Curso: Sistemas de Data Warehousing

Caso de Estudio:

Arte Espectacular

Ada Caorsi
Horacio Paggi
Guillermo Pérez

02/05/2002 10:31

I. Planteo del problema

Realidad Planteada

La empresa *Arte Espectacular* desea instalar un sistema de DW para estudiar su funcionamiento.
La empresa se encarga de traer al país espectáculos artísticos (cine, teatro, música, exposiciones, etc.) y organizar su exhibición. Para ello alquila salas adecuadas para exhibir el espectáculo, en diferentes regiones del país. En una misma sala pueden exhibirse varios espectáculos (por ejemplo en el cine *Milla* esta tarde se exhibió una película infantil y para la noche está programado un recital folklórico). A su vez un mismo espectáculo puede exhibirse en varias salas (por ejemplo una película se puede exhibir en varios cines simultáneamente).
Para asistir a un espectáculo, la persona debe asociarse a *Arte Espectacular*. Existen varios planes para asociarse:

- *Socio común*: Paga una cuota mensual y obtiene descuentos en los espectáculos.
- *Socio estudiante*: Paga una cuota mensual más baja que la del socio común y obtiene los mismos descuentos.
- *Socio familiar*: Se obtiene si se asocian 3 o más personas de un núcleo familiar. La cuota es más baja, y obtiene los mismos descuentos que el socio común.
- *Socio ocasional*: No paga cuota mensual, pero no obtiene descuentos en los espectáculos.

Al asociarse cada socio debe llenar un formulario muy pequeño donde detalla: nombre, apellido, dirección, teléfono, sexo, edad, profesión y nivel de ingreso. El formulario indica también el tipo de plan al que se está afiliando. Se llena un formulario por persona.
Todos los socios están habilitados a asistir a todos los espectáculos cuantas veces quieran.
Cada exhibición de un espectáculo (por ejemplo cada noche que se realiza un recital determinado) tiene un único precio para todos los espectadores, y un único porcentaje de descuento. Pero tanto el precio como el descuento pueden variar de una exhibición a otra (por ejemplo: las salas de cine cobran más caro los fines de semana).

Requerimientos

- **Recaudación.**

1. Se desea estudiar los importes recaudados por espectáculo y por sala. De esta forma se puede analizar si hay algún tipo de espectáculo (cine, música, etc.) que no esté siendo redituable en alguna sala. Interesa estudiar las salas de acuerdo a su tipo (teatro, anfiteatro, cine, etc.) y su ubicación geográfica (localidad, departamento).
2. Interesa también comparar lo recaudado de acuerdo a los diferentes descuentos (porcentajes de descuentos), estudiando si se logra recaudar más en las exhibiciones que tienen mayores descuentos.
3. También interesa estudiar lo recaudado mensualmente a cada socio para cada tipo de espectáculo.

- **Estudio de Mercado.**

4. Interesa contabilizar la cantidad de exhibiciones a las que asiste cada socio (Si asiste a dos exhibiciones de un mismo espectáculo se lo cuenta dos veces). Cruzando diferentes clasificaciones de los socios (sexo, edad, profesión, nivel de ingreso) y las demás variables (espectáculos, salas) se puede segmentar el mercado y orientar mejor las propagandas.
5. Interesa estudiar que rangos de edades tienen los socios que asisten a cada tipo de espectáculo, y en qué fechas asisten: día, mes, año y día de la semana (lunes, martes ...)

- **Asignación de las Salas.**

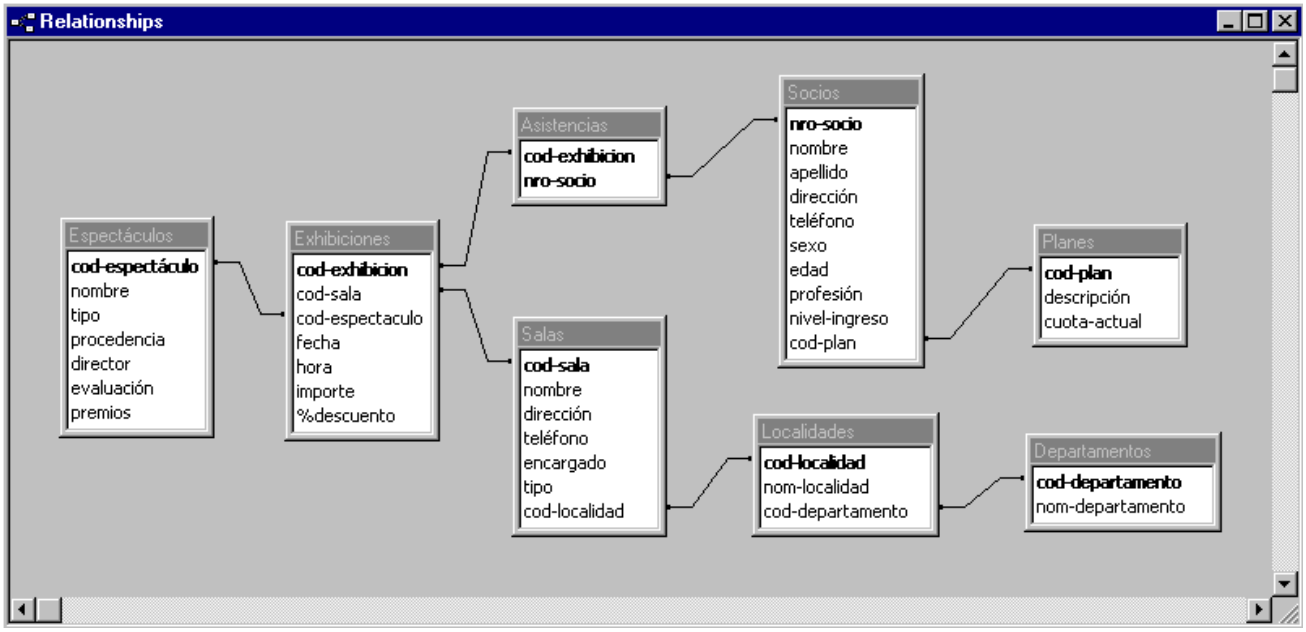
6. Se quiere estudiar la cantidad de espectáculos diferentes que se han exhibido en cada sala. Interesan también totales por fecha y ubicación geográfica de la sala.

Bases Fuentes

Se tienen las siguientes tablas:

- departamentos (cod-departamento, nom-departamento)
- localidades (cod-localidad, nom-localidad)
- salas (cod-sala, nombre, dirección, teléfono, encargado, tipo, cod-localidad)
- planes (cod-plan, descripción, cuota-actual)
- socios (nro-socio, nombre, apellido, dirección, teléfono, sexo, edad, profesión, nivel-ingreso, cod-plan)
- espectáculos (cod-espectáculo, nombre, tipo, procedencia, director, evaluación, premios)
- exhibiciones (cod-exhibición, cod-sala, cod-espectáculo, fecha, hora, importe, %descuento)
- asistencias (cod-exhibición, nro-socio)

Relaciones entre tablas:

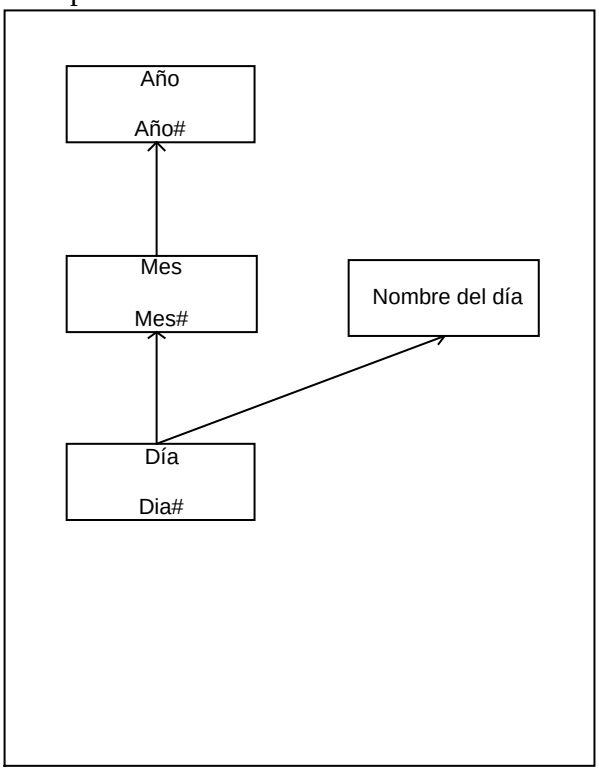


II. Diseño conceptual.

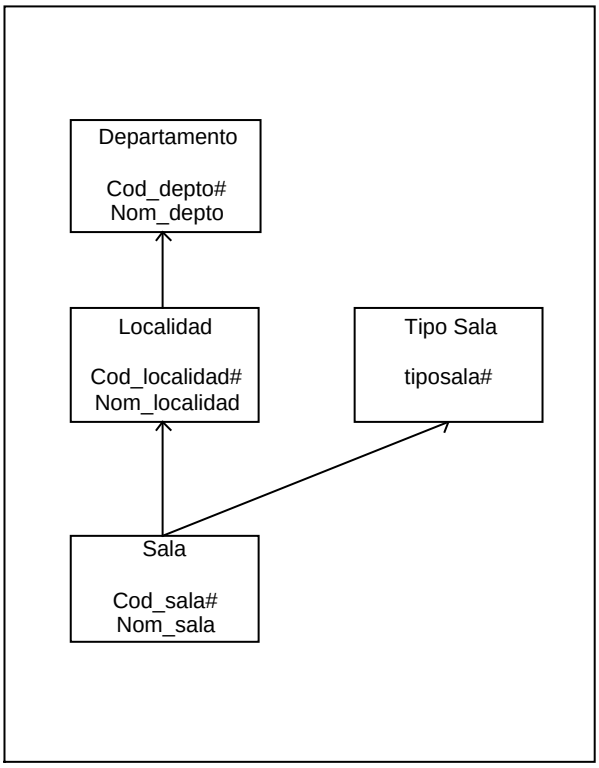
Se describe a continuación el esquema conceptual multidimensional obtenido a partir de los requerimientos planteados, junto con un estudio de aditividad de las medidas identificadas.

Definición de dimensiones y medidas

Tiempo



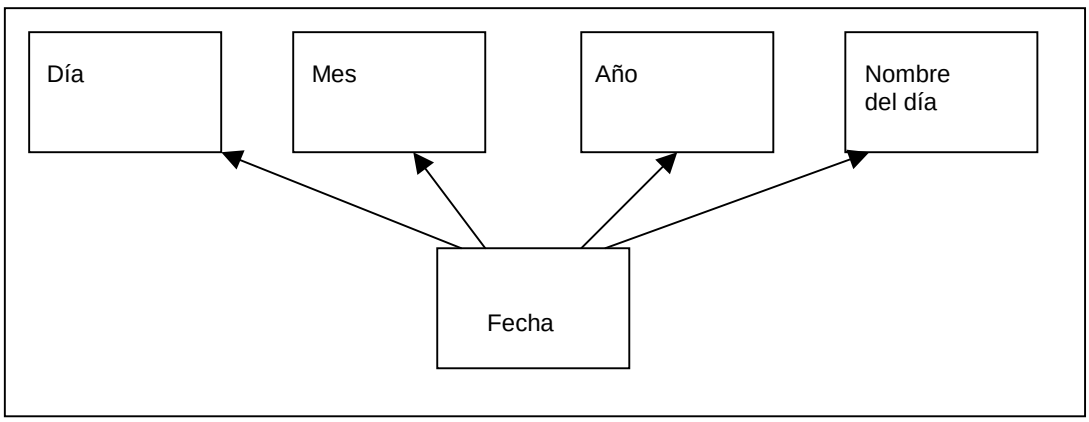
Sala



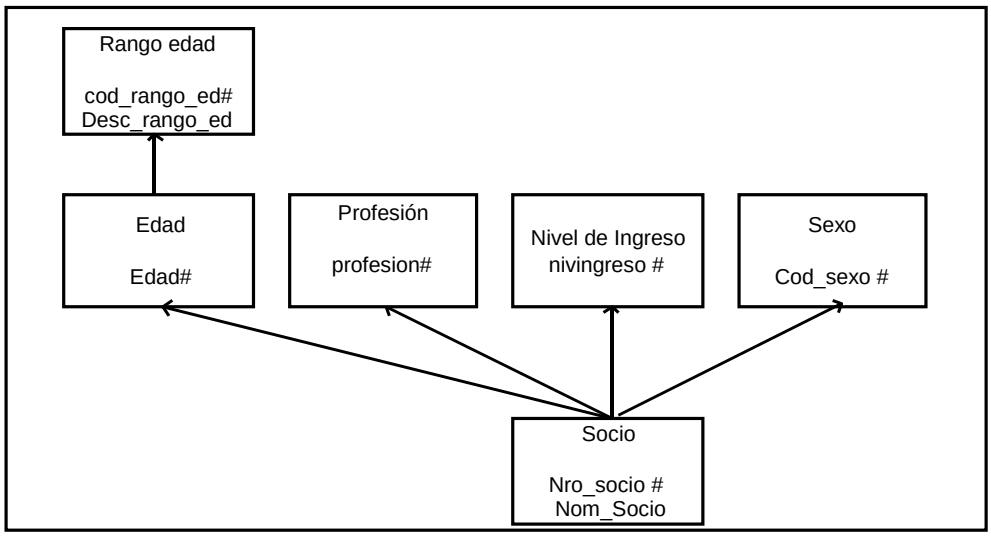
Observación: Día, Mes y Año.

