

2011

**Implementación de Datawarehouse**

**Nombre del Caso: “Data Warehousing y OLAP para la Industria de Comidas Rápidas”**

**Sistemas de Gestión 2 – Abraham, Botta, Fratte**

**ÍNDICE**

# DESCRIPCIÓN DEL CASO

## Introducción

La industria de comida rápida es altamente competitiva, un cambio muy pequeño en las operaciones pueden tener un impacto significativo en la línea de operaciones. Por esta razón, el acceso rápido a la información completa para la presentación de informes estándar y bajo demanda, es esencial.

DecisionSoft ha diseñado e implementado un Data Warehouse para hacer frente a esta exigencia de Springwood Corporation, una franquicia de comida rápida que opera aproximadamente 20 Taco TABASCO y restaurantes Kentucky Fried Chicken alrededor del país. El Springwood Data Warehouse proporciona ahora soporte de decisiones estratégicas y tácticas para todos los niveles de gestión dentro de Springwood.

El almacén de datos se implementa en Microsoft SQL Server 2000, e incorpora los datos de dos fuentes principales:

* Todos los días información de ventas es suministrada de forma automática por el sistema de facturación.
* Información contable desde la base de datos de contabilidad Dynamics (Microsoft Great Plains).

Estos datos multidimensionales, se actualizan automáticamente de forma periódica (o bajo demanda, si es necesario) y se mantienen históricamente durante varios años con fines comparativos.  
A efectos de información y análisis, los datos en el almacén se transforman en cubos OLAP. Los cubos se acceden a través de Excel utilizando MD BusinessQuery.

Los datos pueden ser analizados por tienda, por empresa, por área y zona, por año, trimestre y el mes, y por la marca y el concepto.

## Áreas Temáticas

1. **Datos de pérdidas y ganancias y flujo de caja para Springwood**. Las cantidades se pueden ver durante un período, trimestre a la fecha, de año a la fecha, y se puede comparar con cualquiera de los dos presupuestos, en comparación con el mismo período del año anterior, o como porcentaje de las ventas.
2. **Mezcla de Ventas**: Contiene las ventas diarias de todos los elementos del menú en todas las tiendas. Además de los parámetros de análisis estándar, estos datos también puede ser analizado por marca, por tipo de posición o elemento de menú, por año calendario, mes y semana, y por el nivel de precios. Este cubo se puede utilizar para calcular los importes de ventas, los costos y la variación de precio de lista.  
   Cubos de Ventas (partes del día). Contiene los importes de ventas y cuenta en intervalos de 15 minutos. Además de los parámetros de análisis estándar, los datos de este cubo también se pueden analizar por año calendario, mes y semana, y por ocho horas, cuatro horas, dos horas, una hora y 15 minutos de intervalo, o específicos comida (por ejemplo, almuerzo, cena, desayuno, entre comidas, etc.)
3. **Resumen de Ventas Diarias**. Contiene resumen diario de ventas de cada tienda. Además de los parámetros de análisis estándar, estos datos también pueden ser dimensionado por un indicador comparable, por año calendario, mes y semana, y por el nivel de precios. Las ventas brutas, las ventas gravadas, las ventas no gravadas. Muchas cantidades se pueden ver, opcionalmente, como las variaciones, como un porcentaje de las ventas, o se resumen en la semana a la fecha, período, mes.
4. **Reportes de Costos de Ventas**. Muestra la cantidad y el costo promedio de artículos, considerando como unidad mínima de análisis, el ítem de menú. Debe realizarse un análisis geográfico de costos, e incluyendo la estructura corporativa.
5. **Análisis de Motivación y satisfacción de empleados**, midiendo respuestas positivas de encuestas de satisfacción. Deberá organizarse por tienda, zona, categoría de empleados, y analizar como varía el hecho en los distintos turnos.

## Estructura de Datos

La estructura de datos implementada para Springwood les permite mantener varias estructuras organizativas distintas con el fin de representar adecuadamente a cada tienda en un

* La estructura corporativa, es decir, la filial de que pertenecen,
* La estructura de operaciones, es decir, la zona o área y
* El concepto de estructura, es decir, KFC, la TB (Taco TABASCO ), Burger King, etc.

