Índice

Consideraciones previas

Modelo y base de datos

Tipos de datos

Decisiones tomadas en la migración de datos

Aplicación desktop

Conexión

Pantalla Principal

ABM rol

Log in y seguridad

ABM Usuario

ABM Cliente

ABM Hotel

ABM Habitación

Régimen de estadía

Generar o modificar reserva

Cancelar reserva

Registrar estadía

Registrar consumibles

Facturar estadía

Listado estadístico

Anexo

CONSIDERACIONES PREVIAS

En este documento se plasmarán las estrategias y decisiones adoptadas en el diseño y desarrollo del sistema completo.

IMPORTANTE: Se considera que quien ejecute la aplicación tenga creada la Base de Datos GD1C2018. Los programas se han desarrollado en distintas máquinas pero se han probado en una VM con Windows 7 x64 y 3GB de RAM. La migración con esta configuración promedia los dos minutos.

MODELO Y BASE DE DATOS

TIPOS DE DATOS

En primer lugar, plasmamos en borrador todas las tablas que debían aparecer sí o sí en el sistema, con sus respectivos campos, ya que formaban parte de la tabla maestra. Si bien no les encontrábamos mucho sentido a algunos tipos de dato, los mantuvimos siempre, ya que era una consigna principal. En el momento que tuvimos que crear tablas y campos nuevos, como por ejemplo usuario que antes no existía, decidimos también mantener los mismos tipos por una cuestión de orden. Por ejemplo, no tiene sentido que un nombre o apellido contenga mas de 50 caracteres, exagerando, pero igualmente lo dejamos en nvarchar(255).

DECISIONES TOMADAS EN LA MIGRACION DE DATOS

La primera condición que se revisa en la corrección del trabajo práctico es que haya persistencia en la base de datos, por eso al iniciar el sp se descarta el mismo, si ya existe, y se crea de nuevo en limpio. La misma estrategia elegimos con respecto a todas las tablas y otros procedures utilizados. De esta forma pudimos probar tranquilamente todos los ABMs, insertando y modificando registros, sin que afecte a la base de datos otorgada por la cátedra.

* Se cargan inicialmente 3 roles: Administrador, Recepcionista y Guest. Se agrega un rol extra, Administrador General. Este contiene todas las funcionalidades y se lo asignamos al usuario SYSADM que lo hemos utilizado durante el desarrollo. Esto fue hecho para no alterar los roles anteriores y no crear uno nuevo cada vez que ejecutábamos la migración. Este rol también es asignado al usuario admin pedido en el enunciado.
* Dos clientes no pueden poseer el mismo tipo y Nro. de Documento, ni tampoco pueden existir clientes con el mismo mail. En la Tabla Maestra de datos, esta condición no se cumple, y aparecen clientes con mismo nro. y Tipo de Documento, así como también con el mismo mail. La técnica utilizada en la resolución fue la creación de un atributo llamado Cliente\_consistente de tipo bit, en el cual al cliente con un código mayor, que cumpla el problema se le coloca 0 en Cliente\_consistente y al otro se le coloca un 1. De esta forma no se pierden los datos de la tabla maestra, y obtenemos una tabla cliente, con un estado que informa su consistencia.
* En la tabla maestra de datos, hay facturas con sus correspondientes ítems, pero la suma de sus ítems no es igual al total de la factura. La técnica utilizada en la resolución fue la creación de un atributo llamado Factura\_consistente de tipo bit, el cual se coloca en 0 cuando la igualdad no se cumple y en caso contrario se coloca 1. De esta forma no se pierden los datos de la tabla maestra, y obtenemos una tabla factura, con un estado que informa su consistencia, las facturas con estado de consistente = 0, no son tomadas en cuenta para los cálculos hechos posteriormente.
* Se plantea la posibilidad de tener distintos tipos de Documento (DNI, LE, Pasaporte, etc.). En la tabla maestra los clientes poseen un campo que solo se refiere al Nro de pasaporte, por lo tanto se realiza la migración de todos los clientes utilizando un tipo de documento ‘Pasaporte’ con su correspondiente número. Para manejar esta problemática, se creó una tabla aparte llamada parámetro (un nombre genérico por si era necesario incorporar otros datos necesarios para el desarrollo. No se le ha dado otro uso pero se deja la posibilidad de incorporar nuevos datos en el futuro)
* Los roles, funciones, usuarios, régimen y habitación tienen un atributo estado, para poder realizar una baja lógica. En el que 1 significa que está activo y 0 lo contrario.
* Observamos que todos los hoteles que están en la tabla maestra son de Argentina, por eso por cuestiones de simplicidad en la migración le pusimos por default Argentina. Solo en la migración, es decir que esto no significa que si se crea un nuevo hotel necesariamente esté en dicho país.
* El estado de cancelación por No-Show de la reserva, se dará cuando el cliente intente ingresar al Hotel fuera de término (o sea tarde).
* A fin de simplificar los estados de reserva, se le asignó un número a cada estado:
* Reserva correcta pasa a ser estado 1
* Reserva modificada pasa a ser estado 2
* Reserva cancelada por Recepción pasa a ser estado 3
* Reserva cancelada por Cliente pasa a ser estado 4
* Reserva cancelada por No-Show pasa a ser estado 5
* Reserva con ingreso (efectivizada) pasa a ser estado 6