El requerimiento 6 (consultas de rangos de edades de asistentes por día, mes, año y día de la semana) se entiende aquí como instanciable, por ejemplo, así: "cuántos socios de entre 20 y 30 años asistieron a un espectáculo de tipo "Teatro" el 20 de julio del 2000", y No: "Cuántos socios...los días 20" , Por lo tanto, día es un día perfectamente identificable dentro de un año, así como mes es un mes del tipo "enero del 2002" y no del tipo "enero". La elcción depende de qué es lo que se quiere modelar: si se van a usar los datos obtenidos con fines primordialmente estadísticos (primera opción) o para fines de otros estudios de márketing(opción 2). Por el planteo dado a los requerimientos, nos pareció que la orientación estadística era la más aproximada, por lo que fue la elegida.
La segunda opción (días: 01..31, meses:enero...diciembre, año:1900...2002) generaría una dimensión tiempo del tipo:

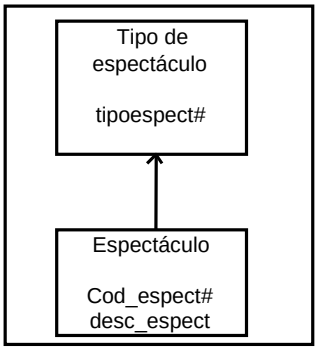
Tiempo:



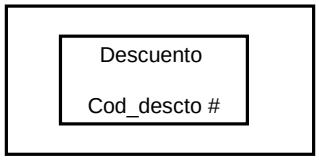
Socio



Espectáculo

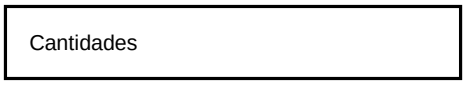


Descuento



Importe

Cantidad

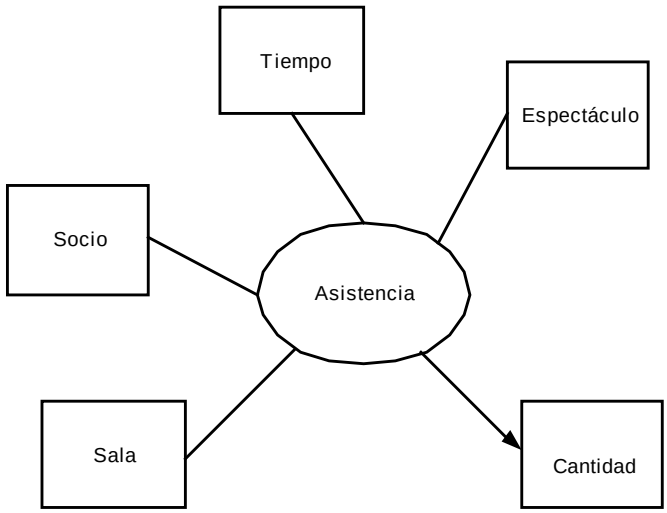


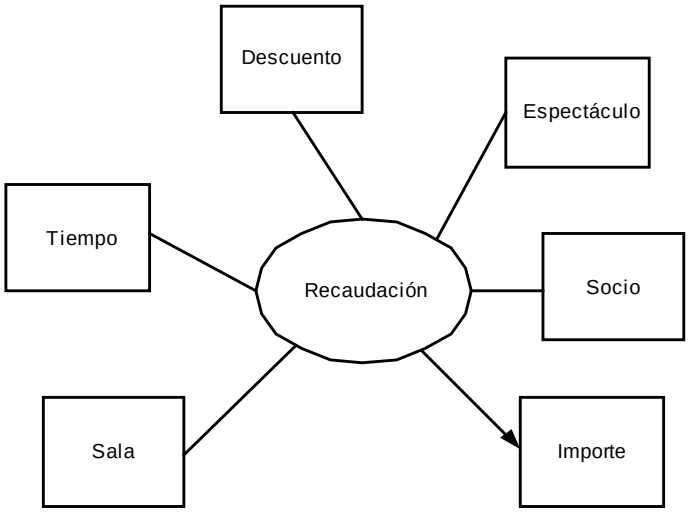
Cantidad de
espectáculos diferentes

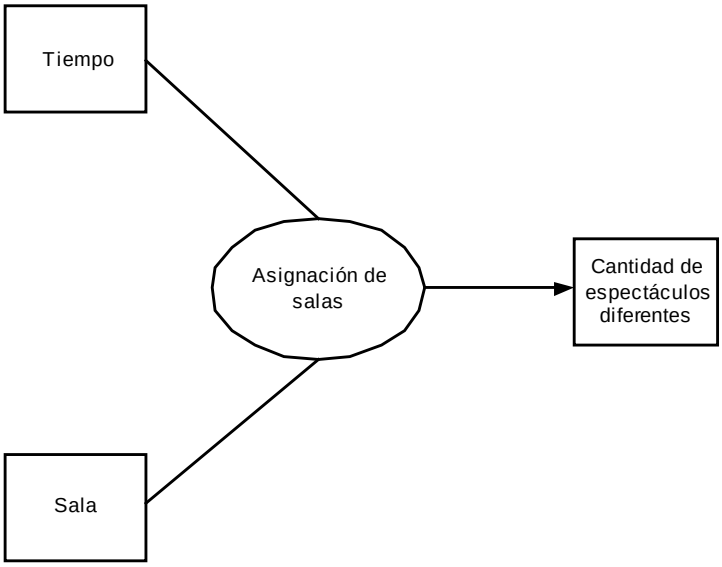
Observaciones:

Usamos solo los códigos sin sus descripciones (p. Ej. de tipo de sala) asumiendo que estos son lo suficientemente autoexplicativos. Además, dichas descripciones no figuran en las bases de datos fuentes.

Relaciones dimensionales







Estudio de aditividad por relación dimensional

Recaudación

	Sala	Espectáculo	Descuento	Tiempo	Socio
Importe recaudado	+	+	+	+	+

Asistencia

	Sala	Espectáculo	Socio	Tiempo
Cantidad	+	+	+	+

Asignación de salas

	Sala	Tiempo
Cantidad de espectáculos diferentes	NA	NA

La no aditividad surge de este contra-ejemplo:

- ***Respecto al tiempo:*** la sala "A" presentó los espectáculos "1" en el mes de enero del 2000, el "2" en febrero del 2000 y el "1", "3" y "5" en marzo y abril del 2000 (los tres en marzo y abril) y ninguno el resto del año. En el año 2000, A no presentó 8 espectáculos diferentes, sino 4.

- ***Respecto a las salas***: La localidad "L " del Depto "D" tiene solo dos salas tipo "cine": "A "y "B". La sala A presentó los espectáculos "1" y "2" durante el mes de enero del 2000 y la B, el "1" y "3" también en enero del 2000. En enero del 2000, las salas de tipo "cine", en la localidad "L" del Depto. "D" exhibieron cada una dos espectáculos distintos, pero en conjunto solo exhibieron 3.

Solución al problema de la no aditividad:

Se podría mantener una serie de tablas, que contendrían todos los valores de la medida cantidad de espectáculos distintos para todos los roll ups posibles:

Asignación_sala_fecha(cod_sala,fecha, cantidad de espectáculos distintos por sala y fecha)
Asignación_sala_mes(cod_sala, mes, cantidad de espectáculos distintos por sala y mes)
Asignación_sala_año(" , año, cantidad de espectáculos distintos por sala y año)

Y análogamente existirían:

Asignación_localidad_fecha,
Asignación_localidad_mes,
Asignación_localidad_año,
Asignación_Departamento_fecha,
Asignación_departamento_mes,
Asignación_departamento_año,

La carga de estas tablas se discute más adelante.

Restricciones del estudio de aditividad

Para hacer roll-up por Rango de edad, dichos rangos deben ser disjuntos

III. Diseño lógico.

Descripción de requerimientos

R1	Importe recaudado por espectáculo
R2	Importe recaudado por sala
R3	Importe recaudado por tipo de descuento
R4	Importe recaudado por mes, tipo de espectáculo y socio
R5	Cantidad de asistencias por socio, espectáculo y sala
R6	Asistencia de socios por rango de edad, tipo de espectáculo y fecha
R7	Cantidad de espectáculos diferentes por sala y fecha

Observaciones:

En el requerimiento 5, se consideró además la FECHA (dimensión Tiempo), ya que se quieren comparar la cantidad de asistencias por espectáculo, socio y salas, pero estas siempre estan asociadas a un intervalo de tiempo.

Dimensiones y medidas por requerimiento

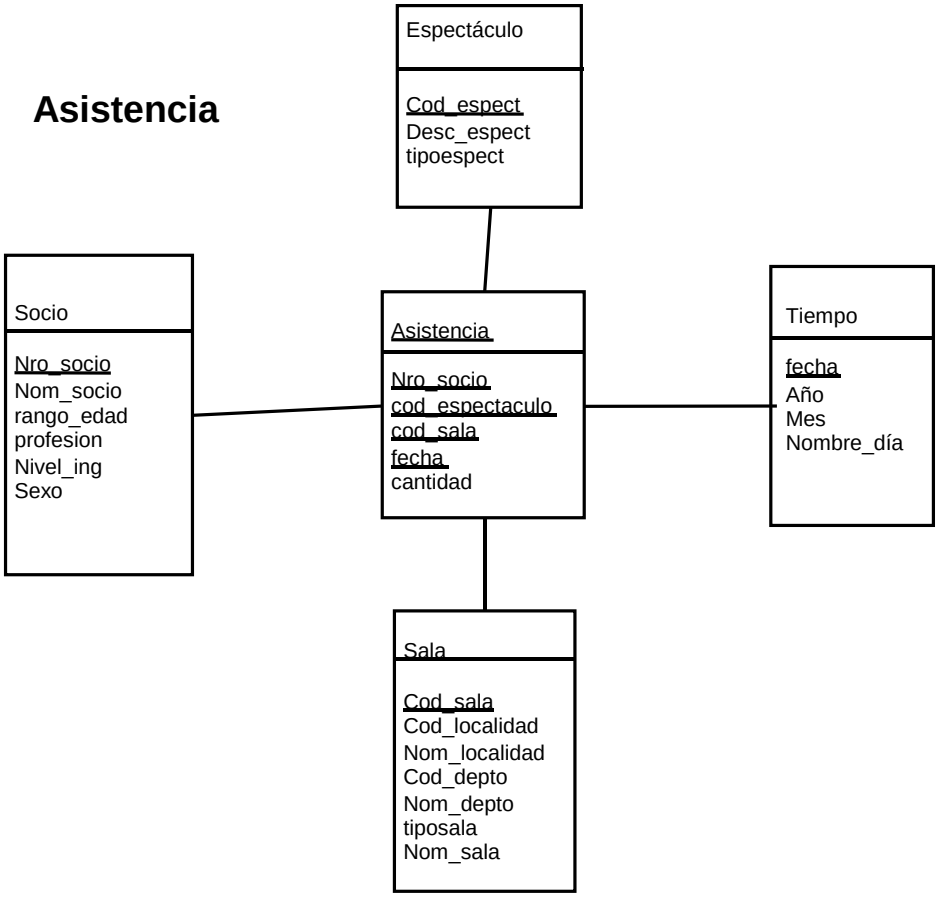
Dimensiones y Medidas	R1	R2	R3	R4	R5	R6	R7
❑ Dimensiones							
Espectáculo	X			X	X	X	
Sala		X			X		X
Descuento			X				
Tiempo				X	X	X	X
Socio				X	X	X	
❑ Medidas							
Importe recaudado	X	X	X	X			
Cantidad					X	X	
Cantidad de espectáculos distintos							X

