por ejemplo, en un área definida cual es la cantidad de empresas que producen gases tóxicos).

e. Modelización: ¿Qué sucede si…?

Si a un sistema planteado se le somete a determinadas modificaciones de sus variables, debemos conocer cómo queda definido el nuevo sistema, cuánto ha cambiado, etc. (por ejemplo, en una determinada zona industrial se desea incrementar el área verde en un 30 % de manera que tengan cierta distribución uniforme, cuáles son los precios que podrían ser considerados en la modificación, etc.).

2.4.4 Componentes de un GIS

1. Equipos (Hardware)

Los programas de GIS se pueden ejecutar en servidores y también en computadores personales, ya sea en red o en modo independiente; es recomendable que el equipo tenga una buena velocidad de procesamiento y capacidad de almacenaje.

2. Programas (Software)

Los programas de GIS cuentan con las funciones y herramientas necesarias para el tratamiento de la información desde la entrada de datos, almacenar, manipular, analizar, procesar y desplegar la información geográfica. Estos programas deben contener interfaces gráficas avanzadas, así como un sistema óptimo que maneje las bases de datos.

3. Base de datos

Es la parte más importante de un GIS y de ella van a depender los resultados que se puedan obtener.

Hoy en día, las empresas e instituciones públicas se están especializando en la recolección de información, ya que supone un activo importante dentro de la empresa. En un GIS, la información se divide en dos tipos básicos:

- **Atributos gráficos**

  Son representaciones de los objetos en un mapa asociados con ubicaciones definidas. La representación de los objetos se hace por medio de puntos, líneas, iconos o áreas.

- **Atributos no gráficos**

  Son datos alfanuméricos, que pueden ser tanto cualitativos como cuantitativos y corresponden a las características que definen a los elementos que intervienen en el sistema.

4. Recurso humano

El recurso humano necesario se puede dividir en dos tipos: uno en el manejo del software y otro en el tratamiento de la información (límites y alcances de un proyecto, manejo de los datos).

5. Procedimientos

Para trabajar con un GIS es necesario contar con una estructura organizada que permita concebir un plan bien diseñado.

2.4.5 Tipos de GIS

Según el ámbito de aplicación se pueden distinguir tres tipos de programas:

1. GIS general

Es un gran sistema informático que gestiona completamente una base de datos geográfica. Es usado en diferentes áreas de una institución, generalmente todas ellas interconectadas vía red.

Las principales funciones son:

- La construcción de datos geográficos asociados con bases de datos alfanuméricas.
- El análisis de los mapas estructurados en combinación con bases de datos asociados.
- Elaboración de aplicaciones a medida.

Las limitaciones son generalmente los costos y el tiempo en la recolección de la información así como la elaboración del software a medida, su implantación es gradual.

Para su desarrollo es necesaria la concurrencia de programadores y profesionales relacionados con el tema en cuestión.

2. Desktop Mapping (DM)

Llamadas también Cartografía de Escritorio, son aplicaciones sencillas de costo moderado que permite la visualización y análisis de datos con componente espacial para sistemas de microordenadores, viene a ser herramientas de explotación de datos que generan un GIS.

Es usado en áreas de marketing, ventas, distribución y reparto, telecomunicaciones, propiedad inmobiliaria, etc.

Un DM es una herramienta que se integra en un escritorio informático, de igual manera que procesadores de texto, hojas de cálculo, etc. Su propósito es el de permitir el análisis y visualización de bases de datos que contienen información espacial.

No es apto para la creación de nuevos mapas ni poder crear nuevos temas combinando datos existentes para análisis de superposición. No se cuenta con funciones avanzadas de manipulación de la topología ni de modelado cartográfico.

3. Sistemas afines al CAD.

Algunos programas relacionados con los sistemas asistidos por computador (CAD), han evolucionado y se han convertido en cierta manera en programas GIS aplicados también denominados CAD Mapping System (CMS), normalmente estos sistemas son los resultados de enlazar un sistema CAD y un RDBMS (relacional DataBase Management System - Sistema de Gestión de base de datos relacional) de manera que se puedan incorporar variables de localización.

2.4.6 Aplicaciones de los GIS

Los sistemas GIS se pueden usar en:

- Análisis catastrales.
- Desarrollo de sistemas ecológicos y ambientales.
- Levantamientos topográficos, mineros, agrícolas, etc.
- Planeamiento forestal.
- Tratamiento de información para censos.
- Planificación urbana y regional.
- Desarrollo de bases cartográficas.
- Análisis económico y social.
- Redes de distribución y transporte,
- Estudio y análisis de las áreas de ventas y marketing de las empresas.
- Explotación de recursos naturales.
- Planificación de negocios.

2.4.7 Software GIS

- ARC/INFO (ESRI) http://www.esri.com/software/arcgis/arcinfo/

  Es un paquete abierto y programable de programas orientados a la captura, análisis, consulta y representación de datos espaciales, cuenta con aplicaciones específicas como ARC NETWORK (modela y analiza redes espaciales, como rutas vehiculares, planeamiento urbano, estudios de mercado, etc.), ARC GRID (modelar y geoprocesar información ráster), ARC TIN (superficies 3D representativas de terrenos) ARC VIEW (análisis de mercado).

- GEOMEDIA http://www.solgrafperu.com/geomedia.htm

  Es el nombre de una tecnología GIS de nueva generación que Intergraph creó en el año 1996. Los criterios básicos que se tuvieron en cuenta desde su génesis fueron:

  - Funcionamiento en ambiente Windows.
  - Acceso en tiempo real y sin necesidad de conversión a todos los formatos GIS y CAD del mercado.
  - Almacenamiento de información gráfica y alfanumérica en Bases de Datos.
  - Programación con lenguajes estándar de mercado.
  - Evolución hacia plataforma Web y dispositivos móviles.

  Actualmente cuenta con una amplia gama de productos GIS, como son: Geomedia, Geomedia Web, G/Technology, IntelliWhere, Fotogrametría, Plantas industriales.

- MAPGUIDE (AUTODESK) http://www.autodesk.es

Esta tecnología permite crear y publicar mapas e información de diseño de forma rápida y sencilla para su distribución interna o en la web. Es una de las soluciones cartográficas para web más fáciles de desarrollar e implantar. Basado en el proyecto MapGuide Open Source, Autodesk MapGuide Enterprise que le dota de todas las ventajas del software colaborativo.

Entre sus productos cuenta con: AutoCad, AutoCad Civil 3D, AutoCad Map 3D y diversos productos Autodesk como Autodesk Maya, con el que se pueden realizar animaciones, efectos visuales, simulación, renderización en 3D.

- AGEMAP http://www.agemap.com

Tiene como objetivo desarrollar productos de CAD y GIS para productos y servicios geográficos en las áreas de distribución, logística, marketing, investigación de mercados, planeamiento urbano y redes urbanas.

Entre sus productos se encuentran aplicaciones de mercadotecnia con una extensión geográfica, censos de población, herramientas para administrar y controlar flotas de vehículos, aplicaciones para realizar desarrollos en Internet orientados a la consulta de información geográfica, de mercado y de distribución, además de un kit para dotar a las aplicaciones de una API con gran cantidad de funciones GIS de gran potencia desarrolladas en lenguajes comunes como vBasic, vFox, Delphi, C++, etc.

2.4.8 Futuro de los GIS

Actualmente muchas disciplinas se están viendo beneficiadas por la tecnología GIS, teniendo una participación directa en gobiernos, universidades, empresas e instituciones que lo han aplicado a sectores como la salud pública, la defensa nacional, la arqueología, urbanismo, transporte, etc.

Una de las aplicaciones con mayor futuro y que más está creciendo actualmente son los servicios de localización, debido al abaratamiento de la tecnología GPS integrada en dispositivos móviles (PDA's, portátiles, teléfonos), permitiendo mostrar la ubicación respecto a algún punto de interés (restaurantes, gasolineras, cajeros, etc.) ó simplemente para mostrar la posición en la que nos encontramos o guiarnos hasta un destino concreto.

Otro sector que está en constante progreso es la cartografía web, cuyo objetivo es mostrar y editar cartografía en entornos web como Google Maps, Bing Maps, etc. Estos sitios web permiten a usuarios el acceso a enormes cantidades de datos geográficos, junto con datos alfanuméricos. Algunos de ellos utilizan software que, a través de una API, permiten a los usuarios crear aplicaciones personalizadas. Estos servicios ofrecen funciones de callejeros, imágenes aéreas, de satélite, geocodificación, búsquedas o funcionalidades de enrutamiento.

Por otro lado, las herramientas y tecnologías emergentes desde la W3C, están resultando útiles para los problemas de integración de datos en los sistemas de información. De igual forma, esas tecnologías se han propuesto como un medio para facilitar la interoperabilidad y la reutilización de datos entre aplicaciones SIG y también para permitir nuevos mecanismos de análisis.

Otro elemento al que se enfrentan los sistemas de información geográfica es de añadir el tiempo a los datos geométricos. Los SIG temporales incorporan las tres dimensiones espaciales (X, Y, Z) añadiendo además el tiempo en una representación 4D que se asemeja más a la realidad. La temporalidad en los

SIG permite recoge los procesos dinámicos de los elementos representados, como por ejemplo, la evolución del suelo en un determinado contexto, los accidentes geográficos que ha sufrido una región concreta, o, en el caso que nos aborda, poder observar la evolución del terreno en cuanto a construcción.

2.4.9. Proyecciones

La representación cartográfica del globo terrestre, ya sea considerado éste como una esfera o un elipsoide, supone un problema, ya que no existe modo alguno de representar toda la superficie desarrollada sin deformarla e incluso llegar a representarla fielmente, ya que la superficie de una esfera no es desarrollable en su conversión a un soporte papel (a una representación plana).

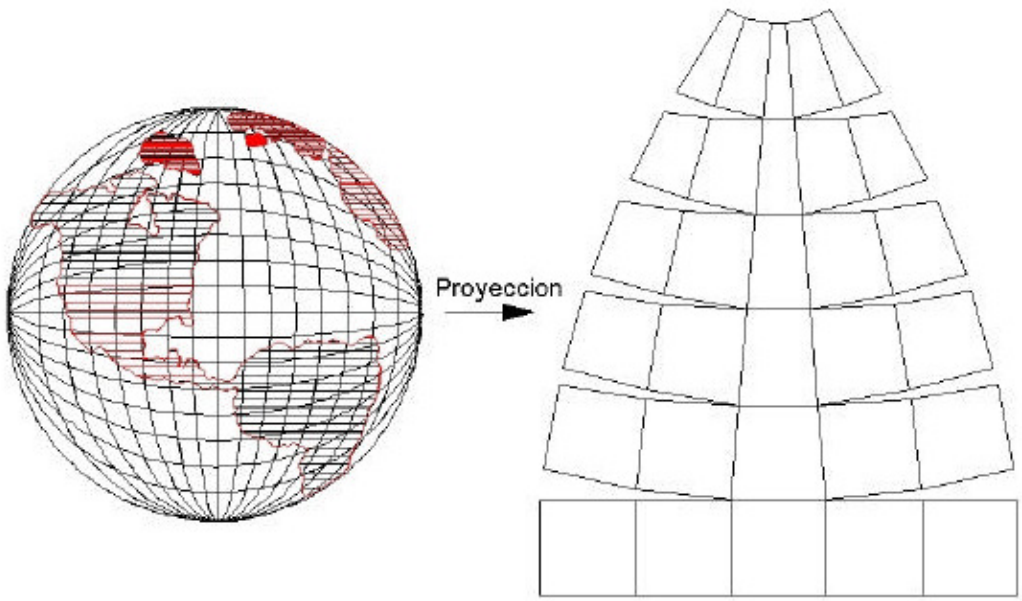


Figura 6: Proyección de la tierra sobre un plano.

Las proyecciones estudian las distintas formas de desarrollar la superficie terrestre minimizando, en la medida de lo posible, las deformaciones sufridas al representar la superficie terrestre.

En todos los casos conservan o minimizan los errores, dependiendo de la magnitud física que se desea conservar; su superficie, las distancias, los ángulos, etc., teniendo en cuenta que únicamente se podrá conservar una de las magnitudes anteriormente descritas y no todas a la vez.

Se recurre a un sistema de proyección cuando la superficie que estemos considerando es tan grande que tiene influencia la esfericidad terrestre en la representación cartográfica. La parte de la tierra entonces representada en papel u otro soporte se denomina "mapa". Esta representación de la tierra entra dentro del campo de la *Geodesia*.

2.4.9.1 Proyecciones planas

Cuando la superficie a representar es pequeña y por lo tanto la esfericidad terrestre no va a influir en la representación cartográfica, por ejemplo en pequeños levantamientos topográficos, se recurre a su representación de forma plana, de forma que todos los puntos representados están vistos desde su perpendicular, tal y como muestra la siguiente figura:

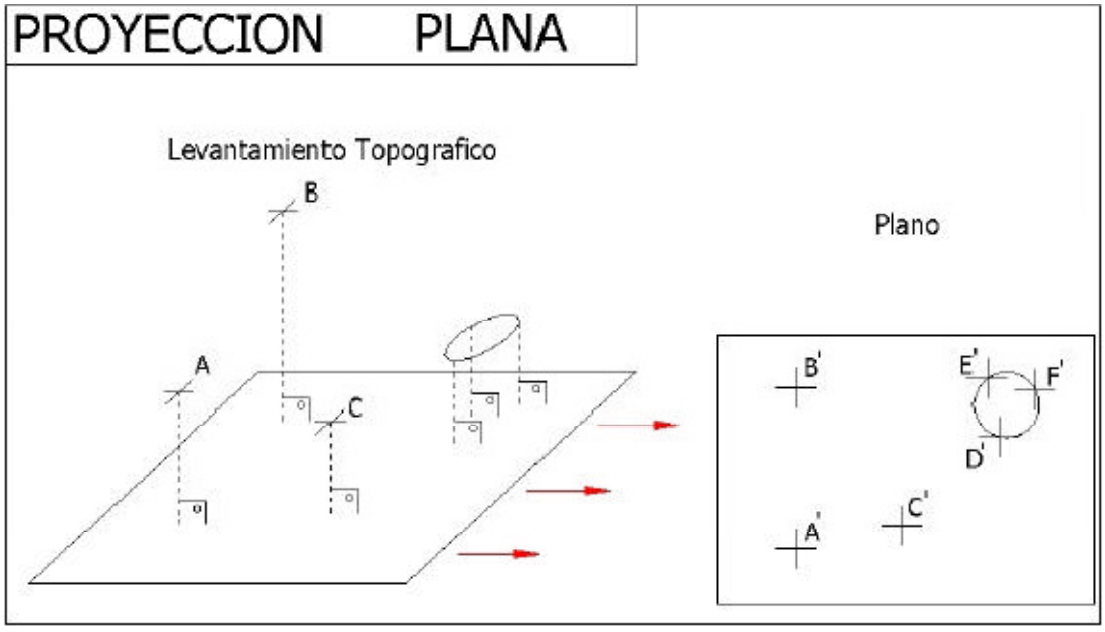


Figura 7: Ejemplo de una proyección de puntos sobre plano.

A la representación cartográfica obtenida, ya sea en soporte papel o en soporte magnético, se le denomina **"plano"**. Esta representación de la superficie, generalmente en el sistema de planos acotados, está dentro del campo de la **Topografía**, la Agrimensura, etc.

2.4.9.2 Proyecciones geodésicas

Las proyecciones geodésicas son proyecciones en las que la esfericidad terrestre tiene repercusión importante sobre la representación de posiciones geográficas, sus superficies, sus ángulos y sus distancias.

El sistema UTM es un sistema de proyección geodésica ideado en 1569 por Gerhard Kremer, denominado Mercator al latinizar su apellido. Es un sistema en el cual se construye geométricamente el mapa de manera que los meridianos y paralelos se transformen en una red regular, rectangular, de manera que se conserven los ángulos originales.

Este tipo de transformación se la denomina **conforme.** Dentro de las transformaciones posibles existen fundamentalmente tres tipos en función de la variable que conservan una vez proyectados: proyecciones conformes, equivalentes y alifáticas.

Una proyección no puede ser a la vez equivalente y conforme, ni a la inversa. En cartografía se emplean sobre todo las conformes, ya que interesa la magnitud angular sobre la superficial.

2.4.9.3 La proyección Mercator – Mercator transversal

La Proyección UTM conserva, por lo tanto, los ángulos pero distorsiona todas las superficies sobre los objetos originales así como las distancias existentes.

La proyección UTM se emplea habitualmente dada su gran importancia militar, y sobre todo, debido a que el Servicio de Defensa de Estados Unidos lo estandariza para su empleo mundial en la década de 1940.

Otra de las formas de clasificar a las proyecciones en función de la figura geométrica empleada al proyectar. La proyección UTM está dentro de las llamadas proyecciones cilíndricas, por emplear un cilindro situado en una determinada posición espacial para proyectar las situaciones geográficas.

El sistema de proyección UTM toma como base la proyección MERCATOR. Este es un sistema que emplea un cilindro situado de forma tangente al elipsoide en el ecuador, tal y como muestra la siguiente figura:

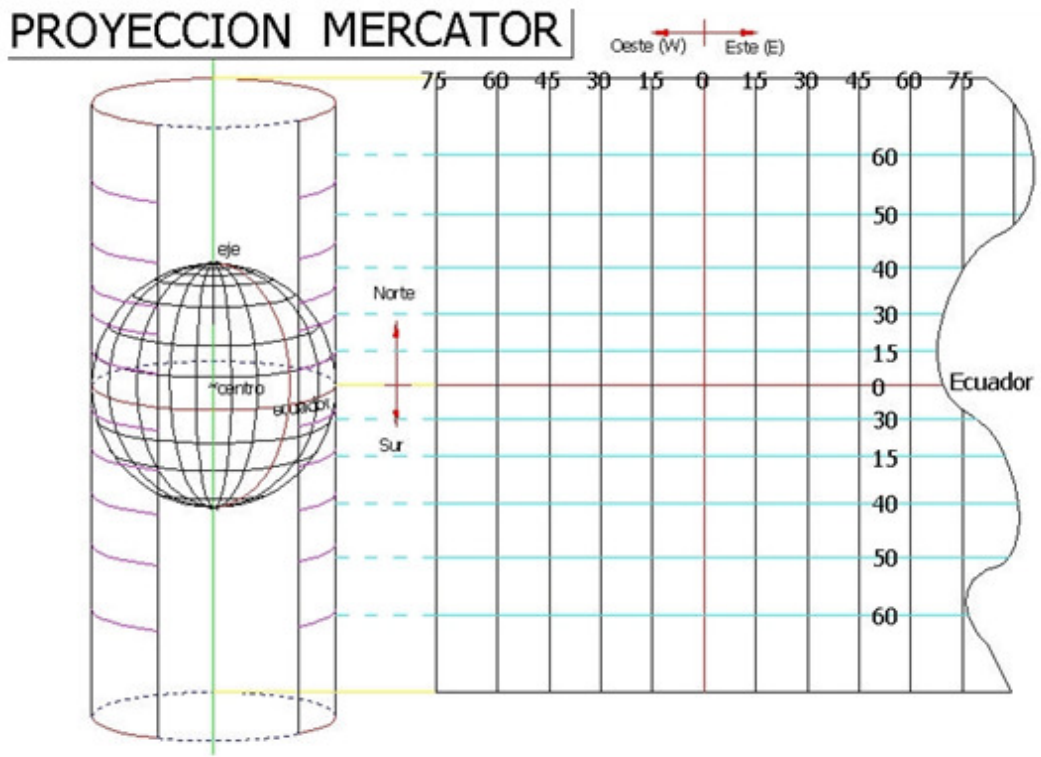


Figura 8: Proyección de mercator. Muestra la proyección tangente de la tierra sobre un cilindro.

La red creada hace que tanto meridianos como paralelos formen una cuadrícula oblicua, "grid" o rejilla, de manera que una recta oblicua situada entre dos paralelos forma un ángulo constante con los meridianos.

Como ejemplo de esta proyección se muestra el desarrollo de todo el globo terráqueo en la proyección mercator:

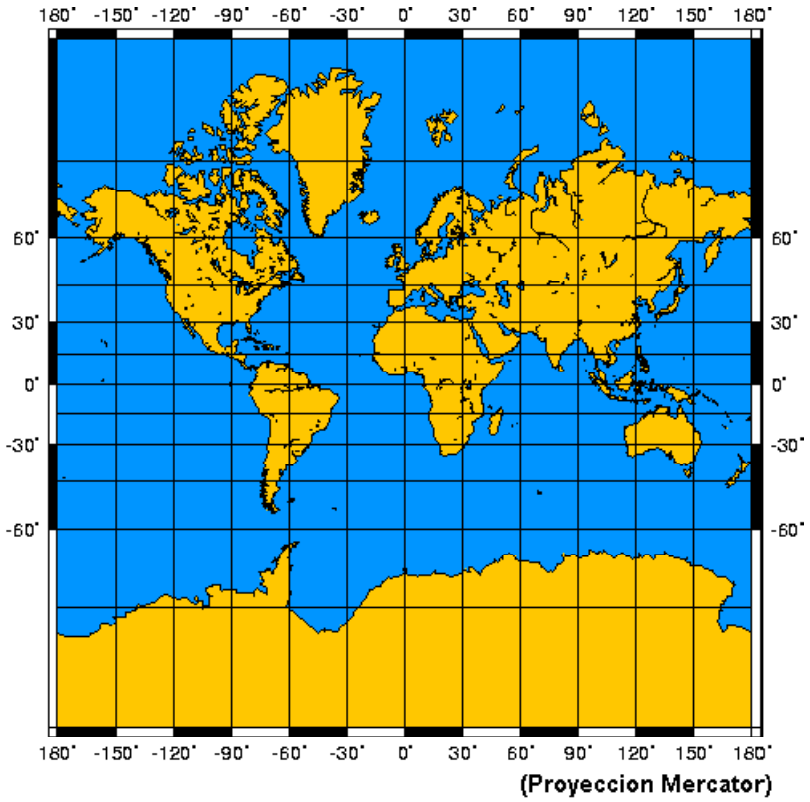


Figura 9: Proyección Mercator de todo el globo terrestre.

La proyección TRANSVERSAL MERCATOR (UTM), toma como base la proyección Mercator, sin embargo la posición del cilindro de proyección es transversal respecto del eje de la tierra:

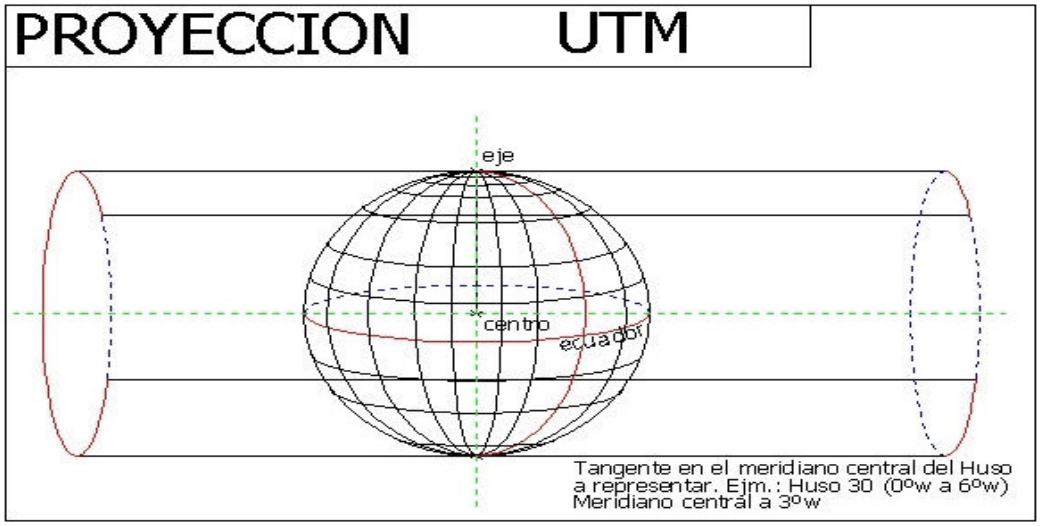


Figura 10: Proyección transversal de Mercator.

2.4.9.4 Sistema UTM: Distribución de husos

El sistema UTM divide el globo terráqueo en un total de 60 HUSOS.

Cada HUSO está nombrado con un número y zona, identificada con una letra. La distribución de los HUSOS es la siguiente:

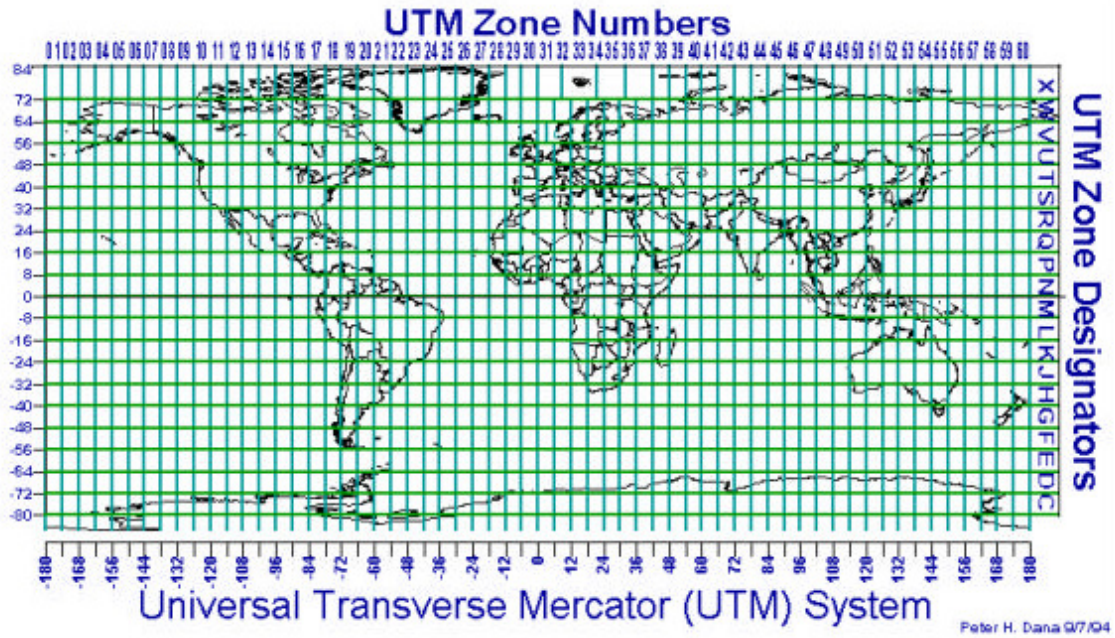


Figura 11: Mapa global dividido en zonas UTM.

Cada HUSO comprende un total de 6 º de LONGITUD, medidos desde el antemeridiano de Greenwich (180º Este), numerados en dirección este. Cada uno de estos sesenta husos se encuentra dividido en 20 zonas. 10 situadas en el hemisferio Norte y 10 situadas en el Hemisferio sur.

Cada una de estas zonas se designa por una letra C, D, E, F, G, H, J, K, L, y M, que corresponden a zonas situadas en el hemisferio sur y las notadas como N, P, Q, R, S, T, U, V, W y X que corresponden a zonas situadas en el hemisferio Norte.

Estas zonas se corresponden a 8º de LATITUD, si está comprendido dentro de las zonas desde la letra C, D, E y F hasta las zonas S, T, U, W, y para las zonas B y X que comprenden 12º de LATITUD.
El Huso 30 identifica una zona de la superficie terrestre situado entre la latitud 0º y 6º W (oeste), y su meridiano central es el de 3º W.

3. OBJETIVOS

Los objetivos que se plantean al desarrollar este trabajo son los siguientes:

- Estudio de las bases de datos espaciales
- Estudio de los GIS
- Selección de un dominio de aplicación
- Selección de herramientas de trabajo
- Diseño e implementación de un sistema basado en el dominio seleccionado.

- **Estudio de las bases de datos espaciales.**

A través del conocimiento teórico y práctico que se tenía inicialmente de ellas, se decidió utilizar el SGBD Oracle 11g, que incorpora un módulo destinado a la gestión de información geométrica.

A través de esta herramienta, que posee una API con funciones espaciales, se han podido desarrollar métodos y consultas que nos permitan conocer las propiedades de la información que hemos almacenado.

- **Estudio de los GIS**

A través de la herramienta ArcView, se ha logrado que, partiendo de la imagen de un mapa concreto, se puedan digitalizar las zonas que, para el dominio de esta aplicación, se han considerado más relevantes a modo de capas.

- **Selección de un dominio de aplicación**

En este caso, la idea inicial era construir un sistema que almacenase información dentro de un contexto social y urbanístico, ya que se dispondrá de datos concretos de personas (edad, dirección, tipo de vivienda, situación laboral, empresa en la que trabaja, cambios de domicilio, etc.), información de inmuebles (barrio al que pertenece, metros cuadrados, número de personas), características de esos barrios (zonas urbanas, rurales e industriales, número de calles, etc.) y características de estas zonas (área, empresas (si procede), barrio y ciudad a la que pertenecen).

- **Selección de herramientas de trabajo**

Además de las citadas anteriormente, se han necesitado otras herramientas para la consecución de este trabajo.

En el caso de Oracle 11g, se reparó en utilizar una sencilla aplicación desarrollada por esta multinacional, llamada shape2SDO y que nos permitía exportar toda la información geométrica (en formato shape) a formato entendible por el sistema gestor (formato SDO_GEOMETRY).

Una vez exportada, necesitábamos cargarla dentro de la base de datos en su tabla correspondiente, por lo que vimos que Oracle también disponía de su herramienta correspondiente, llamada SQL Loader, que viene ya integrada en la propia BBDD y no necesitaba instalación previa.

Ambas aplicaciones se utilizan por línea de comandos y disponen de una interfaz sencilla para su ejecución, lo que ha ayudado enormemente a desarrollar con mayor rapidez este trabajo.

4. ENTORNO DE TRABAJO

En este apartado se explicarán las utilidades que han servido para la consecución de este proyecto. Entre ellas están aplicaciones informáticas para el almacenamiento, representación y consulta de datos geométricos y alfanuméricos y conceptos básicos como los tipos de datos espaciales, junto con sus respectivas funciones.

4.1 Oracle Spatial

Oracle Spatial es un conjunto integrado de funciones y procedimientos que permiten guardar, acceder y analizar datos espaciales de forma rápida y eficiente en el SGBD Oracle.

Los datos espaciales representan las características esenciales de localización de objetos reales o conceptuales que se relacionan con el espacio real o conceptual en el cual ellos existen.

Oracle Spatial proporciona un esquema SQL y funciones que facilitan el almacenamiento, recuperación, actualización y consultas de grupos de características espaciales en el SGBD de Oracle. Oracle Spatial tiene los siguientes componentes:

- Un esquema (MDSYS) que determina el almacenamiento, sintaxis y semántica de los tipos de datos geométricos soportados.

- Un mecanismo de indización espacial.

- Un conjunto de operadores y funciones para realizar consultas sobre áreas de interés, consultas sobre relaciones espaciales y otras operaciones espaciales.

- Utilidades administrativas.

La característica de los elementos espaciales es la representación geométrica de sus límites en un espacio de coordenadas.

Oracle Spatial soporta diversos tipos de elementos geométricos, tanto simples como compuestos. Entre los diferentes tipos de elementos que se pueden implementar están los siguientes elementos de dos dimensiones:

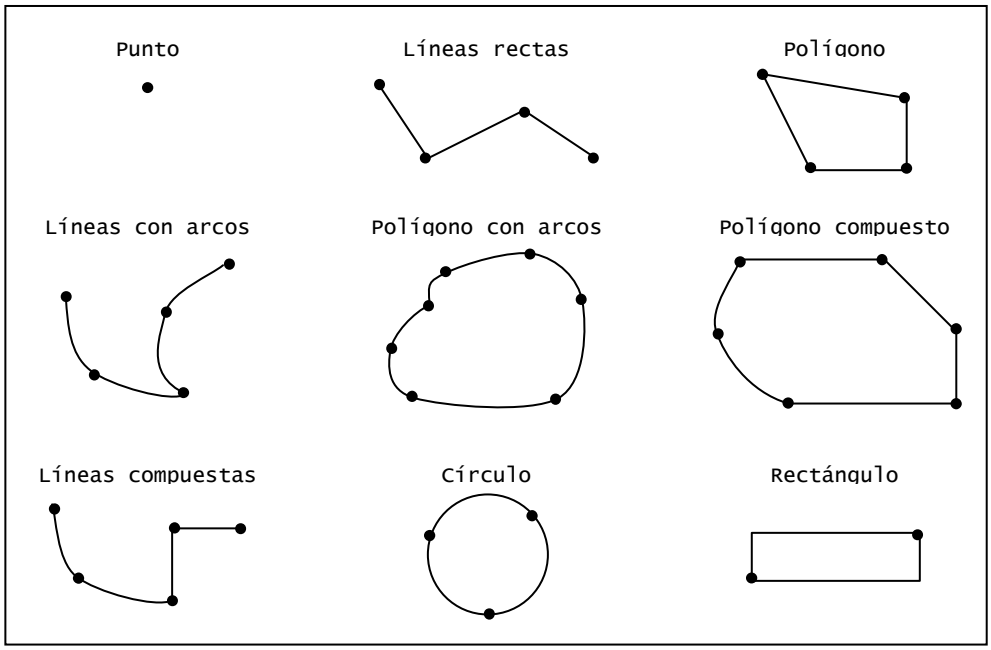


Figura 12: Tipos de geometrías de dos dimensiones

Los puntos de dos dimensiones son elementos compuestos por dos coordenadas X e Y, que a menudo corresponden con la longitud y la latitud.

Las líneas están compuestas por uno o más pares de puntos que definen segmentos que pueden ser rectos o arcos. Los polígonos están compuestos por líneas conectadas que forman un anillo cerrado, el área de los polígonos está implícita.

Para la representación de este sistema de coordenadas, se usa el tipo de dato SDO_GEOMETRY.

