Trabajo Práctico: Bases de Datos I

UNGS – Segundo Semestre 2024

1 Modelo de Datos

A continuación se presenta el modelo de datos que se utiliza para almacenar información relativa a la gestión de los llamados que realizan al Centro de Atención Telefónica (CAT), les clientes del proveedor de servicios de internet *Skynet*.

```
cliente(
    id_cliente:int,
    nombre:text,
    apellido:text,
    dni:int,
    fecha_nacimento:date,
    telefono:char(12),
    email:text --válido
)

operadore(
    id_operadore:int,
    nombre:text,
    apellido:text,
    dni:int,
    fecha_ingreso:date,
    disponible:boolean
)

cola_atencion(
    id_cola_atencion:int,
    id_cliente:int,
    f_inicio_llamado:timestamp,
    id_operadore:int, --se informa al iniciar la atención
    f_inicio_atencion:timestamp,
    f_fin_atencion:timestamp,
    estado:char(15) --`en espera', `en linea', `finalizado', `desistido'
)

tramite(
    id_tramite:int,
    id_cliente:int,
    id_cola_atencion:int,
    tipo_tramite:char(10), --`consulta',`reclamo'
    f_inicio_gestion:timestamp,
    descripcion:text, --transcripción de lo solicitado por le cliente
    f_fin_gestion:timestamp,
    respuesta:text, --transcripción de lo respondido por Skynet
    estado:char(15) --`iniciado', `solucionado', `rechazado'
)
```

```
rendimiento_operadore(
    id_operadore:int,
    fecha_atencion:date, --se sumariza por fecha de atención
    duracion_total_atenciones:interval,
    cantidad_total_atenciones:int,
    duracion_promedio_total_atenciones:interval,
    duracion_atenciones_finalizadas:interval,
    cantidad_atenciones_finalizadas:int,
    duracion_promedio_atenciones_finalizadas:interval,
    duracion_atenciones_desistidas:interval,
    cantidad_atenciones_desistidas:int,
    duracion_promedio_atenciones_desistidas:interval
)

error(
    id_error:int,
    operacion:char(15),
           --`nuevo llamado', `baja llamado', `atencion llamado'
           --`fin llamado', `alta tramite', `cierre tramite'
    id_cliente:int,
    id_cola_atencion:int,
    tipo_tramite:char(10),
    id_tramite:int,
    estado_cierre_tramite:char(15),
    f_error:timestamp,
    motivo:varchar(80)
)

envio_email(
    id_email:int,
    f_generacion:timestamp,
    email_cliente:text,
    asunto:text,
    cuerpo:text,
    f_envio:timestamp,
    estado:char(10) --`pendiente', `enviado'
)

-- Esta tabla *no* es parte del modelo de datos, pero se incluye para
-- poder probar la funcionalidad del sistema.
datos_de_prueba(
    id_orden:int, --en qué orden se ejecutarán las transacciones
    operacion:char(15),
           --`nuevo llamado', `baja llamado', `atencion llamado'
           --`fin llamado', `alta tramite', `cierre tramite'
    id_cliente:int,
    id_cola_atencion:int,
    tipo_tramite:char(10),
    descripcion_tramite:text,
    id_tramite:int,
    estado_cierre_tramite:char(15),
    respuesta_tramite:text
)
```

El sistema debe administrar la cola de espera de los llamados recibidos, la atención de les operadores a los llamados, el alta y el cierre de los trámites, y mantener toda la información de les clientes y de les operadores.

Les clientes deben ser informades vía email cuando ocurran eventos importantes sobre su trámite. Además, es necesario mantener, en tiempo real, información sumarizada sobre la duración promedio de los llamados atendidos por cada operadore.

2 Creación de la Base de Datos

La base de datos deberá nombrarse con el apellido de cada integrante del equipo en orden alfabético, separados con underscores, y seguidos del string `_db1`, por ejemplo:

`giunta_maradona_palermo_riquelme_db1`

No respetar esto, automáticamente implica la desaprobación del trabajo práctico.

Se deberán crear las tablas respetando ***exactamente*** *los nombres de tablas, atributos, y tipos de datos especificados.*

Se deberán agregar las PK's y FK's de todas las tablas, por separado de la creación de las mismas. Además, se deberá tener la posibilidad de borrar todas las PK's y FK's.

3 Instancia de los Datos

Los datos de les clientes y de les operadores, así como los datos de entrada para las transacciones de prueba, deberán cargarse en las tablas correspondientes a partir de los siguientes documentos JSON:

- `clientes.json`
- `operadores.json`
- `datos_de_prueba.json`

Estos archivos `json` se encuentran en el directorio compartido de Google Drive de la materia, junto a este archivo `pdf`.

4 Stored Procedures y Triggers

El trabajo práctico deberá incluir los siguientes stored procedures ó triggers:

- **ingreso de llamado:** se deberá incluir la lógica que reciba un id de cliente, y que devuelva el id de cola de atención si se logró ingresar al cliente en la cola de espera, ó `-1` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar el ingreso:
  - Que el id de le cliente exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de cliente no válido`.