Obtención del Esquema Relacional del DW a partir del modelo conceptual

Introducción

Dado que no se especifica la volatilidad (es decir, la frecuencia de cambio) que presentan los datos de las dimensiones (y por ende la optimización que se requiere en su mantenimiento) así como tampoco sobre la performance de las consultas, utilizaremos una representación relacional directa del modelo multidimensional, usando el star schema para cada relación dimensional. Con este esquema se logran estructuras más fáciles de entender y consultar dado que se disminuye en número de tablas respecto al de la otra opción (snowflake schema), lográndose así una mejor performance. A su vez, cada star schema consistirá en una tabla central (fact table) y para cada dimensión, una única tabla, con las jerarquías embebidas (representadas implícitamente) y por lo tanto, desnormalizada. La fact table tiene por clave la concatenación de las claves primarias de todas las tablas dimensión y las medidas.

Esquemas



Función de roll up para la medida (cantidad):

Se utiliza la función suma para realizar todos los roll ups de **cantidad**

Dependencias funcionales (comunes con los otros star schemas):

Socio

Nro_socio → (nom_socio, rango_edad, profesión, nivel_ingreso, sexo)

Sala

Cod_depto → (nom_depto)
Cod_localidad → (nom_localidad, cod_departamento, nom_departamento)
Cod_sala → (cod_localidad, nom_localidad, cod_departamento, nom_departamento, tipo_sala, nom_sala)

Espectáculo

Cod_espectáculo → (desc_espectáculo, tipo_espectáculo)

Tiempo

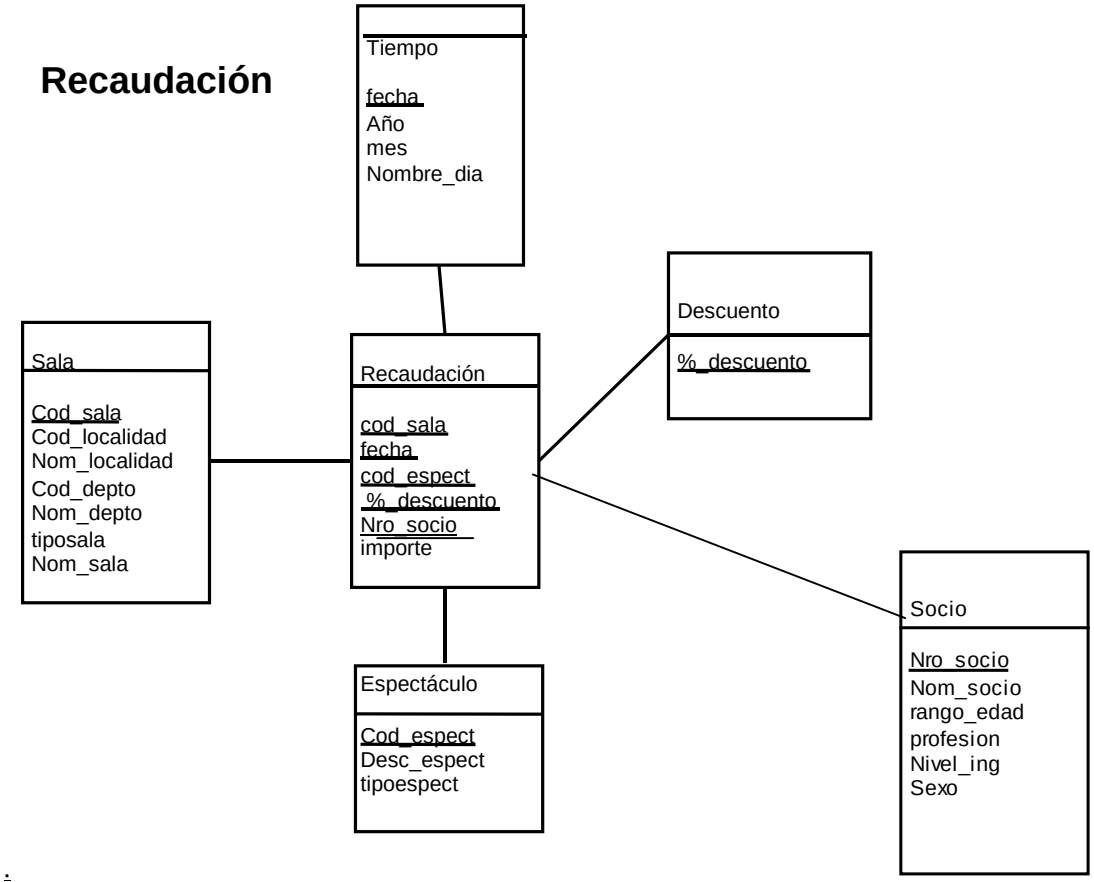
Fecha → (año, mes, nombre_día)

Propias de Asistencias:

Asistencia

(nro_socio, cod_espectáculo, cod_sala, fecha) → cantidad

Recaudación



.

Dependencias funcionales:

Las comunes con Asistencias y además

Recaudación

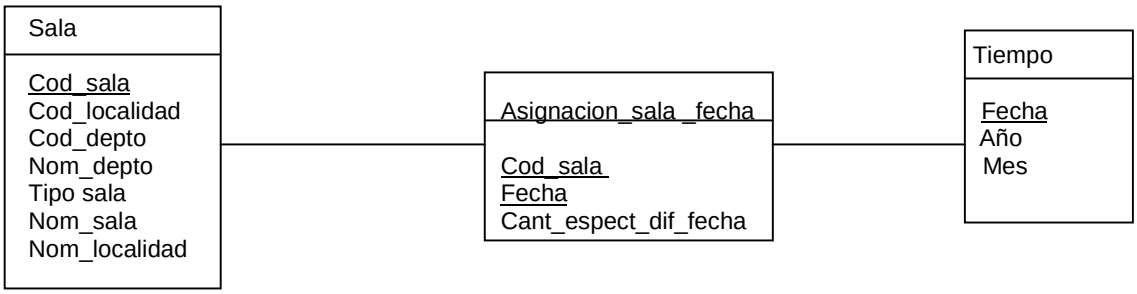
(cod_sala, fecha, cod_espectáculo,%_descuento,nro_socio)→ importe

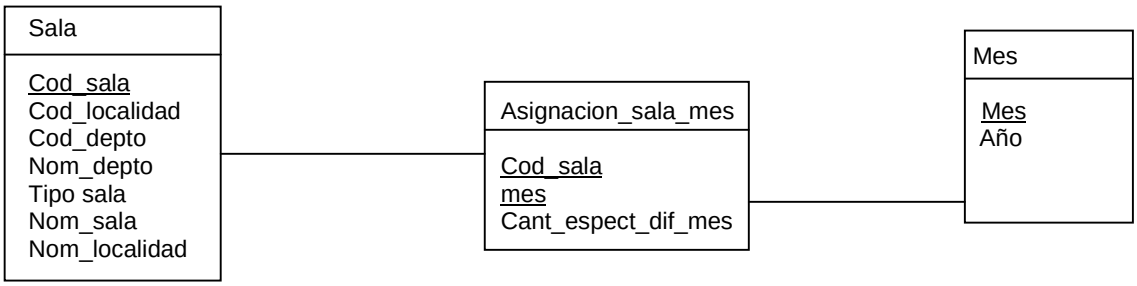
Función de roll up para la medida (importe):

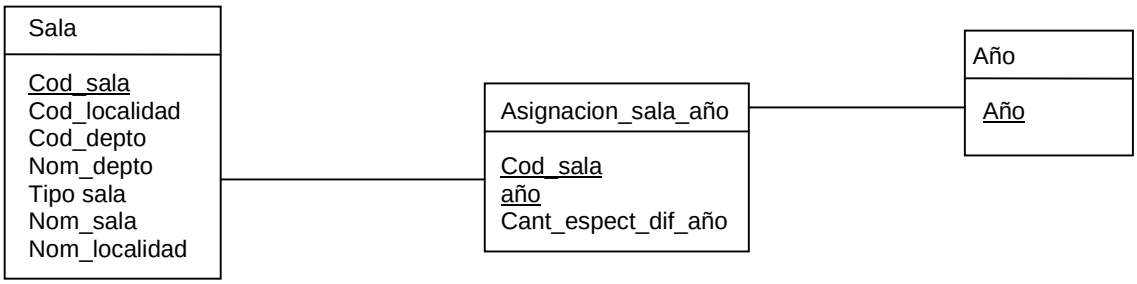
Se utiliza la función suma para realizar todos los roll ups de **importe**

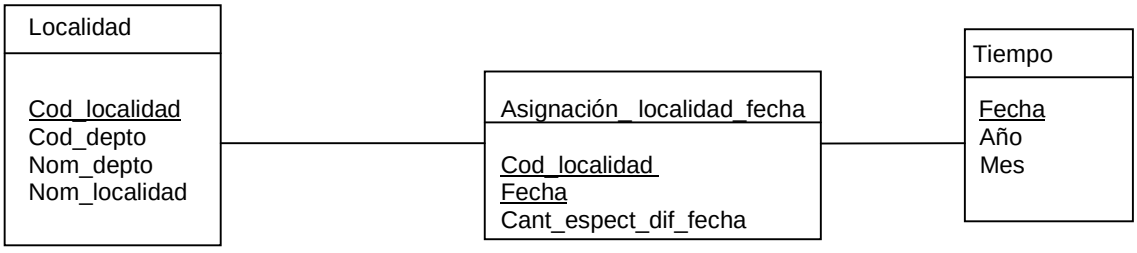
Asignación de sala

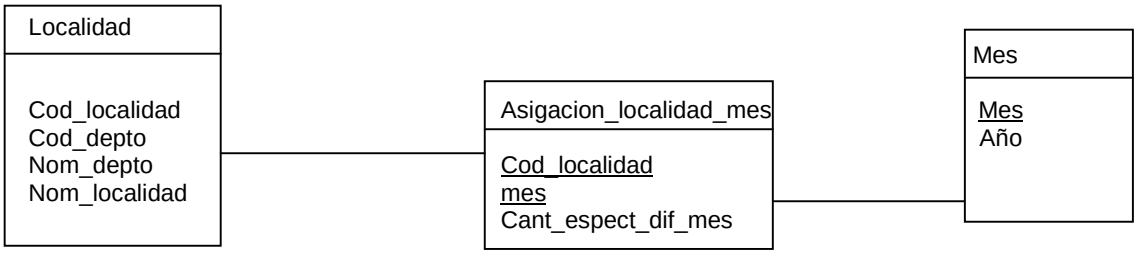
Debido a los problemas de aditividad encontrados, debemos plantear un star schema para cada una de las tablas que se crea para resolver el mencionado problema: asignación_sala_año, asignación_sala_mes, asignación_ sala_fecha, etc.