El almacén de datos Springwood y la consiguiente estructura de cubos OLAP puede navegarse a lo largo de cualquiera de estas jerarquías corporativas - es decir, por la compañía operadora, según la zona o zonas, o por marca o concepto. Esto permite comparaciones entre los conceptos, por ejemplo, o de todas las tiendas dentro de un concepto. Del mismo modo, es fácil hacer comparaciones de zona a zona, o para ver el desempeño de todas las tiendas dentro de un área.  
El almacén de datos tiene una dimensión de tiempo. Este calendario se ha construido en el almacén, lo que permite una fácil comparación de cualquier período del ejercicio anterior o el mismo período de un año antes. En lugar de comparar a nivel período, las comparaciones y las tendencias se pueden hacer a nivel trimestral o anual. (Excepto para los casos de decisiones de más bajo nivel, por ejemplo, comparar semana a semana e incluso día a día).  
En caso de Beneficios y Pérdidas se compara cada período durante los últimos cinco años, hasta el nivel de cuenta. Esto hace que sea fácil para Springwood para evaluar las tendencias en cualquier categoría de gasto, comparando tienda a tienda, período a período, una zona a zona, o concepto-concepto.

Los cubos de Ventas Diarias se actualizan durante la noche con la información capturada por las cajas registradoras en cada tienda. Esto permite a los gerentes para evaluar y comparar las tendencias en la rapidez del servicio, el uso de mano de obra, las compras de alimentos de los empleados, etc., por el almacén, zona, área, concepto, etc. Debido a las ventas se registran en intervalos de 60 minutos, llamadas Partes del día, los administradores pueden usar esto para encontrar patrones de ventas extraños, posiblemente indicativos de robo de los empleados, durante las horas de la medianoche.

## Requisitos Particulares

Las ventas diarias se analizan en el segundo Nivel de la organización y podría darse a cada gerente de Tienda la autorización para el análisis de sus propios datos.

## Índices a Resolver

1. Mostrar Análisis de costos, ya sea histórico y además geográfico.
2. Analizar históricamente las ventas, podría desagregarse por Estructura Corporativa (para determinar si hay mayores o menores incidencias de tiendas)
3. Ranking de Ventas por Estructura de Productos de manera de estudiar preferencias.
4. Análisis de Tendencia para pérdidas y ganancias, Ranking de gastos por estructura corporativa.
5. Análisis de Importes de Ventas por intervalos de tiempo, para Gerentes de segundo nivel de la organización.
6. Reporte de cifras de Ventas gravadas, no gravadas, etc.
7. Tabla dinámica de Mix de Ventas, detallando hasta el Ítem de Menú, posición de menú y sus costos.
8. Relacionar en una tabla dinámica, Tiendas (y correspondiente estructura corporativa) con las estructura de productos vendidos.
9. Comparación entre distintos periodos de pérdidas y ganancias según las cuentas contables. Tabla y gráfico.
10. Total de ventas realizadas a Empleados. (horarios y productos).
11. Análisis geográfico de ventas mensuales, para estudiar posibilidades de abrir nuevas sucursales
12. Rentabilidad Económica de Productos del último mes, comparando contra un valor objetivo.

# SOLUCIÓN PROPUESTA

## Diseño del Datawarehouse

COLOCAR DIBUJOS DE CUBOS AQUÍ!

### Esquema

Como puede observarse en los diagramas de la sección anterior, se ha optado por un esquema de Datawarehouse del tipo **Estrella**.

La implementación de este tipo de esquema fue posible debido al tamaño del modelo y datos en cuestión. Al no ser un volumen grande de datos, la performance de este tipo de esquema es aceptable.