  Si las validaciones pasan correctamente, se deberá insertar una fila en la tabla `cola_atencion` con los datos de le cliente, y la fecha y hora de inicio del llamado, dejando su estado como `en espera`.

- **desistimiento de llamado:** se deberá proveer la lógica que permita desistir del llamado a une cliente. Se debe recibir un id de cola de atención, y retornar `true` si se logra marcar al llamado como desistido, ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar la baja:

- Que el id de cola de atención exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de cola de atención no válido`.

- Que el estado del llamado sea `en espera` ó `en linea`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?el llamado no está en espera ni en línea`.

Si las validaciones pasan correctamente, se deberá actualizar la fila correspondiente de la tabla `cola_atencion` con el estado `desistido`. En caso de que el llamado se encontrara previamente en el estado `en linea`, deberá además actualizarse la fecha y hora de fin del llamado.

- **atención de llamado en espera:** se deberá incluir la lógica que permita asignar une operadore disponible al llamado que se encuentre en el primer lugar de la cola de espera. Se debe retornar `true` si se logra asignar a le operadore, ó `false` si no. El procedimiento deberá validar los siguientes elementos antes de confirmar la asignación:

  - Que exista al menos un llamado en estado `en espera`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no existe ningún llamado en espera`.

  - Que exista al menos une operadore en estado disponible. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no existe ningune operadore disponible`.

  Si las validaciones pasan correctamente, se deberá asegurar que se ejecuten las siguientes acciones de forma completa (en caso de que se produzca algún error o inconveniente, las acciones que se hayan realizado deberán deshacerse):

  - Se seleccionará a une de les operadores que estén disponibles.

  - Se actualizará como no disponible a le operadore seleccionade.

  - En la fila de la tabla `cola_atencion`, correspondiente al primer llamado de la cola de espera, deberá grabarse el id de le operadore seleccionade, y la fecha y hora de inicio de la atención, y su estado deberá actualizarse como `en linea`.

- **alta de trámite:** se deberá proveer la lógica que reciba un id de cola de atención, un tipo de trámite y una descripción, y que devuelva el id de trámite si se logró crear el nuevo trámite, ó `-1` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar el alta:

  - Que el tipo de trámite recibido como parámetro sea `consulta` o `reclamo`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?tipo de trámite no válido`.

  - Que el id de cola de atención exista, y que su estado no sea `en espera`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de cola de atención no válido`.

  Si las validaciones pasan correctamente, se deberá insertar una fila en la tabla `tramite` con el id de le cliente asociado al llamado, el id de cola de atención, el tipo y la descripción del trámite, y la fecha y hora de inicio de la gestión, dejando su estado como `iniciado`.

- **finalización de llamado:** se deberá proveer la lógica que permita dar por finalizado el llamado de une cliente. Se debe recibir un id de cola de atención, y retornar `true` si se logra marcar al llamado como finalizado, ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar la finalización del llamado:

  - Que el id de cola de atención exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de cola de atención no válido`.

  - Que el estado del llamado sea `en linea`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?el llamado no está en línea`.

Si las validaciones pasan correctamente, se deberá actualizar la fila correspondiente de la tabla `cola_atencion` con la fecha y hora de fin de atención, y dejando su estado como `finalizado`.

- **cierre de trámite: (opcional)** se deberá proveer la lógica que permita cerrar un trámite. Se debe recibir un id de trámite, un estado de cierre y un texto de respuesta, y retornar `true` si se logra cerrar el trámite, ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar el cierre:
  - Que el estado de cierre recibido como parámetro sea `solucionado` o `rechazado`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?estado de cierre no válido`.
  - Que el id de trámite exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de trámite no válido`.
  - Que el estado del trámite sea `iniciado`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?el trámite se encuentra cerrado`.

  Si las validaciones pasan correctamente, se deberá actualizar la fila correspondiente de la tabla `tramite` con la fecha y hora de fin de gestión, el estado de cierre y el texto de respuesta.
- **reporte de rendimiento de operadores:** se deberá mantener en tiempo real—en la tabla `rendimiento_operadores`—, información sumarizada del tiempo promedio de atención de les operadores, cuando sucedan las siguientes novedades:
  - Cada vez que un llamado asignado a une operadore se marque como `desistido`, se debe sumar la duración de ese llamado, tanto a la duración total de atenciones de le operadore como a la duración de sus atenciones desistidas, incrementar en uno las respectivas cantidades de atenciones, y recalcular ambas duraciones promedio.
  - Cada vez que un llamado se marque como `finalizado`, se debe sumar la duración de ese llamado, tanto a la duración total de atenciones de le operadore como a la duración de sus atenciones finalizadas, incrementar en uno las respectivas cantidades de atenciones, y recalcular ambas duraciones promedio.