4.2 Tipo de dato SDO_GEOMETRY

Con Oracle Spatial, la descripción de los objetos espaciales es guardada en una fila simple, en una columna de tipo SDO_GEOMETRY de una tabla definida por el usuario. Cualquier tabla que tenga columnas del tipo SDO_GEOMETRY debe tener otra columna, o conjunto de columnas, que definan una única clave primaria para la tabla. Las tablas de este tipo algunas veces son llamadas tablas espaciales o tablas geométricas espaciales.

Oracle Spatial define el tipo SDO_GEOMETRY como sigue:

```
CREATE TYPE sdo_geometry AS OBJECT(
 SDO_GTYPE NUMBER,
 SDO_SRID NUMBER,
 SDO_POINT SDO_POINT_TYPE,
 SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
 SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Oracle Spatial también define los tipos SDO_ELEM_INFO_ARRAY, SDO_ORDINATE_ARRAY y SDO_POINT_TYPE los cuales son usados en la definición del tipo SDO_GEOMETRY como sigue:

```
CREATE TYPE sdo_point_type AS OBJECT(
     X NUMBER,
     Y NUMBER,
     Z NUMBER);
CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
CREATE TYPE sdo_ordinate_array AS VARRAY (1048576) of NUMBER;
```

El tipo SDO_GEOMETRY tiene métodos que proporcionan acceso a algunos de los atributos.

A continuación se describe la semántica de cada atributo de SDO_GEOMETRY y algunas consideraciones de uso.

4.2.1 SDO_GTYPE

El atributo SDO_GTYPE indica el tipo de la geometría. Los tipos válidos de geometría corresponden con aquellos especificados en el Geometry Object Model for the OGIS Simple Features for SQL (Modelo de Objetos Geométricos para el OGIS, Open Geographic Information System, para Características Simples de SQL) (con la excepción de Superficies). Los valores numéricos difieren de los dados por las especificaciones del OGIS, pero tienen una correspondencia directa entre los nombres y la semántica donde es aplicable.

El valor SDO_GTYPE tiene 4 dígitos con el formato **dltt**, dónde:

- **d** identifica el número de dimensiones (2, 3 o 4). El número de dimensiones refleja el número de coordenadas usadas para representar cada vértice (por ejemplo, *X, Y* para objetos en dos dimensiones). Los puntos y las líneas son considerados objetos en dos dimensiones.

  En una columna dada, todos los elementos geométricos deben tener el mismo número de dimensiones. Por ejemplo, no se puede mezclar datos de dos dimensiones con datos de tres dimensiones en la misma columna.

- **l** identifica la dimensión de la medida de referencia linear. Para un sistema de referencia linear (LRS) de tres dimensiones geométricas, puede ser de dimensión (3 o 4) conteniendo este valor. Para una geometría que no sea LRS, o para aceptar el que por defecto tiene Spatial como última dimensión medida para una geometría LRS, hay que especificar 0.

  - **tt** identifica el tipo de geometría (00 hasta 07, con 08 hasta 99 reservados para usos futuros).

La

Tabla 1 muestra los valores válidos para SDO_GTYPE. El tipo de geometría y la descripción reflejan la especificación del OGIS.

Tabla 1: Valores válidos par SDO_GTYPE

Valor	Tipo de Geometría	Descripción
*d*700	GEOMETRÍA DESCONOCIDA	Oracle Spatial ignora esta geometría
*d*701	PUNTO	La geometría contiene un punto
*d*702	LÍNEA O CURVA	La geometría contiene una línea quebrada que puede contener líneas rectas, segmentos de arco circulares, o ambas. (LÍNEA y CURVA son sinónimos en este contexto).
*d*703	POLÍGONO	La geometría contiene un polígono con o sin agujeros[1].
*d*704	COLECCIÓN	La geometría es una colección heterogénea de elementos[2]. COLECCIÓN es un súper conjunto que incluye todos los otros tipos.
*d*705	MULTIPUNTO	La geometría tiene uno o más puntos (MULTIPUNTO es un súper conjunto de PUNTO).
*d*706	MULTILÍNEA o MULTICURVA	La geometría tiene uno o más líneas. (MULTILÍNEA o arcos MULTICURVA son sinónimos en este contexto, y cada una es un súper conjunto de LÏNEAS y CURVAS)
*d*707	MULTIPOLÍGONO	La geometría puede tener múltiples polígonos disjuntos (más que un límite exterior). (MULTIPOLÍGONO es un súper conjunto de POLÍGONOS)

[1] Para un polígono con agujeros, se introduce primero el límite exterior, seguido por cualquier límite interior.
2 Los polígonos en la colección pueden ser disjuntos.

Los siguientes métodos están disponibles para devolver los componentes individuales del *dltt* del SDO_GTYPE para un objeto geométrico: GET_DIMS, GET_LRS_DIMS, y GET_GTYPE.

4.2.2 SDO_SRID

El atributo SDO_SRID puede ser usado para identificar un sistema de coordenadas (Sistema espacial de referencia) que será asociado con la geometría. Si SDO_SRID es nulo (null), ningún sistema de coordenadas será asociado con la geometría. Si SDO_SRID no es nulo, debe contener un valor de la columna SRID de la tabla MDSYS.CS_SRS y este valor debe ser insertado dentro de la columna SRID de la vista USER_SDO_GEOM_METADATA.

Todos los elementos geométricos en la misma columna deben tener el mismo valor de SDO_SRID.

4.2.3 SDO_POINT

El atributo SDO_POINT es definido usando el tipo SDO_POINT_TYPE, que tiene los atributos (*X*,*Y*,*Z*) todos de tipo NUMBER. Si los arrays SDO_ELEM_INFO y SDO_ORDINATES son ambos nulos, y el atributo SDO_POINT es no nulo, entonces los valores de la *X* y la *Y* se consideran las coordenadas de un punto geométrico. Si sucede lo contrario, el atributo SDO_POINT será ignorado por Oracle Spatial.

Para un óptimo almacenaje, se deberían guardar los puntos geométricos en el atributo SDO_POINT, esto es, si solamente se tienen puntos en esa columna, es muy recomendable que se guarden los puntos geométricos en este atributo.
No usar el atributo SDO_POINT en la definición de puntos en un sistema de referencia linear (LRS).

4.2.4 SDO_ELEM_INFO

El atributo SDO_ELEM_INFO es definido usando un array de longitud variable de números. Este atributo permite interpretar las coordenadas guardadas en el atributo SDO_ORDINATES.
Cada conjunto triple de números es interpretado como sigue:

- SDO_STARTING_OFFSET - Indica la distancia desde el inicio (offset) del array. SDO_ORDINATES - donde está guardada la primera coordenada de este elemento. El valor de comienzo es normalmente 1 y no 0. Así, la primera coordenada para el primer elemento estará en SDO_GEOMETRY.SDO_ORDINATES(1).

  Para un segundo elemento, la primera coordenada estará en SDO_GEOMETRY.SDO_ORDINATES(*n*), donde *n* refleja la posición dentro de la definición de SDO_ORDINATE_ARRAY (por ejemplo, 19 para el decimonoveno número).

- SDO_ETYPE – Indica el tipo de elemento. Los valores válidos se mostrarán en la Tabla 2.

Los valores de SDO_ETYPE, 1, 2, 1003 y 2003 son considerados *elementos simples*. Son definidos por una sola entrada triple de datos en el array SDO_ELEM_INFO. Para los valores SDO_ETYPE 1003 y 2003, el primer dígito indica exterior (1) o interior (2):

- 1003: anillo de polígono exterior (los puntos deben ser especificados en sentido contrario a las agujas del reloj)
- 2003: anillo de polígono interior (los puntos deben ser especificados en el sentido de las agujas del reloj)

El uso del valor 3 como un valor de SDO_ETYPE para un elemento de polígono cerrado en una única geometría es desaconsejado. Se debería especificar 3 solamente si no se conoce si el polígono simple es exterior o interior y se podría entonces actualizar la tabla o columna para el formato actual usando el procedimiento SDO_MIGRATE.TO_CURRENT.

No se puede mezclar valores de 1 dígito y de 4 dígitos de SDO_ETYPE en una única geometría. Si se usa un valor de 4 dígitos de SDO_ETYPE, se sebe usar un valor de 4 dígitos para SDO_GTYPE.

Los valores 4, 1005 y 2005 son considerados *elementos compuestos*. Éstos contienen al menos una cabecera triple con una serie de valores triples que corresponden al elemento compuesto. Para los valores de SDO_ETYPE 1005 y 2005, el primer dígito indica exterior (1) o interior (2):

- 1005: anillo de polígono exterior (los puntos deben ser especificados en sentido contrario a las agujas del reloj)
- 2005: anillo de polígono interior (los puntos deben ser especificados en el sentido de las agujas del reloj)

El uso del valor 5 como un valor de SDO_ETYPE para un elemento de polígono cerrado en una única geometría es desaconsejado. Se debería especificar 5 solamente si no se conoce si el polígono compuesto es exterior o interior y se podría entonces actualizar la tabla o columna para el formato actual usando el procedimiento SDO_MIGRATE.TO_CURRENT.

Los elementos de un elemento compuesto son contiguos. El último punto de un subelemento de un elemento compuesto es el primer elemento del siguiente subelemento. El punto no es repetido.

- SDO_INTERPRETATION – significa una o dos cosas, dependiendo si SDO_ETYPE es o no un elemento compuesto.

Si SDO_ETYPE es un elemento compuesto (4, 1005 o 2005), este campo especifica cuántas subsecuencias de valores triples forman parte del elemento.
Si SDO_ETYPE no es un elemento compuesto (1, 2, 1003 0 2003), la interpretación del atributo determina cómo será interpretada la secuencia de coordenadas de ese elemento. Por ejemplo, una línea o un polígono cerrado pueden ser fabricados como una secuencia de segmentos de líneas rectas o arcos circulares.

La descripción y las parejas de valores válidos para SDO_ETYPE y SDO_INTERPRETATION se representan en la Tabla 2.
Si un elemento geométrico consta de más de un elemento, entonces la última coordenada para un elemento está siempre en una posición menos que la del siguiente elemento. El último elemento en la geometría es descrito por las coordenadas desde su principio de posición en el array de longitud variable SDO_ORDINATES hasta el final.

Para elementos compuestos (valores de SDO_ETYPE 4, 1005 o 2005), un conjunto de *n* elementos (uno por cada subelemento), compuesto cada uno por tres valores, es usado para describir el elemento compuesto. Es importante

recordar que los subelementos de un elemento compuesto son contiguos. El último punto de un subelemento es el primer punto del siguiente subelemento.

Para los subelementos de 1 hasta *n*-1, el punto final de un subelemento es el mismo que el punto inicial del siguiente subelemento. El punto inicial de los subelementos 2…*n*-2 es el mismo que el punto final del subelemento 1…*n*-1. La última coordenada del subelemento *n* es cada posición *n* del array. SDO_ORDINATES menos uno del siguiente elemento de la geometría o la última coordenada en el array de longitud variable SDO_ORDINATES.

El tamaño usado de un array de longitud variable puede ser determinado usando la función varray_variable. Que se encuentra en PL/SQL o OCICollSize en la interfaz de llamadas de Oracle (Oracle Call Interface, OCI).

La semántica de cada elemento SDO_ETYPE y la relación entre el SDO_ELEM_INFO y el array de longitud variable SDO_ORDINATES para cada uno de los elementos SDO_ETYPE dados se muestra en la Tabla 2.

Tabla 2: Valores y semántica en SDO_ELEM_INFO

SDO_ ETYPE	SDO_ INTERPRETACIÓN	Significado
0	(ningún valor numérico)	Elemento Tipo 0 (cero). Usado para tipos de modelos de geometría no soportados por Oracle Spatial.
1	1	Tipo punto.
1	*n* > 1	Agrupación de puntos con *n* puntos.
2	1	Línea cuyos vértices están conectados por líneas rectas.
2	2	Línea construida por una secuencia conectada de arcos circulares. Cada arco, está descrito usando tres coordenadas: el punto de comienzo de arco, un punto cualquiera del arco y el punto final del arco. Las coordenadas para un punto designado como final de un arco y comienzo del siguiente no son repetidas. Por ejemplo, se usan cinco coordenadas para describir una línea construida por dos arcos conectados. Los puntos 1, 2 y 3 definen el primer arco y los puntos 3, 4 y 5 el segundo arco, el punto 3 es guardado solamente una vez.

SDO_ ETYPE	SDO_ INTERPRETACIÓN	Significado
1003 2003	o 1	Un polígono simple cuyos vértices son conectados por segmentos de líneas rectas. Se debe especificar un punto por cada vértice y el último punto especificado debe ser el exactamente el mismo punto que el primero (para cerrar el polígono), a pesar del valor de tolerancia. Por ejemplo, para un polígono de 4 lados, se especifican 5 puntos, con el punto 5 igual al punto 1.
1003 2003	o 2	Polígono construido por una secuencia de arcos conectados que se cierran. El punto final del último arco será el mismo que el punto inicial del primer arco. Cada arco, está descrito usando tres coordenadas: el punto de comienzo de arco, un punto cualquiera del arco y el punto final del arco. Las coordenadas para un punto designado como final de un arco y comienzo del siguiente no son repetidas. Por ejemplo, se usan cinco coordenadas para describir una línea construida por dos arcos conectados. Los puntos 1, 2 y 3 definen el primer arco y los puntos 3, 4 y 5 el segundo arco, el punto 3 es guardado solamente una vez. Las coordenadas de los puntos 1 y 5 deben ser las mismas (la tolerancia no es considerada) y el punto 3 no debe ser repetido.
1003 2003	o 3	Tipo rectángulo (a veces llamado *rectángulo optimizado*). Un rectángulo que para representarlo tan solo son necesarios dos puntos, el inferior izquierdo y el superior derecho. El tipo rectángulo puede ser usado con datos geodésicos o no geodésicos. Sin embargo, con datos geodésicos, solo se puede usar este tipo para crear una ventana de consulta (no para guardar elemento en la base de datos).
1003 2003	o 4	Tipo círculo. Descrito por tres puntos distintos no alineados, todos en la circunferencia del círculo.
4	*n* > 1	Línea compuesta con algunos vértices conectados con segmentos rectos y otros con arcos. El valor *n* en la columna SDO_Interpretación especifica el número de subelementos contiguos que construyen la línea. Las siguientes *n* tripletas de valores en el array SDO_ELEM_INFO describen cada uno de los subelementos. Los subelementos pueden ser solamente del SDO_ETYPE 2. El último punto de un subelemento es el primer punto del siguiente subelemento y no debe ser repetido.
1005 2005	o *n* > 1	Polígono compuesto con algunos vértices conectados por segmentos rectos y otros por arcos. El valor *n* en la columna SDO_Interpretación especifica el número de subelementos contiguos que construyen el polígono. Las siguientes *n* tripletas de valores en el array SDO_ELEM_INFO describen cada uno de los subelementos. Los subelementos pueden ser solamente del SDO_ETYPE 2. El último punto de un subelemento es el primer punto del siguiente subelemento y no debe ser repetido. Los puntos inicial y final del polígono deben ser exactamente el mismo punto (la tolerancia es ignorada).

4.2.5 SDO_ORDINATES

El atributo SDO_ORDINATES es definido usando un array de longitud variable (1048576) de tipo NUMBER que almacena los valores de las coordenadas que construyen los límites de un objeto espacial.

Este array debe ser usado en conjunción con el array de longitud variable SDO_ELEM_INFO. Los valores en el array son ordenados por dimensión. Por ejemplo, un polígono cuyos límites tengan cuatro puntos de dos dimensiones se guardará como {X1, Y1, X2, Y2, X3, Y3, X4, Y4, X1, Y1}.

Si los puntos fueran de tres dimensiones, entonces se almacenarían como {X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3, X4, Y4, Z4, X1, Y1, Z1}. Los índices espaciales, operadores y funciones ignoran el valor Z porque esta versión del producto soporta solamente objetos espaciales en dos dimensiones. El número de dimensiones asociadas con cada punto es almacenado como metadato en la vista xxx_SDO_GEOM_METADATA.

Los valores en el array SDO_ORDINATES deben ser todos validos y no nulos. No se usan valores especiales para delimitar elementos en un multielemento geométrico. Los puntos de inicio y fin para la descripción de una secuencia que especifica un elemento son determinados por el valor del SDO_START_OFFSET para cada elemento y el elemento siguiente, en el array SDO_ELEM_INFO, como se ha explicado previamente.

El valor de comienzo del array 1, SDO_ORDINATES(1) es la primera coordenada del primer punto del primer elemento.

4.2.6 Consideraciones de uso

Se deberían usar los valores de SDO_GTYPE que se muestran en la Tabla 1; sin embargo, Oracle Spatial no comprueba ni fuerza todas las restricciones de consistencia geométrica. Oracle Spatial sí comprueba lo siguiente:

- Para los valores de SDO_GTYPE *d*001 y *d*005, cualquier subelemento que no sea de SDO_ETYPE 1 es ignorado.
- Para los valores de SDO_GTYPE *d*002 y *d*006, cualquier subelemento que no sea de SDO_ETYPE 2 es ignorado.
- Para los valores de SDO_GTYPE *d*003 y *d*007, cualquier subelemento que no sea de SDO_ETYPE 2 o 5 es ignorado (esto incluye las variantes de SDO_ETYPE 1003, 2003, 1005 y 2005).

La función SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT puede ser usada para evaluar la consistencia de un objeto geométrico simple o para todos los objetos geométricos en una tabla especificada

4.3 ANÁLISIS GEOMÉTRICOS

Para realizar análisis geométricos eficientes, se hará uso de tres elementos básicos de la tecnología de Oracle Spatial:

- Índices espaciales
  Análogamente a un índice en árbol B, los índices espaciales facilitan una ejecución rápida de operadores espaciales en columnas SDO_GEOMETRY de tablas de Oracle.

  Un árbol B indexa la columna del identificador de la tabla que se indique pudiendo facilitar búsquedas basadas en este tipo de identificadores. De forma similar, un índice espacial en la columna localización de tipo SDO_GEOMETRY de una tabla podría facilitar una ejecución más rápida de operadores como SDO_WITHIN_DISTANCE.

- Operadores espaciales
  Igual que se puede especificar operadores relacionales en un sentencia de SQL, como < (menor que), > (mayor que), o = (igual que), etc., se pueden usar operadores espaciales para buscar la localización de las columnas SDO_GEOMETRY de una tabla por proximidad con respecto a una consulta de localización.

- Funciones de procesamiento geométrico
  Estas funciones representan una importante variedad de operaciones, incluyendo el cálculo de interacciones, distancias, áreas y diversos análisis rigurosos de los datos espaciales.

La mayoría de las funcionalidades de los índices y operadores espaciales son parte del localizador de Oracle (Oracle Locator) que está incluido en todas las ediciones de bases de datos de Oracle. Esto significa que todas las aplicaciones de Oracle pueden potenciar la funcionalidad descrita.

4.3.1 Índices Espaciales

Para usar operadores espaciales se necesita primero crear un índice espacial en la columna donde vayamos a almacenar los datos geométricos.

Tal como sucede en una tabla convencional que contuviese un índice en árbol B en una columna de datos estándar, como por ejemplo, *nombre,* en dónde el índice optimizaría las consultas que se realizaran a través de este dato, un índice espacial creado en una columna SDO_GEOMETRY se requiere para asegurar la respuesta efectiva en tiempo en búsquedas que usan operadores espaciales. Es uno de los primeros pasos a acometer para dicho fin.

Es importante recordar, que antes de crear un índice espacial hay que haber introducido los metadatos de la columna de la tabla en la base de datos.

4.3.1.1 Sintaxis para creación de índices espaciales

La creación de índices espaciales, no tiene mayor dificultad que la de escribir correctamente la sintaxis en SQL. La sintaxis correcta sería la siguiente:

```
CREATE INDEX <nombre_índice> ON <nombre_tabla>
(<nombre_columna>) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

4.3.1.2 Inserción de metadatos para tablas espaciales

Uso de rutinas que requieren que se cree una vista con metadatos sobre columnas SDO_GEOMETRY.

Los metadatos para una capa espacial (identificada por <table_name, column_name>) son insertados en la vista USER_GEOM_METADATA.
La Vista de metadatos es creada para todos los usuarios en la instalación de Oracle Spatial.

Por cada columna SDO_GEOMETRY, se inserta una fila en la vista USER_SDO_GEOM_METADATA.

Esta vista la presenta la siguiente estructura:

```
Name                          Null             Type
------------------            --------         -------------------
TABLE_NAME                    NOT NULL         VARCHAR2(32)
COLUMN_NAME                   NOT NULL         VARCHAR2(1024)
DIMINFO                                        MDSYS.SDO_DIM_ARRAY
SRID                                           NUMBER
```

```
MDSYS.SDO_DIM_ARRAY                          VARRAY(4) OF SDO_DIM_ELEMENT
MDSYS.SDO_DIM_ELEMENT object
      SDO_DIMNAME                                 VARCHAR2(64)
      SDO_LB                                      NUMBER
      SDO_UB                                      NUMBER
      SDO_TOLERANCE                               NUMBER
```

En donde:

- SDO_DIMNAME

  Nombre de la dimensión.

- SDO_LB

  Valor mínimo para esta dimensión.

- SDO_UB

  Valor máximo para esta dimensión.

- SDO_TOLERANCE

  Redondeo que usa Oracle Spatial para índices, operadores y funciones. Rellena con 0 la precisión decimal y añade un 5. (p.ej. 0.00005).

- SRID

  SRID específico, un sistema de coordinadas geodésicas. Nulo para la versión 8i.

Relleno de la tabla USER_SDO_GEOM_METADATA

```
INSERT INTO USER_SDO_GEOM_METADATA
 (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES (
 'NOMBRE_TABLA',
 'COLUMNA_SDO_GEOMETRY',
 MDSYS.SDO_DIM_ARRAY (
 MDSYS.SDO_DIM_ELEMENT('Long', -180, 180, .005),
 MDSYS.SDO_DIM_ELEMENT('Lat', -90, 90, .005)),
       NULL);
```

Notas sobre las dimensiones

• Todas las capas a cruzar deben tener los mismos límites en los ejes de las dimensiones.

• Se pueden definir más de 2 dimensiones en SDO_DIM_ARRAY, pero Oracle Spatial sólo trabajará con las dos primeras definidas.

Vistas de sistema *_SDO_GEOM_METADATA

Hay 3 vistas de sistema *_SDO_GEOM_METADATA:

- USER_SDO_GEOM_METADATA.
  Para insertar la información.

- ALL_SDO_GEOM_METADATA
  Para seleccionar la información.

- DBA_SDO_GEOM_METADATA
  Para ver todos los metadatos espaciales.

4.3.2 Funciones de procesamiento geométrico

En este punto, se describirán los diferentes operadores espaciales soportados por Oracle Spatial para la realización de los análisis espaciales.

Oracle Spatial posee una API de funciones geométricas que nos permitirán manipular estos datos para realizar cualquier tipo de análisis.

Al igual que los operadores relacionales <, >, y =, los operadores espaciales pueden ser usados en la cláusula `WHERE` de una expresión regular de SQL. Se examinará la sintaxis y la semántica de estos operadores.

4.3.2.1 Semántica de operadores espaciales

Cuando se especifica un operador espacial en una declaración de SQL, Oracle selecciona aquellas filas para las cuales los valores del operador son verdaderas (`TRUE`).

De esta manera el operador selecciona solo aquellas filas de la tabla donde los valores correspondientes a la columna SDO_GEOMETRY satisfacen un operador específico (localización de la consulta).

Este tipo de selección es análoga a la selección con operadores relacionales usando un predicado específico como `id>=10`.

4.3.2.2 Evaluación de operadores espaciales

A continuación, se abordará brevemente cómo son evaluados los operadores espaciales. Entender este proceso ayudará a asegurar una mejor estrategia en la ejecución de los operadores espaciales.

Puesto que el operador espacial está ligado al índice espacial, en muchos casos, son evaluados en dos etapas de mecanismo de filtrado usando el índice espacial, como se muestra en la siguiente figura:

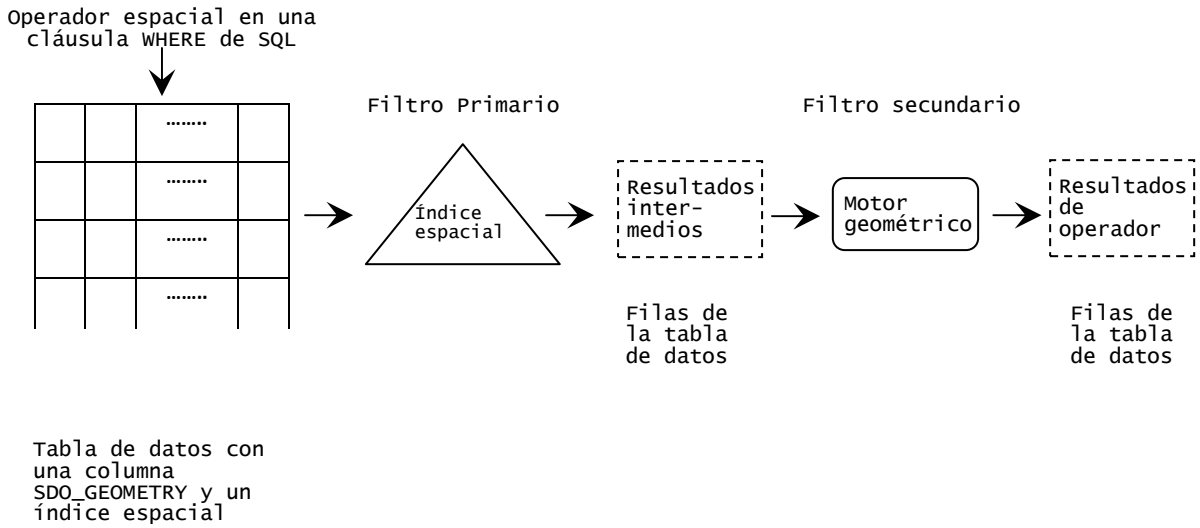


Figura 13: Evaluación de un operador espacial usando un índice espacial asociado

Primero es evaluado el operador espacial usando el índice asociado. Esta evaluación está referida como *filtro primario*. Esta aproximación con el índice es usada para identificar un conjunto candidato de filas que satisfagan la relación con respecto a la localización de la consulta.

Las filas identificadas son entonces pasadas a través del motor geométrico, referido como *filtro secundario*, para devolver el conjunto correcto de filas para

el operador especificado. Comentar que todo este proceso es transparente al usuario, ya que éste solamente especifica el operador en la cláusula `WHERE` de una declaración de SQL y Oracle realiza internamente el plan de ejecución de la consulta.

En algunos casos, sin embargo, el optimizador puede decidir pasar por alto el índice espacial. Entonces invoca el filtro secundario directamente obteniendo las filas correctas de la tabla.

4.3.2.3 Funciones geométricas en Oracle Spatial

Oracle Spatial dispone de una API de funciones geométricas que pueden ser invocadas en cualquier consulta, procedimiento, función o trigger para el tratamiento de nuestra información espacial. A continuación se introducirán brevemente aquellas que han sido utilizadas para la consecución de este trabajo.

- **Función SDO_GEOM.SDO_AREA**

  Devuelve el área de un polígono de dos dimensiones.

Formato

```
SDO_GEOM.SDO_AREA(
          geom IN MDSYS.SDO_GEOMETRY,
          dim IN MDSYS.SDO_DIM_ARRAY
          [, unit IN VARCHAR2]
          ) RETURN NUMBER;
     O también:
     SDO_GEOM.SDO_AREA(
          geom IN MDSYS.SDO_GEOMETRY,
          tol IN NUMBER
          [, unit IN VARCHAR2]
```

```
) RETURN NUMBER;
```

Dónde los parámetros especifican:

geom

La columna que contiene el tipo de dato geométrico de la tabla correspondiente.

dim

Corresponde a la geometría del objeto. Se puede consultar en las vistas xxx_SDO_GEOM_METADATA.

unit

Unidad en la que queremos que calcule el área. Los valores disponibles se pueden consultar en la tabla MDSYS.SDO_AREA_UNITS.

Si no se especifica nada, la unidad por defecto que tomará Oracle son los km cuadrados.

tol

Margen de error.

- **SDO_GEOM.SDO_DISTANCE**

Calcula la distancia entre dos objetos. Ésta es computada como la distancia entre el par de puntos más cercanos de las dos geometrías.

Formato

```
SDO_GEOM.SDO_DISTANCE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     dim1 IN MDSYS.SDO_DIM_ARRAY,
     geom2 IN MDSYS.SDO_GEOMETRY,
     dim2 IN MDSYS.SDO_DIM_ARRAY
     [, unit IN VARCHAR2]
     ) RETURN NUMBER;
```

```
O también:

SDO_GEOM.SDO_DISTANCE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     geom2 IN MDSYS.SDO_GEOMETRY,
     tol IN NUMBER
     [, unit IN VARCHAR2]
     ) RETURN NUMBER;
```

Donde los parámetros especifican:

geom1

Geometría cuya distancia a la geometría 2 queremos evaluar.

dim1

Dimensión de la geometría 1, cuyo valor podemos consultar en las vistas xxx_SDO_GEOM_METADATA.

geom2

Geometría cuya distancia a la geometría 1 queremos evaluar.

dim2

Dimensión de la geometría 2, cuyo valor podemos consultar en las vistas xxx_SDO_GEOM_METADATA.

unit

Unidad en la que queremos que calcule el área. Los valores disponibles se pueden consultar en la tabla MDSYS.SDO_AREA_UNITS.
Si no se especifica nada, la unidad por defecto que tomará Oracle son los km cuadrados.

tol

Margen de error.

- **SDO_GEOM.WITHIN_DISTANCE**

Determina si dos objetos espaciales están dentro de una distancia especificada. Devuelve TRUE o FALSE según corresponda.

Formato

```
SDO_GEOM.WITHIN_DISTANCE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     dim1 IN MDSYS.SDO_DIM_ARRAY,
     dist IN NUMBER,
     geom2 IN MDSYS.SDO_GEOMETRY,
     dim2 IN MDSYS.SDO_DIM_ARRAY
     ) RETURN VARCHAR2;
O también:
SDO_GEOM.WITHIN_DISTANCE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     dist IN NUMBER,
     geom2 IN MDSYS.SDO_GEOMETRY,
     tol IN NUMBER
     ) RETURN VARCHAR2;
```

Los parámetros indican lo mismo que las anteriores funciones, salvo que ahora se requiere la distancia a la que se quiere hacer el cálculo en el parámetro `dist`.

- **SDO_GEOM.SDO_INTERSECTION**

Devuelve un objeto geométrico que es la intersección topológica (operación AND) de dos geometrías especificadas. En la siguiente figura se muestra un ejemplo gráfico de esta función.

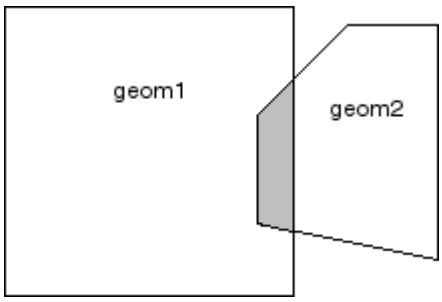


Figura 14: Intersección de dos geometrías.

Formato

```
SDO_GEOM.SDO_INTERSECTION(
     geom1 IN MDSYS.SDO_GEOMETRY,
     dim1 IN MDSYS.SDO_DIM_ARRAY,
     geom2 IN MDSYS.SDO_GEOMETRY,
     dim2 IN MDSYS.SDO_DIM_ARRAY
     ) RETURN MDSYS.SDO_GEOMETRY;
O también:
SDO_GEOM.SDO_INTERSECTION(
     geom1 IN MDSYS.SDO_GEOMETRY,
     geom2 IN MDSYS.SDO_GEOMETRY,
     tol IN NUMBER
     ) RETURN MDSYS.SDO_GEOMETRY;
```

Dónde los parámetros tienen las mismas características que los descritos en las funciones anteriores.

- **SDO_GEOM.SDO_BUFFER**

Esta función construye un buffer alrededor de un objeto geométrico especificado o un conjunto de objetos geométricos.

Por ejemplo, se puede usar esta función para crear un buffer de medio kilómetro alrededor de una localización especial. La geometría del buffer será un círculo alrededor de la localización de medio kilómetro de radio.

Asimismo, un buffer alrededor de una línea en L será el área de la combinación de dos óvalos aplanados, uno por cada segmento de línea. Algunos ejemplos de buffer en diferentes formas geométricas básicas se pueden ver a continuación:

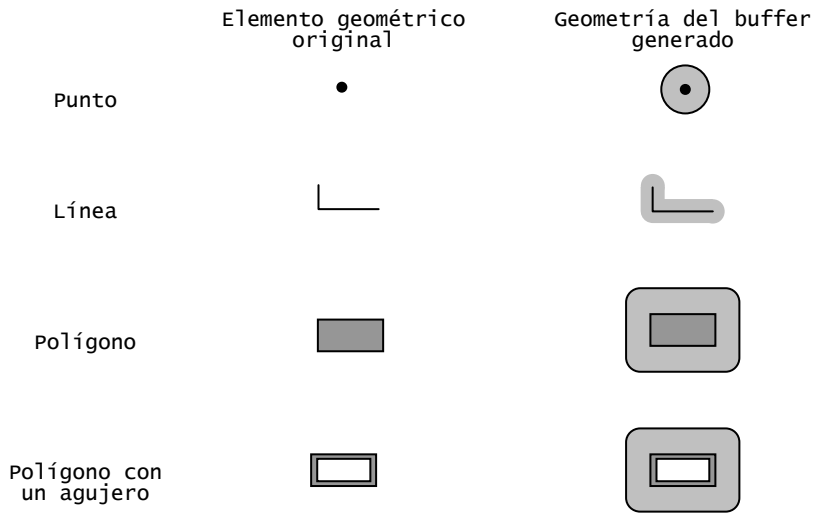


Figura 15: Objetos geométricos simples y geometría de sus buffers generados

Formato

```
SDO_GEOM.SDO_BUFFER(
      geom IN MDSYS.SDO_GEOMETRY,
      dim IN MDSYS.SDO_DIM_ARRAY,
      dist IN NUMBER
      [, params IN VARCHAR2]
```

```
    ) RETURN MDSYS.SDO_GEOMETRY;
```

O también:

```
SDO_GEOM.SDO_BUFFER(
    geom IN MDSYS.SDO_GEOMETRY,
    dist IN NUMBER,
    tol IN NUMBER
    [, params IN VARCHAR2]
    ) RETURN MDSYS.SDO_GEOMETRY;
```

Dónde los parámetros siguen las mismas características que las anteriores funciones, salvo el parámetro opcional `params` que especifica dos parámetros: `unit`, que especifica las unidades en las que se quiere calcular el buffer, y `arc_tolerance`. Este parámetro es requerido si la geometría es geodésica (esto es, si el SRID es 8307 o 8265). En el espacio geodésico, los arcos no están permitidos. En cambio, se pueden representar usando una línea recta de aproximación. El parámetro arc_tolerance especifica la máxima distancia entre un arco y su línea de aproximación.