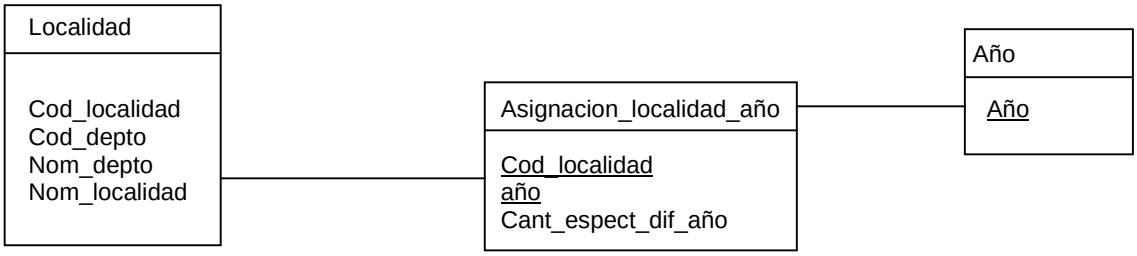


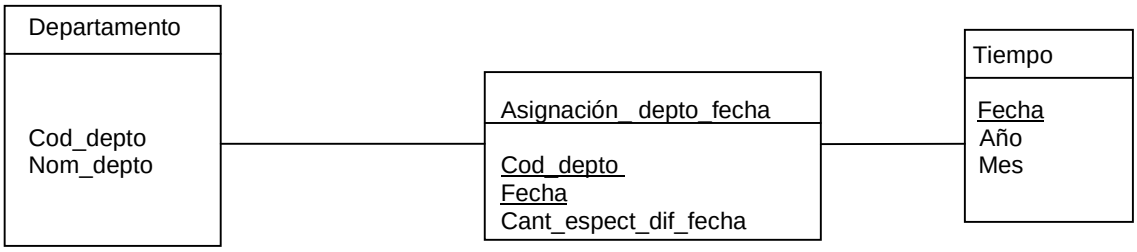


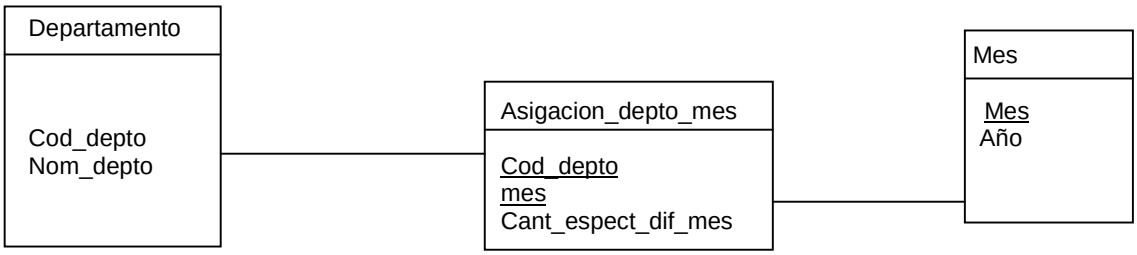


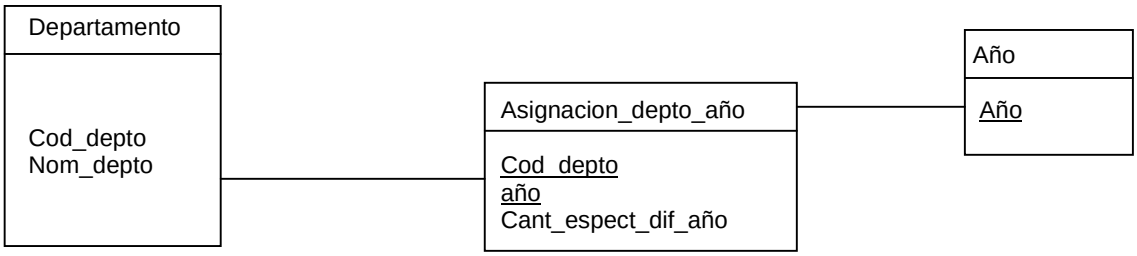












Nota:

En realidad, en el DW real no vamos a tener unas dimensionesn Mes, Año, Departamento y Localidad por separado, sino que en su lugar se utilizarán las dimensiones Tiempo y Sala . En los cubos generados con el Cognos-Transformer se volverán a ver las dimensiones Mes y Año por separado.

Dependencias funcionales:

Las comunes (para Sala y Tiempo) con Asistencia.

Función de roll up para la medida (cant_espect_dif):

Debido a los problemas de aditividad ya vistos (no se dispone de una función aritmética de cálculo para los roll ups de la medida) se mantiene la misma materializada en distintas tablas, correspondientes a todos los posibles roll ups

.

Comparación con el Esquema Relacional del DW obtenido aplicando el mecanismo de Kortink & Moody

Clasificación de entidades

Entidades Transacción

- Asistencias
- Exhibiciones

Entidades Componente

- Socios
- Espectáculos
- Salas

Entidades Clasificación

- Departamentos
- Localidades
- Planes

El método de Kortink y Moody es orientado a obtener el esquema lógico a partir de los datos (modelo E/R) por lo que ‘tiempo’ y ‘% de descuento’ no aparecen como entidades sino como atributos contrastando con los requerimientos que solicitan el estudio de la información por estas entidades, a su vez en este modelo surge la entidad ‘planes’ que en los requerimientos no se menciona. Además, quedarían dos star schemas en vez de tres. Por otra parte, si las bases fuentes tuvieran mucha más información que la necesaria para satisfacer los requerimientos, podría llegarse a generar un modelo demasiado complejo para los requerimientos planteados.

Optimización de las consultas por sexo, edad, etc.

Como para un socio la edad, profesión, nivel de ingreso y sexo cambian poco o nada en períodos breves, y como a la vez tienen unos pocos valores significativos cada uno, se puede mejorar significativamente la performance del las consultas del requerimiento 5 si definimos una mini-dimensión:

Mini(tipo_socio, sexo, edad, profesión, nivel_ingresos)

Hemos agrupado los atributos en uno solo, y por lo tanto bajado el número de joins entre tablas que tendría el cubo asociado a la consulta.

La fact table Asistencia quedaría:

Asistencia(nro_socio, cod_espectáculo;cod_sala, fecha, cantidad, tipo_socio)

y Socio:

Socio(Nro_socio, Nom_socio, tipo_socio)

Historización de los datos de Socio.

Para manejar los cambios en el nivel de ingresos, edad, profesión o sexo del socio, modificaríamos Socio para obtener:

Socio(nro_socio, fecha_de_validez, nom_socio, tipo_socio_actual)

donde fecha_validez es la fecha desde la cual es válido el cambio. En la sección de primitivas, esta tabla fue llamada Socios_His1.

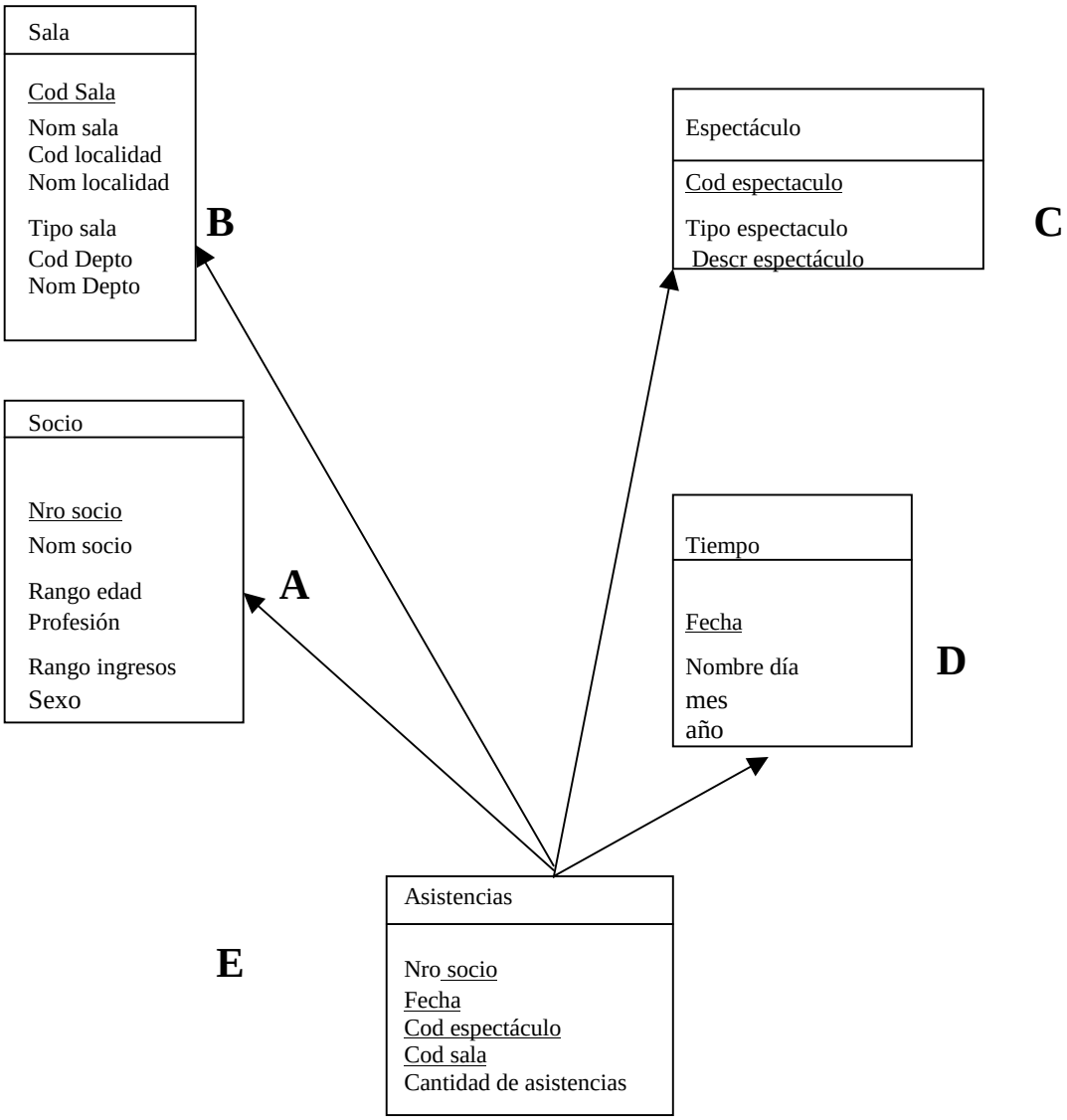
Variando la fecha de validez, podemos obtener todos los valores de Mini en la historia de ese socio.

No se precisa agregar ninguna fecha_his a Asistencia ya que la propia tabla ya contiene el campo fecha que le da una marca temporal .

Opcionalmente, como los cambios en el tipo de socio deben ser muy pocos (alrededor de 1 por año), si se quisiera acelerar el proceso de recuperación de datos históricos del socio, agregando un campo tipo_socio_ant a Socios se cubriría un año más de historia, lo que puede llegar a ser todo lo que se necesite.

Primitivas a usar para construir el Esquema Relacional

a. Para el star schema de ASISTENCIAS:



Asumimos que cod tipo sala es auto explicativo, y que no precisamos usar descr tipo sala (que no figura en las BD fuentes y complicaría el proceso de ETL) P. Ej. Un tipo de sala sería "CINE" , otro "TEATRO" y no X1 o C2.

En esta etapa no se toman en cuenta los cambios producidosen las dimensiones por la historización ni por la optimización de las consultas.

A (Socio) la obtengo a partir de Socios

Socios (nro socio, nombre, apellido, dirección, teléfono, sexo, edad, profesión, nivel de ingresos, codplan)

Asumimos que el nivel de ingresos ya se encuentra almacenado como rango (Es raro que en los formularios se pregunte directamente por el monto de los ingresos).