### Hechos

Los Hechos definidos son:

* Hecho 1
* Hecho 2
* …

### Dimensiones

Las dimensiones definidas son:

* Hecho 1
* Hecho 2
* …

## Herramientas Utilizadas

|  |  |
| --- | --- |
| Herramienta | Propósito |
| RAD Studio Delphi 2007 | Con este IDE se realizó una aplicación para generar los datos aleatorios sobre una Base de Datos en Microsoft Access. |
| Microsoft Access 2010 | Utilizado como motor de base de datos genérico. Se optó por este motor ya que la mayoría de los motores de base de datos del mercado permiten importar/exportar desde/hacia Access. |
| Microsoft SQL Server Migration Assistant for Access v4.2 | Para migrar una base de datos Access a SQL Server |
| Microsoft SQL Server 2008 R2 Developers Edition:   * Analysis Services * SQL Server Business Intelligence Development Studio * SQL Server Management Studio | Creación de cubos, índices y reportes. |

## Proceso de ETL (Extracción, Transformación y Carga)

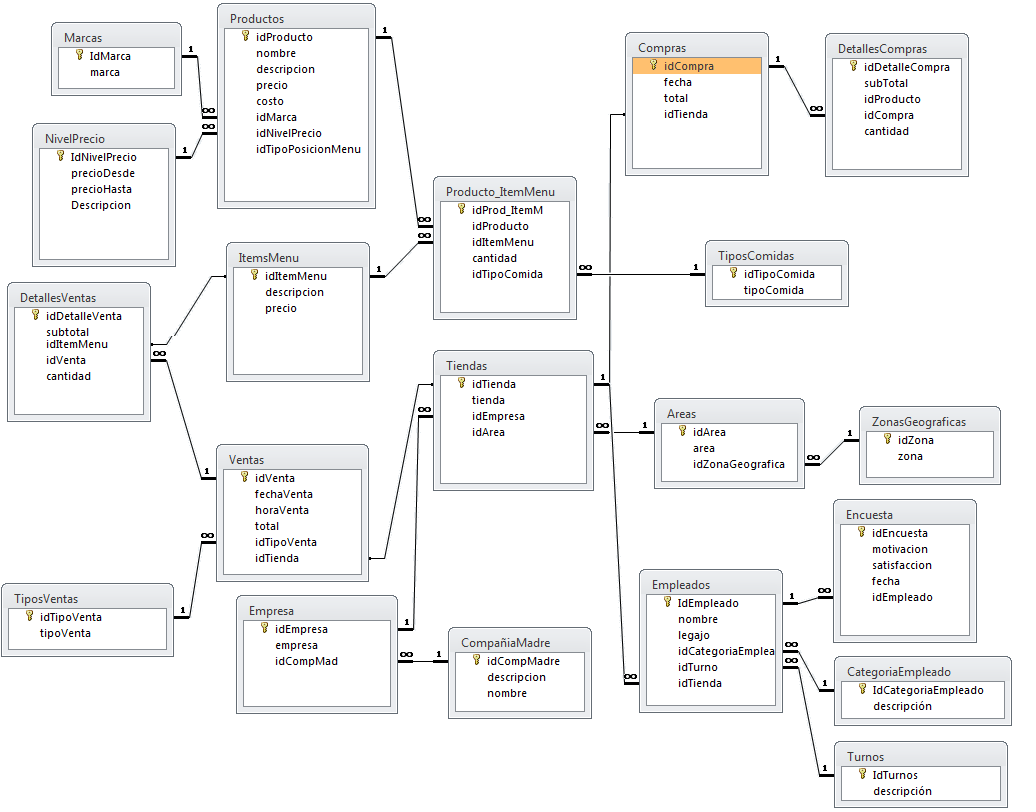
ETL es el proceso que organiza el flujo de los datos entre diferentes sistemas en una organización y aporta los métodos y herramientas necesarias para mover datos desde múltiples fuentes a un *almacén* de datos, reformatearlos, limpiarlos y cargarlos en otra base de datos.



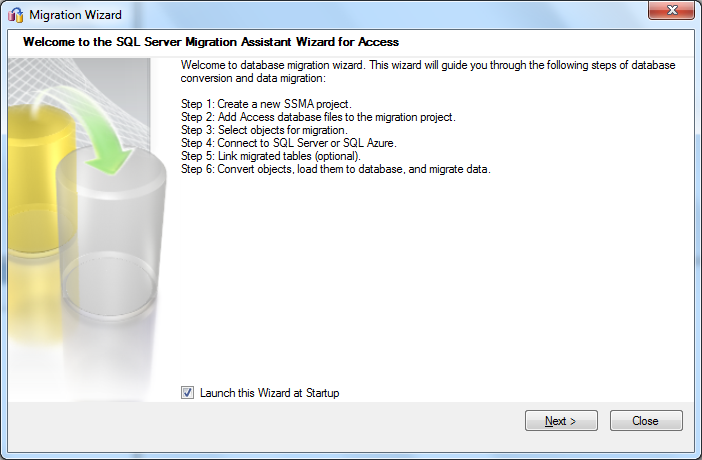
Para este caso en particular, se crearon datos aleatorios en una BD Relacional en Microsoft Access utilizando una aplicación desarrollada en Delphi 2007, y luego se migraron a SQL Server, para poder crear los cubos utilizando Analysis Services.