Hay que tener en cuenta que:

- El arc_tolerance siempre debe ser mayor que la tolerancia para la geometría.
- La tolerancia se especifica en unidades de metro para los datos geodésicos. El arc_tolerance, sin embargo, siempre se representa en las unidades especificadas en el `unit=<valor_cadena>`.
- Si se especifica el parámetro `unit` las unidades se aplican a la distancia del buffer y al `arc_tolerance`, pero no así a las unidades del

parámetro tolerancia que seguirá siendo en metros al ser geodésica su geometría.

- **SDO_RELATE**

Una vez creados los buffers alrededor de los elementos geométricos, se pueden efectuar análisis de relaciones para identificar objetos dentro de regiones o zonas. Esta operación se realiza usando el operador SDO_RELATE o la función RELATE del paquete SDO_GEOM.

Formato

```
SDO_GEOM.RELATE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     dim1 IN MDSYS.SDO_DIM_ARRAY,
     mask IN VARCHAR2,
     geom2 IN MDSYS.SDO_GEOMETRY,
     dim2 IN MDSYS.SDO_DIM_ARRAY
     ) RETURN VARCHAR2;
O también:
SDO_GEOM.RELATE(
     geom1 IN MDSYS.SDO_GEOMETRY,
     mask IN VARCHAR2,
     geom2 IN MDSYS.SDO_GEOMETRY,
     tol IN NUMBER
     ) RETURN VARCHAR2;
```

- Dónde los parámetros especifican lo mismo que en las funciones anteriores, salvo el parámetro `mask`, que provee de una lista de relaciones de las cuales se puede tomar uno o varios de los siguientes valores: DETERMINE, INSIDE, COVEREDBY, COVERS, CONTAINS, EQUAL, OVERLAPBDYJOINT, OVERLAPBDYINTERSCT, ON, TOUCH, ANYINTERACT, DISJOINT.

- Si se especifican varios valores, la función devuelve el nombre de la relación que se cumple para las dos geometrías, es decir, la que devuelva TRUE.

4.4 SQL*Plus

SQL*Plus es una herramienta que permite establecer conexión con el servidor de base de datos Oracle y comenzar el trabajo con la información de la base de datos. Los comandos SQL para el trabajo con la base de datos se introducen en la línea de comandos.

Para acceder a esta aplicación se debe poseer una cuenta y una contraseña en un servidor de base de datos Oracle 10g, que se introducirán antes de comenzar a utilizar la herramienta en la siguiente ventana:

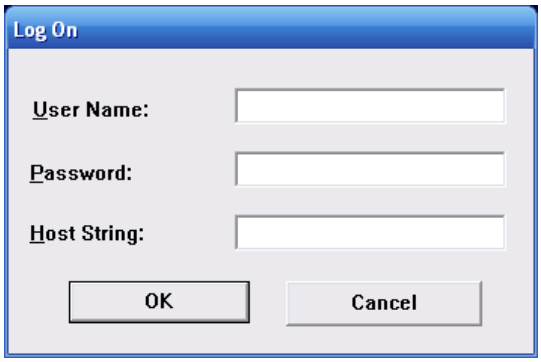


Figura 16: Acceso a SQL*Plus

Una vez comprobado la cuenta y la contraseña, el programa nos presentará la siguiente pantalla:

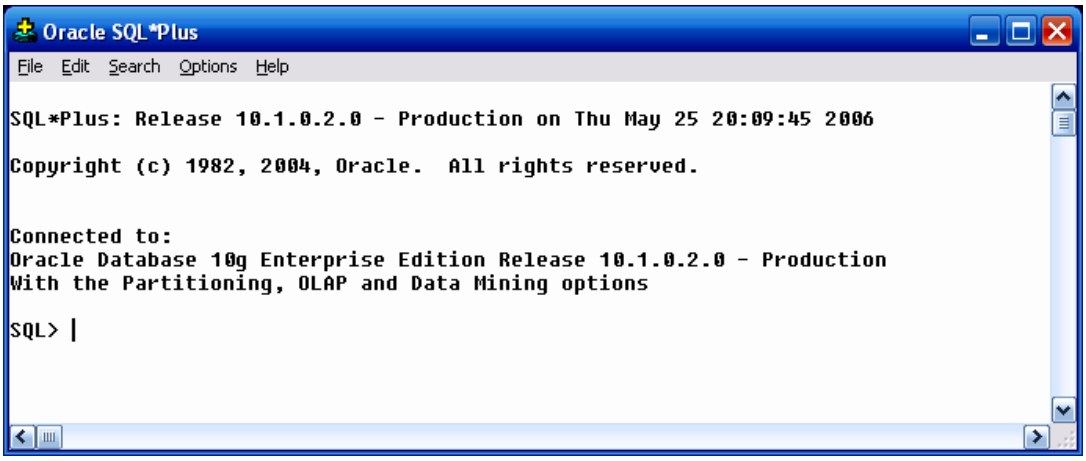


Figura 17: Oracle SQL*Plus

Se puede utilizar SQL*Plus para manejar el lenguaje SQL y su extensión PL/SQL, que permite enlazar varios comandos SQL a través de procedimientos lógicos. SQL*Plus permite manipular comandos SQL y bloque PL/SQL, y ejecutar tareas adicionales entre las que destacan las siguientes:

- Introducir, editar, almacenar, recuperar y ejecutar comandos SQL y bloques PL/SQL.
- Dar formato, ejecutar cálculos, almacenar e imprimir resultados de consultas en formato de informe.
- Listar definiciones de columnas para cualquier tabla.
- Acceder y copiar datos entre bases de datos SQL.
- Enviar mensajes a usuarios y recibir respuestas.
- Ejecutar tareas de administración de bases de datos.

4.5 Aplicación shp2sdo

Esta sencilla herramienta de Oracle, ha permitido exportar las capas generadas desde un programa de edición de geometrías a un archivo .sql que contiene la definición de tabla, junto con otros archivos para la carga de información geográfica y alfanumérica.

Esta aplicación se ejecuta por línea de comandos bajo las plataformas Windows NT, Sun Solaris y Linux. Dependiendo de cuál sea, la sintaxis para invocar al programa modifica ligeramente.

Si se está utilizando una versión de Oracle superior a la 9i, y una plataforma Windows NT, el formato para ejecutar *shp2sdo* será el siguiente:

```
shp2sdo.exe states states -g geom -d -x (-180,180) -y (-90,90) -s 8307
-t 0.5 -v
```

Donde:

states

El archive shape a convertir (puede ser archivos con la extensión .dbf, .shp, y .shx).

states

El nombre que se dará a la table a crear en Oracle.

-**g Geom**

Geom el nombre que se le va a dar a la columna de tipo SDO_GEOMETRY en la tabla a crear.

-**d**

Coloca los datos en el archivo de control generado por la aplicación.

-**x**

Los límites de la primera dimensión en el sistema de coordenadas. Éstos serán (-180,180).

-**y**

Los límites de la segunda dimensión en el sistema de coordenadas. Éstos serán (-90,90).

-**s**

El SRID (Spatial Reference System ID) utilizado. Por defecto se asigna el 8307.

-t

Margen de error permitido. Por defecto 0.5.

-v

Modo de salida de la aplicación. En este caso muestra mensajes informativos de éxito/error.

Genera los siguientes archivos:

Archivos .sql

Generan la tabla en sintaxis de SQL con los tipos de datos que se han asignado en el programa de edición de geometrías (ArcView 3.3). También crea los metadatos para la tabla, cuyo formato se ha explicado en el apartado 4.3.1.2.

Archivos .ctl

Son los que contienen los datos que han sido insertados en ArcView y que serán introducidos en la tabla. Se cargarán con otra herramienta de Oracle, SQL Loader, que se explicará más adelante.

4.6 SQL Loader

Se trata de otra herramienta de Oracle, con la que podemos cargar los datos después de haber creado la tabla con los archivos generados por *shp2sdo.*

También se ejecuta por línea de comandos, y su sintaxis es bastante sencilla:

```
sqlldr usuario/contraseña archivo.ctl
```

De esta forma, cargaremos los datos introducidos en ArcView en la tabla creada en la base de datos Oracle. Inmediatamente se generarán dos tipos de archivos por cada tabla geométrica insertada:

Archivos .bad

Son aquellas geometrías que no se han insertado debido a alguna incorrección en sus coordenadas, como por ejemplo, no haberse cerrado correctamente o no haber respetado los límites de las dimensiones.

Archivos .log

En ellos se especifica el número de aciertos o errores que la carga de datos haya podido ocasionar. En este caso, los errores pueden ser no sólo de tipo geométrico, sino también de tipo constraint (violación de clave primaria, clave foránea no encontrada, checks, etc.)

4.7 ESRI ArcView 3.3

ArcView es un Sistema de Información Geográfica (GIS) de escritorio completo y de fácil uso diseñado para explorar, analizar, visualizar, y consultar datos geográficos. ArcView proporciona a los analistas, proyectistas, usuarios especializados, y a la población general una interfaz amigable e intuitiva.

Descripción de ArcView

ArcView está disponible para computadores basados en MS-Windows, Macintosh, y UNIX. Opera en un modo personal o en un ambiente de red. ArcView puede comunicarse también con otras aplicaciones, tales como Sistemas de Posicionamiento Global (GPS), o utilizar ARC/INFO como un servidor para desempeñar un análisis sofisticado. Dispone un ambiente de desarrollo y lenguaje de programación Orientado a Objetos, *Avenue*, que se utiliza para extender las capacidades básicas de ArcView y personalizar aplicaciones específicas.

Características más importantes de ArcView 3.1

- Acceso integral de datos
- Integra mapas, cuadros, tablas y gráficos.
- Enlaces geográficos con todos los formatos de datos soportados
- Integra imágenes, CAD, datos de mapa, tablas y bases de datos SQL.
- Permite mostrar los símbolos en formatos diferentes (densidad de puntos, tamaño graduado, color graduado, cuadros, y color único).
- Importa y exporta formatos de imagen tales como TIFF, WMF, BMP, PICT, EPS, y JPEG.
- Realiza consultas espaciales.
- Utiliza el mouse o un digitalizador para ingresar/editar los datos

Con esta herramienta se ha logrado digitalizar por capas una imagen *.jpg* para posteriormente poder importarse a una base de datos espacial en Oracle.

El proceso de digitalización de cada capa así como los resultados obtenidos se detallarán en el apartado Método de resolución.

El programa devuelve varios archivos que serán utilizados a la hora de crear la tabla e introducir sus datos, así como de controlar la ejecución a través de un log.

Archivos .shp

Contienen las coordenadas que se han asignado a cada objeto geométrico. Son totalmente necesarios para transformar capa editada en ArcView a formato entendible por Oracle Spatial.

Archivos .dbf

Son archivos data base file. Contienen únicamente los datos alfanuméricos que se han introducido en ArcView.

Archivos .shx

Son archivos de tipo de letra específicos.

Estos archivos son imprescindibles para poder utilizar la aplicación shp2SDO e importar las geometrías y sus datos a una base de datos.

5. MÉTODO DE RESOLUCION

En este apartado se detallará paso a paso cómo se ha realizado el proceso de análisis, diseño e implementación de la base de datos, así como de los subprogramas, disparadores y consultas realizadas sobre la misma.

Se justificará el dominio escogido para la representación de la información, el proceso de digitalización de una imagen, la carga de los datos sobre Oracle y la extracción de información que se ha realizado sobre el sistema.

5.1 Requisitos

Para una correcta y cómoda utilización de la aplicación, se recomendarán una serie de requisitos software, hardware y de usuario.

5.1.1 Requisitos hardware

Se recomienda que la máquina en la que se vaya a instalar una aplicación de procesamiento de imágenes y una base de datos Oracle versión 10/11g, cumpla las siguientes características:

- Procesador 550Mhz.
- 1 Gb de memoria RAM.
- Memoria Virtual El doble de tamaño que la RAM (2Gb)
- Espacio en disco:
    - Instalación Básica: 4.55Gb
    - Instalación Avanzada: 4.92Gb
- Adaptador de vídeo de 256 colores.
- Tarjeta aceleradora de vídeo (recomendado).
- Monitor de 19 pulgadas (mínimo recomendado).
- Un procesador para compresión de imágenes (recomendado).

5.1.2 Requisitos software

Instalación de Oracle 11g en Windows 32-Bit. Sólo se tocaran los puntos más importantes para dejar corriendo los servicios básicos de Oracle.

Arquitectura de procesador	Intel (x86), AMD64, o Intel EM64T
Sistema Operativo	• Windows 2000 Service Pack 1 o superior • Windows Server 2003 o Superior • Windows XP Professional • Windows Vista Businnes o Superior *Windows NT no está soportado
Compilador	• ACUCOBOL-GT version 6.2 • Micro Focus Net Express 5.0
Protocolo de red	• TCP/IP ó TCP/IP con SSL

Navegadores web soportados:

- Netscape Navigator 7.2 o Superior
- Mozilla version 1.7 o Superior
- Microsoft Internet Explorer 6.0 SP2 o Superior
- Firefox 1.0.4 o Superior

5.1.3 Instalación bajo UNIX

Oracle y ESRI ArcGIS pueden instalarse bajo otro sistema operativo tipo UNIX como Linux y Sun Solaris, siendo el paquete ArcGIS mucho más selectivo en cuanto a requisitos. A continuación se detallan los indispensables para que éste último pueda instalarse y ejecutarse correctamente.

Arquitectura de procesador	Arquitectura x86.
Sistema Operativo	• Red Hat Linux v.4/5. • SUSE Linux 10. • SUN Solaris.
Kernel	• Red Hat Linux 4: AS/ES/WS 4.0 Kernel version 2.6.9 o superior. • Red Hat Linux 5: AS/ES/WS 5.0 Kernel version 2.6.18 o superior. • SUSE 10: Kernel version 2.6.16 o superior.
Protocolo de red	• TCP/IP • TCP/IP con SSL

5.1.4 Requisitos de usuario

Es recomendable que el usuario de estas aplicaciones disponga de conocimiento teórico y práctico en bases de datos relacionales, así como un concepto introductorio como mínimo de alguna aplicación de procesamiento de imágenes, como Photoshop, AutoCAD, ArcGIS, etc. También debería conocer alguna técnica de modelado de bases de datos para poder interpretar correctamente el diseño escogido para la misma.

5.2 Dominio de la base de datos

El dominio escogido para representar la información en este proyecto han sido los datos geográficos de una ciudad, de la que conocemos su ubicación real en el mapa y con la que hemos asociado un sistema de referencia UTM y una zona específica dentro del cuadrante en que está colocada (Ver Anexo Coordenadas UTM, apartado 2).

De esta imagen, se han seleccionado los barrios, que cuentan con zonas rurales, industriales y urbanas (manzanas) y a su vez están asociados con sus respectivas calles. También se ha delimitado gráficamente el perímetro de la ciudad.

Respecto a la información alfanumérica, se han registrado datos de las personas que viven en esta ciudad (datos personales, estilos de vida, bienes inmuebles, etc.) estando asociadas a una determinada vivienda dentro de una calle y, por lo tanto, de un barrio dentro de una zona urbana. También se han recogido las empresas que están ubicadas en cada barrio, de las que se conoce su actividad empresarial y su ubicación, que puede ser en un polígono industrial o en una zona urbana.

Con todos estos datos se procederá a hacer un estudio que proporcione información diversa como áreas, densidad de población, distancias, porcentajes de cada zona, etc., sin descuidar información adicional de cada habitante y cada empresa.

Esta información trata de reflejar una aproximación a lo que podríamos encontrarnos en una ciudad determinada, por lo que no se trata de datos reales recogidos con alguna técnica destinada para tal fin.

5.3 Diseño de la Base de datos

Una vez establecido el dominio de la base de datos, se procede a realizar un modelo conceptual, el modelo Entidad/Interrelación del mismo, explicando las principales entidades y relaciones, para posteriormente realizar un modelo lógico y por último uno físico.

5.3.1 Modelo conceptual: modelo E/R

Es el primer paso a abordar para la construcción de lo que será la base de datos espacial. Este modelo abstracto nos permite expresar aquellas entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. En el siguiente diagrama se presenta el modelo E/R:

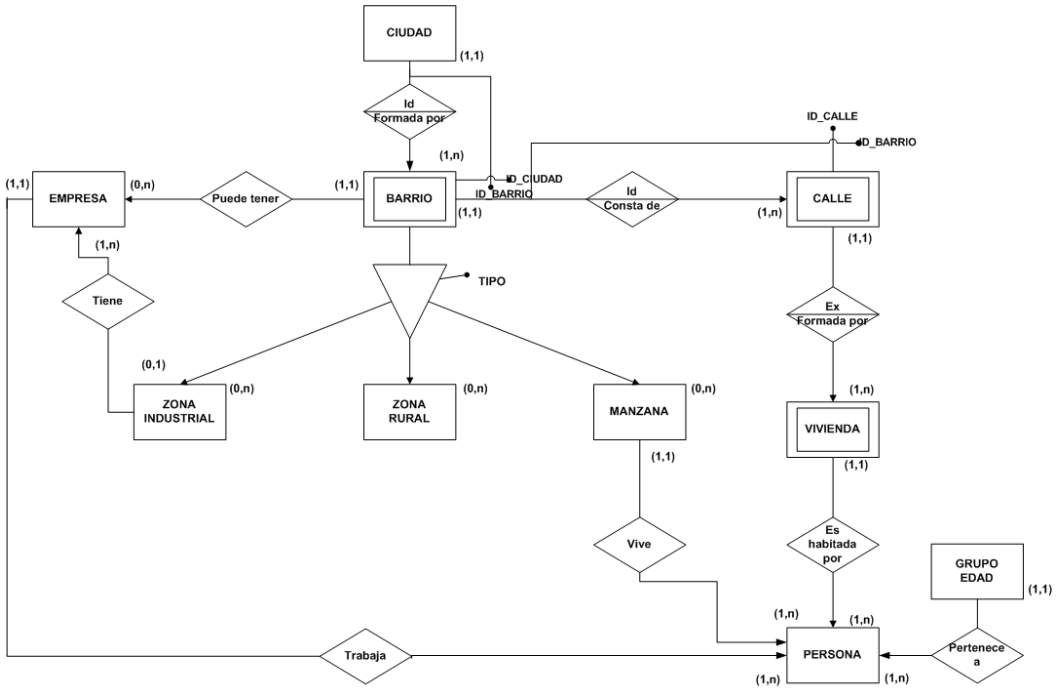


Figura 18. Diagrama E/R.

Supuestos semánticos asociados al diagrama E/R

SS1. La entidad BARRIO depende de la entidad CIUDAD en identificación, ya que se ha considerado que si no existe una ciudad, no pueden existir barrios y además, en caso de que exista, una ciudad puede contener barrios que se llamen de la misma forma en otra ciudad.

SS2. La entidad CALLE depende en identificación de la entidad BARRIO, ya que si no existe un barrio, no puede existir una calle, según la cardinalidad que une a ambas. Además, un barrio de una determinada ciudad puede tener calles que se llamen igual que otro barrio de otra ciudad.

SS3. La entidad VIVIENDA depende únicamente en existencia de la entidad CALLE porque si no existe una calle, no podrá haber una vivienda en ella. No se

ha asumido una dependencia en identidad porque una vivienda se identificará por su clave primaria de manera unívoca; Además, tiene sentido suponer que una vivienda es única en cualquier parte del mundo.

SS4. En todas las viviendas registradas residirá al menos una persona.

SS5. El atributo "Tipo" de la entidad BARRIO, toma valores en el dominio tipo lugar= {Manzanas, Zona Rural, Zona Industrial}.

SS6. El dominio del atributo "segunda propiedad" en la entidad PERSONA, toma valores en el dominio valores = {S, N}.

SS7. El atributo "régimen vivienda" en la entidad PERSONA, toma valores en el dominio tipo_reg_vivienda = {Propia, En alquiler}.

SS8. El atributo "situación laboral" en la entidad PERSONA, toma valores en el dominio tipo_sit_laboral = {En paro, Trabajando}.

Supuestos semánticos no recogidos en el E/R

SS9. En las viviendas registradas, residirá al menos una persona.

SS10. Las personas que no trabajen en Getafe, tendrán un id de empresa nulo.

SS11. Si el atributo ""Situación_laboral" de la tabla PERSONA es "EN PARO", el atributo "Ciudad_trabajo" deberá ser nulo.

SS12. Si el atributo ""Situación_laboral" de la tabla PERSONA es "EN PARO", el atributo "id_empresa_fk" deberá ser nulo.

5.3.2 Transformación al modelo físico-lógico

Una vez terminado el modelo E/R, se procede a la implementación del modelo físico-lógico de la base de datos, respetando las reglas establecidas para ello. Los supuestos también sufrirán esta transformación, que darán como resultado una serie de restricciones que tiene que cumplir este diseño.

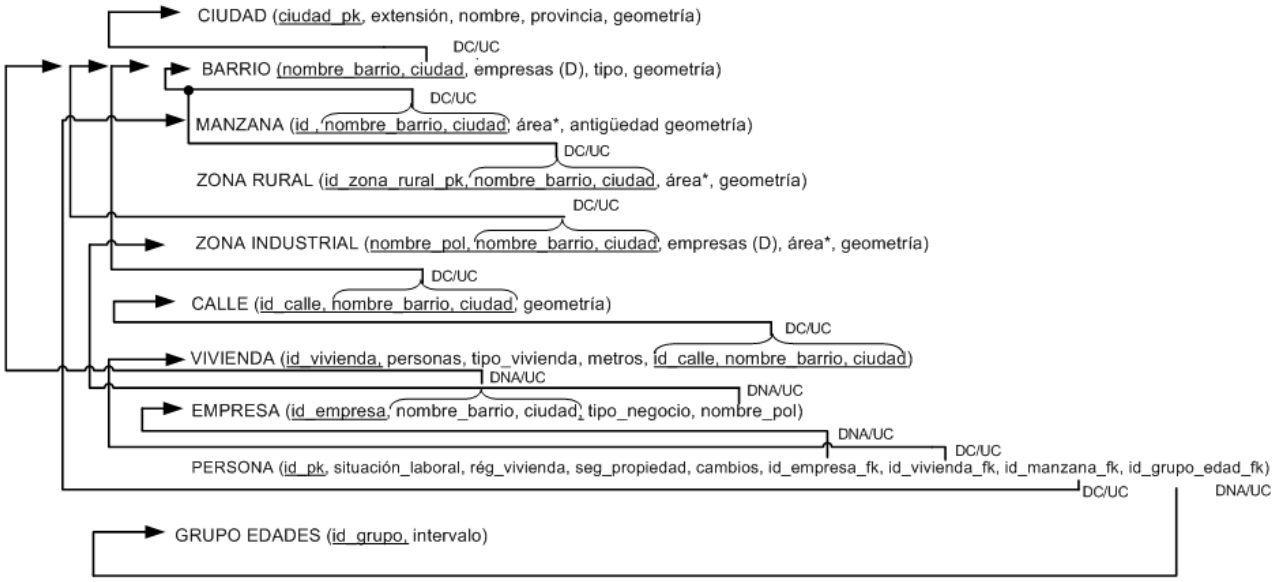


Figura 19. Modelo relacional.

Transformación de los supuestos semánticos asociados al modelo E/R

SUPUESTO SEMÁNTICO E/R	TRANSFORMACIÓN GRAFO RELACIONAL
SS1. La entidad BARRIO depende de la entidad CIUDAD en identificación, ya que se ha considerado que si no existe una ciudad, no pueden existir barrios y además, en caso de que exista, una ciudad puede contener barrios que se llamen de la misma forma en otra ciudad.	Propagación de clave principal de la entidad CIUDAD hacia la entidad BARRIO.
SS2. La entidad CALLE depende en identificación de la entidad BARRIO, ya que si no existe un barrio, no puede existir una calle, según la cardinalidad que une a ambas. Además, un barrio de una determinada ciudad puede tener calles que se llamen igual que otro barrio de otra ciudad.	Propagación de clave principal de la entidad BARRIO hacia la entidad CALLE.
SS3. La entidad VIVIENDA depende únicamente en existencia de la entidad CALLE porque si no existe una calle, no podrá haber una vivienda en ella. No se ha asumido una dependencia en identidad porque una vivienda se identificará por su clave primaria de manera unívoca.	Restricción de integridad DC/UC para la relación entre VIVIENDA y CALLE.
SS4. El atributo "Tipo" de la entidad BARRIO, toma valores en el dominio tipo_lugar= {Manzanas, Zona Rural, Zona Industrial}.	CHECK sobre la tabla BARRIO.

SS5. En las viviendas registradas, siempre residirá al menos una persona.	CHECK sobre la tabla PERSONA.
SS6. El dominio del atributo "segunda propiedad" en la entidad PERSONA, toma valores en el dominio valores = {S, N}.	CHECK sobre la tabla PERSONA.
SS7. El atributo "régimen_vivienda" en la entidad PERSONA, toma valores en el dominio tipo_reg_vivienda = {Propia, En alquiler}.	CHECK sobre la tabla PERSONA.
SS8. El atributo "situación laboral" en la entidad PERSONA, toma valores en el dominio tipo_sit_laboral = {En paro, Trabajando}.	CHECK sobre la tabla PERSONA.
SS9. En las viviendas registradas, residirá al menos una persona.	CHECK sobre la tabla VIVIENDA.
SS10. Las personas que no trabajen en Getafe, tendrán un id de empresa nulo**.**	ASERCIÓN sobre la tabla PERSONA.
SS11. Si el atributo "'"Situación_laboral" de la tabla PERSONA es "EN PARO", el atributo "id_empresa_fk" deberá ser nulo.	ASERCIÓN sobre la tabla PERSONA.
SS12. El atributo empresas en la entidad BARRIO es un campo calculado.	Disparador que calcula el número de empresas por cada barrio.

5.3.3 Modelo físico

Una vez diseñado el modelo relacional o físico-lógico, pasamos a desarrollar el modelo físico, especificando los atributos y sus tipos en cada tabla, las restricciones tanto de tipo constraint (primary/foreing key, check) como de integridad referencial.

El lenguaje utilizado es SQL, en concreto DDL (Data Definition Language), que nos permitirá crear las tablas definidas en el modelo físico a un SGBD, en concreto Oracle v. 10g.

A continuación se define, de una manera teórica, la estructura de las tablas de la base de datos que posteriormente se traducirá en un script de SQL.

CIUDAD

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_CIUDAD	INTEGER	No	Clave principal de ciudad.
NOMBRE	VARCHAR2(25)	No	Nombre de la ciudad.
ÁREA	REAL	Sí	Área de la ciudad.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.

BARRIO

Atributo	Tipo de dato	NULLABLE?	Observaciones
BARRIO	VARCHAR2(25)	No	Clave principal de barrio.
CIUDAD	INTEGER	No	Clave principal de barrio y foreign key para ciudad.
TIPO	VARCHAR2(25)	Sí	Atributo discriminante.
EMPRESAS	INTEGER	Sí	Número de empresas en el barrio.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.

MANZANA

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID	INTEGER	No	Clave principal de Manzana.
BARRIO	VARCHAR2(25)	No	Clave principal de Manzana y foreign key para Barrio.
CIUDAD	INTEGER	No	Clave principal de Manzana y foreign key para Barrio.
ANTIGÜEDAD	INTEGER	Sí	Años de antigüedad del inmueble.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.

CALLE

Atributo	Tipo de dato	NULLABLE?	Observaciones
NOMBRE_CALLE	VARCHAR2(25)	No	Clave candidata de Calle.
ID_CALLE	INTEGER	No	Clave principal de Calle.
CIUDAD	INTEGER	No	Clave principal de Calle y foreign key para Barrio.
NOMBRE_BARRIO	VARCHAR2(25)	No	Clave principal de Calle y foreign key para Barrio.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.

ZONA INDUSTRIAL

Atributo	Tipo de dato	NULLABLE?	Observaciones
NOMBRE_POL	VARCHAR2(25)	No	Clave principal de la tabla.
NOMBRE_BARRIO	VARCHAR2(25)	No	Clave principal de la tabla y foreign key para Barrio.
CIUDAD	INTEGER	No	Clave principal de la tabla y foreign key para Barrio.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.
ÁREA	REAL	Sí	Área de la zona industrial.
EMPRESAS	INTEGER	Sí	Número de empresas.

ZONA RURAL

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_RUR_PK	INTEGER	No	Clave principal de la tabla.
NOMBRE_BARRIO	VARCHAR2(25)	No	Clave principal de la tabla y foreign key para Barrio.
CIUDAD	INTEGER	No	Clave principal de la tabla y foreign key para Barrio.
AREA	REAL	Sí	Área de la zona rural.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.

VIVIENDA

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_VIVIENDA	INTEGER	No	Clave principal de la tabla.
PERSONAS	INTEGER	No	Número de personas que residen.
CIUDAD	INTEGER	No	Foreign key para Calle.
AREA	REAL	Sí	Área de la zona rural.
GEOM	SDO_GEOMETRY	No	Coordenadas geométricas.
TIPO_VIVIENDA	VARCHAR2 (20)	No	{Piso, chalet, etc.}
METROS	REAL	No	Metros cuadrados.
ID_CALLE	INTEGER	No	Foreign key para Calle.
BARRIO	VARCHAR2(25)	No	Foreign key para Calle.

GRUPO EDAD

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_GRUPO	INTEGER	No	Clave principal de la tabla.
INTERVALO	VARCHAR2(10)	No	Intervalo de edad.

EMPRESA

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_EMPRESA_PK	INTEGER	No	Clave principal de la tabla.
NOMBRE_EMPRESA	VARCHAR2(25)	No	Clave alternativa.
TIPO_NEGOCIO	VARCHAR2(40)	No	Actividad empresarial.
BARRIO	VARCHAR2(25)	No	Foreign key para Barrio y Zona Industrial.
CIUDAD	INTEGER	No	Foreign key para Barrio y Zona Industrial.
NOMBRE_POL	VARCHAR2(20	Sí	Polígono en el que se encuentra.

PERSONA

Atributo	Tipo de dato	NULLABLE?	Observaciones
ID_PK	INTEGER	No	Clave principal de la tabla.
ID_VIVIENDA	INTEGER	No	Clave principal de la tabla y foreing key para Vivienda.
SITUACION_LABORAL	VARCHAR2(10)	No	{En paro, trabajando}
CIUDAD_TRABAJO	VARCHAR2(20)	No	Nombre de la ciudad de trabajo.
REGIMEN_VIVIENDA	VARCHAR2(11)	No	{Propia, en alquiler}.
SEGUNDA_PROPIEDAD	CHAR	Sí	{S, N}
CAMBIOS_VIVIENDA	INTEGER	Sí	Número de cambios de vivienda.
ID_EMPRESA	INTEGER	No	Id de la empresa registrada ó cero en caso de no estar en Getafe.
ID_GRUPO_EDAD	INTEGER	No	Foreign key para la tabla Grupo edad.
ID_MANZANA_FK	INTEGER	No	Manzana en la que vive.
CAMBIOS_VIVIENDA	INTEGER	Sí	Número de cambios de vivienda.
ID_EMPRESA	INTEGER	No	Id de la empresa registrada ó cero en caso de no estar en Getafe.

5.4 Metadatos e índices espaciales

Como ya se introdujo en el apartado 3, "Método de resolución", para la realización de operaciones con datos geométricos es necesario construir índices espaciales sobre las columnas de las tablas que posean tipos de datos SDO_GEOMETRY. Como paso previo, se deben de introducir los metadatos y

posteriormente, necesitamos migrar el formato del tipo de datos de la columna geométrica a la versión actual de la base de datos.

La siguiente secuencia muestra los pasos necesarios a seguir:

1. Creación de la tabla.
2. Inserción de los metadatos.
3. Migración de la columna GEOM a la versión actual de la base de datos.
4. Creación del índice espacial.