1) Datafilter (saco de Socios los atributos innecesarios: diriección, cod_plan y teléfono)
   R= Socios

X={dirección, teléfono,cod_plan}
R'=Socio1(nro-socio, nombre, apellido, sexo, edad, profesión,nivel ingresos,)

2) DD Adding 1-1 a Socios1 (para calcular el rango de edad)
R= Socios
F=f(edad)=⌊edad/10⌋ (un nro correspondiente al rango de edad)
R'=Socios2(nro socio, nombre, apellido, sexo, edad, profesión, nivel de ingresos, rango-edad)

3) Data Filter a Socios2 (para sacar edad)
R=Socios2
X={edad}
R'=Socios2.5(nro socio, nombre, apellido, sexo, rango-edad, profesión, nivel ingreso)

4) DD Adding 1-1 a Socios 2.5 (para concatenar el nombre y el apellido)
R=Socios2.5
F=f(nombre, apellido)=Nombre+apellido
R'=Socio3(nro. socio, nombre, apellido, sexo,rango-edad, profesión, nivel de ingreso, nombre+apellido)

5) Data Filter a Socios3 (para sacar nombre y apellido como campos por separado)
R=Socios3
X={nombre, apellido}
R'=Socio(nro. socio, nombre+apellido, sexo, rango edad, profesión,nivel de ingreso)

La supresión del nombre y apellido como dos campos separados es opcional, y se lo hace ya que nada indica que se utilicen esos campos individualmente.

B (Sala) Se obtiene a partir de Salas, Localidades, Departamentos.

Salas(cod sala, nombre, dirección, teléfono, encargado, tipo, cod_ localidad)
Departamentos (cod departamento,nom_departamento)
Localidades(cod localidad, nom localidad, cod departamento)

Tenemos que llegar a

Sala (cod sala, nom sala, cod localidad, nom localidad, tipo sala, cod depto, nom depto)

1) Datafilter a Salas (para eliminar atributos innecesarios)
R=Salas
X={Dirección, Teléfono, encargado}
R'=Sala1(cod sala, nombre, cod_localidad, tipo)

2) DD Adding N-1 entre Sala1 y Localidades (para agregar nom_localidad)
R1= Sala1
R2= Localidades
f()= Localidades.nom_localidad
C1=Nom_localidad
A= Cod_localidad
Is_fk = F
R'1= Sala2(cod sala, nombre, cod_localidad,nom_localidad,tipo)

3) DD Adding N-1 entre Sala2 y localidades (para agregar cod_departamento)
R1 = Sala2
R2 = Localidades
f()= Localidades.cod_departamento
C1= cod_depto
A= cod localidad
Is fk = F
R'= Sala3(Cod_sala, nombre, cod_localidad, nom_localidad,tipo,cod_depto)

4) DD Adding N-1 entre Sala3 y Departamentos (para agregar el nombre del departamento)

R1=Sala2
R2=Departamentos
F()=departamentos.nom_departamento
A=cod_depto
Is FK= F
C1=Nom_depto
R'= **Sala** (cod sala, nom sala, cod localidad, nom localidad, tipo sala,
Cod_depto, nom depto)

C (Espectáculo) la obtengo de

Espectaculos (cod espectaculo, nombre, tipo, procedencia, director,evaluacion, premios)

1) Data filter
R = Espectáculos
X={procedencia,director,evaluación, premios}
R'= **C**(cod espectáculo, nombre, tipo espectáculo)

Las mismas consideraciones para el tipo de espectáculo que para el tipo de sala.
Sino, habría que hacer DD Adding N-1 con alguna tabla auxiliar (cod tipo espectáculo, descr tipo espectáculo)

D (Tiempo) la obtengo a partir de Exhibiciones

Exhibiciones(cod-exhibición, cod sala, cod espectáculo,fecha, hora, importe, %_descuento)

1) Data filter (para dejar solo fecha, sin hora)
R=Exhibiciones
X={cod_exhibición, cod_sala, cod_espectáculo, importe, %_descuento, hora}
R'=Fecha1(fecha)

2) DD Adding 1-1 (para generar mes-año)
R=Fechas2
f=f(fecha)=mes/año con mes = mes de la fecha y año ídem
R'=fechas3(fecha,mes)

3) DD Adding 1-1 (para generar el año)
R=Fechas3
f=f(fecha)=año con año= año de la fecha
R'=fechas4 (fecha,mes,año)

4) DD Adding 1-1 (para generar el nombre del día)
R=Fechas4
f=f(fecha)=nombre del día (lunes...domingo)
R'=fechas4 (fecha,mes,año,nombre del día)

E (Asistencia) La obtengo a partir de Exhibiciones y Asistencias.

1) DD Adding N-1 (para agregar cod_espectáculo a Asistencias)
R1=Asistencias
R2=Exhibiciones
f=Exhibiciones.cod_espect
A=cod_exhibición
C1= cod_espect

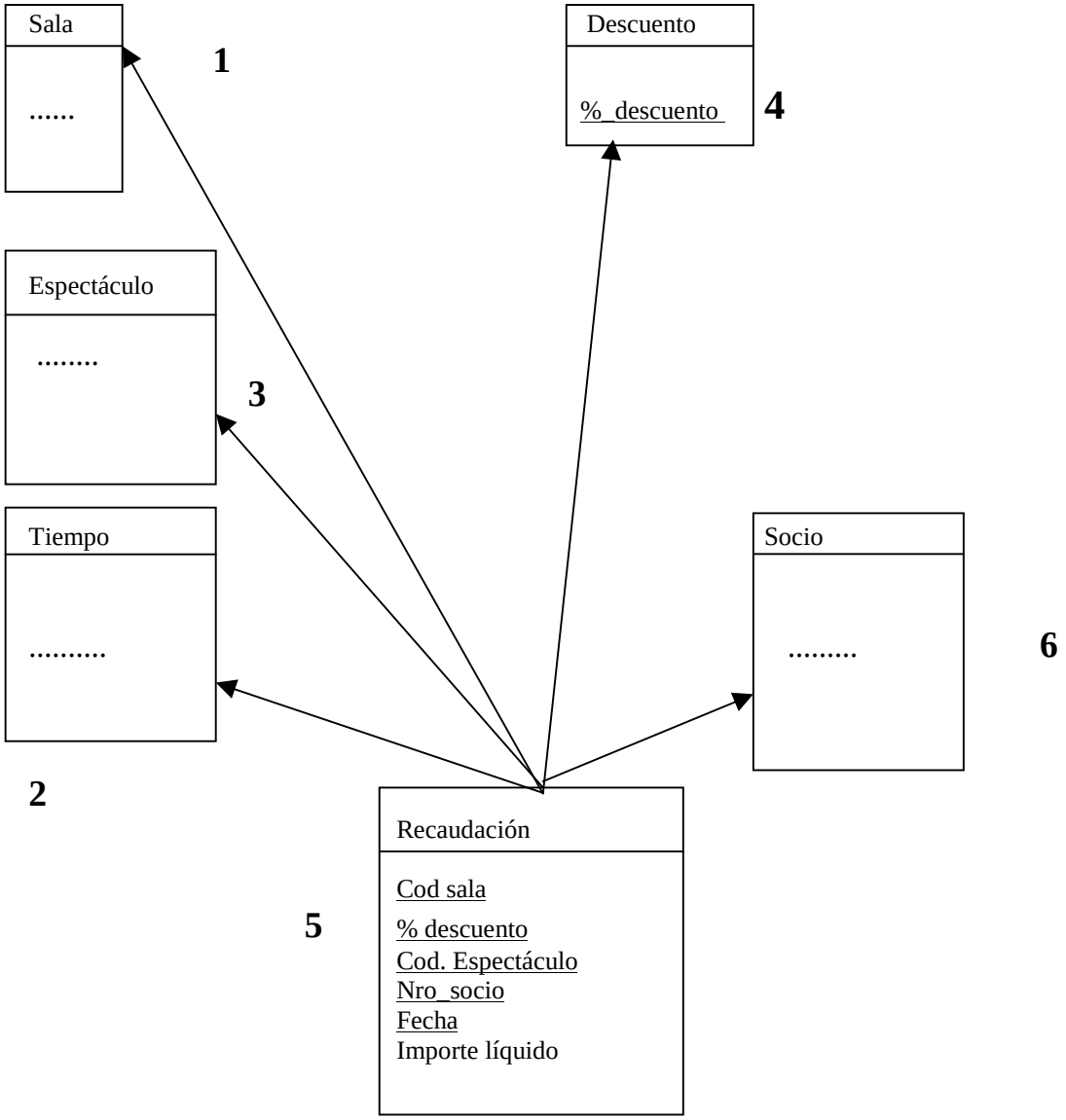
Is fk = F
R'=Asistencias1(cod_exhib, nro_socio, cod_espectáculo)

2) DD Adding N-1 (para agregar cod_sala)
R1=Asistencias1
R2=Exhibiciones
F=exhibiciones.cod_sala
A=cod_exhibición
C1=cod_sala
R'=Asistencias2(cod_exhibición, nro_socio, cod_espectáculo, cod_sala)

3) DD Adding N-1 entre Asistencias2 y exhibiciones (para agregar fecha)
R1=Asistencias2
R2=Exhibiciones
F=exhibiciones1.fecha
A=cod_exhib
C1=%Dcto
R'=Asistencias3(cod_exhibición, nro_socio, cod_espectáculo, cod_sala, fecha)

4) Aggregate generation (para sacar cod_exhibición)
R=Asistencias3
Z={} k=0
Y={cod_exhibición}
{e1..ek}={count(nro_socio, cod_espectáculo, cod_sala, fecha)}
R'=Asistencia=**E**

b. Para el star schema de RECAUDACION



1, 2 ,3, 6 se obtienen igual que para Asistencias

4 **(Descuento**) Se obtiene a partir de Exhibiciones

1) Datafilter (para dejar solo el %_descuento)
   R=Exhibiciones
   X={cod_exhibición, cod_sala, cod_espectáculo, importe, fecha, hora}
   R'=Descuento(%_descuento)

5 **(Recaudación)** Lo obtengo a partir de Exhibiciones, Socios y Asistencias.

*Dado que los socios tienen, en lo que respecta a esta tabla, dos tipos de planes (**ocasional**, es decir, la persona que va ver un espectáculo pero no es socio, y **común**, es decir, socio en el sentido ordinario de la palabra), y como el descuento que tuviese un espectáculo se le aplica solo a los socios comunes, no ocasionales, cuando se habla de importe, para que dicha medida sea significativa, deberá ser considerada como importe líquido, y no nominal.*

1) DD Adding N-1 (para agregar cod_sala)
   R1= Asistencias
   R2=Exhibiciones
   F=Exhibiciones.cod_sala
   A=cod_exhibición
   C1=cod_sala
   R'=Asistencias1(cod_exhibición, nro_socio,cod_sala)

2) DD Adding N-1 (para agregar %_descuento)
   R1= Asistencias1
   R2=Exhibiciones
   F=Exhibiciones.%_descuento
   A=cod_exhibición
   C1=%_descuento
   R'=Asistencias2(cod_exhibición, nro_socio,cod_sala, %_descuento)

3) DD Adding N-1 (para agregar el cod_espectáculo)
   R1= Asistencias2
   R2=Exhibiciones
   F=Exhibiciones. cod_espectáculo
   A=cod_exhibición
   C1=cod_espectáculo
   R'=Asistencias3(cod_exhibición, nro_socio, cod_sala, %_descuento, cod_espectáculo)

4) DD Adding N-1 (para agregar fecha)
   R1= Asistencias3
   R2=Exhibiciones
   F= Exhibiciones.fecha
   A=cod_exhibición
   C1=fecha
   R'=Asistencias4(cod_exhibición, nro_socio, cod_sala, %_descuento, cod_espectáculo, fecha)

5) DD Adding N-1 (para agregar importe)
   R1= Asistencias4
   R2=Exhibiciones
   F=Exhibiciones.importe
   A=cod_exhibición
   C1=importe
   R'=Asistencias5(cod_exhibición, nro_socio, cod_sala, %_descuento, cod_espectáculo, fecha, importe)