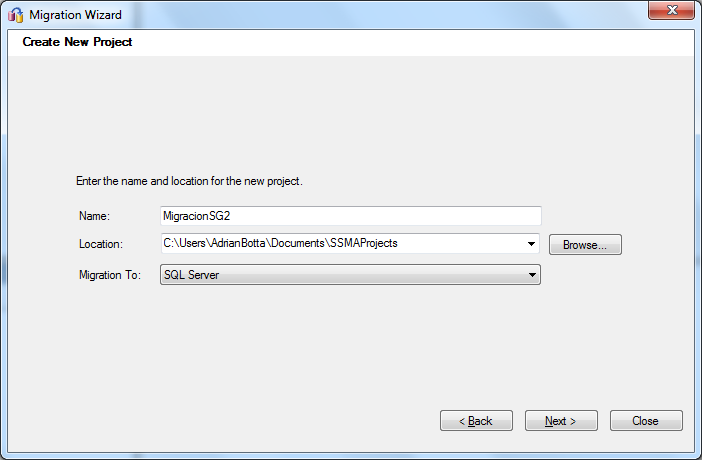


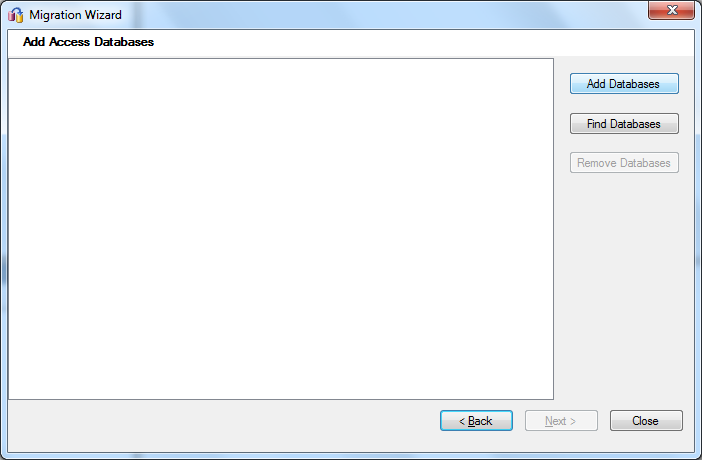
1. **Generación de Datos**: El código para generar los datos puede verse en el Anexo Adjunto al final del presente informe. En el mismo se muestran las funciones y porciones de código más representativas del generador. (Nota: El generador no tiene interfaz gráfica)
2. **Almacén Intermedio**: El diseño de la Base de Datos intermedia, realizado en Access, puede apreciarse en la figura siguiente:



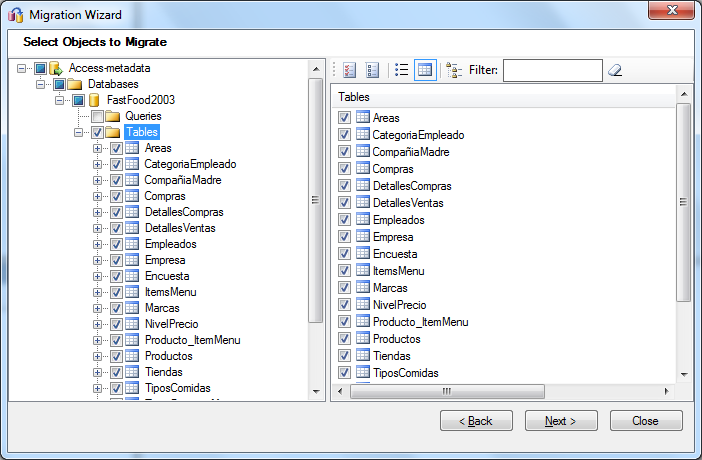
1. **Microsoft SQL Server Migration Assistant for Access v4.2:**