Se muestra un ejemplo con la tabla CIUDAD de la base de datos de este proyecto:

1. Creación de la tabla.

```
Drop table CIUDAD;

CREATE TABLE CIUDAD (

  ID_CIUDAD  INTEGER,

  NOMBRE     VARCHAR2(25) NOT NULL,

  AREA          REAL,

  PROVINCIA     VARCHAR2(50) NOT NULL,

  GEOM     MDSYS.SDO_GEOMETRY NOT NULL,

  CONSTRAINT CIUDAD_PK PRIMARY KEY (ID_CIUDAD));
```

2. Inserción de metadatos.

```
  DELETE FROM USER_SDO_GEOM_METADATA

  WHERE TABLE_NAME = 'CIUDAD' AND COLUMN_NAME = ‘GEOM’;


INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)

  VALUES ('CIUDAD', 'GEOM',

    MDSYS.SDO_DIM_ARRAY
```

```
        (MDSYS.SDO_DIM_ELEMENT('X', -180.000000000, 180.000000000,
0.500000000),

        MDSYS.SDO_DIM_ELEMENT('Y', -90.000000000, 90.000000000,
0.500000000)

      ), 8307);
```

3. Migración de la columna GEOM a la versión actual de la base de datos.

```
EXECUTE SDO_MIGRATE.TO_CURRENT('CIUDAD','GEOM');
```

4. Creación del índice espacial.

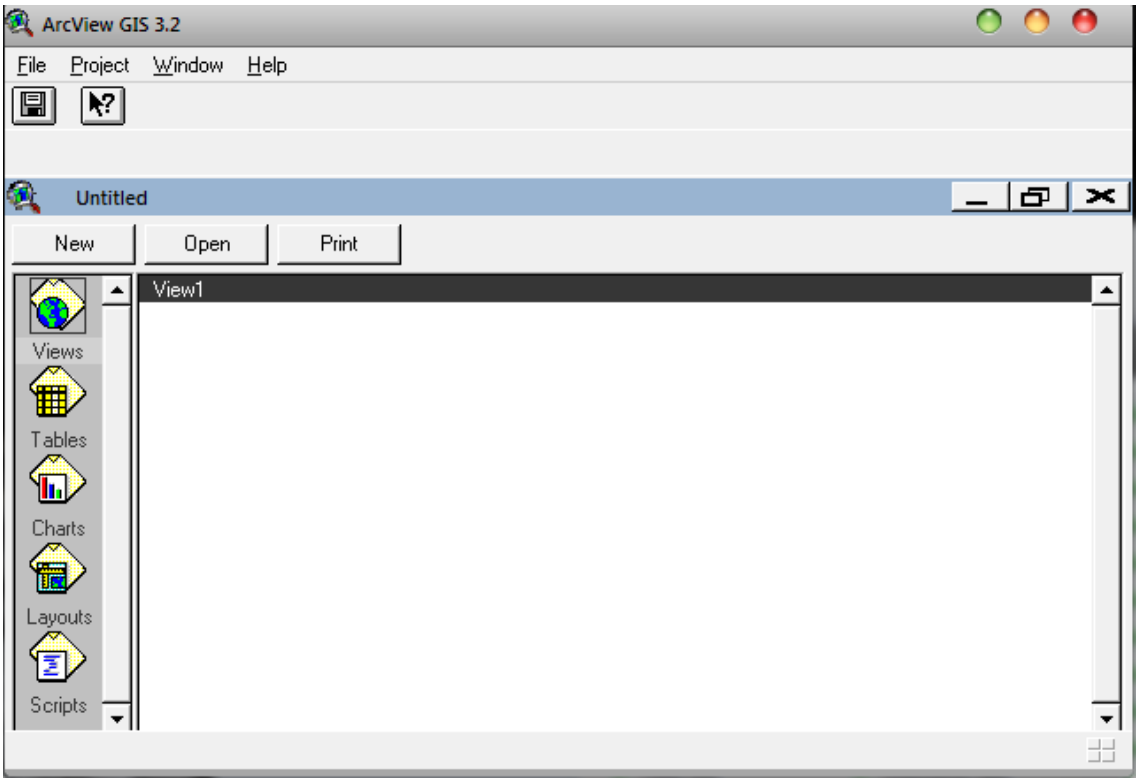
```
CREATE INDEX INDX_CIUDAD ON CIUDAD(GEOM) INDEXTYPE IS
MDSYS.SPATIAL_INDEX;
```

5.5 Digitalización de una imagen con ArcView 3.2

Como ya se ha introducido en apartados anteriores, ArcView es una herramienta de procesamiento de datos, donde podemos crear, modificar y consultar todo tipo de datos geográficos. Estos datos pueden ser importados desde ubicaciones externas, siempre y cuando se respete el formato para que el programa pueda leerlo correctamente.

En este caso, decidimos partir de la imagen de un mapa, obtenida a través de Google Maps, de la cual se quieren dibujar las figuras que van a formar la base de datos. A continuación se va a recrear la construcción de un proyecto en ArcView, explicando los pasos que se han llevado a cabo hasta obtener una imagen digitalizada por capas que se podrá importar a una base de datos.

Abrimos el programa y creamos un nuevo proyecto. Nos aparecerá a la izquierda un cuadro de herramientas que contiene Vistas, tablas, gráficos, scripts, etc. En este caso escogemos Vistas (Views).



Al hacer doble click sobre la vista que se ha creado, se abre una nueva ventana con opciones distintas; ahora es cuando podemos importar nuestra imagen.

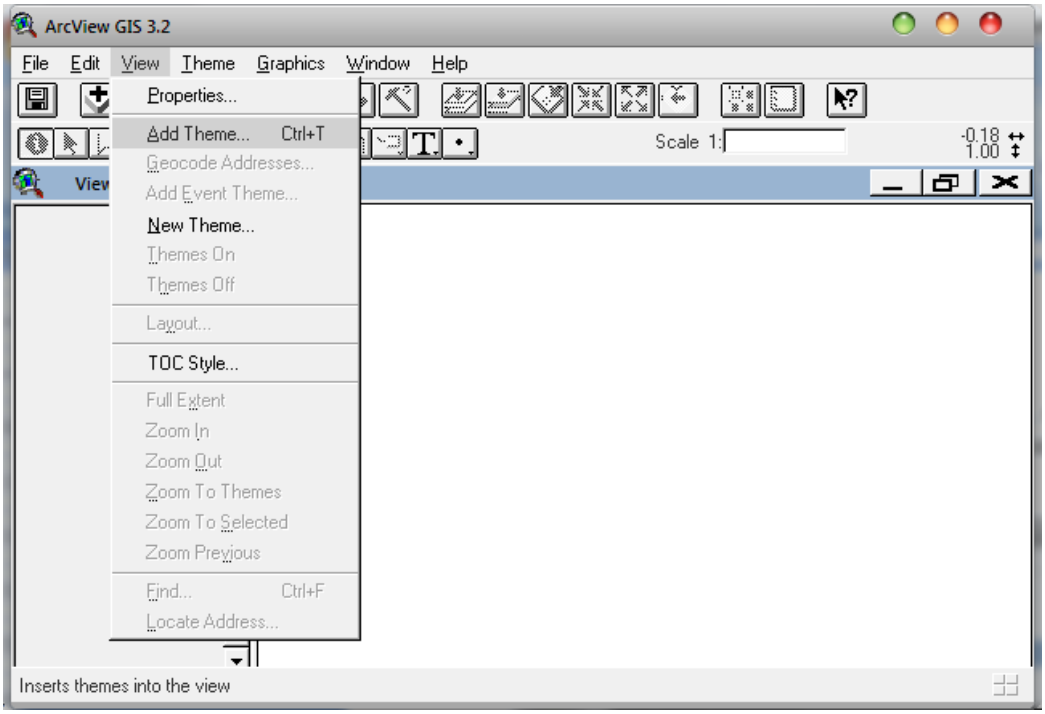


Figura 20: Importación de la imagen

Una vez seleccionada la ruta donde se encuentra la imagen, el programa la muestra dentro de la vista que estamos creando:

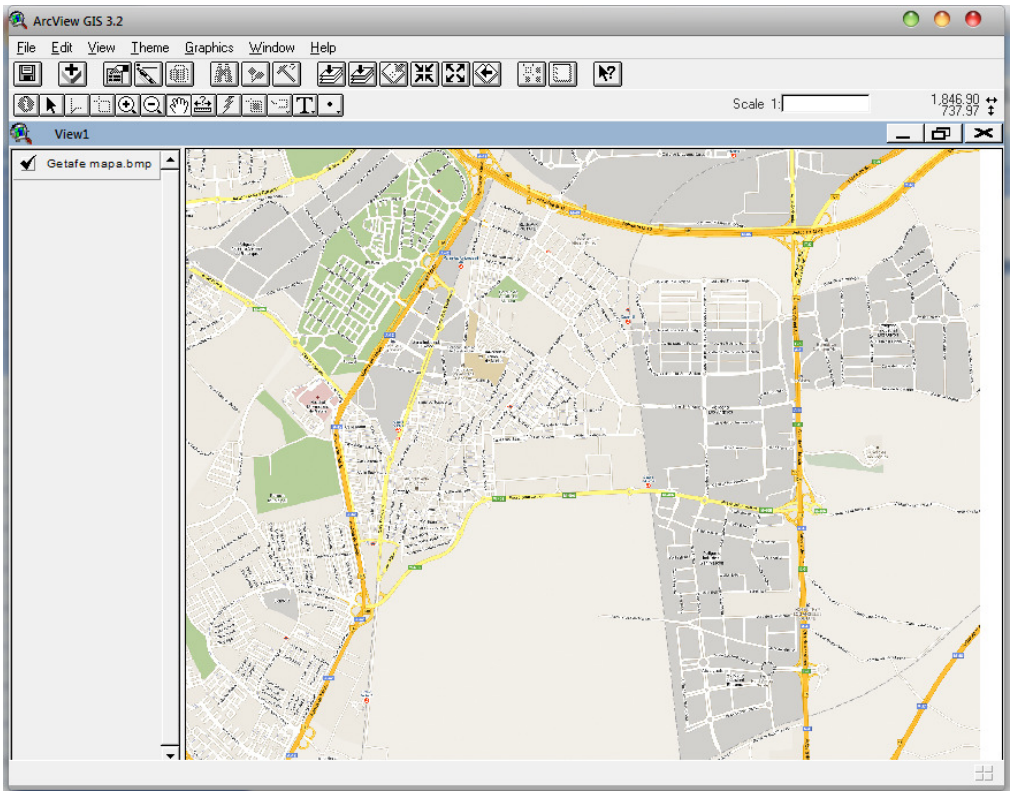


Figura 21: Imagen .bmp del mapa de Getafe

El mapa de la figura muestra una zona que contiene a la ciudad de Getafe, la cual vamos a comenzar a digitalizar para representar geométricamente los lugares que nos interese tener almacenados para el contexto de este trabajo.

Comenzamos añadiendo la primera capa, por ejemplo, la más externa, la que delimita el perímetro de la ciudad:

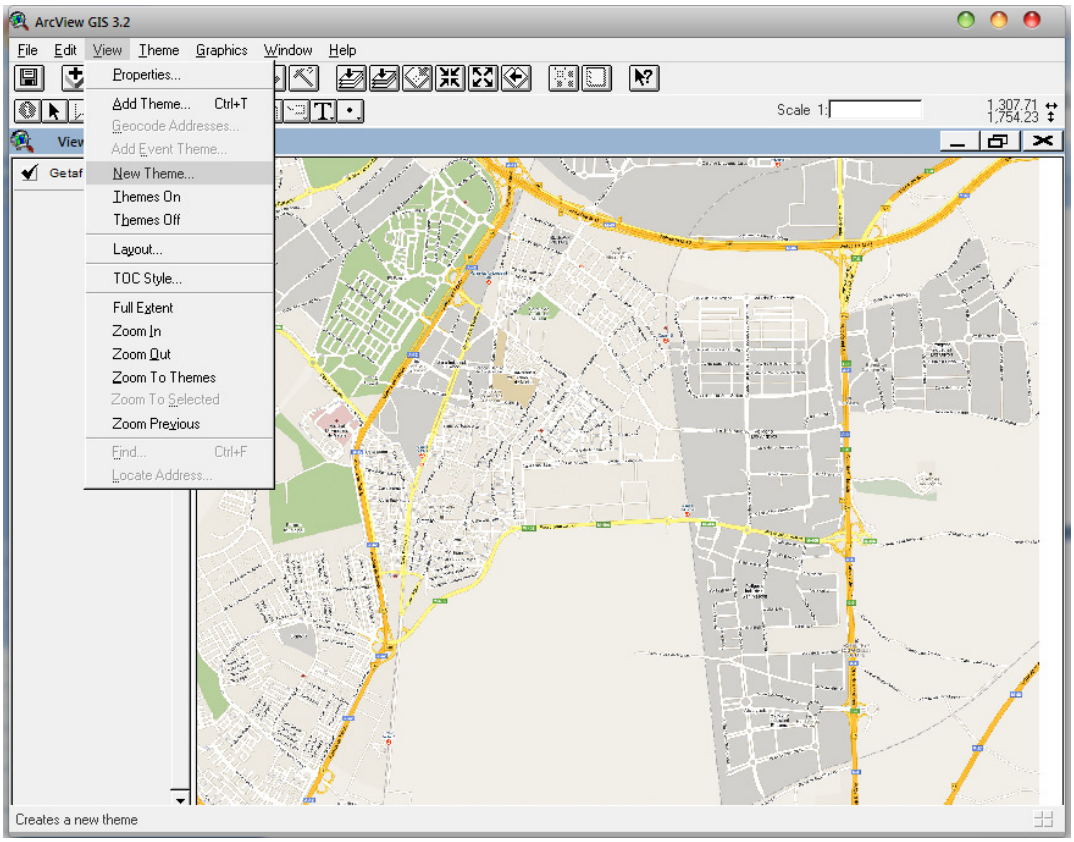


Figura 22. Adición de una nueva capa.

El programa solicitará qué tipo de geometría va a contener esta capa. Es importante saber que una capa no puede almacenar geometrías de distintos tipos. Para esta capa, seleccionamos Polígonos.

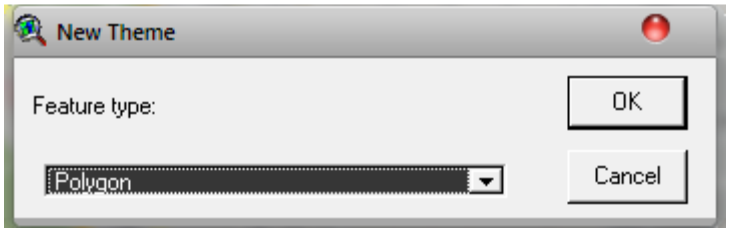


Figura 23. Selección del tipo de geometría.

Una vez escogida, deberemos guardar la capa en la ruta donde estemos almacenando nuestro nuevo proyecto. Nótese que ArcView le asigna la extensión .shp (Shape), que es uno de los formatos más conocidos y estándar para las geometrías. Lo guardamos con el nombre de ***Ciudad.shp***.

Podremos observar que ahora nos aparecen dos capas a la izquierda, la imagen del mapa y la capa que acabamos de crear, a la que se le asigna un color por defecto, que puede cambiarse si se desea. Comenzamos a digitalizar nuestra capa con las herramientas de dibujo que nos proporciona ArcView.

Dependiendo del tipo de geometría que hayamos escogido, nos aparecerán unas opciones u otras, en este caso, sólo podremos dibujar círculos, cuadrados y polígonos (regulares e irregulares). Una vez pintada, la imagen del perímetro de Getafe quedaría así a una escala de 10,937:

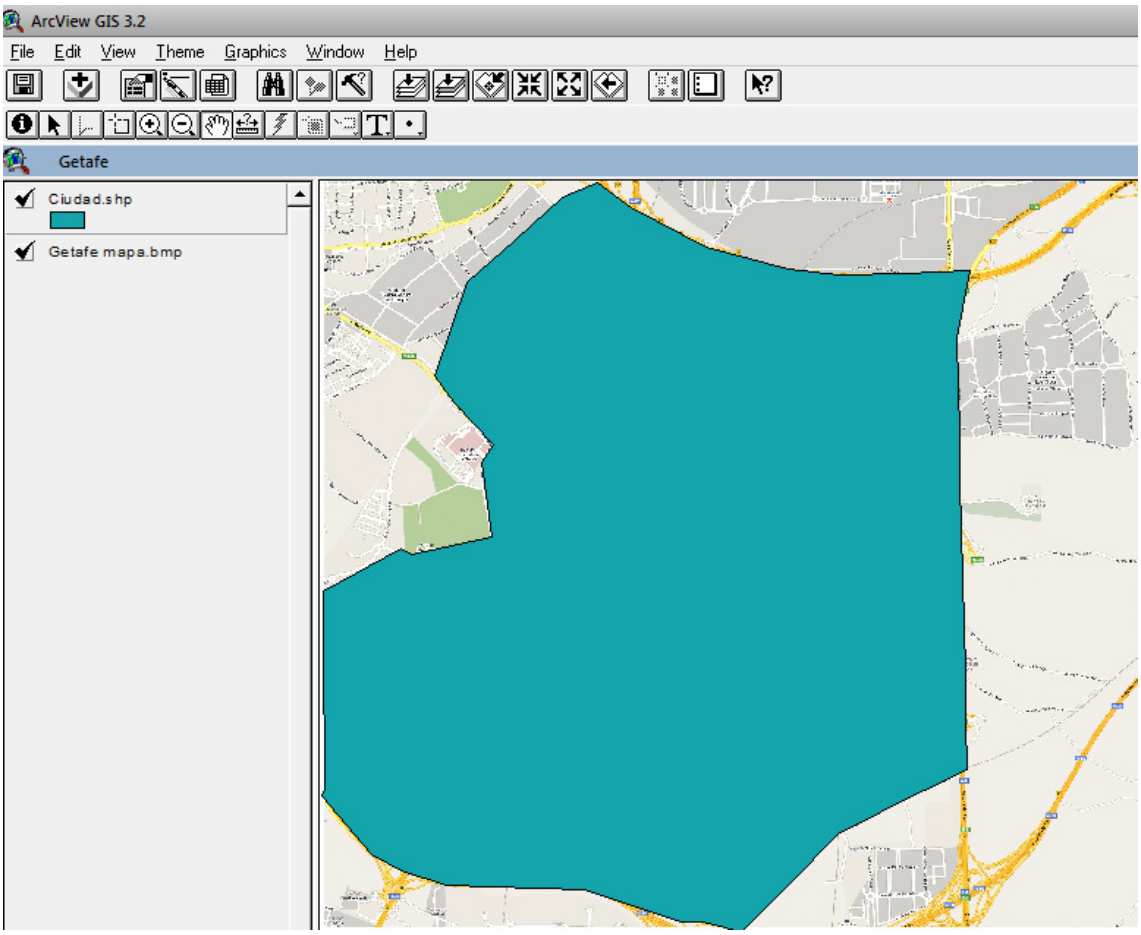


Figura 24. Digitalización del perímetro de Getafe.

Encima o debajo de esta capa, podemos seguir dibujando otras. En este caso, se hará encima, para poder visualizar correctamente todas al finalizar.

Seguimos con los barrios. Deshabilitamos temporalmente la capa (cliqueando sobre el nombre) y añadimos la nueva, que también serán polígonos. El resultado será el siguiente:

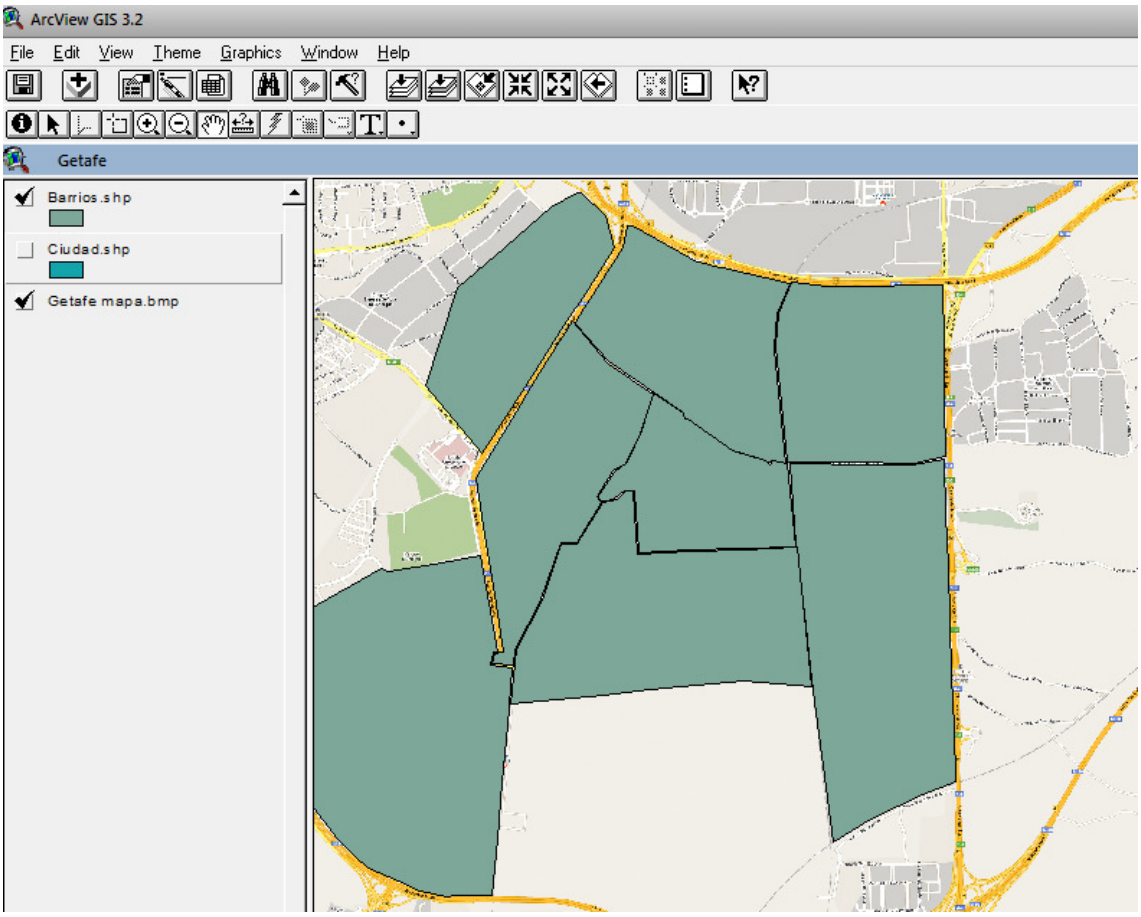


Figura 25. Digitalización de barrios.

A continuación, queremos dibujar lo que será la zona urbana de la ciudad. En este caso, utilizamos el concepto de *Manzana,* que va a representar a un conjunto de edificios delimitados por calles. Recordamos que, por simplicidad, se ha considerado que una Manzana sólo pertenece a una calle, por tanto, terminarán dentro del límite de ésta. La digitalización de las manzanas da como resultado la siguiente imagen:

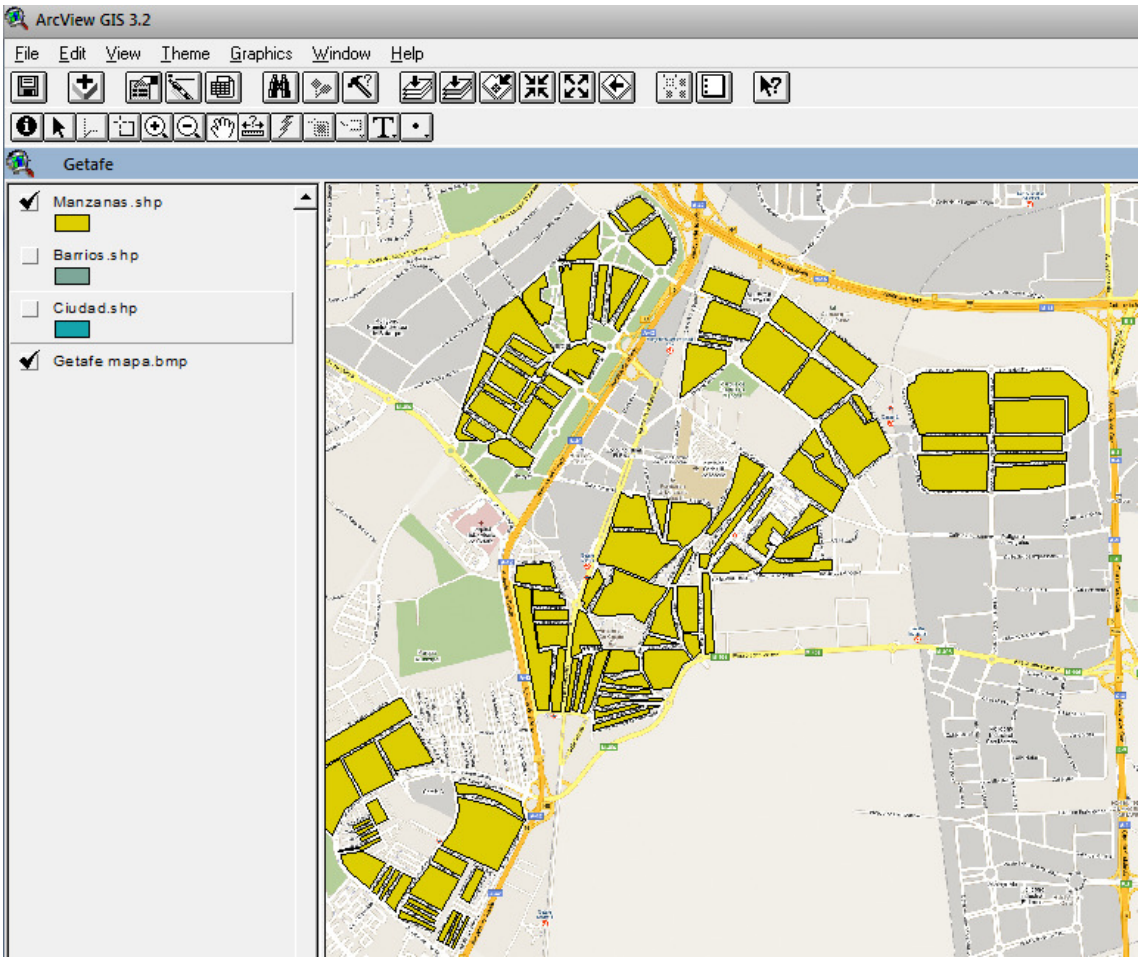


Figura 26. Digitalización de manzanas.

Nótese que las capas que vamos editando están acumuladas en el panel de la izquierda, pudiendo marcarse o desmarcarse en caso de querer visualizarlas o no. Como nuestro estudio también va a tener en cuenta las calles, zonas industriales y rurales de la ciudad, procedemos a su digitalización.

En el caso de las calles, no se trata de polígonos, sino de líneas, de las cuales podemos escoger su grosor y color. Las calles estarán dentro de la zona urbana o manzanas, por tanto, no se representarán las que estén en otras zonas como pueden ser la industrial o rural. El resultado es el siguiente:

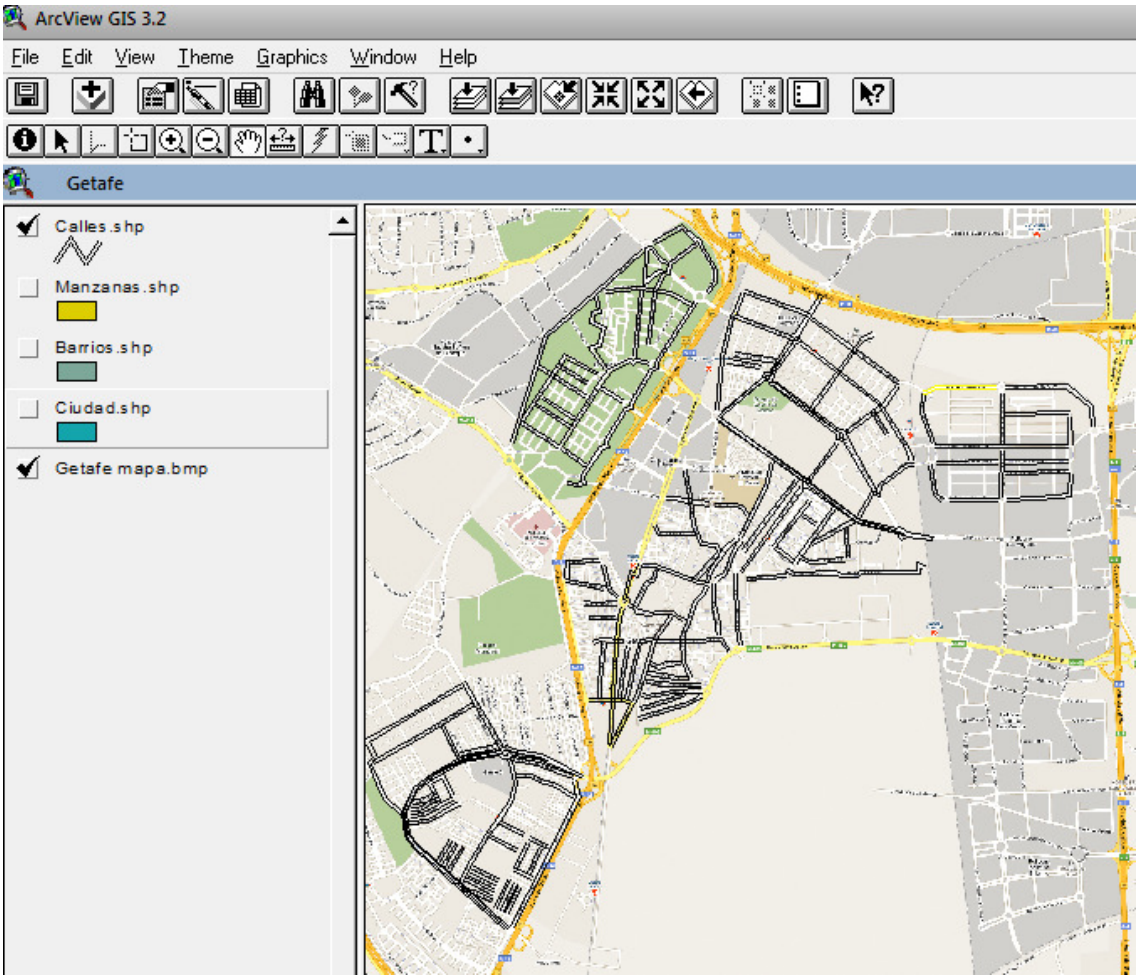


Figura 27. Digitalización de calles.

Por último, sólo quedan las zonas industriales y rurales. Sabemos, por consideraciones del diseño, que un barrio puede contener zonas urbanas, industriales y rurales al mismo tiempo. El resultado es el siguiente:

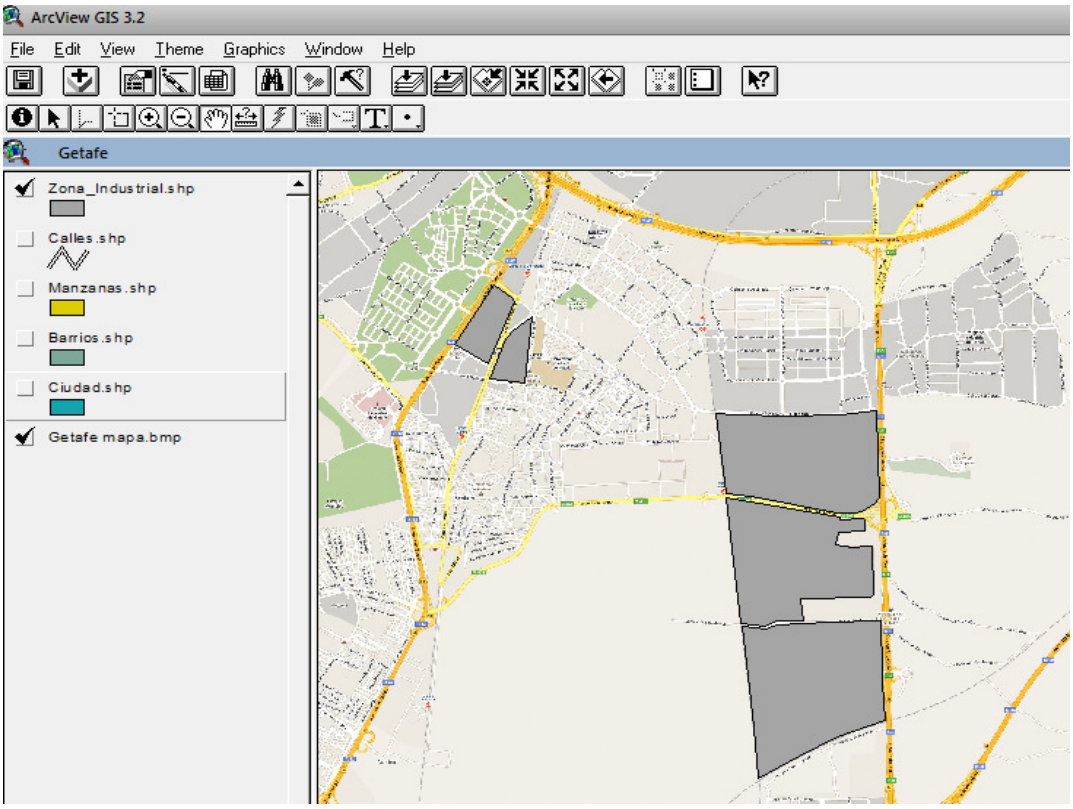


Figura 28. Digitalización de zona industrial.

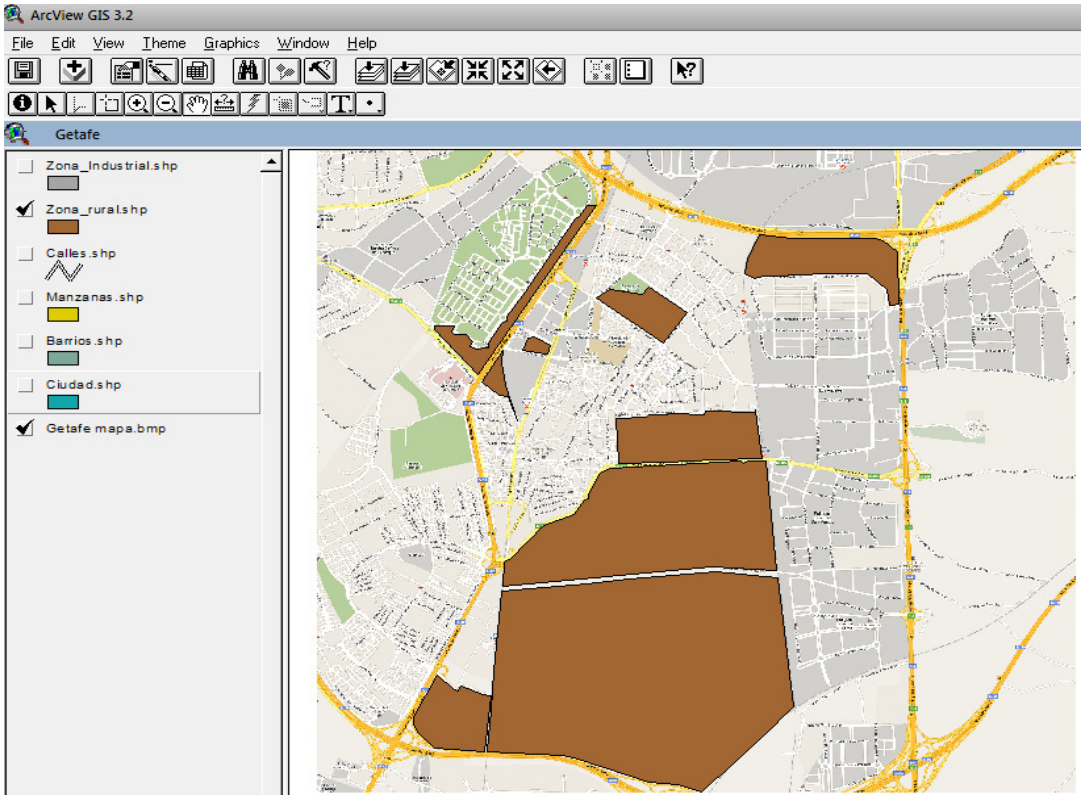


Figura 29. Digitalización de zona urbana.