6) DD Adding N-1 (para calcular líquido)
   R1=Asistencias5
   R2=Socios
   F={*Si Socios.cod_plan <>= "ocasional" entonces líquido=importe*(1-%_descuento) sino líquido = importe*}
   A=nro_socio
   C1=líquido
   R'=Asistencias6(cod_exhibición, nro_socio, cod_sala, %_descuento, cod_espectáulo, fecha, importe,líquido)

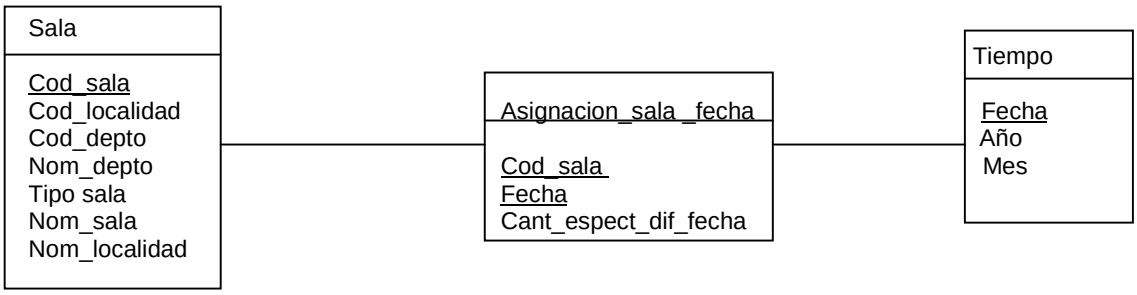
7) Data filter (para sacar importe)
   R=Asistencias6
   X={importe}
   R'=Asistencias6(cod_exhibición, nro_socio, cod_sala, %_descuento, cod_espectáulo, fecha, líquido)

8) Aggregate generation (para sacar cod_exhibición)

R=Asistencias8
Z={líquido} k=1=card(Z)
{e1}=sum(líquido)
Y={cod_exhibición}
R'=**5**

c. Para el star schema de ASIGNACIÓN DE SALA

Solo desarrollaremos las primitivas para Asignación_sala_fecha, Asignación_sala_mes y Asignación_sala_año. Para los demás casos es totalmente análogo.



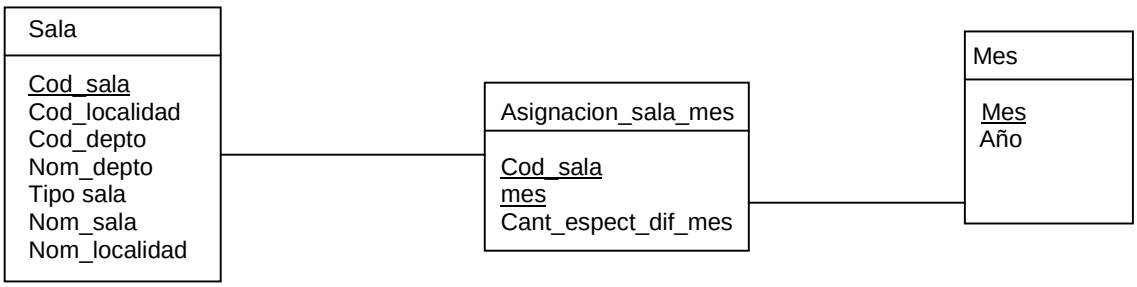
Tenemos que llegar a Asignación_sala_fecha(cod_sala, fecha, cantidad_espect_sala_fecha)

1) Aggregate generation (para eliminar cod_espectáculo)

R=Exhibiciones
Z={} card(Z)=0
{e1..ek}={}
Y={hora, cod_exhibición, importe, %_descuento}
R'=Asignacion1(cod_sala, fecha, cod_espectáculo)

2) Aggregate generation (para contar los registros)

R=Asignacion1
Z={} card(z)=0
{e1...ek}={count(*)}
Y={cod_espectáculo}
R'=Asignación(cod_sala, fecha, cantidad_espect_sala_fecha)



1) DD Adding 1-1 (para añadir mes y año)

R=Exhibiciones
F=Formato(fecha,"mmyyyy")
R'=Exhibiciones10

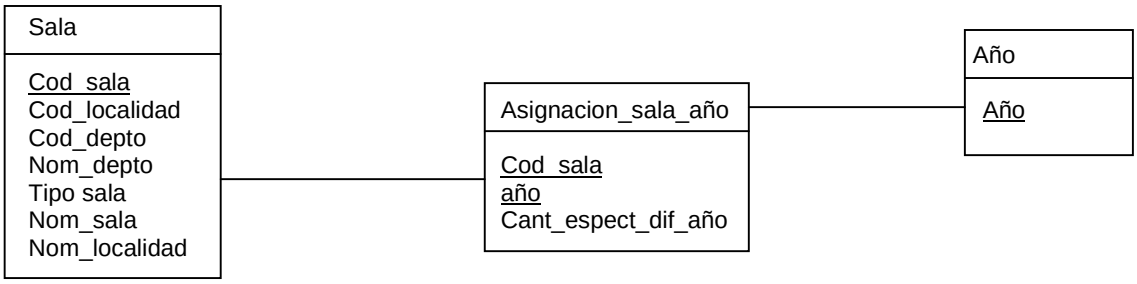
2) Aggregate generation (para eliminar cod_espectáculo)

R=Exhibiciones10

Z={} card(Z)=0
{e1..ek}={}
Y={hora, cod_exhibición, importe, %_descuento}
R'=Asignacion10(cod_sala, fecha, cod_espectáculo, mes)

3) Aggregate generation (para contar los registros)

R=Asignacion10
Z={} card(z)=0
{e1...ek}={count(*)}
Y={cod_espectáculo, fecha}
R'=Asignación(cod_sala , mes, cantidad_espect_sala_mes)



1) DD Adding 1-1 (para añadir año)

R=Exhibiciones
F=Formato(fecha,"yyyy")
R'=Exhibiciones100

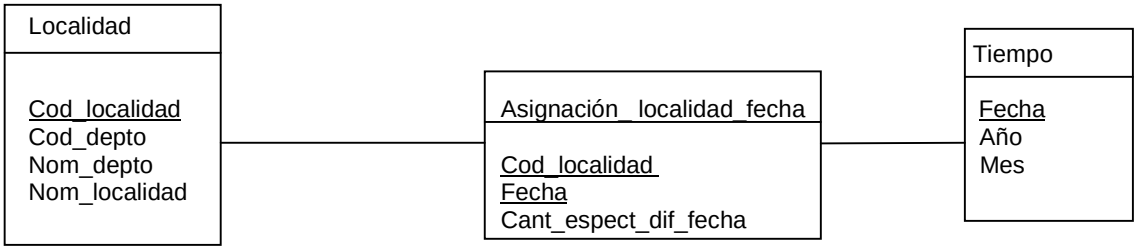
2) Aggregate generation (para eliminar cod_espectáculo)

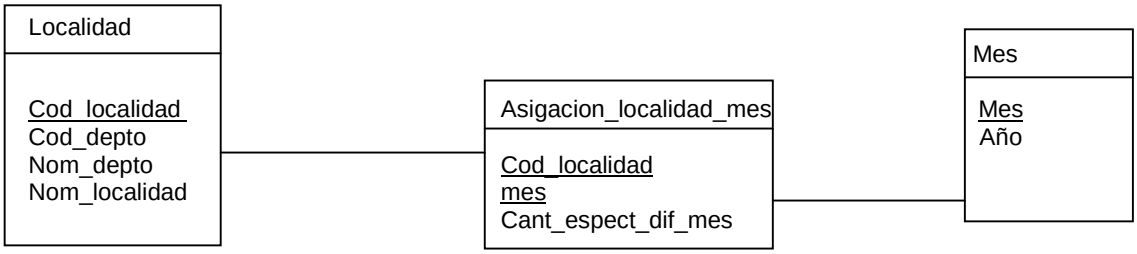
R=Exhibiciones100
Z={} card(Z)=0
{e1..ek}={}
Y={hora, cod_exhibición, importe, %_descuento}
R'=Asignacion10(cod_sala, fecha, cod_espectáculo, año)

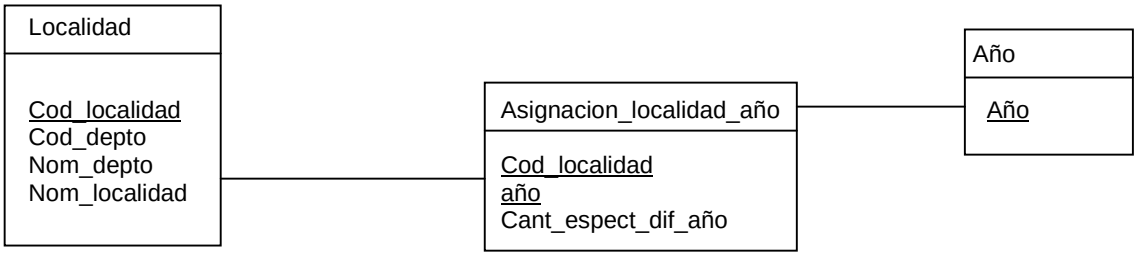
3) Aggregate generation (para contar los registros)

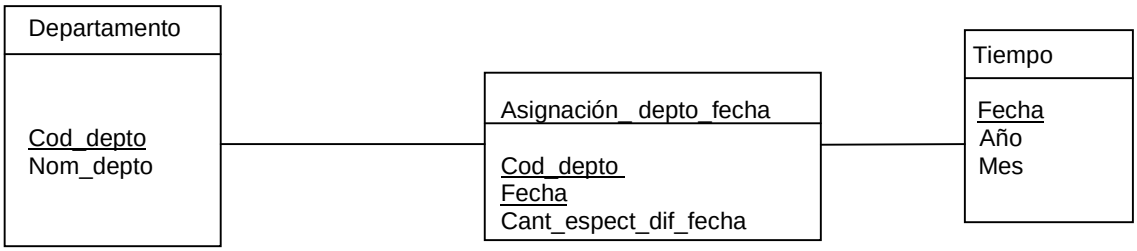
R=Asignacion10
Z={} card(z)=0
{e1...ek}={count(*)}
Y={cod_espectáculo, fecha}
R'=Asignación(cod_sala , año, cantidad_espect_sala_año)

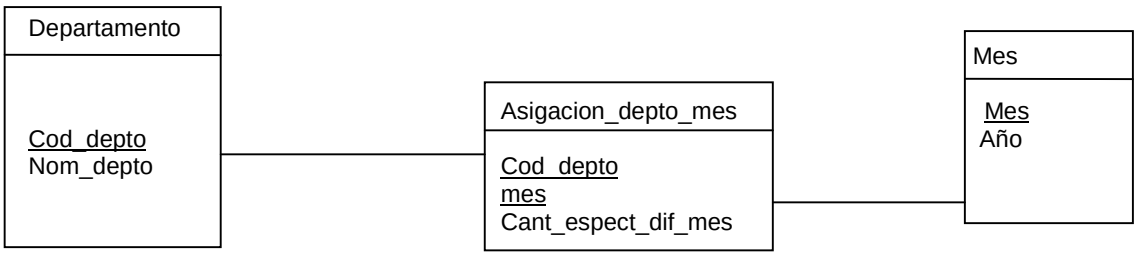
Y los demás casos son análogos:

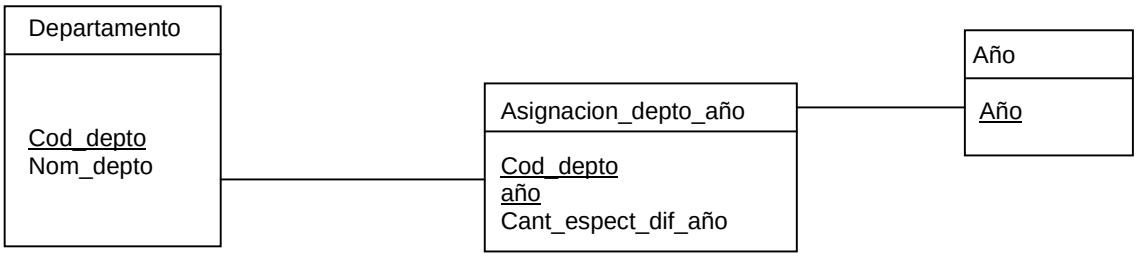



Localidad
Cod_localidad
Cod_depto
Nom_depto
Nom_localidad
Asigacion_localidad_mes
Cod_localidad
mes
Cant_espect_dif_mes
Mes
Mes
Año


Localidad
Cod_localidad
Cod_depto
Nom_depto
Nom_localidad
Asignacion_localidad_año
Cod_localidad
año
Cant_espect_dif_año
Año
Año


Departamento
Cod_depto
Nom_depto
Asignación_ depto_fecha
Cod_depto
Fecha
Cant_espect_dif_fecha
Tiempo
Fecha
Año
Mes


Departamento
Cod_depto
Nom_depto
Asigacion_depto_mes
Cod_depto
mes
Cant_espect_dif_mes
Mes
Mes
Año


Departamento
Cod_depto
Nom_depto
Asignacion_depto_año
Cod_depto
año
Cant_espect_dif_año
Año
Año

Primitivas a usar para permitir la historización de los datos de Socio

Se debería aplicar:

1) Horizontal partition a Socio obteniendo Socio_DW y Socio His

R=Socio, X={Nro_socio}

2) Temporalization a Socio_His obteniendo Socio-His1

R=Socio_His
T=fecha del momento
Key=T

Socio_His1(nro_socio, fecha, ...demás atributos de Socio...)