# ANEXO

## Código de la Generación de Datos

1. **const**
2. Apellidos: **array**[1..70] **of** **string** = (
3. 'Rodríguez','Gómez','Fernández','López','Díaz','Martínez','Pérez','García','Sánchez','Romero',
4. 'Sosa','Álvarez','Torres','Ruiz','Ramírez','Flores','Acosta','Benítez','Medina','Suárez',
5. 'Herrera','Aguirre','Pereyra','Gutiérrez','Giménez','Molina','Silva','Castro','Rojas','Ortíz',
6. 'Núñez','Luna','Juárez','Cabrera','Ríos','Ferreyra','Godoy','Morales','Domínguez','Moreno',
7. 'Peralta','Vega','Carrizo','Quiroga','Castillo','Ledesma','Muñoz','Ojeda','Ponce','Vera',
8. 'Vázquez','Villalba','Cardozo','Navarro','Ramos','Arias','Coronel','Córdoba','Figueroa','Correa',
9. 'Cáceres','Vargas','Maldonado','Mansilla','Farías','Rivero','Paz','Miranda','Roldán','Lucero');
11. Nombres: **array**[1..70] **of** **string** = (
12. 'Santiago','Sebastián','Diego','Nicolás','Samuel','Alejandro','Daniel','Mateo','Ángel','Matías',
13. 'Gabriel','Tomás','David','Emiliano','Andrés','Joaquín','Carlos','Alexander','Adrián','Lucas',
14. 'Benjamín','Leonardo','Rodrigo','Felipe','Francisco','Pablo','Martín','Fernando','Isaac','Manuel',
15. 'Juan Pablo','Emmanuel','Emilio','Vicente','Eduardo','Juan','Javier','Jorge','Aaron','José',
16. 'Erick','Luis','Cristian','Ignacio','Christopher','Jesús','Kevin','Juan José','Agustín','Juan David',
17. 'Simón','Joshua','Maximiliano','Miguel Ángel','Juan Sebastián','Bruno','Iván','Gael','Miguel','Thiago',
18. 'Jerónimo','Hugo','Ricardo','Antonio','Ian','Anthony','Pedro','Rafael','Jonathan','Esteban');
19. Categoria: **array**[1..7] **of** **integer** = (1,2,3,2,3,2,3) ;
21. Marca: **array** [1..50] **of** **string**=(
22. 'AppleCrings','Arnott','ArnottTimTam','Ben&Jerry','Bisc&Mars','BlueRiband',
23. 'Bob','Brannigans','Bugles','BurtonMaryland','Specials','Burtons','Cadbury',
24. 'Calypso','Cheeselets','CheesyToolBag','Cremosa','Discos','Disney','Domino',
25. 'DoubleTake','EskimoPie','FarmerBrown','Fish','Fishfingers','FlaminHotWotsits',
26. 'FootballCrazy','FoxBiscuits','Frazzles','Gold','GoldenWonder','GoodCrunchy',
27. 'Griddles','HobNob','HorrorBags','HulaHoops','Humdinger','JaffaCake','JuicyDropPop',
28. 'Kinder','Kins','KitKat','Kits','KPNuts','Kracks','LayStax','LemonCheesecakeKitKat',
29. 'Loacker','LyonsMaid','Magnum');
31. ---COMPRAS----------------
33. forIdCompra:=1;
35. **for** forIdTienda:=1 **to** 26  **do**   *// 26 tiendas*
36. **begin**
37. **for** forFechaCompraAño:=2010 **to** 2011  **do**   *// del 2010 a 2011*
38. **for** forFechaCompraMes:=1 **to** 12 **do**   *// Enero a Diciembre*
39. **begin**
40. totalCompra:= 0;
41. *// Siempre se compra el día 1*
42. fechaCompra := StrToDate(''+inttostr(forFechaCompraMes)+'/01/'+inttostr(forFechaCompraAño));
44. **with** ADOQuery\_Compras **do**
45. **begin**
46. Close;
47. Parameters.ParamByName('idCompra').Value:= forIdCompra;
48. Parameters.ParamByName('fecha').Value:= fechaCompra;
49. Parameters.ParamByName('total').Value:= 0; *// Despues hacemos Update...*
50. Parameters.ParamByName('idTienda').Value:= forIdTienda;
51. ExecSQL;
52. **end**;
54. *// Detalles de la Compra, 1 PARA CADA PRODUCTO*
55. **for** forIDproducto:=1 **to** 17 **do**
56. **begin**
58. **with** ADOQuery\_DetallesCompras **do**
59. **begin**
60. Close;
61. Parameters.ParamByName('idCompra').Value:= forIdCompra;
62. Parameters.ParamByName('idProducto').Value:= forIDproducto;
64. **if** forIDproducto<=8 **then**  *// 540 al mes*
65. cantidadD:= random(400)+300
66. **else**
67. cantidadD:= random(200)+100;