Como se puede observar, la ciudad contiene más zona rural que industrial. Estas percepciones visuales se constatarán con cálculos geométricos posteriores sobre estos datos.

La imagen que reúne todas las capas activadas representa la información que vamos a almacenar en nuestra base de datos Oracle.

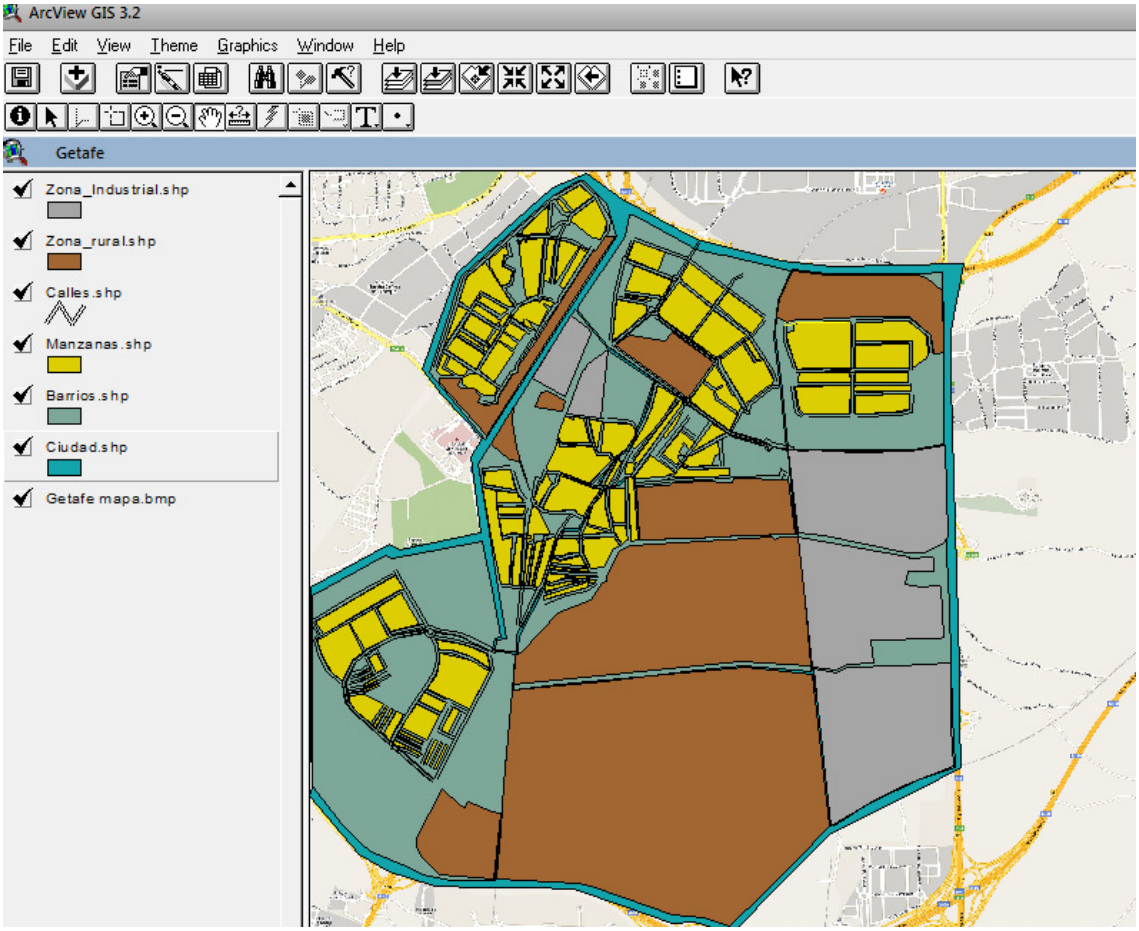
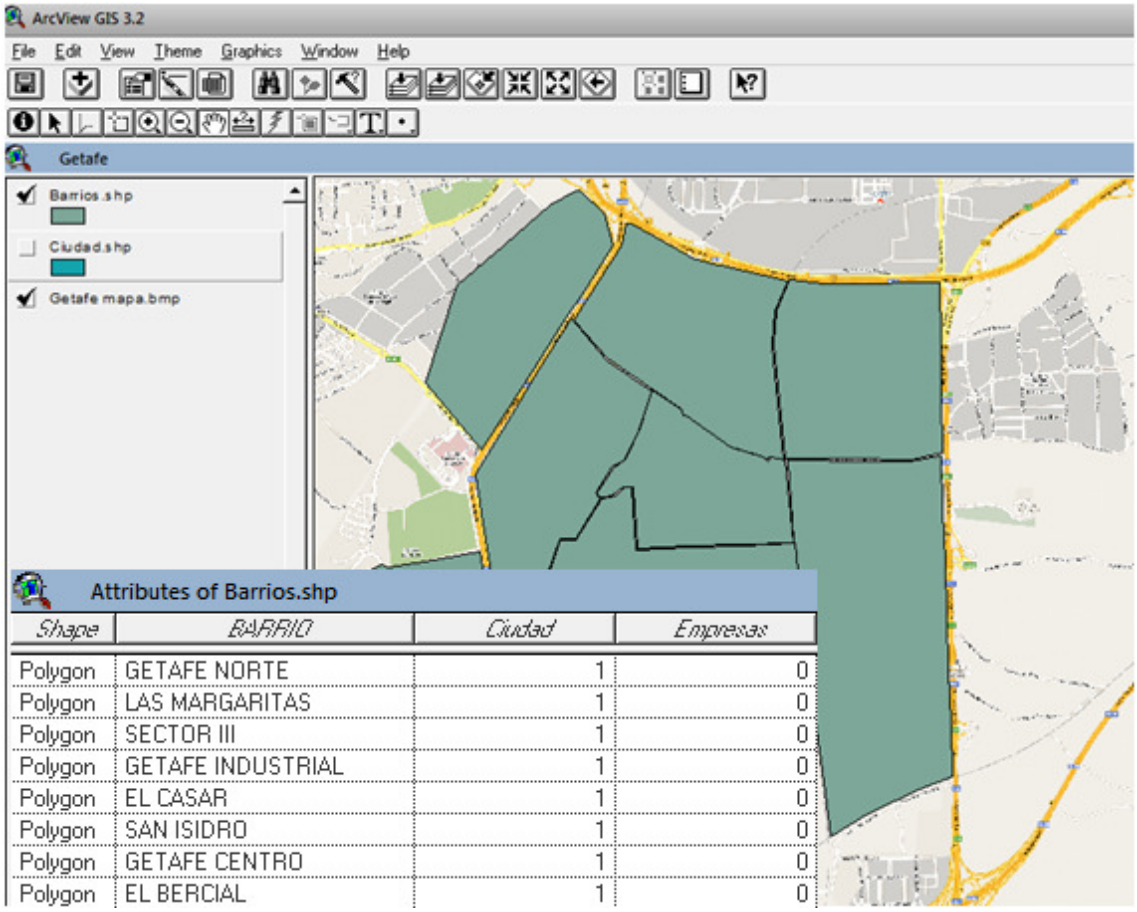


Figura 30. Digitalización completa de Getafe.

A medida que hemos ido creando las capas, se han ido añadiendo los atributos que formarán las tablas que las contengan. Esta opción está disponible por cada capa, y editándola podemos introducir la información con los campos pertinentes.

Por capa geometría que creemos, se añadirá un registro en la tabla de la capa correspondiente. Por ejemplo, para la capa Barrios.shp, se ha creado una tabla con los siguientes campos:



Shape	BARRIO	Ciudad	Empresas
Polygon	GETAFE NORTE	1	0
Polygon	LAS MARGARITAS	1	0
Polygon	SECTOR III	1	0
Polygon	GETAFE INDUSTRIAL	1	0
Polygon	EL CASAR	1	0
Polygon	SAN ISIDRO	1	0
Polygon	GETAFE CENTRO	1	0
Polygon	EL BERCIAL	1	0

Figura 31. Muestra de la tabla asociada a la capa Barrios.shp.

Nótese que por cada geometría de barrio que hemos digitalizado en la imagen, aparece un registro en la tabla.

Tenemos la opción de rellenar dentro del programa cada campo o dejar los que vayamos a utilizar como resultado de algún cálculo, como es el caso del atributo derivado *Empresas*. Este valor será el resultado de una función creada en la base de datos.

5.6 Importación de las capas a la Base de datos Oracle

Después de haber realizado los pasos descritos, disponemos de unas capas en formato shape que contienen figuras con sus respectivas coordenadas y una serie de atributos alfanuméricos asociados a cada una de ellas.

Queremos importar toda esa información a una base de datos, donde ya disponemos de un diseño de tablas para albergar dicha información. La aplicación de Oracle llamada *shp2SDO*, nos permitirá llevar a cabo esta etapa. Se dispone de una versión gratuita junto con un pequeño manual donde se explica la sintaxis a emplear dependiendo de la máquina dónde queramos ejecutarlo. Toda esta información está disponible en su página web.

Recordamos que en el apartado Entorno de trabajo, se explicó de manera detallada la forma de utilizar esta aplicación, así como los archivos que generaba ArcView, que son, en este caso, imprescindibles para trabajar con esta herramienta

La transformación de cada tabla obedece a la siguiente llamada:

- **CAPA Ciudad.shp**

  Si escribimos en la consola de comandos la sentencia

```
Shp2sdo ciudad ciudad –g geom –d –x (–180,180) –y (–90,90) –s 8307 –t 0.5 v
```

Obtenemos el siguiente resultado:

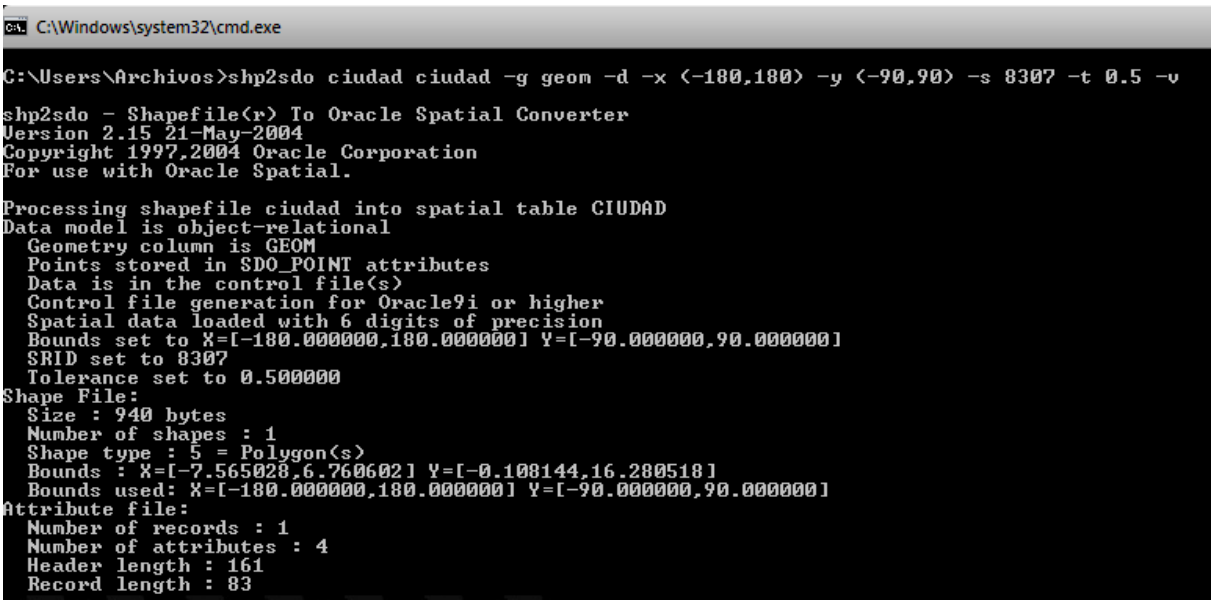


Figura 32. Transformación del formato shape de Ciudad a SDO.

Se procederá de la misma manera para cada capa, siendo su sintaxis de la siguiente manera:

- **CAPA Barrios.shp**

```
Shp2sdo barrios barrio -g geom -d -x (-180,180) -y (-90,90) -s 8307 -t 0.5 v
```

- **CAPA Calles.shp**

```
Shp2sdo calle calles -g geom -d -x (-180,180) -y (-90,90) -s 8307 -t 0.5 v
```

- **CAPA Manzanas.shp**

```
Shp2sdo manzanas manzana -g geom -d -x (-180,180) -y (-90,90) -s 8307 -t 0.5 v
```

- **CAPA Zona_industrial.shp**

```
Shp2sdo zona_Industrial zona_ind -g geom -d -x (-180,180) -y (-90,90) -s 8307
-t 0.5 v
```

- **CAPA Zona_Rural.shp**

```
Shp2sdo zona_Rural zona_rural -g geom -d -x (-180,180) -y (-90,90) -s 8307 -t
0.5 v
```

Para todos los casos, la aplicación procesa cada capa detectando información del shape file tales como su longitud (Size), los límites máximo y mínimo que se han dibujado (Bounds) y los establecidos al exportar la capa (en la llamada a shp2sdo).

También hace un resumen de la información alfanumérica escrita en ArcView. En este caso, el archivo que contiene los datos tiene 9 registros en total y la tabla está formada por 4 atributos. Incluye además información del tamaño de cabecera y de registro.

Lo más importante de esta exportación a formato SDO son los archivos que esta aplicación ha generado. En concreto son dos:

Archivos .sql

Contienen la estructura de la tabla creada en ArcView en formato entendible por Oracle Spatial así como sus metadatos.

En el apartado 5.3 se explica en detalle el porqué de la necesidad de tener creadas estas estructuras.

Archivos .ctl

Aparte de este script, es necesario que haya exportado un archivo con todos los datos alfanuméricos introducidos en ArcView. De aquí se cargarán las coordenadas de cada geometría y la información adicional escrita en forma de tabla, con sus respectivos atributos, indicados en el script de arriba.

Se trata de archivos de control o Control file; No se muestran por ser demasiado extensos y tener una estructura poco legible para el usuario.

5.7 Carga de datos a Oracle Spatial

Una vez creadas las tablas con el script generado por shp2SDO, procedemos a llenar la tabla con los archivos que contienen la información contenida en cada una de ellas, los archivos.ctl.

La herramienta que se debe utilizar en este caso es SQL Loader, también desarrollada por Oracle Corporation. La sintaxis de llamada es tan sencilla como loguearse con un usuario de Oracle que tenga privilegios para crear, borrar y editar objetos en la base de datos (al menos tablas) y que tenga acceso al esquema donde vayan a almacenarse estas tablas. La llamada al programa sería la siguiente:

```
sqlldr username/password archivo.ctl
```

NOTA IMPORTANTE: ArcView exporta los datos numéricos de coordenadas y otros valores que contengan decimales utilizando el punto como separador. Al intentar importar estos datos a Oracle se obtiene un error, ya que no los reconoce como tal.

Para subsanar este problema, basta con sustituir el punto por una coma en todos los casos. Además, es importante resaltar que para cargar los datos de cada tabla, se deben ir creando individualmente y no ejecutando un script completo de todas las tablas, ya que SQL Loader proporciona el siguiente tipo de error: ORA-02266: claves únicas/primarias en la tabla referidas por claves ajenas activadas, en el caso de que a una tabla le estén referenciando otras en ese momento.

Una vez cargados todos los datos a las tablas, tenemos nuestra base de datos completa para poder trabajar con ella. Lógicamente, no todas las tablas se han creado dentro de ArcView. Disponemos de tablas que no contienen datos geométricos y por tanto, hay que crear e insertar datos en ellas manualmente. En este caso se trata de las tablas Vivienda, Empresa, Grupo_Edades y Persona. El script llamado "Inserciones" contiene todas las inserciones que se han hecho sobre ellas.

5.8 Funciones implementadas

Se hacía necesario el uso de subprogramas en pl/sql para realizar cálculos más complejos que no se podían resolver mediante una consulta en SQL.

5.8.1 Densidad de población por Barrio

La siguiente función calcula la densidad de población por cada barrio. En concreto recoge el número de personas que tenemos registradas en cada uno y lo divide entre su área.

```
    CREATE OR REPLACE FUNCTION CALCULA_DENSIDAD (P_BARRIO
BARRIO.BARRIO%TYPE)
    RETURN  NUMBER
     IS
             v_personas   NUMBER;
             v_barrio            VARCHAR2(25);
             v_area              NUMBER;
             v_result            NUMBER;
    BEGIN
                SELECT COUNT(*) AS personas INTO v_personas
                FROM PERS_BARRIO
                WHERE barrio = p_barrio;


SELECT SDO_GEOM.SDO_AREA(BARRIO.GEOM, 0.005, 'UNIT=SQ_KM')

 INTO v_area

            FROM barrio
            WHERE barrio.barrio = p_barrio;
               v_result := round( v_personas / v_area, 6);

DBMS_OUTPUT.PUT_LINE('LA DENSIDAD DE ' ||p_barrio ||  ' ES:' ||
TO_CHAR(v_result) || ' HABITANTES POR KM CUADRADO');
       RETURN v_result;
```

5.8.2 Porcentaje de zona urbana de la ciudad

Esta función podríamos haberla calculado también por barrio. En este caso se ha optado por ciudad por tener un mayor número de datos. Primero calcula el área en km cuadrados de la ciudad y luego el total de las áreas de cada manzana. El resultado es la división del total entre el área de la ciudad.

```
CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_URB (P_CIUDAD CIUDAD.NOMBRE%TYPE)
RETURN NUMBER
IS
       v_area_ciudad     NUMBER;
       v_area_zonas      NUMBER;
```

```
        v_result          NUMBER;
  BEGIN
        SELECT   SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM')
        INTO v_area_ciudad
            FROM ciudad
            WHERE ciudad.nombre = p_ciudad;
        SELECT SUM(SDO_GEOM.SDO_AREA(manzana.GEOM, 0.005, 'UNIT=SQ_KM' ))
         INTO v_area_zonas
            FROM manzana;
        v_result := round( v_area_zonas /v_area_ciudad,6) * 100;
        DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA URBANA DE ' || p_ciudad
        ||  ' ES: ' ||TO_CHAR(v_result) || ' POR CIENTO');
   RETURN v_result;
   END PORCENTAJE_ZONA_URB;
```

5.8.3 Porcentaje zona despoblada de la ciudad

Al igual que la anterior, hacemos el cálculo por ciudad. Esta función utiliza el concepto de zona rural para considerarla zona despoblada, ya que en nuestro diseño, no se contemplan viviendas dentro de estas zonas. Es similar a la anterior, ya que el área de la ciudad es dividido entre el total de las áreas de cada zona rural.

```
CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_RURAL
P_CIUDAD(CIUDAD.NOMBRE%TYPE)
  RETURN NUMBER
  IS
     v_area_ciudad     NUMBER;
     v_area_zonas      NUMBER;
     v_result          NUMBER;
  BEGIN
      SELECT   SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM')
      INTO v_area_ciudad
     FROM ciudad
     WHERE ciudad.nombre = p_ciudad;

     SELECT SUM(SDO_GEOM.SDO_AREA(zona_rural.GEOM, 0.005, 'UNIT=SQ_KM' ))
      INTO v_area_zonas
     FROM zona_rural;
       v_result := round( v_area_zonas /v_area_ciudad,6) * 100;
     DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA DESPOBLADA DE ' || p_ciudad
     ||  ' ES: ' ||TO_CHAR(v_result) || ' POR CIENTO');
   RETURN v_result;
   END PORCENTAJE_ZONA_RURAL;
```

5.8.4 Porcentaje zona industrial de la ciudad

Ahora el objeto de estudio es el porcentaje de zona industrial que posee nuestra ciudad. Este porcentaje será menor que el de la zona rural según se vislumbró en el mapa, por lo que esta función debería constatarlo.

El cálculo es similar a los anteriores. En este caso, el área de la ciudad es dividida entre el total del área de las zonas industriales.

```
CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_IND (P_CIUDAD CIUDAD.NOMBRE%TYPE)
 RETURN NUMBER
 IS
    v_area_ciudad     NUMBER;
    v_area_zonas      NUMBER;
    v_result          NUMBER;
 BEGIN
    SELECT   SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM')
     INTO v_area_ciudad
    FROM ciudad
    WHERE ciudad.nombre = p_ciudad;
    SELECT SUM(SDO_GEOM.SDO_AREA(zona_ind.GEOM, 0.005, 'UNIT=SQ_KM' ))
     INTO v_area_zonas
      FROM zona_ind;
    v_result := round( v_area_zonas /v_area_ciudad,6) * 100;
     DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA INDUSTRIAL DE ' || p_ciudad
    ||  ' ES: ' ||TO_CHAR(v_result) || ' POR CIENTO');
  RETURN v_result;
  END PORCENTAJE_ZONA_IND;
  /
```

5.9 Disparadores

5.9.1 Disparador COMPRUEBA_PERSONA

Este disparador tiene por objetivo asegurar la consistencia de los datos en la tabla PERSONA, que no ha sido posible recoger mediante el diseño físico-lógico de la base de datos.

```
  CREATE OR REPLACE TRIGGER INSERTA_PERSONA
  BEFORE INSERT OR UPDATE
  ON PERSONA
  FOR EACH ROW
DECLARE
   ID_PER         TEMP_PERSONA.ID_PK%TYPE;
   ID_VIV         TEMP_PERSONA.ID_VIVIENDA_FK%TYPE;
   SIT_LAB        TEMP_PERSONA.SITUACION_LABORAL%TYPE;
```

```
   ID_EMP        TEMP_PERSONA.ID_EMPRESA_FK%TYPE;
   CIUDAD_TRAB   TEMP_PERSONA.CIUDAD_TRABAJO%TYPE;
   ERROR_EMPRESA EXCEPTION;
BEGIN
        INSERT INTO TEMP_PERSONA VALUES (
                :NEW.ID_PK,
                :NEW.ID_VIVIENDA_FK,
                :NEW.SITUACION_LABORAL,
                :NEW.ID_EMPRESA_FK,
                :NEW.CIUDAD_TRABAJO  );

 SELECT *
         INTO ID_PER, ID_VIV, SIT_LAB, ID_EMP,  CIUDAD_TRAB
         FROM TEMP_PERSONA
        WHERE ID_PK = :NEW.ID_PK;

   IF (SIT_LAB = 'EN PARO' AND ID_EMP != 0)
   THEN     RAISE_APPLICATION_ERROR (-20001,  'La Situación laboral de esta
persona no permite tener asociada ninguna empresa');
   END IF;

   IF (SIT_LAB = 'EN PARO' AND CIUDAD_TRAB IS NOT NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20002,
         'La Situación laboral de esta persona no permite tener asociada
ninguna ciudad de trabajo');
   END IF;

   IF (CIUDAD_TRAB != 'GETAFE' AND ID_EMP IS NOT NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20003,
         'Las personas que no trabajen en Getafe no tendrán asociado ningún ID
de empresa');
   END IF;

   IF (CIUDAD_TRAB = 'GETAFE' AND SIT_LAB = 'TRABAJANDO' AND ID_EMP IS NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20004,
         'Las personas que trabajen en Getafe tienen que tener un ID de
empresa asociado');
   END IF;

END;
```

El disparador utiliza una tabla auxiliar llamada TEMP_PERSONA, con una estructura parecida a la tabla PERSONA. Esta tabla ha tenido que implementarse debido a que se producía error de **tabla mutante*[1]** en relación a la tabla PERSONA.

1. **Tabla mutante es una tabla que está siendo modificada por una sentencia SQL (insert, update, delete) o por el efecto de un DELETE CASCADE asociado a la sentencia SQL**

Por cada inserción que se hace en la tabla PERSONA, se hace otra en la tabla TEMP_PERSONA. De esta forma, luego podemos hacer una consulta sobre ella para obtener los campos que se quieren insertar en un momento determinado.

5.9.2 Disparadores para el cálculo de empresas en cada barrio.

Para llevar un control de las empresas que se van insertando, actualizando y borrando, se han creado 3 disparadores.

El primer trigger sólo introduce un registro en la tabla TMP_EMPRESA por cada valor introducido en la tabla original, es decir, EMPRESA. Es similar a la primera parte del disparador anterior, ya que luego se necesita hacer una consulta para saber qué empresa se quiere borrar.

```
CREATE OR REPLACE TRIGGER Llena_temp_empresas_barrio
AFTER INSERT OR UPDATE ON EMPRESA
FOR EACH ROW
BEGIN
   INSERT INTO TMP_EMPRESA VALUES (:new.id_empresa_pk, :new.barrio,
   :new.ciudad, :new.nombre_pol);
END;
```

Este disparador se encarga de actualizar la table BARRIO. En este caso, de restar un valor al campo "empresas" del barrio al que pertence la empresa que se quiere eliminar.

```
CREATE OR REPLACE TRIGGER Borrado_empresas_barrio
AFTER DELETE ON EMPRESA
FOR EACH ROW
DECLARE
    v_emp INTEGER;
    v_barrio barrio.barrio%TYPE;
    v_ciudad barrio.ciudad%TYPE;
BEGIN
    SELECT id_empresa_pk, barrio, ciudad INTO v_emp, v_barrio, v_ciudad
    FROM tmp_empresa
    WHERE id_empresa_pk = :old.id_empresa_pk;

            UPDATE BARRIO SET empresas = empresas - 1
     WHERE barrio.barrio = v_barrio AND
           barrio.ciudad = v_ciudad;

END;
```

Por último, éste es el disparador que incrementa una unidad en el campo empresas por cada registro introducido en la tabla. Utiliza un cursor para recoger los datos que hay actualmente en la tabla y suma un valor comprobando que el barrio y la ciudad correspondan con la tabla BARRIO, que es a la que EMPRESA hace referencia.

```
CREATE OR REPLACE TRIGGER Inserta_empresas_barrio
AFTER INSERT ON EMPRESA
FOR EACH ROW
DECLARE
    v_emp INTEGER;
    v_barrio barrio.barrio%TYPE;
    v_ciudad barrio.ciudad%TYPE;

   CURSOR c_empresas IS
      SELECT id_empresa_pk, barrio, ciudad
      FROM temp_empresa;

  BEGIN

 UPDATE BARRIO SET empresas=0;
    OPEN c_empresas;

       LOOP

          FETCH c_empresas INTO v_emp, v_barrio, v_ciudad;

          EXIT WHEN c_empresas%NOTFOUND;


          UPDATE BARRIO SET empresas = empresas + 1
           WHERE barrio.barrio = v_barrio AND
                 barrio.ciudad = v_ciudad;

       END LOOP;

    CLOSE c_empresas;

END;
/
```

5.10 Consultas

Las consultas para este trabajo resultan de vital importancia. A través de ellas hemos ido recopilando información que luego se ha podido utilizar en las funciones, ya que todas han sido creadas como vistas. Se han abordado siguiendo 3 fases:

- **Consultas Fase I**

  Son consultas sencillas que nos aportan datos informativos, en su mayor parte referidos a personas, como su situación laboral, la ciudad de trabajo, segundas propiedades, número de personas por barrio y empresas, número de empresas que tiene cada barrio, etc.

Consulta 1 - Tipo de vivienda que más predomina

Esta consulta cuenta los tipos de vivienda en los que residen las personas por cada calle, barrio y ciudad determinada. Posteriormente realizamos una consulta sobre la vista creada para sacar el máximo.

```
CREATE VIEW tipo_vivienda AS
SELECT tipo_vivienda, barrio.barrio, COUNT(*) tipo
FROM vivienda, calle, barrio
WHERE vivienda.id_calle = calle.id_calle AND
      vivienda.barrio = calle.nombre_bar AND
      vivienda.ciudad = calle.ciudad AND
      calle.nombre_bar = barrio.barrio AND
      calle.ciudad = barrio.ciudad
GROUP BY barrio.barrio, tipo_vivienda
ORDER BY barrio;

SELECT tipo_vivienda, barrio
FROM tipo_vivienda
WHERE tipo = (SELECT MAX(tipo) FROM tipo_vivienda);
```

Consulta 2 - Número de personas por barrio con segunda vivienda

En esta consulta se cuenta el número de personas que residen en una determinada vivienda, calle, barrio y ciudad y que poseen una segunda propiedad.

```
CREATE OR REPLACE SEG_VIV AS
SELECT COUNT(*) AS personas, barrio.barrio
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
      vivienda.id_calle = calle.id_calle AND
vivienda.barrio = calle.nombre_bar AND
vivienda.ciudad = calle.ciudad AND
      calle.nombre_bar = barrio.barrio AND
calle.ciudad = barrio.ciudad AND
            persona.segunda_propiedad = 'S'
GROUP BY barrio.barrio;
```

Consulta 3 - Situación laboral que más predomina por barrio

Esta consulta hace un resumen de la anterior, ya que saca la situación laboral más predominante en cada barrio. Se calcula el máximo de el campo situación_laboral de la tabla PERSONA y se agrupa al final por barrio.

```
CREATE OR REPLACE VIEW SIT_LAB_BARRIO AS
SELECT MAX(situacion_laboral)as situacion, barrio.barrio
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
      vivienda.id_calle = calle.id_calle AND
      vivienda.barrio = calle.nombre_bar AND
      vivienda.ciudad = calle.ciudad AND
      calle.nombre_bar = barrio.barrio AND
      calle.ciudad = barrio.ciudad
GROUP BY barrio.barrio;
```

Consulta 4 - Situación laboral que más predomina

Basta hacer una consulta sobre la vista anterior para sacar el máximo de todos los barrios:

```
SELECT situacion_laboral, barrio
FROM sit_lab_barrio
WHERE situacion = (SELECT MAX(situacion) FROM sit_lab_barrio);
```

Consulta 5 - Régimen de vivienda que más predomina por barrio

Al igual que las anteriores, se obtiene el régimen de vivienda que tienen en la actualidad las personas que viven en una determinada vivienda, situada en una calle, un barrio y una ciudad en concreto.

```
CREATE VIEW reg_vivienda AS
SELECT regimen_vivienda , barrio.barrio, COUNT(*) regimen
FROM persona, vivienda, calle, barrio
WHERE  persona.id_vivienda_fk = vivienda.id_vivienda AND
            vivienda.id_calle = calle.id_calle AND
            vivienda.barrio = calle.nombre_bar AND
            vivienda.ciudad = calle.ciudad AND
            calle.nombre_bar = barrio.barrio AND
            calle.ciudad = barrio.ciudad
GROUP BY barrio.barrio, regimen_vivienda;
```

Consulta 6 - Régimen de vivienda que más predomina

Basta con hacer una select a la vista creada anteriormente, sacanado el máximo del campo régimen.

```
CREATE VIEW REG_MAX AS
SELECT barrio, regimen_vivienda
FROM reg_vivienda
WHERE regimen = (SELECT MAX(regimen) FROM reg_vivienda);
```

Consulta 7 – Número de personas que han cambiado de vivienda

Se crea esta vista para contar el número de personas que hemos registrado con el campo cambios_vivienda de la tabla PERSONA distinto de cero. Una vez más se tienen que comparar las tablas PERSONA, VIVIENDA, CALLE y BARRIO.

```
CREATE REPLACE VIEW cambios_viv AS
SELECT COUNT(*) AS personas, barrio.barrio
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
      vivienda.id_calle = calle.id_calle AND
      vivienda.barrio = barrio.barrio AND
      vivienda.ciudad = barrio.ciudad AND
      calle.nombre_bar = barrio.barrio AND
      calle.ciudad = barrio.ciudad AND
      persona.cambios_vivienda != 0
GROUP BY barrio.barrio;
```

Consulta 8 – Barrio con el número mayor de personas que han cambiado de vivienda

Nos valemos de la anterior vista creada para sacar el barrio en el que más personas han cambiado de vivienda.

```
CREATE OR REPLACE VIEW CAMBIOS_VIV_MAX AS
SELECT personas, barrio
FROM cambios_viv
WHERE personas = (SELECT MAX(personas) FROM cambios_viv);
```

Consulta 9 – Número de personas por barrio que trabajan en Getafe

Esta vista muestra las personas que tienen a Getafe como su ciudad de trabajo. Para un estudio sobre la ciudad en la que más personas trabajan

en una determinada ciudad puede resultar muy útil. En este caso se agrupa por barrio ya que sólo estudiamos una ciudad.

```
CREATE OR REPLACE VIEW trabaja_getafe AS
 SELECT  (barrio.barrio), COUNT(*) AS personas
 FROM persona, vivienda, calle, barrio
 WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
       vivienda.id_calle = calle.id_calle AND
       vivienda.barrio = calle.barrio AND
       vivienda.ciudad = calle.ciudad AND
       calle.nombre_bar = barrio.barrio AND
       persona.ciudad_trabajo = 'GETAFE'
 GROUP BY barrio.barrio;
```

Consulta 10 – Barrio en el que más personas trabajan en Getafe

Igual que en consultas anteriores, podemos utilizar la vista creada para sacar el barrio en el que más personas trabajan en Getafe.

```
CREATE OR REPLACE VIEW TRAB_GETAFE_MAX AS
SELECT barrio, personas
FROM trabaja_getafe
WHERE personas = (SELECT MAX(personas) FROM trabaja_getafe);
```

Consulta 11 – Número de personas por barrio

Vista que proporciona el número de personas que tenemos registradas como residentes en un determinado barrio.

```
CREATE OR REPLACE VIEW PERS_BARRIO AS
 SELECT COUNT(*) AS personas, barrio.barrio
 FROM persona, vivienda, calle, barrio
 WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
       vivienda.id_calle = calle.id_calle AND
       vivienda.barrio = calle.nombre_bar AND
       vivienda.ciudad = calle.ciudad AND
       calle.nombre_bar = barrio.barrio AND
       calle.ciudad = barrio.ciudad
 GROUP BY barrio.barrio;
```

Consulta 12 – Número de empresas por barrio

Consulta que muestra las empresas asociadas a un determinado barrio.

```
 CREATE VIEW empresas_barrio AS
 SELECT COUNT(*) AS empresas, barrio.barrio
 FROM empresa, barrio
 WHERE empresa.barrio = barrio.barrio AND
```

```
        empresa.ciudad = barrio.ciudad
GROUP BY barrio.barrio;
```

Consulta 13 – Barrio que más empresas tiene

Para sacar este máximo, hacemos una consulta sobre la vista anterior, consultando el máximo del campo "empresas", en este caso.

```
CREATE OR REPLACE VIEW EMP_BARRIO AS
SELECT empresas, barrio
FROM empresas_barrio
WHERE empresas = (SELECT MAX(empresas) FROM empresas_barrio);
```

Consulta 14 - Número de calles por barrio

En esta consulta se obtiene el número de calles asociadas a cada barrio. Recordamos que una calle está asociada directamente con barrio, por lo que la unión es directa entre ambas.

```
 CREATE VIEW barrio_calles AS
SELECT COUNT(*) AS calles, barrio
FROM calle, barrio
WHERE calle.nombre_bar = barrio.barrio AND
      calle.ciudad = barrio.ciudad
GROUP BY barrio.barrio;
```

Consulta 15 - Barrio que más calles tiene

Para sacar el barrio que más calles tiene, sólo tenemos que hacer una consulta sobre la vista creada, sacando el máximo del atributo que contiene el número de calles, en este caso "calles".

```
CREATE OR REPLACE MAX_BARRIO_CALLES AS
SELECT calles, barrio
FROM barrio_calles
WHERE calles = (SELECT MAX(calles) FROM barrio_calles);
```

- **Consultas Fase II**

En esta etapa se realizan consultas utilizando funciones geométricas propias de la API de Oracle Spatial. Se calculan áreas y distancias entre barrios y calles.