IV. Carga y actualización

Carga inicial de datos

Las sentencias SQL para las tareas de carga inicial de datos desde la base fuente al DW se describen a continuación, siguiendo la correspondencia *primitiva → sentencia SQL*, y se agrega la(s) instrucción(es) SQL necesaria(s) para hacer la carga directamente, a efectos comparativos.

Orden de las tareas de la carga inicial

Básicamente, para cada star schema se cargan primero sus dimensiones y luego la fact table asociada. Por lo tanto, los órdenes serían:

Cargar Asistencia:

1) Cargar Espectáculo, Socio, Tiempo y Sala (en cualquier orden)
2) Cargar Asistencia

Cargar Recaudación

1) Cargar Descuento (las otras dimensiones ya se cargaron antes)
2) Cargar Recaudación

Cargar las fact tables de Asignación (sala_año, sala_fecha, sala_mes, etc)

Para Asignación_sala_año, Asignación_sala_mes, Asignación_sala_fecha :

En la implementación no se cargan las dimensiones Año y Mes por separado, así como tampoco Localidad, Departamento. Por lo tanto, la carga se reduce a la de las otras fact tables, ya que Tiempo se había cargado previamente.

Para las otras fact tables de Asignación y las dimensiones (Localidad, Departamento, etc):

Vale lo mismo.

A SOCIO

Generación según las primitivas:

1) socio1=

```
SELECT      nro-socio,
            Nombre,
            Apellido,
            Sexo,
            Edad,
            Profesion,
            Nivel-ingreso
FROM socios
```

2) socio2=

```
SELECT      nro-socio,
            Nombre,
            Apellido,
            Sexo,
            Edad,
            Profesion,
            Nivel-ingreso,
            f_parte_entera(edad/10) rango-edad
FROM socio1
```

I

3) Socio2.5=

```
SELECT      nro-socio,
            Nombre,
            Apellido,
            Sexo,
            Rango-edad
            Profesion,
            Nivel-ingreso

FROM socio2
```

4) socio3=

```
SELECT      nro-socio,
            Nombre,
            Apellido,
            Sexo,
            Rango-edad
            Profesion,
            Nivel-ingreso,
            f_concatenar(nombre,apellido) nom-socio

FROM socio2.5
```

5) SOCIO =

```
SELECT      nro_socio,
            Nom_socio,
            Sexo,
            Rango_edad
            Profesion,
            Nivel_ingreso
FROM socio3
```

Generación directa a partir de la base fuente:

```
SOCIO =
SELECT  nro_socio,
         f_concatenar(nombre,apellido)  nom_socio,
        sexo,
        f_parte_entera(edad/10) rango_edad,   * si hacemos rango de 10 años
                                              (parte_entera=función que devuelve parte entera
                                               del cociente)
        profesión
        nivel_ingreso

FROM    socios
```

B SALA

Generación según las primitivas:

1) sala1=

```
SELECT cod_sala,nombre ,tipo ,cod_localidad
```

```
        FROM salas
```

2) sala2=

```
        SELECT cod_sala,nombre ,tipo ,cod_localidad ,nom_localidad

        FROM sala1,localidades
        WHERE sala1.cod_localidad = localidades.cod_localidad
```

3) salas3

```
        SELECT cod_sala,nombre ,tipo ,cod_localidad ,nom_localidad ,cod_departamento
        FROM sala2,localidades
        WHERE sala2.cod_localidad = localidades.cod_localidad
```

4) sala=

```
        SELECT cod_sala,nom_sala,cod_localidad,nom_localidad,tipo_sala,cod_departamento,
                Nom_departamento
        FROM salas3,departamentos
        WHERE salas3.cod_depto = departamentos.cod_departamento
```

Generación directa a partir de la base fuente:

```
sala=
SELECT  cod_sala,
          Nombre nom_sala,
          l.cod-localidad,
          nom-localidad,
          tipo tipo_sala,
          d.cod-departamento cod_depto
          nom-departamento nom_depto
FROM     salas s,
          Localidades l,
          Departamentos d
WHERE   s.cod-localidad = l.cod-localidad
   AND   l.cod-departamento = d.cod-departamento
```

C ESPECTACULO

Generación según las primitivas

1) espectaculo=

```
        SELECT cod_espectaculo,tipo tipo_espectaculo,nombre descr_espectaculo,
        FROM espectaculos
```

Generación a partir de la base fuente

espectaculo=

```
SELECT cod_espectaculo,
          Tipo tipo_espectaculo,
          Nombre descr_espectaculo

FROM     espectaculos
```

D TIEMPO

Generación según las primitivas:

1) fechas1=

```
        SELECT unique fecha dia
        FROM exhibiciones
```

2) fechas2=

```
        SELECT dia,f_mesaño(dia) mes
```

FROM fechas1

3) fechas3=
   SELECT dia,mes,f_año(dia) año
   FROM fechas2

4) tiempo=
   SELECT f_nomdia(dia) nombre_dia,dia,mes,año
   From fechas3

Generación directa a partir de la base fuente:
Tiempo=
SELECT f-nomdia(dia) * f-nomdia= función que dada una fecha devuelve nombre del día
unique fecha dia,
f_mesaño(dia) * f_mesaño= funcion que dada una fecha en formato ddmmaaaa devuelve mmaaaa
f_año(dia) * f_año = funcion que dada una fecha en formado ddmmaaa devuelve aaaa
FROM exhibiciones

E ASISTENCIAS

Generación según las primitivas:

1) asistencias1=
   SELECT cod_exhibicion,nro_socio,cod_espectaculo
   FROM exhibiciones, asistencias
   WHERE exhibiciones.cod_exhibicion = asistencias.cod_exhibicion
2) asistencias2=
   SELECT cod_exhibicion,nro_socio,cod_espectaculo, cod_sala,
   FROM asistencias1,exhibiciones
   WHERE exhibiciones.cod_exhibicion = asistencias1.cod_exhibicion
3) asistencias3=
   SELECT cod_exhibicion,nro_socio,cod_espectaculo,cod_sala,fecha
   FROM asistencias2,exhibiciones
   WHERE exhibiciones.cod_exhibicion = asistencias2.cod_exhibicion
4) asistencia =
   SELECT nro_socio,cod_espectaculo,cod_sala,fecha, count(*) cantidad
   FROM asistencias3
   GROUP BY 1,2,3,4

Generación directa a partir de las base fuente:
Asistencias=
SELECT nro-socio,
fecha,
Cod-espectaculo,
Cod-sala,
Count(*) cantidad

FROM exhibiciones e,

Asistencias a
WHERE e.cod-exhibicion = a.cod-exhibicion
GROUP BY nro-socio,fecha,cod-espectaculo,cod-sala

4 DESCUENTO

Generación según las primitivas

1) descuento=
   SELECT %descuento
   FROM exhibiciones

Generación directa a partir de las base fuente:

Descuento=
   SELECT unique %descuento
   FROM exhibiciones

5 RECAUDACION

Generación según las primitivas:

1) asistencias1=
   SELECT cod_exhibicion,
   Nro_socio,
   Cod_sala
   FROM asistencias,exhibiciones
   WHERE asistencias.cod_exhibicion = exhibiciones.cod_exhibicion

2) asistencias2=
   SELECT cod_exhibicion,
   Nro_socio,
   Cod_sala,
   %descuento
   FROM asistencias1,exhibiciones
   WHERE asistencias1.cod_exhibicion = exhibiciones.cod_exhibicion

3) asistencias3=
   SELECT cod_exhibicion,
   Nro_socio,
   Cod_sala,
   %descuento,
   cod_espectaculo
   FROM asistencias2,exhibiciones
   WHERE asistencias2.cod_exhibicion = exhibiciones.cod_exhibicion

4) asistencias4=
   SELECT cod_exhibicion,
   Nro_socio,
   Cod_sala,
   %descuento,
   cod_espectaculo,
   fecha
   FROM asistencias3,exhibiciones
   WHERE asistencias3.cod_exhibicion = exhibiciones.cod_exhibicion

5) asistencias5=
   SELECT cod_exhibicion,
   Nro_socio,
   Cod_sala,
   %descuento,
   cod_espectaculo,

```
                fecha,
                importe
        FROM asistencias4,exhibiciones
        WHERE asistencias4.cod_exhibicion = exhibiciones.cod_exhibicion
```

6) asistencias6=

```
        SELECT      cod_exhibicion,
                    Nro_socio,
                    Cod_sala,
                    %descuento,
                    cod_espectaculo,
                    fecha,
                    importe,
                    f_liquido(cod_plan,importe,%descuento) liquido
        FROM asistencias5,socios
        WHERE asistencias5.nro_socio = socios.nro_socio
```

7) asistencias7=

```
        SELECT      cod_exhibicion,
                    Nro_socio,
                    Cod_sala,
                    %descuento,
                    cod_espectaculo,
                    fecha,
                    liquido
        FROM asistencias6
```

8) recaudacion=

```
        SELECT      Nro_socio,
                    Cod_sala,
                    %descuento,
                    cod_espectaculo,
                    fecha,
                    sum(liquido)
        FROM asistencias7
        GROUP BY    Nro_socio,
                    Cod_sala,
                    %descuento,
                    cod_espectaculo,
                    fecha
```

Generación directa a partir de las base fuente:

```
    Recaudacion=
        SELECT      cod-sala,
                    %descuento,
                    cod-espectaculo,
                    nro_socio
                    fecha,
                    sum(f_liquido(cod_plan,importe,%descuento)) liquido
        FROM    exhibiciones,asistencias,socios
        WHERE asistencias.nro_socio = socios.nro_socio
            AND asistencias.cod_exhibicion = exhibiciones.cod_exhibicion

        GROUP BY    Nro_socio,
                    Cod_sala,
                    %descuento,
                    cod_espectaculo,
                    fecha
```

Nota: f_liquido: funcion que dado un código de plan,un importe y un % devuelve el importe líquido

6 ASIGNACION_SALA_FECHA

1) Asignacion1=
```
SELECT  cod_sala, fecha, cod_espectáculo
FROM  Exhibiciones
GROUP BY cod_sala, fecha, cod_espectáculo
```

2)
```
SELECT cod_sala, fecha, count(*)
FROM Asignacion1
GROUP BY cod_sala, fecha
```

Generación directa a partir de las bases fuentes:

1)
```
SELECT cod_sala,fecha,cod_espectaculo
FROM exhibiciones
GROUP BY cod_sala,fecha,cod_espectaculo ;
```