71. *////////////////////////////////////////////////////////ACA CUANTOS====???*
72. Parameters.ParamByName('cantidad').Value:= cantidadD;
74. *//Busco el costo en la tabla Productos*
75. ADOTable\_Productos.Locate('idProducto',forIDproducto,[]);
76. costo:= ADOTable\_Productos.FieldByName('costo').AsFloat;
78. subtotal:= costo\*cantidadD;
80. Parameters.ParamByName('subtotal').Value:= FormatFloat('0.00', subtotal);
82. ExecSQL;
83. **end**;
85. totalCompra:= totalCompra + costo \* CantidadD;
86. **end**;
88. **with** ADOQuery\_UpdateCompras **do**
89. **begin**
90. Close;
91. Parameters.ParamByName('total').Value:= totalCompra;
92. Parameters.ParamByName('idCompra').Value:= forIdCompra;
93. ExecSQL;
94. **end**;
96. inc (forIdCompra);
98. BarraProgresoFechas.Progress:= BarraProgresoFechas.Progress+1;
100. **end**; *// end For Fechas*
101. **end**; *// end for IDtienda*


105. --------------VENTAS---------------------
107. inicialForIDVentas:= 1;
108. randomize;
109. ADOConnection1.Connected:= **true**;
110. ADOTable\_ItemsMenu.Active:= **true**;

113. **for** forIdTienda:=1 **to** 26  **do**   *// 26 tiendas*
114. **begin**
115. **for** forFechaVentaAño:=2010 **to** 2011  **do**   *// del 2010 a 2011*
116. **for** forFechaVentaMes:=1 **to** 12 **do**   *// Enero a Diciembre*
117. **for** forFechaVentaDia:=1 **to** 28 **do**     *// Días 1 a 28*
118. **begin**
119. **if** (forFechaVentaDia **mod** 6=0) **or** (forFechaVentaDia **mod** 7=0) **then** *//fin de semana + ventas*
120. RNDCantVentasDia:= random(30)+1
121. **else**
122. RNDCantVentasDia:= random(15)+1;
124. **for** forIdVenta := inicialForIDVentas **to** (inicialForIDVentas+RNDCantVentasDia-1) **do** *// Uno por cada venta diaria*
125. **begin**
127. totalVenta:= 0;
128. fechaVenta := StrToDate(inttostr(forFechaVentaMes)+'/'+inttostr(forFechaVentaDia)+'/'+inttostr
130. (forFechaVentaAño));
132. *//Calculo de la hora*
133. **if** random(100)>30 **then**
134. **begin**
135. *// 70% de probabilidades de que sea al mediodia o noche*
136. **if** random(100)>50 **then**
137. *// Venta al mediodia*
138. horaVenta:= StrToTime(inttostr(random(2)+12)+':'+inttostr(random(50)+10))
139. **else**
140. *// Venta a la noche*
141. horaVenta:= StrToTime(inttostr(random(2)+21)+':'+inttostr(random(50)+10));
142. **end**
143. **else**
144. **begin**
145. *// 30 % de probabilidades que sea a la tarde*
146. horaVenta:= StrToTime(inttostr(random(15)+9)+':'+inttostr(random(50)+10));
147. **end**;