Consulta 1 - Área de un barrio en Km cuadrados

Creamos esta vista para calcular el área de cada barrio. Oracle da la posibilidad de escoger la unidad de medida en la que quiere realizarse dicho cálculo. Más abajo se indica cómo saber qué medidas están disponibles.

```
CREATE OR REPLACE AREA_BARRIO AS

SELECT BARRIO.BARRIO, ROUND(SDO_GEOM.SDO_AREA(BARRIO.GEOM, 0.005,
'UNIT=SQ_KM' ),2) AS Área

FROM barrio;
```

-- Para sacar las unidades de medida disponibles

```
SELECT * from MDSYS.SDO_AREA_UNITS;
```

Consulta 2 – Área de los polígonos industriales

Área de los polígonos que forman parte de las zonas industriales de los barrios.

```
CREATE OR REPLACE AREA_ZONA_IND AS

SELECT NOMBRE_POL, ROUND(SDO_GEOM.SDO_AREA(ZONA_IND.GEOM, 0.005,
'UNIT=SQ_KM' ),2) AS Área

FROM ZONA_IND;
```

Consulta 3 – Área de las zonas rurales

Área de las zonas rurales de cada barrio. Se hace una suma ya que Oracle realiza el cálculo por separado para cada polígono digitalizado.

```
CREATE OR REPLACE VIEW AREA_ZONA_RURAL AS

SELECT BARRIO, ROUND(SUM(SDO_GEOM.SDO_AREA(ZONA_RURAL.GEOM, 0.005,
'UNIT=SQ_KM')),2)  AS Area

FROM ZONA_RURAL

GROUP BY BARRIO;
```

Consulta 4 – Área de las zonas urbanas (manzanas)

Área de la zona urbana que tiene cada barrio. Al igual que en la consulta anterior, necesitamos sumar para que Oracle devuelva un registro único por cada barrio. En caso contrario, mostraría cada manzana y su área correspondiente.

```
CREATE OR REPLACE VIEW AREA_ZONA_URB AS

 SELECT BARRIO, ROUND(SUM(SDO_GEOM.SDO_AREA(MANZANA.GEOM, 0.005,
 'UNIT=SQ_KM')),2)  AS Area

FROM MANZANA

GROUP BY BARRIO;
```

Consulta 5 - Distancia entre barrios y calles

Una vez más hacemos uso de una función de Oracle Spatial, en concreto de cálculo de distancias. Como se puede observar, se permite dicho cálculo entre geometrías de distinto tipo, como son los barrios (polígonos) y calles (líneas).

En concreto se toma un barrio y una calle para ilustrar el ejemplo, pero quitando esta restricción, la consulta devolvería la distancia de todas las calles a todos los barrios.

Cabe destacar que se observó que para las calles pertenecientes a un barrio la distancia se considera cero.

También se ofrece la opción de escoger la unidad de medida en la que se desea mostrar la distancia. En este caso, seguimos decantándonos por los km cuadrados.

```
CREATE OR REPLACE DIST_BARRIO_CALLE AS
SELECT ROUND(SDO_GEOM.SDO_DISTANCE(barrio.geom, calle.geom,0.005),2)
FROM barrio, calle
WHERE barrio.barrio = 'EL BERCIAL' AND
           calle.nombre_cal = 'YELMO';
```

- **Consultas Fase III**

En esta última fase se realizan consultas más completas, todas de ellas empleando un número alto de condiciones y utilizando las funciones geométricas requeridas para cada caso.

Consulta 1 - Necesidad de vivienda por barrios

En este caso, se han planteado preguntas como la demanda de vivienda por barrios condicionada por varios factores, como la pertenencia a un grupo de edad determinado, con empleo, residiendo en Getafe, sin posesión de segunda vivienda y con una antigüedad de la manzana de un número de años mayor a 20.

Se ha considerado que las personas que cumplan estos requisitos pueden ser posibles demandantes de vivienda, si bien estos factores pueden cambiar.

```
CREATE OR REPLACE DEMANDA_VIV AS
SELECT COUNT(*) AS personas, barrio.barrio
FROM grupo_edades, persona, vivienda, manzana, barrio, calle
WHERE (persona.id_grupo_edad_fk > 1) AND
       (persona.id_grupo_edad_fk <= 3) AND
        persona.id_vivienda_fk = vivienda.id_vivienda AND
        vivienda.id_calle_fk = calle.id_calle AND
        calle.nombre_bar = barrio.barrio AND
        persona.situacion_laboral = 'TRABAJANDO' AND
```

```
            persona.ciudad_trabajo= 'GETAFE' AND
            persona.segunda_propiedad = 'N' AND
            persona.cambios_vivienda <= 2  AND
            persona.id_manzana_fk = manzana.id AND
            manzana.antiguedad > 20
  GROUP BY barrio.barrio;
```

Consulta 2 - Posibles demandantes de vivienda por cercanía a empresas

En esta consulta se utiliza una vez más el cálculo de distancias entre dos geometrías para tenerlo en cuenta a la hora de seleccionar a los posibles demandantes de vivienda en una zona.

En esta consulta se tienen menos restricciones en cuanto a las características que deben cumplir las personas, pero se añaden otras en cuanto a distancia.

Para este caso, queremos saber qué personas viven a una determinada distancia de su centro de trabajo y pueden estar interesados en encontrar vivienda en los alrededores.

Para ello, se selecciona la manzana en la que viven las personas que están en un determinado rango de edad y que trabajan en la empresa seleccionada.

Dado que todas las empresas registradas ahora mismo en nuestra BBDD son de Getafe, con añadir esta condición será suficiente.

Después, se calcula la distancia entre el barrio en el que está su empresa con la manzana en la que viven, y seleccionándose dichas personas.

```
DROP VIEW DIST_TRABAJO;
     CREATE OR REPLACE VIEW DIST_TRABAJO AS
     SELECT DISTINCT persona.id_manzana_fk, persona.id_pk, empresa.barrio
     FROM persona, manzana, empresa, barrio
     WHERE   (persona.id_grupo_edad_fk > 1) AND
             (persona.id_grupo_edad_fk <= 4) AND
              persona.id_manzana_fk = manzana.id AND
```

```
        persona.ciudad_trabajo = 'GETAFE' AND
        persona.id_empresa_fk = empresa.id_empresa_pk AND
        empresa.barrio = barrio.barrio   AND
SDO_GEOM.SDO_DISTANCE(BARRIO.GEOM, MANZANA.GEOM, 0.005, 'UNIT=KM') > 40
```

6. EXPERIMENTACIÓN

Este apartado tiene como objetivo probar el correcto funcionamiento de las herramientas que se necesitan para el desarrollo del proyecto.

Las pruebas consistirán en introducir geometrías, validarlas y cargar sus datos, así como el funcionamiento de los disparadores, funciones y consultas creadas.

6.1. Introducción de geometrías en ArcView

Vamos a comprobar la correcta introducción de geometrías en esta aplicación. Para ello, se crea un nuevo proyecto y se introducen algunas figuras de prueba. Como en este caso sólo trabajamos con polígonos y líneas, las capas sólo contendrán estas formas.

6.1.1 Digitalización de polígonos

Para esta versión de ArcView, podemos crear polígonos irregulares, cuadrados y círculos. Para los polígonos, también tenemos la opción de dividirlos y juntarlos mediante dentro de su propia capa.

Para insertar polígonos, se debe hacer click por cada punto que vaya a contener la figura y cerrarse con doble click en el último punto.

En caso de que no se haya cerrado bien, automáticamente ArcView lo descarta, borrando la geometría del lienzo de dibujo.

Los polígonos que estén divididos por líneas, serán considerados como dos figuras independientes.

En la siguiente figura, se muestran todos los tipos de polígonos que podemos crear.

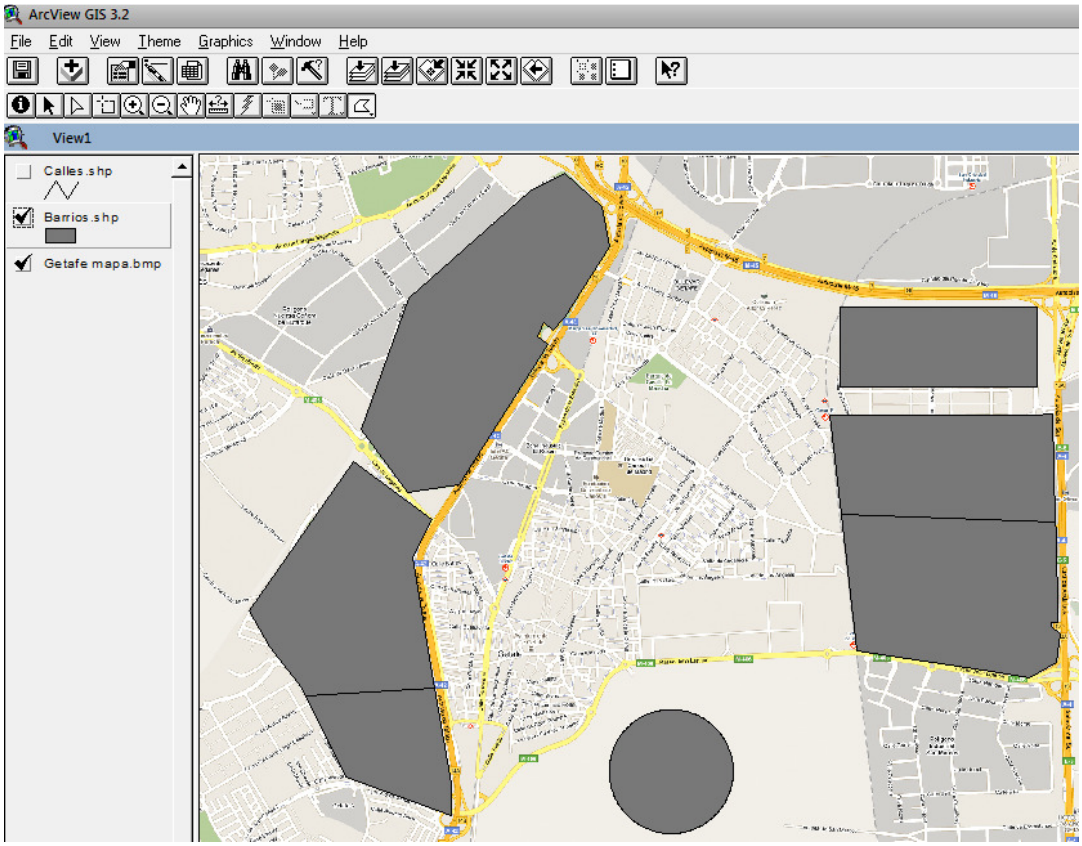


Figura 33. Tipos de polígonos en ArcView 3.2

6.1.2 Digitalización de líneas

El caso de las líneas es similar al de los polígonos. Pueden crearse líneas formadas por varios puntos, lo que ArcView llama poli líneas y también pueden ser cortadas o divididas por otras.

Para dibujar poli líneas, basta con ir haciendo click por cada punto que formará la línea y doble click al final de la misma. Lógicamente, como mínimo deben ser dos.

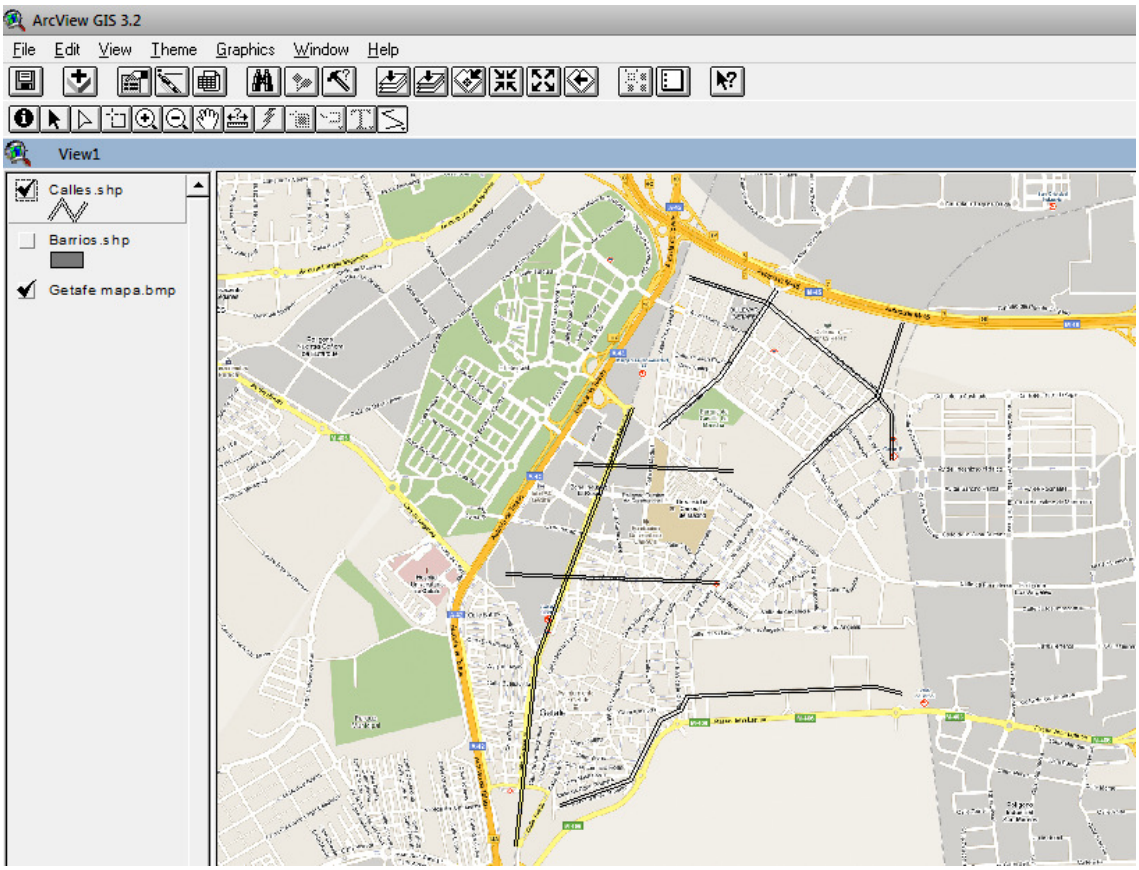


Figura 34. Tipos de líneas en ArcView 3.2

6.1.3 Carga de datos con SQL Loader

Como ya se comentó en el apartado "Método de resolución", el siguiente paso a abordar una vez dibujadas y exportadas las geometrías, se cargan los datos que se han introducido en ArcView mediante tabla.

Una vez creadas las tablas con el archivo .sql que Shp2sdo ha creado para cada capa, se cargan los datos con SQL Loader invocando al archivo .ctl correspondiente.

6.1.3.1 Carga de datos para la capa Barrios

En el ejemplo para Barrios, disponemos de su archivo Barrios.ctl con las coordenadas introducías y demás campos que hayamos querido considerar.

Se procede a su carga mediante el siguiente mandato por línea de comandos:

```
Sqlldr usuario/pwd barrios.ctl
```

El programa no muestra nada en su salida:

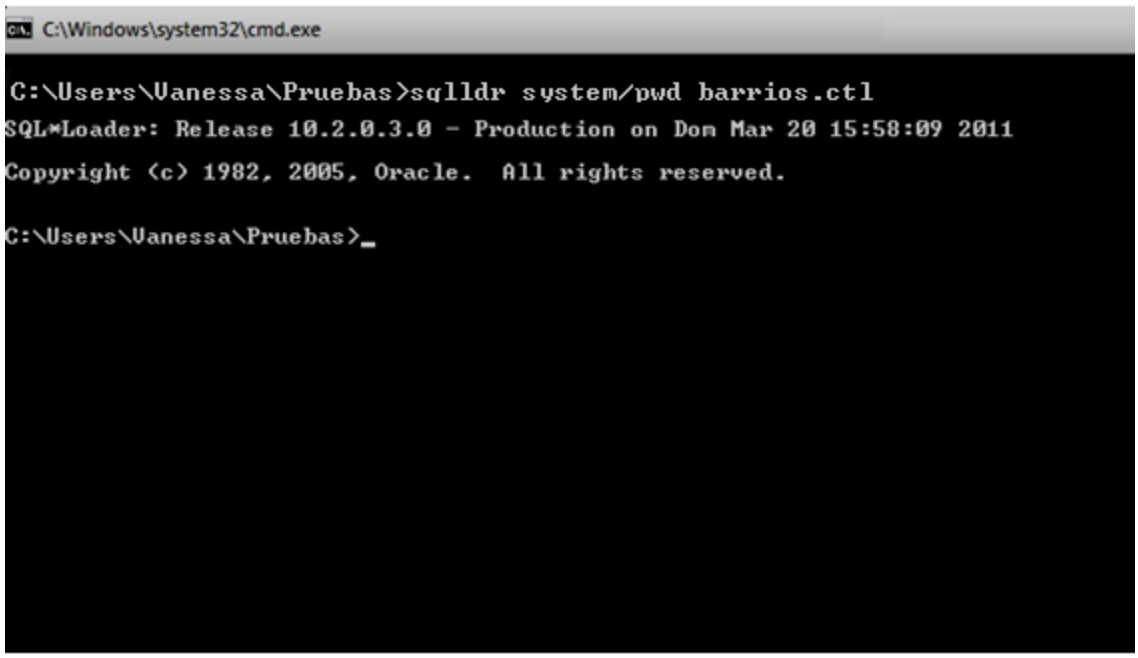


Figura 35: Ejecución errónea de SQL Loader para Barrios.ctl

Inmediatamente observamos que SQL Loader ha creado dos archivos para esta capa: barrios.log y barrios.bad.

Este último archivo sólo aparece en el caso de que haya habido algún problema a la hora de hacer la carga. Efectivamente, nos damos cuenta de que los decimales que representan las coordenadas geométricas de cada barrio están representados mediante un punto; de este modo, Oracle no los reconoce como tal y los ignora. La solución es reemplazar estos puntos por comas y volver a guardar el archivo barrios.ctl.

Una vez hecho esto, la carga funciona correctamente y la salida del programa verifica el número de polígonos que se han insertado:

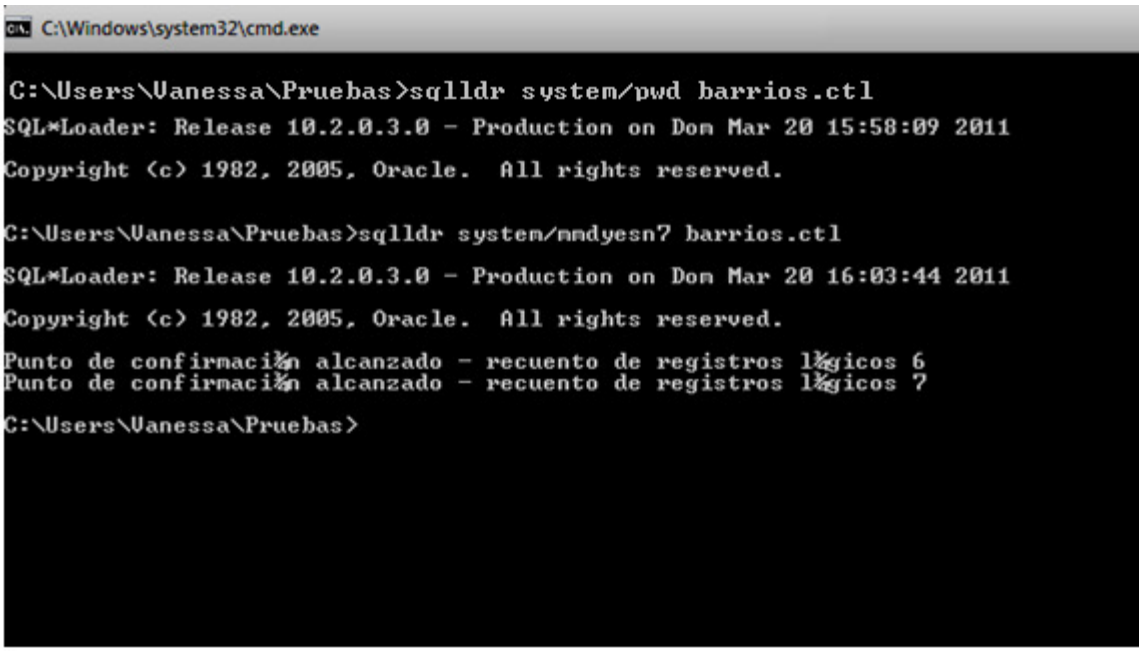


Figura 36: Ejecución correcta de SQL Loader para Barrios.ctl

Efectivamente, si nos damos cuenta, el número de polígonos que queremos insertar para representar a los barrios, son 7, contando los que están divididos. Por tanto, la carga se ha llevado a cabo correctamente. Esto también puede verificarse mediante una consulta a la tabla Barrios.

6.1.3.2 Carga de datos para la capa Calles

Procederemos de la misma manera para estas geometrías. Dadas las calles dibujadas para el proyecto de prueba creado, exportadas con shp2sdo y creada su tabla, ejecutamos el mandato correspondiente:

```
Sqlldr usuario/pwd calles.ctl
```

Al igual que el caso anterior, el programa no muestra nada en su salida.

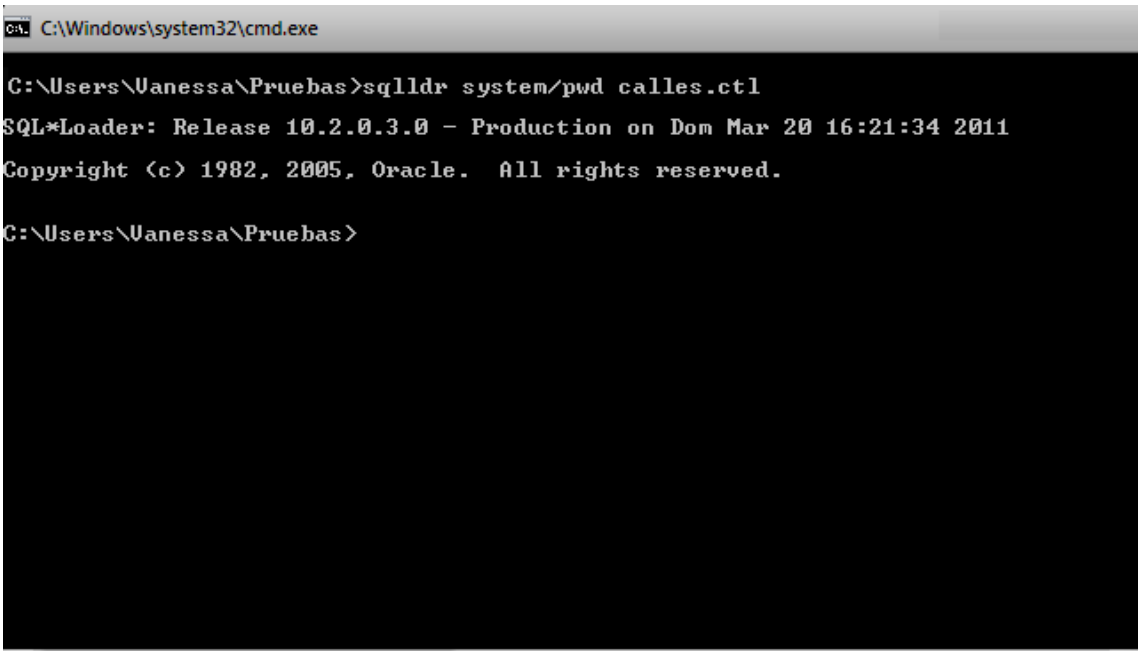


Figura 37: Ejecución errónea de SQL Loader para Calles.ctl

Ahora ya sabemos que el problema está en los puntos de los decimales, por lo que volvemos a actuar de la misma manera, sustituyendo los puntos por las comas, guardando nuevamente el archivo y ejecutando de nuevo el mandado:

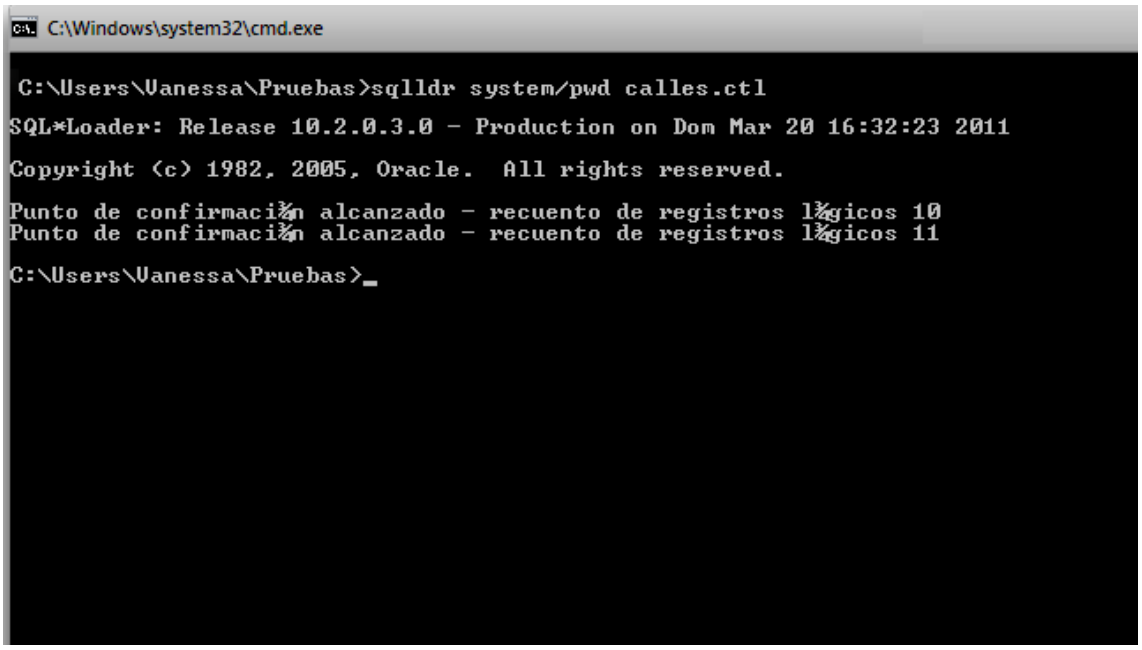


Figura 38: Ejecución correcta de SQL Loader para Calles.ctl.

Observamos que SQL Loader ha identificado 11 calles. Esto no es un error del programa, ya que al dividir algunas de las calles por otras, ArcView las considera como calles independientes. Por ejemplo, en la zona remarcada en la figura, tendremos 5 calles, las tres por las que queda dividida la calle vertical y las dos calles que la cortan en horizontal.

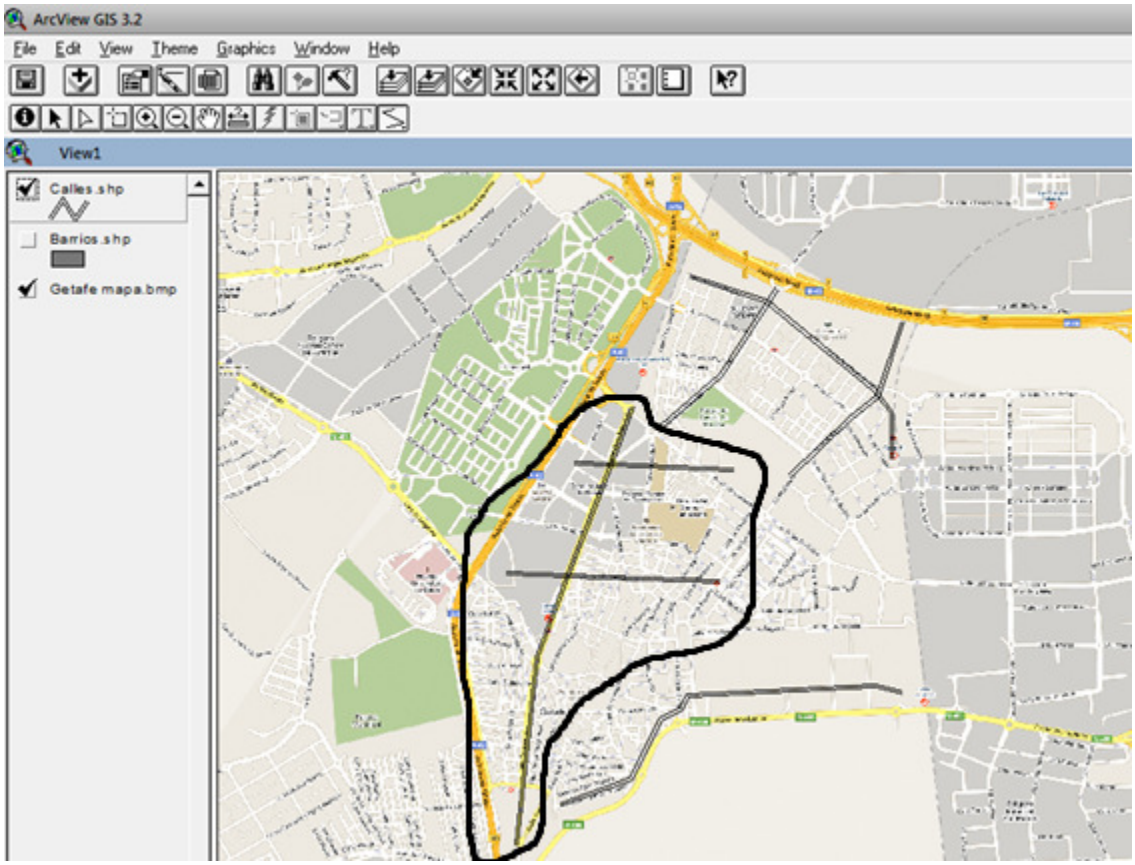


Figura 39: Ejecución correcta de SQL Loader para Calles.ctl.

NOTA IMPORTANTE: Para que ArcView considere una calle dividida en otras, tiene que utilizarse la opción de línea correspondiente a la hora de dibujarla: *"Draw line to split Feature"*. En el caso de la figura mostrada, todas las calles que cortan a otras han sido dibujadas utilizando esta opción.

El hecho de dibujar dos líneas cruzadas sin utilizar este modo, es ignorado por el programa y considerado como dos calles independientes.

6.2 Funciones

Se han implementado funciones sobre todo para el cálculo de datos estadísticos, como densidad de población, porcentaje de zona urbana, industrial y rural, así como otras para calcular algún atributo derivado, como son el número de empresas de cada barrio.

6.2.1 Función CALCULA_DENSIDAD

```
CREATE OR REPLACE FUNCTION CALCULA_DENSIDAD (P_BARRIO   BARRIO.BARRIO%TYPE)
RETURN  NUMBER
     IS
              v_personas   NUMBER;
              v_barrio            VARCHAR2(25);
              v_area              NUMBER;
              v_result            NUMBER;
BEGIN

   SELECT COUNT(*) AS personas INTO v_personas
   FROM PERS_BARRIO
   WHERE barrio = p_barrio;

   IF v_personas = 0 THEN
     RAISE_APPLICATION_ERROR(-20002, 'El barrio no existe o no tiene
habitantes' );
   END IF;

   SELECT SDO_GEOM.SDO_AREA(BARRIO.GEOM, 0.005, 'UNIT=SQ_KM') INTO v_area
   FROM barrio
   WHERE barrio.barrio = p_barrio;
   v_result := round( v_personas / v_area, 6);
DBMS_OUTPUT.PUT_LINE('LA DENSIDAD DE ' ||p_barrio ||  ' ES:' ||
TO_CHAR(v_result) || ' HABITANTES POR KM CUADRADO');
       RETURN v_result;
  END CALCULA_DENSIDAD;
```

- **Objetivo**: Calcular el porcentaje de habitantes por kilómetro cuadrado en un determinado barrio.
- **Argumentos de entrada**: Barrio para el que se quiere calcular la densidad de habitantes.
- **Argumentos de salida**: Número de habitantes por kilómetro cuadrado.
- **Requisitos**: Para esta función debemos contar con una tabla BARRIO y una vista creada llamada PERS_BARRIO con datos insertados en ellas.
- **Batería de pruebas**

  A continuación exponemos una serie de casos para probar el funcionamiento de esta función.