2)
```
SELECT cod_sala,fecha,count(*)
FROM  asig_sala_fecha_aux
GROUP BY  cod_sala,fecha
```

7 ASIGNACION_SALA_MES

1) Exhibiciones10=
```
SELECT *, FORMAT(fecha,”mmyyyy”) mes
FROM Exhibiciones
```

2) Asignacion10=
```
SELECT  cod_sala, fecha, mes, cod_espectáculo
FROM  Exhibiciones10
GROUP BY cod_sala, fecha, mes,  cod_espectáculo
```

3)
```
SELECT cod_sala, mes, count(*)
FROM Asignacion10
GROUP BY cod_sala, mes
```

Generación directa a partir de las bases fuentes:

1)
```
SELECT cod_sala,  format(fecha,'mmyyyy'),cod_espectaculo
FROM exhibiciones
GROUP BY cod_sala, format(fecha,'mmyyyy'),cod_espectaculo;
```

2)
```
SELECT cod_sala,mes,count(*)
FROM  asig_sala_mes_aux
GROUP BY cod_sala,mes
```

8 ASIGNACION_SALA_AÑO

1) Exhibiciones100=
```
SELECT *, FORMAT(fecha,’yyyy’) año
FROM Exhibiciones
```

2) Asignacion100=
```
SELECT  cod_sala, fecha, año, cod_espectáculo
FROM  Exhibiciones100
GROUP BY cod_sala, fecha, año,  cod_espectáculo
```

3) SELECT cod_sala, año, count(*)
FROM Asignacion100
GROUP BY cod_sala, año

Generación directa a partir de las bases fuentes:

1) SELECT cod_sala, format(fecha,'yyyy'), cod_espectaculo
FROM exhibiciones
GROUP BY cod_sala, format(fecha,'yyyy'), cod_espectaculo;

2) SELECT cod_sala,año , count(*)
FROM asig_sala_año_aux
GROUP BY cod_sala, año

Nota:

Para las otras fact tables: Asignación_localidad_fecha, Asignación_localidad_mes, etc, es análogo.

Carga periódica (refresque) de los datos

Para la detección de los cambios en los atributos de los socios existen dos posibilidades: verificar que ellos (sexo, edad, profesión y nivel de ingresos) hayan cambiado al realizar la carga del DataWarehouse, y si cambiaron, proceder como más adelante se indica, o que los sistemas operacionales aporten la información de los cambios (en la formade movimientos de cambios) a los programas de carga.

Cuando se detecta un cambio (por alguna de las dos vías):

a) se determina el nuevo tipo de socio, *tipo_nuevo*, correspondiente a la nueva combinación de atributos en MINI

b) se genera un registro nuevo en SOCIO, con tipo_socio = *tipo_nuevo* y fecha_validez la que determinen los sistemas operacionales o la de la fecha de carga de datos. En la implementación no se tomó en cuenta la historización ni la minidimensión por estar fuera del alcance de este trabajo.

V. Consideraciones finales

De la metodología usada

En términos generales, se realizaron reuniones de unas dos o tres horas cada una, entre los tres integrantes del equipo. Como se vio, se realizaron unas seis reuniones de coordinación y consultas con los docentes responsables.

a. Diseño conceptual.

El modelo CMDM permitió un diseño conceptual de calidad al permitir tener una documentación completa del mismo y enfatizando la importancia de los requerimientos del usuario; consideramos que constituye un buen medio de comunicación entre el analista y los usuarios. El encare del desarrollo del Data Warehouse basado en los requerimientos parece adecuado en la medida en que nos estamos refiriendo a sistemas a ser usados por usuarios que tienen muy claro la información que necesitan para tomar sus decisiones. Además, cuanto más tengamos presente al usuario, hay más posibilidades de incluir en el diseño aspectos futuros que el usuario nos adelanta a través de algunos requerimientos, y prolongamos así la vida útil del diseño actual. Podemos pues decir que el CMDM hace aportes a la calidad en el diseño del Data Warehouse, aunque si bien aun no sabemos bien como medir bien la misma en estos casos, intuitivamente se ve que la documentación y consideración de los requerimientos del usuario son básicas.

Las primitivas también proporcionan esa buena documentación tan necesaria, permitiendo además que el usuario vea en forma clara como se generan los datos de Data Warehouse, con los cuales él puede tomar decisiones de mucho peso, a partir de sus sistemas operacionales. Sin embargo, la documentación de su semántica y ejemplificación es algo que sin duda vale la pena trabajar en todos los aspectos.

b. Diseño lógico.

Se hizo un diseño lógico-relacional. Dado que no teníamos información sobre cómo cambiaban los datos, como ya se vio, se utilizó la solución del galaxy schema, es decir, varios star schemas con tablas de hechos independientes, por considerarse la más balanceada entre performance y facilidad del mantenimiento de datos. Por otra parte, ya se vio que la metodología de Kortink y Moody hubiese llevado a un Datawarehouse sobredimensionado, si no se hubiesen considerado los requerimientos.
Se pretendió aclarar la relación entre los datos de las bases fuentes y las del Datawarehouse usando la traza que proveen las primitivas de transformación de esquemas. Adicionalmente, las primitivas nos ofrecieron dos ventajas: por un lado, permitieron obtener una especificación abstracta , independiente del lenguaje de consultas de bases de datos a usar, de cómo se podría realizar la carga inicial del Data Warehouse, lo cual nos trae a la mente ideas tales como "portabilidad", "facilidad de mantenimiento" del diseño del Data Warehouse, etc. ; y por el otro, también dan la especificación " a la SQL" (mecánicamente) de los pasos que habría que seguir obtener esas transformaciones, lo que puede ser de ayuda en el caso de trabajar con SQL como lenguaje de consulta de bases de datos.

c. En la implementación.

Las bases de datos siempre fueron almacenadas como bases Access 97, por simplicidad de manejo, aunque si los datos hubiesen tenido un volúmen importante, se debería haber elegido un manejador de bases de datos más performante y con más opciones de seguridad, tuning, etc. El diseño lógico-relacional sirvió de base a la implementación en PowerPlay Ver 6.0.221. En cuanto a la carga inicial de datos, se utilizó SQL Server (Data Transformation Services) Ver 7.0. Todo se ejecutó sobre un cliente NT 4.0. No se diseñó ninguna estrategia de mantenimiento de datos, porque quedaba fuera del alcance de este trabajo.

Con respecto al "feeling" que tuvimos con las herramientas, si bien no tuvimos una experiencia significativa con ellas, creemos que, en la carga inicial, el DTS resultó muy atractivo visualmente, pero demasiado inestable en su utilización. Para citar algunos ejemplos:

- ejecutar varias veces el mismo package que daba errores, eliminaba el error

- frente a un error cuyo origen desconocíamos y no podíamos descubrir, optábamos por hacer un nuevo package con la misma estructura y copiando las sentencias de carga. El nuevo package funcionaba correctamente.

- Al modificar el nombre de una columna de una tabla, el package daba error- indicando el nombre anterior como no existente en las tablas- aunque todas las transformaciones habían sido ajustadas. Nuevamente en este caso, la generación de un nuevo package copiando todas las definiciones del anterior ejecutaba exitosamente.

En esos casos no encontramos trazas en los archivos de log que nos dieran una pista del momento en que se suscitaba el problema. El manual dela herramienta no permitía un acceso rápido a temas específicos de manejo, y pasaba más tiempo explicando la teoría del datawarehousing que diciendo como funcionaba el producto en sí, lo que resultaba en un libraco de poca utilidad.

Las herramientas del PowePlay, el Transformer y el Impromptu, demostraron ser atractivas visualmente, estables y versátiles. Se utilizaron las versiones existentes en el InCo, aunque podríamos haber trabajado con versiones posteriores de nuestra propiedad, pero no lo hicimos por compatibilidad.

Ventajas y desventajas de las herramientas comparadas con programas

Dados los problemas vistos, para el ejercicio planteado y en la carga inicial, nos hubiera resultado más eficiente realizar un programa o script con sentencias SQL, en particular, porque todos los datos eran extraidos de la misma fuente y no requeríamos forzosamente de procesos en paralelo.

Respecto a las consultas en sí, la herramienta de Cognos sería difícilmente superada por programas ad hoc, al menos en este caso y con los volúmenes de datos actuales.

Del desarrollo en el tiempo del proyecto

El proyecto siguió aproximadamente el siguiente cronograma

Mes	***Nro. de reuniones***	***Cantidad de horas prom p/reunión***	***Etapa del proyecto***
Dic/01	2	2	Def. del modelo conceptual
Ene/02	8	2	Idem
Feb/02	9	2	Idem y Escritura de primitivas
Mar/02	10	2 ½	Carga inicial, uso del DTS
Abr/02	15	3	DTS y generación de cubos
May/02	2	2 ½	Armado final de los cubos, etc.

Total de horas :

Notas:

- *Se debe agregar el trabajo de una persona, 6 hs. para la carga y diseño de las bases de datos fuentes.*
- *No se tuvo en cuenta el tiempo de documentación ni de las reuniones de consulta y coordinación con los docentes. Se realizaron alrededor de 6 reuniones de este tipo, de 2 horas cada una.*
- *Cada reunión solía incluir a los tres integrantes del equipo de trabajo, pero en algunas solo trabajaron dos. Se ignora este hecho, por haber estado cada uno de los integrantes ausentes dos semanas en distintos momentos.*
- *El uso del DTS se hizo especialmente dificultoso, bien sea por desconocimiento de la herramienta como por bugs de la misma, lo que justifica la gran cantidad de horas asignadas al diseño de la carga inicial de datos.*

De la utilización de las herramientas informáticas

Documentación de los paquetes y cubos

Se detallan a continuación los nombres de los componentes implementados y su uso.
En términos generales, las bases de datos auxiliares estan en el directorio ARTE junto con las fuentes, y tienen extensión AUX. Las bases finales del DataWarehouse se hallan en ARTEDW.

De los packages DTS:

* Asig_sala: Es el Package que realiza la carga inicial de Asignación_sala_fecha, Asignación_sala_mes y Asignación_sala_año

* Arte_dw_asist_recaud carga Asistencia y Recaudación

De los cubos del Impromptu-Transformer:

* Asistencia implementa las consultas sobre Recaudación y Asistencia

* Asig_sala implementa las consultas sobre Asignación_sala_fecha, Asignación_sala_mes y Asignación_sala_año.

Ventajas y desventajas de cada herramienta comparadas con programas

Con respecto a las herramientas usadas, si bien no tenemos una experiencia significativa con ellas, creemos que, respecto a la carga inicial, el DTS resultó ser muy atractivo visualmente, pero demasiado inestable en los resultados obtenidos. No permitía establecer órdenes de ejecución de los distintos packages, y tampoco permitía establecer las relaciones de precedencia entre las tareas dentro de un package que se ejecutan en paralelo. La documentación (Manual) pasaba más tiempo explicando la teoría de datawarehousing que diciendo como funcionaba el producto en sí, lo que resultaba en un mamotreto impreso de poca utilidad. Esperemos que estos inconvenientes desaparezcan en el futuro. Francamente, tal vez hubiese sido más práctico, para las limitadas funcionalidades de la herramienta que utilizó este caso, hacer un programa (script) con las sentencias SQL necesarias para la carga inicial de datos.

Las herramientas del Power Play, el Transformer y el Impromptu, nos mostraron ser atractivas visualmente, estables y versátiles. Se utilizaron las versiones existentes en el InCo, aunque podríamos haber trabajado con versiones posteriores, pero no lo hicimos para tener compatibilidad.

VI. Indice

VII. Bibliografía.

++++++Trabajo de Marotta
++++++Apuntes de clase
++++++Revista Computer, IEEE, dic/2001
++++++Biblioteca Digital de la IEEE

++++++Panel on Data Warehouse Quality Issues (Yannis Vassiliou, Mathias Jarke, Timos Sellis, Eric Simon). Biblioteca Digital de la IEEE. **Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems** .*1998.*