  En todos los casos, se debe actualizar la fila de la tabla `rendimiento_operadores` correspondiente a le operadore asignade y a la fecha de fin del llamado. De no existir la fila, deberá insertarse con los valores iniciales de acuerdo a la novedad producida.
- **envío de emails a clientes: (opcional)** el trabajo práctico deberá proveer la funcionalidad de generar emails para ser enviados a la dirección de email de le cliente—en la tabla `envio_emails`— cuando sucedan las siguientes novedades:
  - Cada vez que se confirme el alta de un trámite, se debe ingresar automáticamente un email con el asunto '`Skynet - nuevo trámite: [id tramite]`', informando en el cuerpo del email los datos de le cliente y los datos importantes del trámite—tipo de trámite, fecha y hora de inicio de gestión, y descripción de lo solicitado—, remplazando en el asunto `[id tramite]` por el valor del id del trámite generado.
  - Cada vez que se confirme el cierre de un trámite, se debe ingresar automáticamente un email con el asunto '`Skynet - cierre de trámite: [id tramite]`', informando en el cuerpo del email los datos de le cliente y los datos importantes del trámite—tipo de trámite, fecha y hora de inicio de gestión, fecha y hora de fin de gestión, descripción de lo solicitado, estado de cierre, y texto de respuesta de Skynet—, remplazando en el asunto `[id tramite]` por el valor del id del trámite cerrado.

Se deberá crear una tabla con datos de entrada de las operaciones para poder probar la funcionalidad del sistema, que deberá contener los siguientes atributos: `id_orden`, `operacion`, `id_cliente`, `id_cola_atencion`, `tipo_tramite`, `descripcion_tramite`, `id_tramite`, `estado_cierre_tramite`,

`respuesta_tramite`. A partir de esta tabla, se deberá hacer un procedimiento de testeo que invoque a las transacciones correspondientes de acuerdo a la operación. Los datos deberán tomarse del archivo `datos_de_prueba.json` provisto en el directorio compartido de Google Drive.

5 JSON y Bases de datos NoSQL

Por último, para poder comparar el modelo relacional con un modelo no relacional NoSQL, se pide guardar los datos de clientes, operadores, llamados (al menos tres) y trámites (al menos tres) en una base de datos NoSQL basada en JSON. Para ello, utilizar la base de datos BoltDB.

6 Aplicación Solicitada

Todo el código SQL escrito para este trabajo práctico, deberá ejecutarse desde una aplicación **simple** command line escrita en Go.

A modo de ejemplo:

```
$ go run .

### Menú ###

1  Crear base de datos
2  Crear tablas
3  Agregar PKs y FKs
4  Eliminar PKs y FKs
5  Cargar datos
6  Crear stored procedures y triggers
7  Iniciar pruebas
8  Cargar datos en BoltDB
0  Salir

opción> 1

Base de datos creada!

opción> 2

Tablas creadas!

opción> 3

(etc ...)
```

Tener en cuenta, que en la opción 5 deben cargarse todos los datos provistos, incluidos los datos de prueba.

Observar, que la aplicación no debe solicitar datos individuales a le usuarie—más allá de los números de las opciones mencionadas.

Por último, el código NoSQL, puede ejecutarse, como se muestra en el ejemplo, desde la opción 8 de este menú, ó si el equipo lo considera, se puede hacer otra aplicación simple en Go por separado.

7 Condiciones de Entrega y Aprobación

No respetar alguna de las siguientes condiciones, representa la desaprobación del trabajo práctico:

- El trabajo es grupal, en grupos de, *exactamente*, cuatro integrantes. El trabajo práctico se debe realizar en un repositorio **privado** `git`, hosteado en **GitLab** con el apellido de les cuatro integrantes, separados con guiones, seguidos del string `-db1` como nombre del proyecto, e.g. `giunta-maradona-palermo-riquelme-db1`. Agregar como `owner`'s a los docentes de la materia, los usernames de Gitlab `hdr` y `danbert` al momento de la creación del repositorio.

- La fecha de entrega *máxima* es el 25 de noviembre de 2024 en horario de clase, con una defensa presencial del trabajo práctico por cada grupo, en la cual se mirará lo que se encuentre en el repositorio `git` hasta ese momento, y se harán distintas preguntas a cada integrante del grupo.

- El informe del trabajo práctico debe incluirse en el directorio principal del repositorio con el nombre de archivo `README.adoc`, y debe realizarse en formato Asciidoc. Para ello, cuentan con una guía en **hdr.gitlab.io/adoc**.

- Usar apropiadamente `git` y GitLab. **No usar branches, no usar rebase, no borrar el repo una vez creado**. Asegurarse de configurar merge, nombre y apellido, y email:

  ```
  $ git config --global pull.rebase false
  $ git config --global user.email 'diego@maradona.com'
  $ git config --global user.name 'Diego Armando Maradona'
  ```

  Durante la creación de la cuenta de GitLab, configurarla para **uso personal**.

  No subir archivos al repo desde Windows (a menos que se aseguren que los archivos tengan el formato Unix).

- *Trabajo de investigación:* En este trabajo práctico van a tener que investigar por su cuenta cómo se hacen algunas cosas en PostgreSQL. **No usen ChatGPT, ni busquen en Stack Overflow ó sitios similares, para eso tienen la documentación oficial de PostgreSQL.**