1. **Introducir un barrio que no existe.**

Para este particular, la función debe mostrar un mensaje en el que indica que no ha encontrado ningún dato para dicho barrio. Si introducimos el siguiente código en SQL Plus invocando a la función con el nombre de un barrio inexistente en nuestra base de datos:

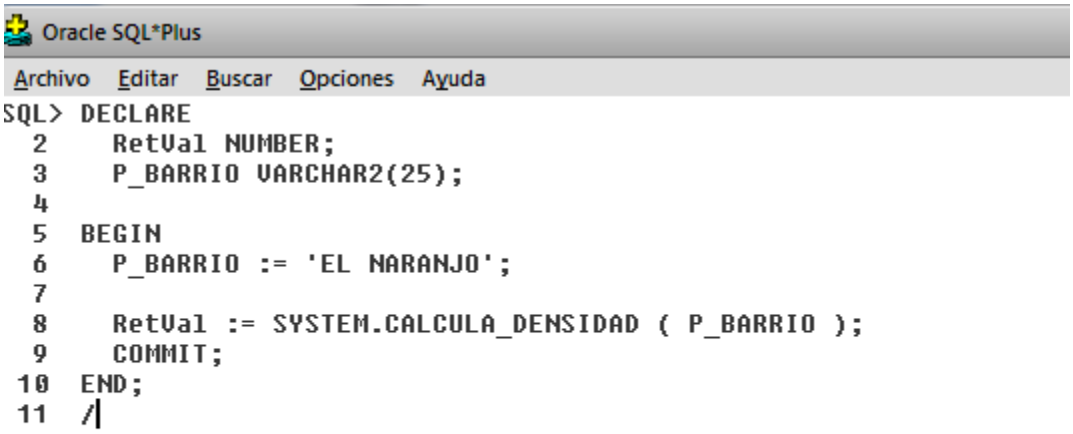


Figura 40 . Invocación a la función Calcula_densidad

SQL Plus nos devuelve el siguiente mensaje:

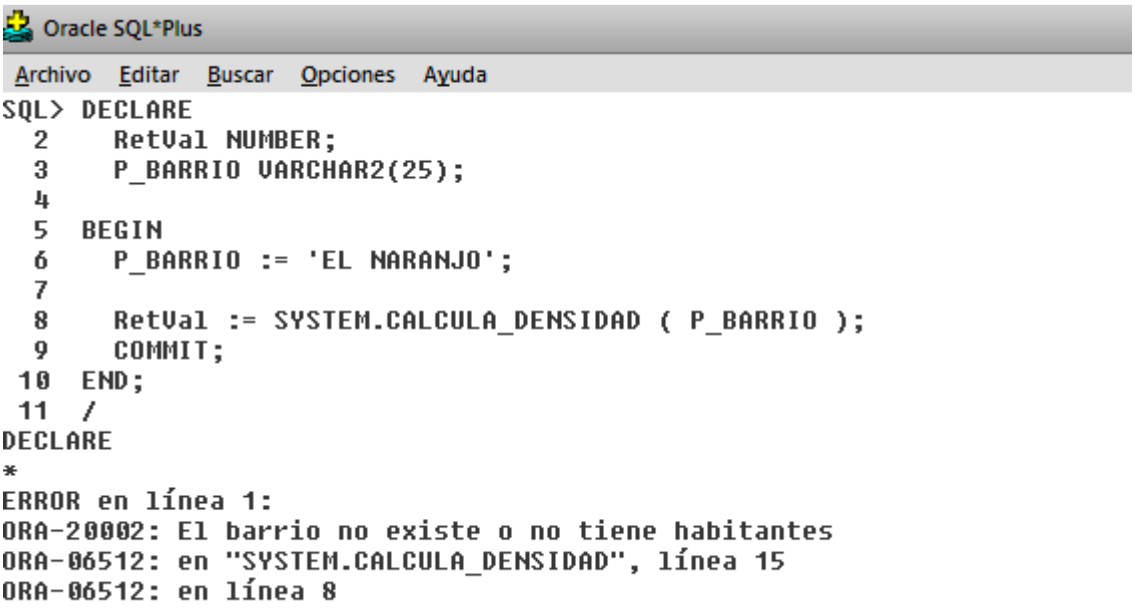


Figura 41. Mensaje de error de la función Calcula_densidad

2. Introducir un tipo de datos erróneo.

En este caso, si se introduce un número, la función devolverá el mismo error que en el apartado anterior.

3. Introducir un barrio correcto.

La función devolverá el número de habitantes por kilómetro cuadrado del barrio que hayamos introducido como parámetro.

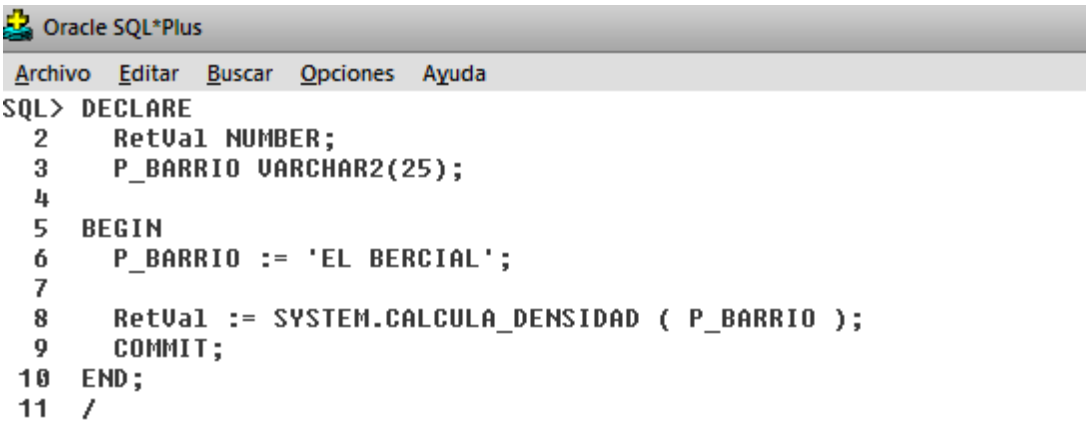


Figura 42. Invocación correcta a la función Calcula_densidad

El resultado de la función será el siguiente:

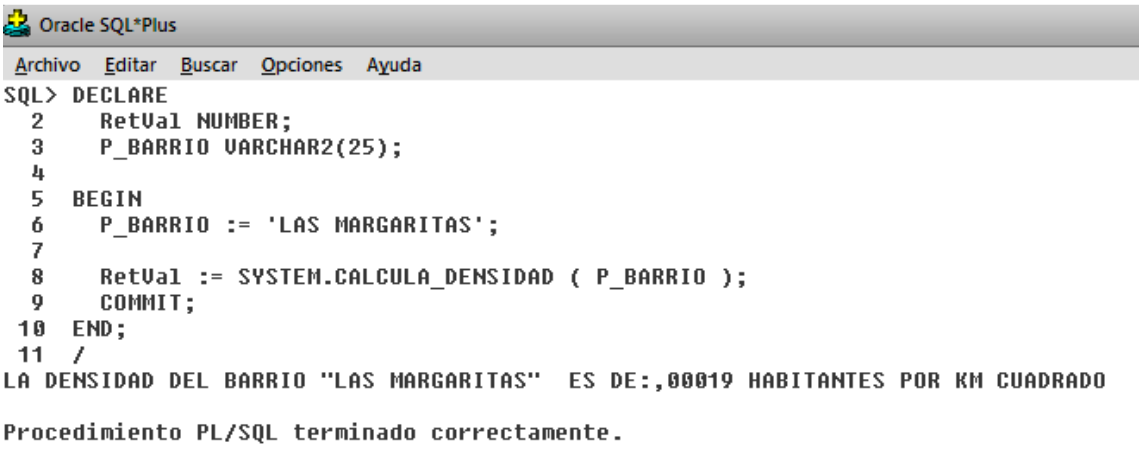


Figura 43. Resultado de la función Calcula_densidad

Este resultado tan bajo es debido a que se han introducido pocos habitantes respecto al área del barrio.

6.2.2 Función PORCENTAJE_ZONA_URB

- **Código**

```
CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_URB (P_CIUDAD CIUDAD.NOMBRE%TYPE)
  RETURN NUMBER
  IS
       v_area_ciudad     NUMBER;
       v_area_zonas      NUMBER;
       v_result          NUMBER;
BEGIN

       SELECT SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM') INTO
       v_area_ciudad
       FROM ciudad
       WHERE ciudad.nombre = p_ciudad;

       SELECT SUM(SDO_GEOM.SDO_AREA(manzana.GEOM, 0.005, 'UNIT=SQ_KM' )) INTO
       v_area_zonas
       FROM manzana;
       v_result := round( v_area_zonas /v_area_ciudad,6) * 100;

DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA URBANA DE ' || p_ciudad ||  ' ES:
' ||TO_CHAR(v_result) || ' %');
RETURN v_result;
END PORCENTAJE_ZONA_URB;
   /
```

- **Objetivo**: Esta función tiene un objetivo similar al anterior, salvo que en este caso, no tenemos en cuenta el número de habitantes, ni los barrios, ya que calculamos el porcentaje de zona urbana que tiene una ciudad determinada.
- **Argumentos de entrada**: Ciudad para la que se quiere realizar el cálculo.
- **Argumentos de salida**: Valor decimal del porcentaje calculado.
- **Requisitos**: Debemos contar con una tabla CIUDAD y una tabla MANZANA con datos insertados en ellas.
- **Batería de pruebas:** Las pruebas que se le pueden hacer a esta función son similares a la anterior, ya que sólo acepta un parámetro de entrada.

1. **Introducir una ciudad inexistente**

Debe mostrar un mensaje de error informando de ello. Probamos con el siguiente código:

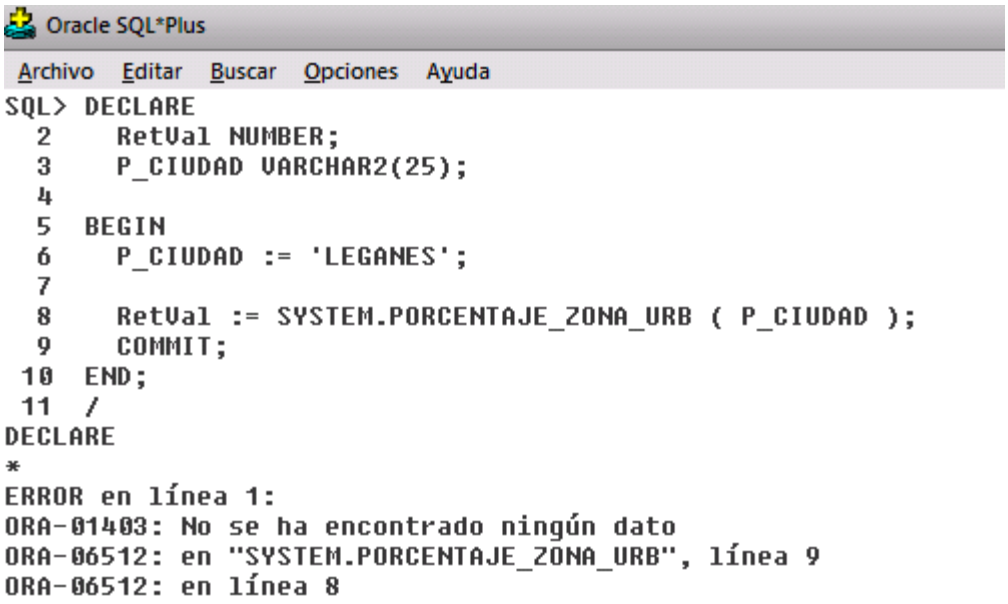


Figura 44. Invocación errónea a la función Porcentaje_Zona_Urb

2. Introducir un tipo de datos erróneo.

En este caso, si se introduce un número, la función devolverá el mismo error que en el apartado anterior.

3. Introducir una ciudad correcta

La función debe devolver el porcentaje de zona urbana de la ciudad introducida. Si se introduce la siguiente llamada:

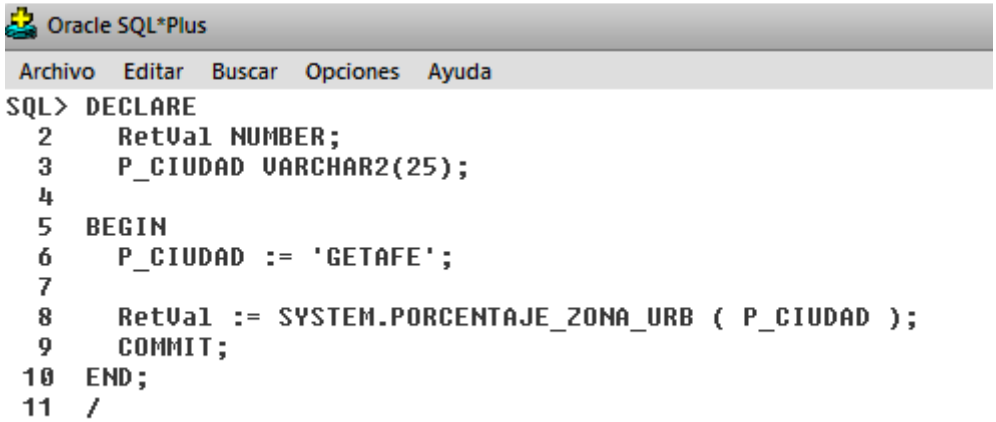


Figura 45. Invocación correcta a la función Porcentaje_Zona_Urb

La función debe mostrar el siguiente mensaje:

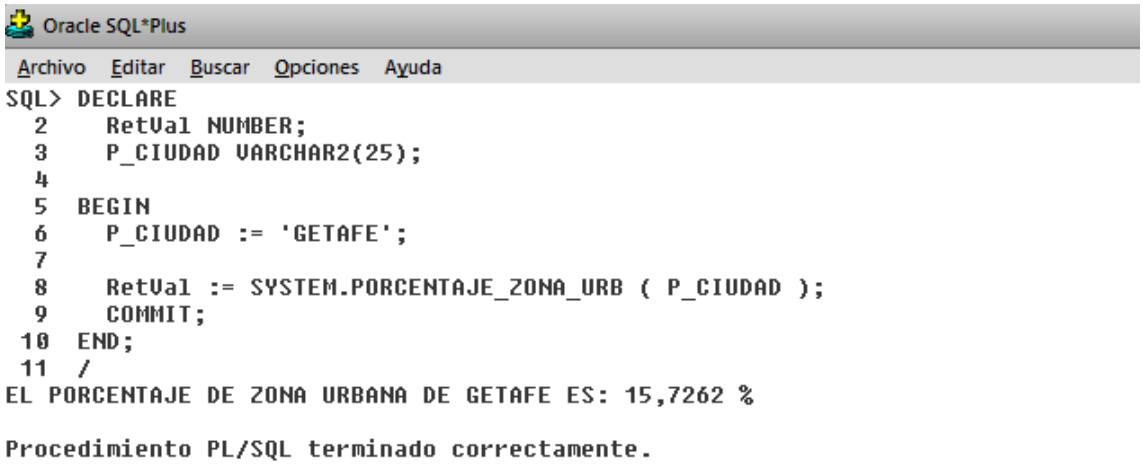


Figura 46. Resultado de la función Porcentaje_Zona_Urb

6.2.3 Función PORCENTAJE_ZONA_RURAL

- **Código**

```
  CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_RURAL (P_CIUDAD
CIUDAD.NOMBRE%TYPE)
  RETURN NUMBER
  IS
       v_area_ciudad     NUMBER;
       v_area_zonas      NUMBER;
       v_result          NUMBER;
  BEGIN
SELECT   SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM') INTO
v_area_ciudad
     FROM ciudad
     WHERE ciudad.nombre = p_ciudad;
SELECT SUM(SDO_GEOM.SDO_AREA(zona_rural.GEOM, 0.005, 'UNIT=SQ_KM' )) INTO
v_area_zonas
     FROM zona_rural;

 v_result := round( v_area_zonas /v_area_ciudad,6) * 100;

 DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA DESPOBLADA DE ' || p_ciudad ||  '
ES: ' ||TO_CHAR(v_result) || '%');
   RETURN v_result;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20000, 'Ciudad inexistente o sin datos');
   END PORCENTAJE_ZONA_RURAL;
 /
```

- **Objetivo:** Consideraremos zona rural a la zona despoblada que aún no ha sido urbanizada en una ciudad. En esta zona, no existen viviendas de ningún tipo.

- **Argumentos de entrada:** El nombre de la ciudad de la que queremos calcular el porcentaje.
- **Argumentos de salida:** El porcentaje calculado de zona despoblada de la ciudad.
- **Requisitos**: Debemos contar con las tablas CIUDAD y ZONA_RURAL con datos insertados en ellas.
- **Batería de pruebas**
  Probamos de nuevo con las posibles alternativas a la hora de invocar a esta función.

1. Ciudad inexistente

Debe mostrar el mismo mensaje de error que las anteriores funciones. Por ejemplo, si invocamos a la función con una ciudad que no tenemos almacenada en nuestra tabla CIUDAD, obtenemos lo siguiente:

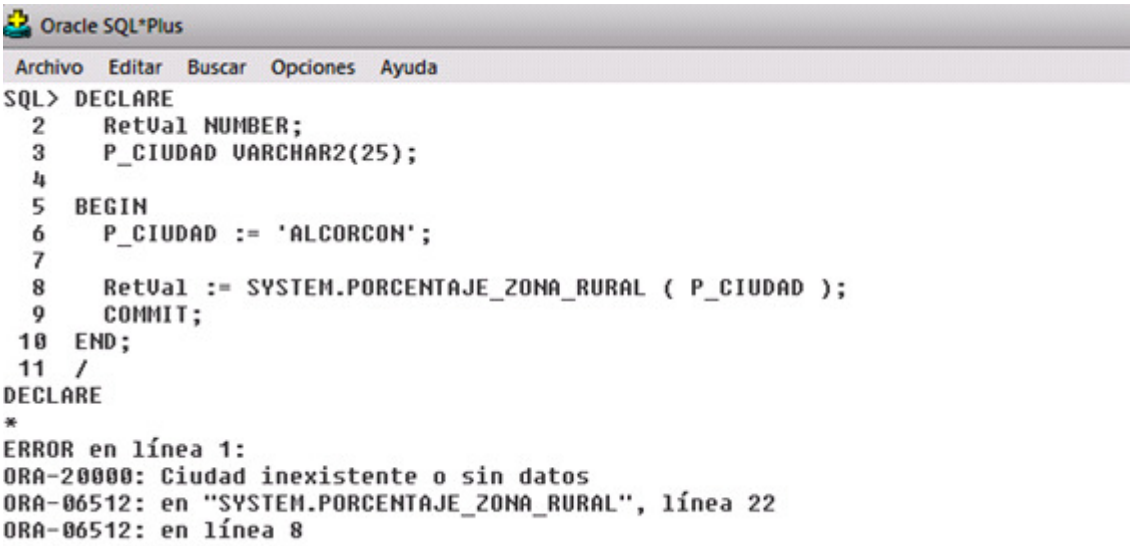


Figura 47. Mensaje de error de la función Porcentaje_Zona_Rural

2. Introducir un tipo de datos erróneo.

En este caso, si se introduce un número, la función devolverá el mismo error que en el apartado anterior.

3. Introducir una ciudad correcta

La función debe devolver el porcentaje de zona rural de la ciudad introducida. Probamos con Getafe, que es la única ciudad que tenemos almacenada, y nos debe devolver su porcentaje correspondiente:

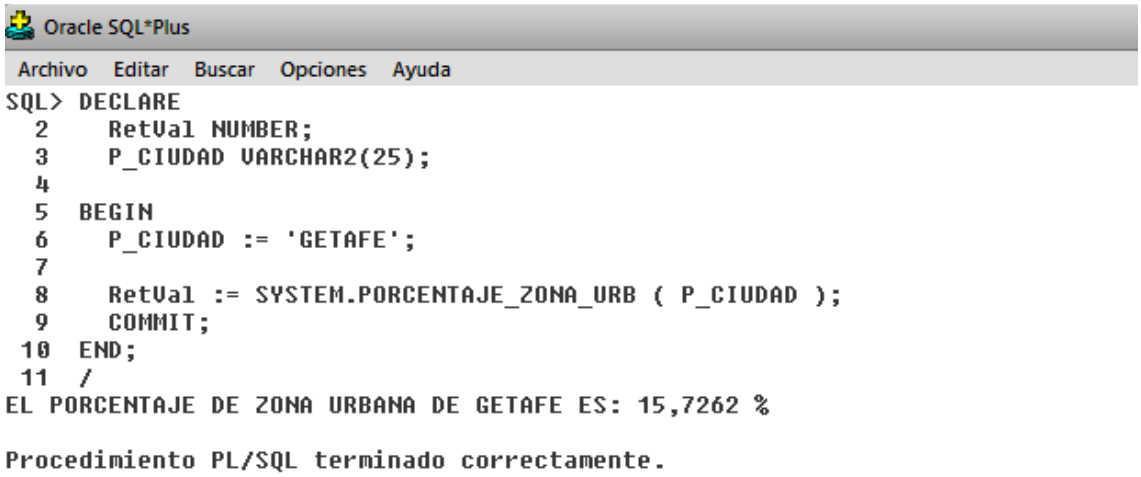


Figura 48. Mensaje devuelto por la función Porcentaje_Zona_Rural

6.2.4 Función PORCENTAJE_ZONA_INDUSTRIAL

- **Código**

```
  CREATE OR REPLACE FUNCTION PORCENTAJE_ZONA_IND (P_CIUDAD CIUDAD.NOMBRE%TYPE)
  RETURN NUMBER
  IS
        v_area_ciudad     NUMBER;
        v_area_zonas      NUMBER;
        v_result          NUMBER;
  BEGIN
   SELECT   SDO_GEOM.SDO_AREA(CIUDAD.GEOM, 0.005, 'UNIT=SQ_KM') INTO
v_area_ciudad
     FROM ciudad
     WHERE ciudad.nombre = p_ciudad;
  SELECT SUM(SDO_GEOM.SDO_AREA(zona_ind.GEOM, 0.005, 'UNIT=SQ_KM' )) INTO
v_area_zonas
      FROM zona_ind;
  v_result := round( v_area_zonas /v_area_ciudad,6) * 100;

  DBMS_OUTPUT.PUT_LINE('EL PORCENTAJE DE ZONA INDUSTRIAL DE ' || p_ciudad ||
' ES: ' ||TO_CHAR(v_result) || ' %');
   RETURN v_result;
   EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Ciudad inexistente o sin datos');
```

```
END PORCENTAJE_ZONA_IND;
```

- **Objetivo:** Esta función proporciona información sobre el porcentaje de zona industrial de una ciudad.
- **Argumentos de entrada:** Ciudad de la que se quiere calcular el porcentaje.
- **Argumentos de salida:** Porcentaje de zona industrial de la ciudad.
- **Batería de pruebas:** Las mismas que las anteriores funciones.

1. Ciudad inexistente

Llamamos a la función con una ciudad no almacenada o vacía:

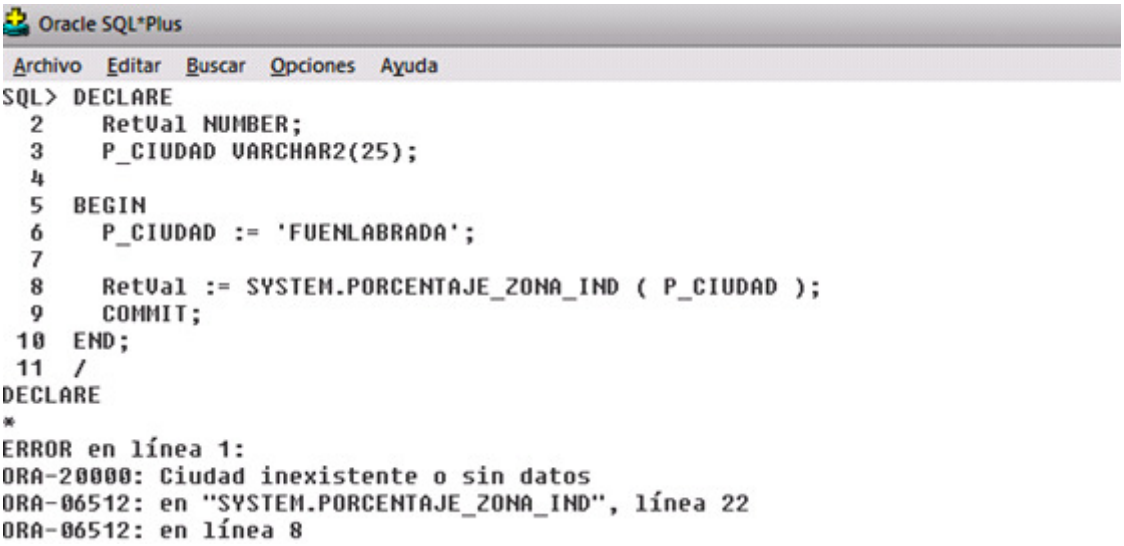


Figura 49. Mensaje de error devuelto por la función Porcentaje_Zona_Industrial

2. Introducir un tipo de datos erróneo.

En este caso, si se introduce un número, la función devolverá el mismo error que en el apartado anterior.

3. Introducir una ciudad correcta

Por último, al introducir una ciudad de la cual disponemos de datos, obtenemos el porcentaje de zona industrial:

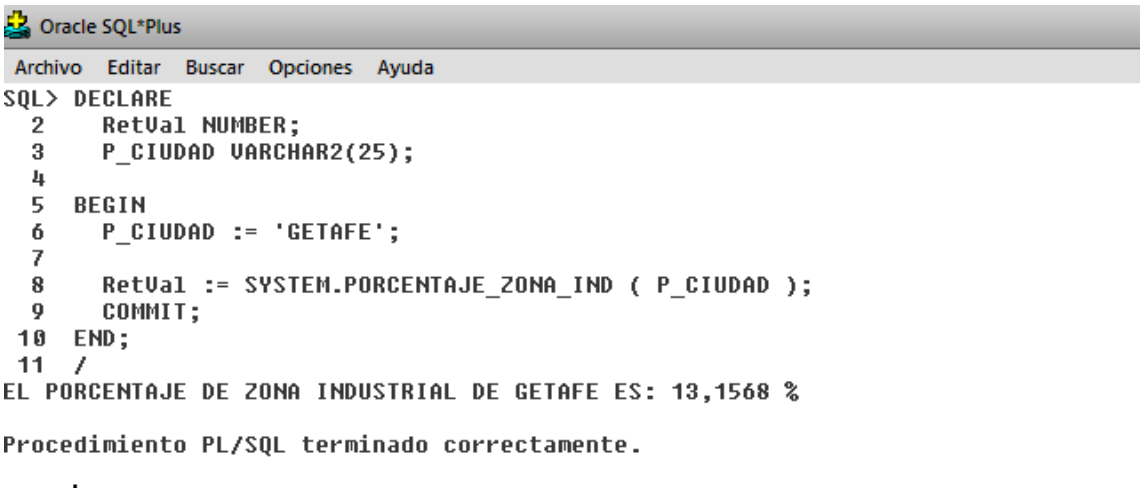


Figura 50. Resultado por la función Porcentaje_Zona_Industrial

6.3 Disparadores

Se han desarrollado disparadores para la comprobación de algunos de los requisitos que no ha sido posible recoger mediante el diseño. Otros se han implementado para la actualización de un atributo derivado como es el número de empresas de cada barrio.

6.3.1 Trigger COMPRUEBA_PERSONA

- **Código**

```
CREATE OR REPLACE TRIGGER COMPRUEBA_PERSONA
   BEFORE INSERT OR UPDATE
   ON PERSONA
   FOR EACH ROW
DECLARE
   ID_PER         TEMP_PERSONA.ID_PK%TYPE;
   ID_VIV         TEMP_PERSONA.ID_VIVIENDA_FK%TYPE;
   SIT_LAB        TEMP_PERSONA.SITUACION_LABORAL%TYPE;
   ID_EMP         TEMP_PERSONA.ID_EMPRESA_FK%TYPE;
   CIUDAD_TRAB    TEMP_PERSONA.CIUDAD_TRABAJO%TYPE;
BEGIN
   INSERT INTO TEMP_PERSONA
        VALUES (:NEW.ID_PK,
                :NEW.ID_VIVIENDA_FK,
                :NEW.SITUACION_LABORAL,
                :NEW.ID_EMPRESA_FK,
                :NEW.CIUDAD_TRABAJO);

   SELECT *
     INTO ID_PER,
          ID_VIV,
          SIT_LAB,
          ID_EMP,
          CIUDAD_TRAB
     FROM TEMP_PERSONA
    WHERE ID_PK = :NEW.ID_PK;
```

```
   IF (SIT_LAB = 'EN PARO' AND ID_EMP != 0)
   THEN
      RAISE_APPLICATION_ERROR (
         -20002,
         'La Situación laboral de esta persona no permite tener asociada
ninguna empresa');
   END IF;

   IF (SIT_LAB = 'EN PARO' AND CIUDAD_TRAB IS NOT NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20003,
         'La Situación laboral de esta persona no permite tener asociada
ninguna ciudad de trabajo');
   END IF;

   IF (CIUDAD_TRAB != 'GETAFE' AND ID_EMP IS NOT NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20004,
         'Las personas que no trabajen en Getafe no tendrán asociado ningún ID
de empresa');
   END IF;

   IF (CIUDAD_TRAB = 'GETAFE' AND SIT_LAB = 'TRABAJANDO' AND ID_EMP IS NULL)
   THEN
      RAISE_APPLICATION_ERROR (
         -20005,
         'Las personas que trabajen en Getafe tienen que tener un ID de
empresa asociado');
   END IF;
END;
```

- **Objetivo**
  Este disparador comprueba los requisitos que deben cumplir las personas registradas, que no ha sido posible recoger mediante el diseño del modelo relacional.
- **Orden:** Insert o Update.
- **Temporización:** Before.
- **Nivel:** Fila.
- **Tabla:** PERSONA.
- **Requisitos:** Por problemas de tabla mutante, se debe tener una tabla creada TEMP_PERSONA, que será actualizada y consultada por el disparador.

- **Pruebas**: Probamos con los siguientes registros a insertar en la tabla PERSONA:

1. Persona en paro con un ID de empresa asociado.

Las personas registradas que estén en situación de desempleo, no podrán tener una empresa asociada. El disparador debe mostrar el aviso cada vez que intentemos insertar un registro con estas características. Probamos con la siguiente persona:

INSERT INTO PERSONA VALUES (sec_per.nextval,226, 'EN PARO','MADRID', 'PROPIA','N', 0, 2, 2, 87);

El disparador muestra el siguiente mensaje de error:

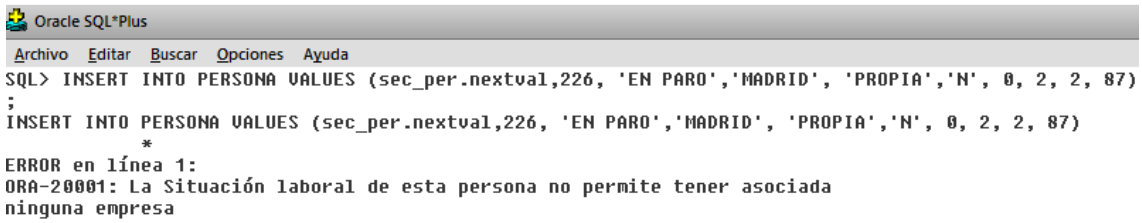
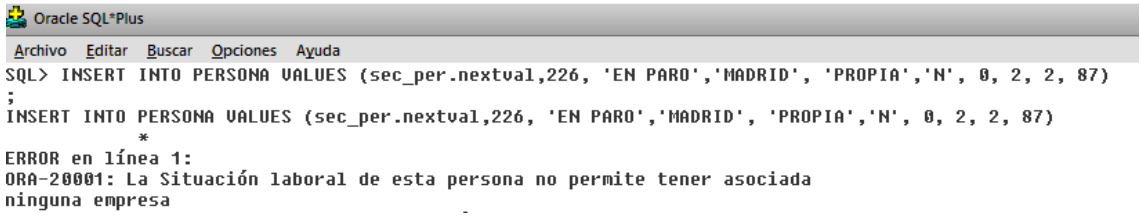


Figura 51. Ejecución del disparador Comprueba_persona

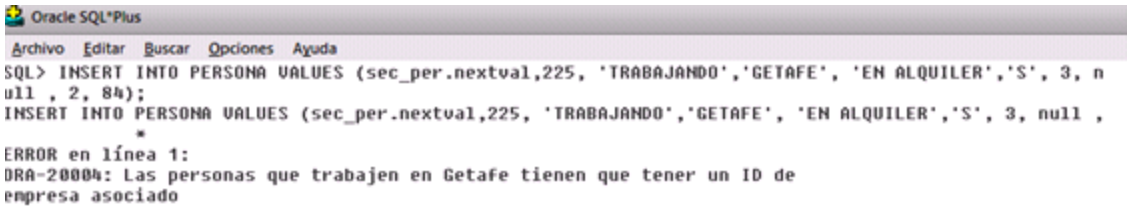
2. Persona en paro con un ciudad de trabajo asociada.

Obviamente, las personas en paro no tendrán ciudad de trabajo, por lo que al insertar un registro con estas características, obtenemos el siguiente error:



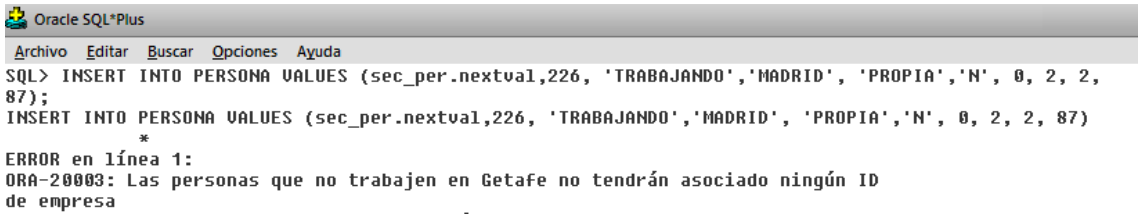
3. Persona trabajando en Getafe sin ID de empresa asociado.