150. **with** ADOQuery\_Ventas **do**
151. **begin**
152. Close;
153. Parameters.ParamByName('idVenta').Value:= forIdVenta;
154. Parameters.ParamByName('fechaVenta').Value:= fechaVenta;
155. *// Parameters.ParamByName('horaVenta').Value:= StrToTime(inttostr(random(15)+9)+':'+inttostr(random(50)+10));*
156. Parameters.ParamByName('horaVenta').Value:= horaVenta;
158. Parameters.ParamByName('total').Value:= 0; *// Despues hacemos Update...*
159. Parameters.ParamByName('idTipoVenta').Value:= random(2)+1; *//gravadas o no gravadas*
160. Parameters.ParamByName('idTienda').Value:= forIdTienda;
161. ExecSQL;
162. **end**;

165. RNDCantElementos:= random(3)+1; *// Cantidad de elementos que compra*
167. *// Detalles de la venta*
168. **for** i:=1 **to** RNDCantElementos **do**
169. **begin**
170. **with** ADOQuery\_DetallesVentas **do**
171. **begin**
172. Close;
173. Parameters.ParamByName('idVenta').Value:= forIdVenta;
175. IDitemMenu:= random(40)+1;
176. Parameters.ParamByName('idItemMenu').Value:= IDitemMenu;
178. cantidadD:= random(3)+1;
179. Parameters.ParamByName('cantidad').Value:= cantidadD;
181. *//Busco el precio en la tabla ItemsMenu*
182. ADOTable\_ItemsMenu.Locate('idItemMenu',IDitemMenu,[]);
183. precio:= ADOTable\_ItemsMenu.FieldByName('precio').AsInteger;
185. Parameters.ParamByName('subtotal').Value:= precio\*cantidadD;
187. ExecSQL;
188. **end**;
189. totalventa:= totalventa + precio \* CantidadD;
190. **end**;
192. **with** ADOQuery\_UpdateVentas **do**
193. **begin**
194. Close;
195. Parameters.ParamByName('total').Value:= totalventa;
196. Parameters.ParamByName('idVenta').Value:= forIdVenta;
197. ExecSQL;
198. **end**;


202. **end**;  *// FIN  for forIdVenta := 1 to RNDCantVentasDia do*
204. inicialForIDVentas:= inicialForIDVentas+RNDCantVentasDia;
206. BarraProgresoFechas.Progress:= BarraProgresoFechas.Progress+1;
207. *//  Showmessage('Venta Cargada'); exit;*
208. **end**;
209. BarraProgresoFechas.Progress:= 0;
210. BarraProgresoVentas.Progress:= BarraProgresoVentas.Progress+1;
211. **end**;


215. ----------EMPLEADOS------------------
217. t:= 3+random(5);
218. **for** forTienda:=1 **to** 75 **do**
219. **for** forTurno:=1 **to** 3 **do**
220. **for** forCategoria:=1 **to** t **do**
221. **begin**
223. **with** ADOQuery\_Empleados **do**
224. **begin**
225. Close;
226. Parameters.ParamByName('nombre').Value:= Nombres[random(70)+1]+' '+Apellidos[random(70)+1];
227. Parameters.ParamByName('legajo').Value:= random(20000)+1;
228. Parameters.ParamByName('idCategoriaEmpleado').Value:= Categoria[forCategoria];
229. Parameters.ParamByName('idTurno').Value:= forTurno;
230. Parameters.ParamByName('idTienda').Value:= forTienda;
231. ExecSQL;
232. **end**;
234. **end**;

237. --------------ENCUESTA----------------------
238. **for** forEmpleados:=901 **to** 1800 **do**
239. **BEGIN**
241. cant:= random(3)+1;
242. **for** t := 1 **to** cant **do**
243. **begin**
244. **with** ADOQuery\_Encuesta **do**
245. **begin**
246. Close;
247. Parameters.ParamByName('motivacion').Value:= random(5)+1;
248. Parameters.ParamByName('satisfaccion').Value:= random(5)+1;
249. Parameters.ParamByName('fecha').Value:=StrToDate(inttostr(random(28)+1)+'/'+inttostr(random(12)+1)+'/'+inttostr
251. (random(11)+2000));
252. Parameters.ParamByName('idEmpleado').Value:= forEmpleados;
253. ExecSQL;
254. **end**;
256. **end**;
257. **END**;