Dado que sólo se almacenan empresas situadas en Getafe, por cada persona trabajando en dicha ciudad tendremos registrada la empresa donde trabaja. En caso de que no se haya asociado ninguna, el disparador muestra el siguiente aviso de error:



4. Persona trabajando fuera de Getafe con un ID de empresa asociado.

Como ya se ha indicado en el punto anterior, las empresas registradas están situadas en Getafe, por lo que una persona que trabaje en otra ciudad distinta, no puede tener ninguna empresa asociada. Al intentar introducir un registro con estas características, obtenemos el siguiente mensaje:



6.3.2 Trigger Borrado_Empresas_Barrio

- **Código**

```
CREATE OR REPLACE TRIGGER Borrado_empresas_barrio
AFTER UPDATE OR DELETE ON EMPRESA
FOR EACH ROW
DECLARE
    v_emp INTEGER;
    v_barrio barrio.barrio%TYPE;
    v_ciudad barrio.ciudad%TYPE;
BEGIN
    SELECT id_empresa_pk, barrio, ciudad INTO v_emp, v_barrio, v_ciudad
    FROM tmp_empresa
    WHERE id_empresa_pk = :old.id_empresa_pk;

     UPDATE BARRIO SET empresas = empresas - 1
     WHERE barrio.barrio = v_barrio AND
           barrio.ciudad = v_ciudad;

      END;
```

- **Objetivo:** Este disparador actualiza el campo de la tabla BARRIO por cada borrado en la tabla empresas. También se ha tenido que utilizar una tabla auxiliar por problemas de tabla mutante con EMPRESA; Por ello, como apoyo a este trigger, se ha creado el disparador Llena_temp_Empresas_barrio, que inserta en la tabla auxiliar TEMP_EMPRESA cada vez que se añade un nuevo registro en la tabla original, EMPRESA.
- **Orden:** Update o Delete.
- **Temporización:** After.
- **Nivel:** Fila.
- **Tabla:** EMPRESA.
- **Requisitos:** Por problemas de tabla mutante, se debe tener una tabla creada TMP_EMPRESA, que es consultada por el disparador.
- **Pruebas:** Basta con ejecutar el bloque anónimo cada vez que queramos actualizar la tabla y comprobar con una consulta que se han insertado el número de empresas correcto.

6.3.3 Trigger Llena_temp_empresas_barrio

- **Código**

```
   CREATE OR REPLACE TRIGGER Llena_temp_empresas_barrio
    AFTER INSERT OR UPDATE ON EMPRESA
    FOR EACH ROW
    BEGIN
       INSERT INTO TMP_EMPRESA VALUES (:new.id_empresa_pk,:new.barrio,
:new.ciudad, :new.nombre_pol);
    END;
```

- **Orden:** Insert o Update.
- **Temporización:** After.
- **Nivel:** Fila.
- **Tabla:** TEMP_EMPRESA.
- **Requisitos:** La tabla debe estar creada y tener la misma estructura que la tabla EMPRESA.
- **Pruebas**

Comprobar que por cada inserción en la tabla persona, se añade un nuevo registro en la tabla TEMP_EMPRESA.

6.3.4 Trigger Inserta_empresas_barrio

- **Código**

```
CREATE OR REPLACE TRIGGER Inserta_empresas_barrio
AFTER INSERT ON EMPRESA
FOR EACH ROW
DECLARE
    v_emp INTEGER;
    v_barrio barrio.barrio%TYPE;
    v_ciudad barrio.ciudad%TYPE;

   CURSOR c_empresas IS
      SELECT id_empresa_pk, barrio, ciudad
      FROM temp_empresa;

  BEGIN

 UPDATE BARRIO SET empresas=0;
    OPEN c_empresas;

       LOOP

          FETCH c_empresas INTO v_emp, v_barrio, v_ciudad;

          EXIT WHEN c_empresas%NOTFOUND;


          UPDATE BARRIO SET empresas = empresas + 1
           WHERE barrio.barrio = v_barrio AND
                 barrio.ciudad = v_ciudad;

       END LOOP;

    CLOSE c_empresas;

END;
```

- **Orden:** Insert.
- **Temporización:** After.
- **Nivel:** Fila.
- **Tabla:** EMPRESA.
- **Requisitos:** La tabla debe estar creada y tener datos.
- **Pruebas**

  Comprobar que por cada inserción en la tabla EMPRESA, se suma un valor al campo "empresas" de la tabla BARRIO.

7. Consultas

A continuación se mostrará el resultado de las consultas creadas para el presente trabajo, divididas por fases según su complejidad o tipo de información que retornan.

7.1 Consultas Fase I

Información específica sobre algunas características de las personas que tenemos registradas en nuestra base de datos.

```
-------------------------------------------------------------------------------
CONSULTA 1 -- TIPO DE VIVIENDA QUE MÁS PREDOMINA
-------------------------------------------------------------------------------

       DROP VIEW tipo_vivienda;

       CREATE OR REPLACE VIEW tipo_vivienda AS
       SELECT tipo_vivienda, barrio.barrio, COUNT(*) tipo
       FROM vivienda, calle, barrio
       WHERE vivienda.id_calle = calle.id_calle AND
                  vivienda.barrio = calle.nombre_bar AND
                  vivienda.ciudad = calle.ciudad AND
                   calle.nombre_bar = barrio.barrio AND
                   calle.ciudad = barrio.ciudad
       GROUP BY barrio.barrio, tipo_vivienda
       ORDER BY barrio;

       SELECT tipo_vivienda, barrio
       FROM tipo_vivienda
       WHERE tipo = (SELECT MAX(tipo) FROM tipo_vivienda);
```

Resultado

```
TIPO_VIVIENDA        BARRIO
-------------------- -------------------------
PISO                 EL BERCIAL
```

```
-------------------------------------------------------------------------------
CONSULTA 2 -- NÚMERO DE PERSONAS POR BARRIO CON SEGUNDA VIVIENDA
-------------------------------------------------------------------------------
             DROP VIEW SEG_VIV;

             CREATE OR REPLACE VIEW SEG_VIV AS
             SELECT COUNT(*) AS personas, barrio.barrio
             FROM persona, vivienda, calle, barrio
             WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
```

```
                    vivienda.id_calle = calle.id_calle AND
               vivienda.barrio = calle.nombre_bar AND
               vivienda.ciudad = calle.ciudad AND
                    calle.nombre_bar = barrio.barrio AND
               calle.ciudad = barrio.ciudad AND
                    persona.segunda_propiedad = 'S'
          GROUP BY barrio.barrio;
```

Resultado

```
  PERSONAS BARRIO
---------- -------------------------
         9 EL BERCIAL
         7 SAN ISIDRO
        11 SECTOR III
         6 LAS MARGARITAS
         9 GETAFE NORTE
         8 EL CASAR
         7 GETAFE CENTRO
```

CONSULTA 3 -- SITUACIÓN LABORAL QUE MÁS PREDOMINA POR BARRIO

```
DROP VIEW SIT_LAB_BARRIO;

CREATE OR REPLACE VIEW SIT_LAB_BARRIO AS
 SELECT MAX(situacion_laboral), barrio.barrio
 FROM persona, vivienda, calle, barrio
 WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
       vivienda.id_calle = calle.id_calle AND
       vivienda.barrio = calle.nombre_bar AND
       vivienda.ciudad = calle.ciudad AND
       calle.nombre_bar = barrio.barrio AND
       calle.ciudad = barrio.ciudad
 GROUP BY barrio.barrio;
```

Resultado

```
SITUACION_ BARRIO
---------- -------------------------
TRABAJANDO EL BERCIAL
TRABAJANDO SAN ISIDRO
TRABAJANDO SECTOR III
TRABAJANDO LAS MARGARITAS
TRABAJANDO GETAFE NORTE
TRABAJANDO EL CASAR
TRABAJANDO GETAFE CENTRO
```

CONSULTA 4 -- SITUACIÓN LABORAL QUE MÁS PREDOMINA EN GENERAL

```
DROP VIEW sit_laboral;

CREATE VIEW sit_laboral AS
SELECT situacion_laboral, barrio.barrio, COUNT(*) situacion
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
```

```
            vivienda.id_calle = calle.id_calle AND
            vivienda.barrio = calle.nombre_bar AND
            vivienda.ciudad = calle.ciudad AND
            calle.nombre_bar = barrio.barrio AND
            calle.ciudad = barrio.ciudad
 GROUP BY barrio.barrio, situacion_laboral
 ORDER BY barrio.barrio;

SELECT situacion_laboral, barrio
FROM sit_laboral
WHERE situacion_laboral = (SELECT MAX(situacion_laboral) FROM
sit_laboral);
```

Resultado

```
SITUACION_ BARRIO
---------- -------------------------
EN PARO    EL BERCIAL
```

CONSULTA 5 -- RÉGIMEN DE VIVIENDA QUE MÁS PREDOMINA POR BARRIO

```
 DROP VIEW reg_vivienda;

 CREATE VIEW reg_vivienda AS
 SELECT regimen_vivienda , barrio.barrio, COUNT(*) regimen
 FROM persona, vivienda, calle, barrio
 WHERE  persona.id_vivienda_fk = vivienda.id_vivienda AND
            vivienda.id_calle = calle.id_calle AND
            vivienda.barrio = calle.nombre_bar AND
            vivienda.ciudad = calle.ciudad AND
            calle.nombre_bar = barrio.barrio AND
            calle.ciudad = barrio.ciudad
 GROUP BY barrio.barrio, regimen_vivienda;
```

Resultado

```
REGIMEN_VIV BARRIO                       REGIMEN
----------- ------------------------- ----------
EN ALQUILER EL BERCIAL                       16
PROPIA      SAN ISIDRO                       13
EN ALQUILER GETAFE CENTRO                     7
PROPIA      GETAFE CENTRO                    13
PROPIA      SECTOR III                       12
EN ALQUILER LAS MARGARITAS                   10
PROPIA      EL BERCIAL                       13
PROPIA      LAS MARGARITAS                   10
EN ALQUILER GETAFE NORTE                     10
PROPIA      GETAFE NORTE                     10
PROPIA      EL CASAR                         17
EN ALQUILER EL CASAR                          7
EN ALQUILER SAN ISIDRO                        7
EN ALQUILER SECTOR III                        8
```

```
-------------------------------------------------------------------------------
CONSULTA 6 --  RÉGIMEN DE VIVIENDA QUE MÁS PREDOMINA EN GENERAL
-------------------------------------------------------------------------------
```

```
DROP VIEW REG_MAX;

CREATE VIEW REG_MAX AS
SELECT barrio, regimen_vivienda
FROM reg_vivienda
WHERE regimen = (SELECT MAX(regimen) FROM reg_vivienda);
```

Resultado

```
BARRIO                    REGIMEN_VIV
------------------------- -----------
EL CASAR                  PROPIA
```

```
-------------------------------------------------------------------------------
CONSULTA 7 --  NÚMERO DE PERSONAS QUE HAN CAMBIADO DE VIVIENDA
-------------------------------------------------------------------------------
```

```
DROP VIEW cambios_viv;

CREATE VIEW cambios_viv AS
 SELECT COUNT(*) AS personas, barrio.barrio
 FROM persona, vivienda, calle, barrio
 WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
           vivienda.id_calle = calle.id_calle AND
           vivienda.barrio = barrio.barrio AND
           vivienda.ciudad = barrio.ciudad AND
           calle.nombre_bar = barrio.barrio AND
           calle.ciudad = barrio.ciudad AND
           persona.cambios_vivienda != 0
 GROUP BY barrio.barrio;
```

Resultado

```
  PERSONAS BARRIO
---------- -------------------------
        15 EL BERCIAL
        12 SAN ISIDRO
        12 SECTOR III
        14 LAS MARGARITAS
        14 GETAFE NORTE
        14 EL CASAR
        11 GETAFE CENTRO
```

```
-------------------------------------------------------------------------------
CONSULTA 8 -- BARRIO EN EL QUE MÁS PERSONAS HAN CAMBIADO DE VIVIENDA
-------------------------------------------------------------------------------
```

```
DROP VIEW CAMBIOS_VIV_MAX;

CREATE OR REPLACE VIEW CAMBIOS_VIV_MAX AS

SELECT personas, barrio
 FROM cambios_viv
 WHERE personas = (SELECT MAX(personas) FROM cambios_viv);
```

Resultado

```
  PERSONAS BARRIO
---------- -------------------------
        15 EL BERCIAL
```

```
-------------------------------------------------------------------------------
CONSULTA 9 -- NÚMERO DE PERSONAS POR BARRIO QUE TRABAJAN EN GETAFE
-------------------------------------------------------------------------------
```

```
DROP VIEW trabaja_getafe;

CREATE VIEW trabaja_getafe AS
SELECT  (barrio.barrio), COUNT(*) AS personas
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
      vivienda.id_calle = calle.id_calle AND
      vivienda.barrio = calle.nombre_bar AND
      vivienda.ciudad = calle.ciudad AND
      calle.nombre_bar = barrio.barrio AND
      persona.ciudad_trabajo = 'GETAFE'
GROUP BY barrio.barrio;
```

Resultado

```
BARRIO                       PERSONAS
------------------------- ----------
EL BERCIAL                          7
SAN ISIDRO                          1
SECTOR III                          4
LAS MARGARITAS                      2
GETAFE NORTE                        2
EL CASAR                            4
GETAFE CENTRO                       1
```

```
-------------------------------------------------------------------------------
CONSULTA 10 -- BARRIO EN EL QUE MÁS PERSONAS TRABAJAN EN GETAFE
-------------------------------------------------------------------------------
```

```
DROP VIEW TRAB_GETAFE_MAX;

CREATE OR REPLACE VIEW TRAB_GETAFE_MAX AS
SELECT barrio, personas
FROM trabaja_getafe
WHERE personas = (SELECT MAX(personas) FROM trabaja_getafe);
```

Resultado

```
BARRIO                       PERSONAS
------------------------- ----------
EL BERCIAL                          7
```

```
-------------------------------------------------------------------------------
CONSULTA 11 -- NÚMERO DE PERSONAS POR BARRIO
-------------------------------------------------------------------------------
```

```
DROP VIEW PERS_BARRIO;

CREATE OR REPLACE VIEW PERS_BARRIO AS
SELECT COUNT(*) AS personas, barrio.barrio
FROM persona, vivienda, calle, barrio
WHERE persona.id_vivienda_fk = vivienda.id_vivienda AND
      vivienda.id_calle = calle.id_calle AND
      vivienda.barrio = calle.nombre_bar AND
      vivienda.ciudad = calle.ciudad AND
      calle.nombre_bar = barrio.barrio AND
      calle.ciudad = barrio.ciudad
GROUP BY barrio.barrio;
```

Resultado

```
  PERSONAS BARRIO
---------- -------------------------
        29 EL BERCIAL
        20 SAN ISIDRO
        20 SECTOR III
        20 LAS MARGARITAS
        20 GETAFE NORTE
        24 EL CASAR
        20 GETAFE CENTRO
```

```
-------------------------------------------------------------------------------
CONSULTA 12 -- NÚMERO DE EMPRESAS POR BARRIO
-------------------------------------------------------------------------------
```

```
DROP VIEW empresas_barrio;

CREATE VIEW empresas_barrio AS
SELECT COUNT(*) AS empresas, barrio.barrio
FROM empresa, barrio
WHERE empresa.barrio = barrio.barrio AND
```

```
            empresa.ciudad = barrio.ciudad
      GROUP BY barrio.barrio;
```

Resultado

```
  EMPRESAS BARRIO
---------- -------------------------
         5 EL BERCIAL
         4 SAN ISIDRO
         6 GETAFE INDUSTRIAL
         2 LAS MARGARITAS
         2 GETAFE NORTE
         8 GETAFE CENTRO
         1 EL CASAR
```

CONSULTA 13 -- BARRIO QUE MÁS EMPRESAS TIENE

```
      DROP VIEW EMP_BARRIO;

      CREATE OR REPLACE VIEW EMP_BARRIO AS
      SELECT empresas, barrio
      FROM empresas_barrio
      WHERE empresas = (SELECT MAX(empresas) FROM empresas_barrio);
```

Resultado

```
  EMPRESAS BARRIO
---------- -------------------------
         8 GETAFE CENTRO
```

CONSULTA 14 -- NÚMERO DE CALLES DE CADA BARRIO

```
        DROP VIEW BARRIO_CALLES;

        CREATE VIEW barrio_calles AS
           SELECT COUNT(*) AS calles, barrio
           FROM calle, barrio
           WHERE calle.nombre_bar = barrio.barrio AND
                 calle.ciudad = barrio.ciudad
           GROUP BY barrio.barrio;
```

Resultado

```
    CALLES BARRIO
---------- -------------------------
        16 EL BERCIAL
        10 SAN ISIDRO
        33 SECTOR III
        14 LAS MARGARITAS
        15 GETAFE NORTE
        14 EL CASAR
        20 GETAFE CENTRO
```

```
-------------------------------------------------------------------------------
CONSULTA 15 --  BARRIO QUE MÁS CALLES TIENE
-------------------------------------------------------------------------------
```

```
DROP VIEW MAX_BARRIO_CALLES;

CREATE OR REPLACE MAX_BARRIO_CALLES AS
SELECT calles, barrio
FROM barrio_calles
WHERE calles = (SELECT MAX(calles) FROM barrio_calles);
```

Resultado

```
    CALLES BARRIO
---------- -------------------------
        33 SECTOR III
```

7.2 Consultas Fase II

Consultas sencillas con algunas funciones geométricas de Oracle Spatial, como cálculo de áreas y distancias entre tablas geométricas.

```
-------------------------------------------------------------------------------
CONSULTA 1 -- ÁREA DE TODOS LOS BARRIOS EN KM CUADRADOS
-------------------------------------------------------------------------------
```

```
DROP VIEW AREA_BARRIO;

CREATE OR REPLACE AREA_BARRIO AS
SELECT BARRIO.BARRIO, ROUND(SDO_GEOM.SDO_AREA(BARRIO.GEOM, 0.005,
'UNIT=SQ_KM' ),2) AS Área
FROM barrio;
```

Resultado

```
BARRIO                         ÁREA
------------------------- ----------
GETAFE NORTE                164553,19
LAS MARGARITAS              104382,44
SECTOR III                   320463,5
GETAFE INDUSTRIAL           273335,63
EL CASAR                    160268,21
SAN ISIDRO                  227677,64
GETAFE CENTRO               156979,58
EL BERCIAL                  130874,39
```

```
--------------------------------------------------------------------------------
-- CONSULTA 2 --  ÁREA DE LOS POLÍGONOS INDUSTRIALES
--------------------------------------------------------------------------------

     DROP VIEW AREA_ZONA_IND;

     CREATE OR REPLACE VIEW AREA_ZONA_IND AS
     SELECT NOMBRE_POL, ROUND(SDO_GEOM.SDO_AREA(ZONA_IND.GEOM, 0.005,
     'UNIT=SQ_KM' ),2) AS Área
     FROM ZONA_IND;
```

Resultado

```
NOMBRE_POL                ÁREA
------------------- ----------
SAN MARCOS             79812,17
EL ROSON               74980,14
LOS ANGELES            93798,59
LOS OLIVOS              9101,52
EL CULEBRO             12849,07
```

```
--------------------------------------------------------------------------------
-- CONSULTA 3 --  ÁREA DE LAS ZONAS RURALES
--------------------------------------------------------------------------------

      DROP VIEW AREA_ZONA_RURAL;

      CREATE OR REPLACE VIEW AREA_ZONA_RURAL AS
      SELECT BARRIO, ROUND(SUM(SDO_GEOM.SDO_AREA(ZONA_RURAL.GEOM, 0.005,
       'UNIT=SQ_KM')),2)  AS Area
      FROM ZONA_RURAL
      GROUP BY BARRIO;
```

Resultado

```
BARRIO                         AREA
------------------------- ----------
EL BERCIAL                  18943,86
SAN ISIDRO                 173245,42
SECTOR III                 420544,86
LAS MARGARITAS              48349,75
GETAFE NORTE                 17587,5
EL CASAR                    41605,84
GETAFE CENTRO                6568,84
```

-- CONSULTA 4 -- ÁREA DE LAS ZONAS URBANAS

```
DROP VIEW AREA_ZONA_URB;

CREATE OR REPLACE VIEW AREA_ZONA_URB AS
SELECT BARRIO, ROUND(SUM(SDO_GEOM.SDO_AREA(MANZANA.GEOM, 0.005,
'UNIT=SQ_KM')),2)  AS Area
 FROM MANZANA
 GROUP BY BARRIO;
```

Resultado

```
BARRIO                         AREA
------------------------- ----------
EL BERCIAL                  57674,87
SAN ISIDRO                  21453,46
SECTOR III                  58638,32
LAS MARGARITAS              20595,56
GETAFE NORTE                59365,97
EL CASAR                    54411,76
GETAFE CENTRO                51237,3
```

CONSULTA 5 -- DISTANCIA ENTRE BARRIOS Y CALLES

```
DROP VIEW DIST_BARRIO_CALLE;

CREATE OR REPLACE DIST_BARRIO_CALLE AS
   SELECT ROUND(SDO_GEOM.SDO_DISTANCE(barrio.geom, calle.geom,0.005),2) AS
   DISTANCIA
   FROM barrio, calle
   WHERE barrio.barrio = 'EL BERCIAL' AND
           calle.nombre_cal = 'YELMO';
```

Resultado

```
 DISTANCIA
----------
 683216,53
```

7.3 Consultas Fase III

Datos que nos proporcionan información más precisa sobre determinadas características y necesidades de la población.

```
--------------------------------------------------------------------------------
CONSULTA 1 -- NECESIDAD DE VIVIENDA POR BARRIOS
       CONDICIONES:
              SITUACION LABORAL: TRABAJANDO
              CIUDAD DE TRABAJO:GETAFE
              REGIMEN VIVIENDA: EN ALQUILER
              SEGUNDA PROPIEDA: NO
              CAMBIOS VIVIENDA DISTINTO A CERO
              ANTIGUEDAD ALTA DE LA VIVIENDA
              EDAD: GRUPOS 1 Y 2.
--------------------------------------------------------------------------------

       DROP VIEW DEMANDA_VIV;

       CREATE OR REPLACE DEMANDA_VIV AS
       SELECT COUNT(*) AS personas, barrio.barrio
       FROM grupo_edades, persona, vivienda, manzana, barrio, calle
       WHERE (persona.id_grupo_edad_fk > 1) AND
              (persona.id_grupo_edad_fk <= 3) AND
               persona.situacion_laboral = 'TRABAJANDO' AND
               persona.ciudad_trabajo= 'GETAFE' AND
               persona.segunda_propiedad = 'N' AND
               persona.cambios_vivienda <= 2  AND
               persona.id_manzana_fk = manzana.id AND
               manzana.antiguedad > 20 AND
               persona.id_vivienda_fk = vivienda.id_vivienda AND
               vivienda.id_calle = calle.id_calle AND
               calle.nombre_bar = barrio.barrio AND
               calle.ciudad = barrio.ciudad
  GROUP BY barrio.barrio;
```

Resultado

```
  PERSONAS BARRIO
---------- -------------------------
        10 EL BERCIAL
         5 SECTOR III
```

Con los datos actuales, observamos que sólo sólo 10 y 5 personas de El BERCIAL y EL SECTOR III respectivamente, cumplen las condiciones de la consulta.

```
-------------------------------------------------------------------------------
-- CONSULTA 2 -- DEMANDANTES DE VIVIENDA POR CERCANÍA A EMPRESAS
 CONDICIONES:
       TRABAJANDO EN  UNA DETERMINADA EMPRESA DE GETAFE.
       EDAD: GRUPOS 1, 2 y 3.
       EL BARRIO DONDE TRABAJA Y LA MANZANA DONDE RESIDE DEBEN GUARDAR LA
       DISTANCIA ESPECIFICADA
-------------------------------------------------------------------------------


DROP VIEW DIST_TRABAJO;

     CREATE OR REPLACE VIEW DIST_TRABAJO AS
     SELECT DISTINCT persona.id_manzana_fk AS MANZANA, persona.id_pk as
       PERSONA, empresa.barrio AS BARRIO_EMPRESA
     FROM persona, manzana, empresa, barrio
     WHERE         (persona.id_grupo_edad_fk > 1) AND
                   (persona.id_grupo_edad_fk <= 4) AND
                    persona.id_manzana_fk = manzana.id and
                    persona.ciudad_trabajo = 'GETAFE' AND
                    persona.id_empresa_fk = empresa.id_empresa_pk AND
                    empresa.barrio = barrio.barrio   AND

    SDO_GEOM.SDO_DISTANCE(BARRIO.GEOM, MANZANA.GEOM, 0.005, 'UNIT=KM') > 40;
```

Resultado

MANZANA	PERSONA	BARRIO_EMPRESA
50	38	SAN ISIDRO
50	32	SAN ISIDRO
84	135	SAN ISIDRO
10	16	SAN ISIDRO
0	5	SAN ISIDRO
0	8	SAN ISIDRO
15	28	SAN ISIDRO
85	142	GETAFE CENTRO

En la imagen se muestran las personas que cumplen los requisitos de la consulta, junto con la manzana en la que viven y el barrio donde está situada la empresa en la que trabajan.

7. CONCLUSIONES

A nivel personal, la realización de este trabajo ha supuesto ampliar el conocimiento sobre todas las tecnologías utilizadas, tanto en bases de datos (relacionales, espaciales, diseño e implementación, recogida de requisitos, etc.), como en su integración con un sistema de información geográfica, área de la que partía con un conocimiento inicial en otra herramienta, y que me ha servido para tener la iniciativa de implementar este sistema.

La idea original que se tuvo al comenzar este trabajo era la de crear una aplicación que almacenara información relevante dentro de un entorno urbanístico, teniendo en cuenta también que queríamos explotar estos datos atendiendo a diversos factores, como por ejemplo características poblacionales como hábitos, estilo de vida, preferencias en cuanto a vivienda, para así poder realizar estudios que nos aportaran el conocimiento que en ese momento necesitáramos o requiriese un cliente que podría ser una inmobiliaria, un ayuntamiento o simplemente una empresa privada que quisiera tener esa información para cualquier actividad.

En definitiva, este estudio pretendía seguir las líneas de lo que sería un sistema de explotación de información y a su vez, decisional, que aportase una foto de lo que en este momento está sucediendo en un lugar determinado.

En el apartado de líneas futuras, se introduce brevemente este concepto, mostrando un ejemplo de la transformación que sufriría el modelo relacional de la base de datos para convertirse en este tipo de sistema.

8. PRESUPUESTO

Para este trabajo, se ha calculado un presupuesto desglosándolo en las distintas fases en las que se ha ido desarrollando.

1. Formación.
2. Análisis y Diseño.
3. Implementación.
4. Experimentación.
5. Documentación.

El detalle de cada una de ellas se incluye en la hoja Excel que acompaña a este proyecto. El coste de cada fase y el total del presupuesto se ha considerado como un proyecto real, en el que inicialmente trabajarían un programador junior y un analista programador, con un coste/hora de 30 y 60 € respectivamente y siempre en función de una jornada laboral de 8 horas.

En base a esto, el presupuesto ha quedado de la siguiente manera:

Total Formación	5.760,00 €
Total Análisis y Diseño	11.700,00 €
Total Implementación	6.240,00 €
Total Experimentación	3.200,00 €
Total Documentación	8.000,00 €
COSTE TOTAL:	34.900,00 €

Tal y como se especifica en la tabla, el coste total de proyecto ascendería a 34.720,00 €.

El tiempo estimado para este primer alcance, se resume en la siguiente tabla:

Total Formación	192
Total Análisis y Diseño	195
Total Implementación	188
Total Experimentación	80
Total Documentación	200
HORAS TOTALES:	855

Teniendo en cuenta que la jornada laboral será de 8 horas, los días que se estiman son 106.9 ≈ 107, que en meses, serían aproximadamente 3 y medio.

9. LÍNEAS FUTURAS Y POSIBLES MEJORAS

Una mejora importante a tener en cuenta en referencia en cuanto al futuro de este trabajo es la introducción de la dimensión tiempo. En este caso, dotaría al proyecto de mayor funcionalidad, ya que podríamos tener distintas versiones a modo de historial, por ejemplo, de la evolución del terreno en cuanto a edificaciones, fenómenos naturales, crecimiento de una ciudad, etc.

Por otro lado, partiendo del modelo del que disponemos ahora, se pueden añadir múltiples análisis además de los que se han hecho. Por ejemplo, se pueden considerar dimensiones nuevas como la cercanía entre viviendas, las zonas libres construibles y no construibles, el nivel de industrialización y de urbanización de una ciudad, el nivel de contaminación, como por ejemplo el ruido, la polución, etc., registrando estos datos para cada ciudad o, en un nivel más bajo, para cada barrio.

A su vez, para enriquecer el carácter de cada funcionalidad o análisis, se pueden utilizar estos valores para la toma de decisiones de varios tipos, como puede ser desde el punto de vista de un ayuntamiento, para obtener información de la conveniencia o no de edificar en una determinada zona o de una constructora, para comprar viviendas en una ciudad, barrio o porción de terreno.

Como mejora final, propondría la transformación del modelo relacional en un modelo decisional, es decir, aquel desarrollado específicamente en proyecto de toma de decisiones o Business Intelligence, para optimizar aquellas consultas que requieran de muchas uniones en una base de datos transaccional. Una posible alternativa sería el diseño de una base de datos ROLAP.

A continuación se muestra lo que sería un sencillo ejemplo de este tipo de modelo, donde Barrio sería la tabla de hechos o central y las tablas que se relacionan con ella serían las que almacenarían los parámetros en función de los cuales varía la magnitud a estudiar, estos parámetros reciben el nombre de dimensiones.

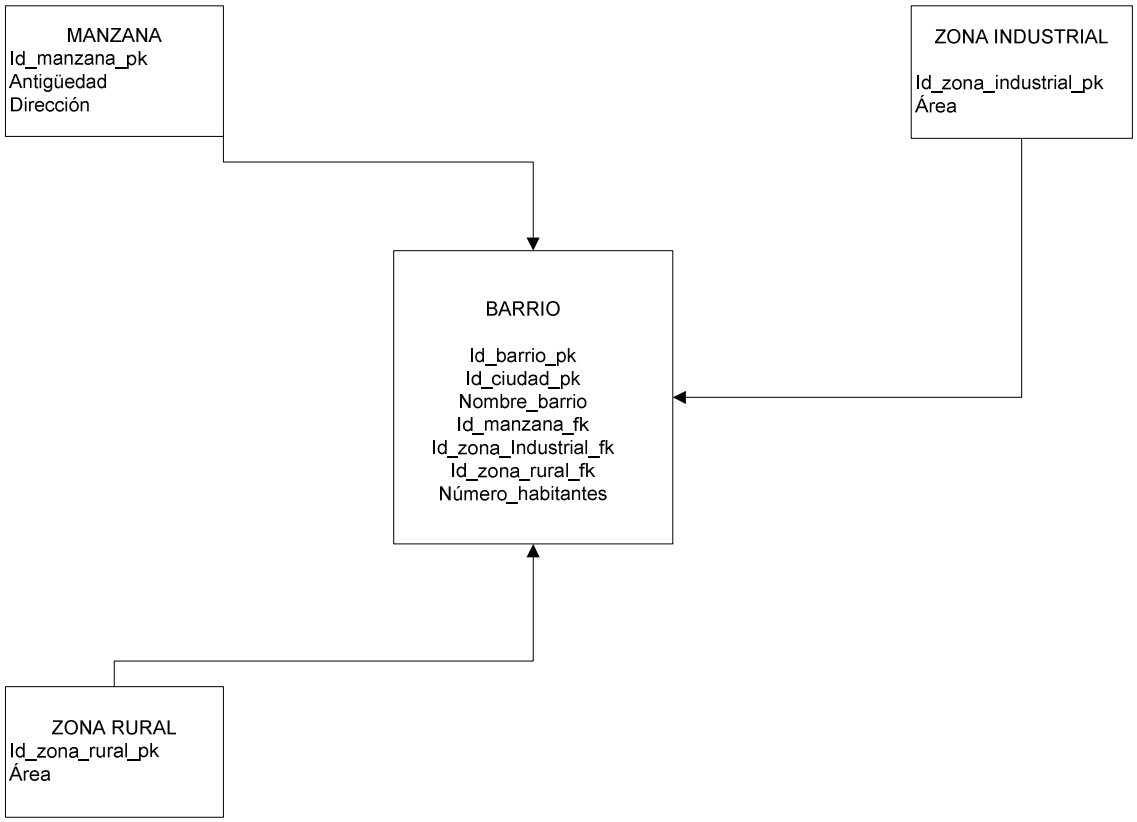


Figura 52. Ilustración de un sistema ROLAP para la tabla BARRIO.

10. REFERENCIAS Y ENLACES DE INTERÉS

[1] Oracle Pl/SQL. Pérez, César. Editorial Ra-Ma.

[2] Oracle Spatial User's Guide and Reference 10g Release 1 (10.1), Oracle 2003.

[3] Oracle Database Administrator's Guide 10g Release 1 (10.1), Oracle 2003.

[4] Oracle Geometry Functions: http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96630/sdo_objgeom.htm

[5] Bases de Datos: Persistencia del Espacio-Tiempo. Elena Navarro, Jesús Damián García-Consuegra, Nikos Lorentzos. Departamento de informática. Escuela politécnica superior. Universidad de Castilla-la Mancha.

[6] ArcView GIS: the geographic information system for everyone. ESRI, cop. 1996.

[7] Página web de ESRI ArcInfo: www.esri.com

[8] Las coordenadas geográficas y la proyección UTM. Ignacio Alonso Fernández-Coppel. Departamento de Ingeniería Agrícola y Forestal Escuela Técnica Superior de Ingenierías Agrarias. Palencia. UNIVERSIDAD DE VALLADOLID

[10] Web dedicada a la cartografía y temas relacionados: www.cartesia.org