DECISIONES TOMADAS EN EL DISEÑO DEL DER (VER DER ADJUNTO)

* La tabla “ClienteXEstadia” va a mantener la relación entre todos los clientes que se alojan, indicando a qué nro de estadía corresponden.
* En “ConsumibleXEstadia” podemos relacionar los consumibles y la cantidad correspondiente que se consumieron en cada estadía. Facilitando el cálculo de la factura.
* Se realiza una tabla de tarjetas de crédito, utilizada para el pago de las facturas a través de esta vía.
* Se genera una tabla llamada Disponibilidad, para un mejor manejo de ella.
* Se genera una tabla llamada ReservaMod, para tener un seguimiento de las modificaciones de la reserva.

DECISIONES TOMADAS EN EL FUNCIONAMIENTO DE LA APLICACIÓN DESKTOP

ABM EN GENERAL

Para simplificar el código en la aplicación, según el botón que se seleccione (alta, baja o modificación) se actualiza el modoABM. Luego, se tiene un switch del modo elegido para realizar las validaciones y operaciones correspondientes.

Con respecto a las validaciones, sólo no permitimos campos vacíos en el modo alta (ya que el enunciado pide que todos los campos sean obligatorios). A nuestro parecer, también sería importante que en modificación no se permitan campos vacíos, pero nos encontramos con que no todos los registros que provengan del sistema antiguo tenían la totalidad de sus campos completos, o bien tenían incompletos los campos nuevos que necesitamos crear nosotros y está claro que no podíamos inventarlos, por lo que si el usuario quisiera hacer alguna modificación a esos registros debería poder sin necesidad de completar los campos que tienen datos “perdidos”, que quedaron en el pasado y no se pueden volver a recuperar.

Si bien el enunciado dice que al dar de alta debe elegirse el estado, pensamos que es lógico que si se le está dando de alta a algo se pretenda que el estado sea activo, por eso al dar de alta la variable estado que se le pasa desde la aplicación es siempre 1. Es decir, el usuario no elige el estado del nuevo registro, pero sí se carga automáticamente, ya que los estados son un tema fundamental en este TP. En el caso de que se quiera dar de baja a un registro se pone directamente en 0 (exceptuando el caso de que el registro ya esté dado de baja, acá simplemente no está permitido). Por último, en el caso de que sea modificación, pensamos también que sería lógico que si se quieren hacer modificaciones es porque el registro debe estar activo, por lo que, si es así, simplemente se modifica, si no, primero se le pregunta al usuario si desea volver a darle de alta, en caso de responder que no, no puede hacer modificaciones, ya que dicho registro “no existe momentáneamente”, es decir, existe pero tiene una baja lógica, que en vida real significaría que no, sólo que no teníamos permitido borrar nada físicamente de la base.

ABM CLIENTES

Nos parecía más lógico que el piso no sea un dato numérico, sino una cadena, ya que una persona que no vive en un edificio tiende a poner un guión medio, o las letras PB correspondientes a planta baja, pero en la tabla maestra ya teníamos que el tipo de dato era numérico, por lo que no teníamos permitido cambiarlo. En este caso la estrategia elegida es, en primero lugar, hacer entender al usuario que no es un dato obligatorio, es decir, si no vive en un edificio, no poner ninguna cosa extraña y simplemente dejarlo vacío. Pero internamente el stored procedure necesita obligatoriamente recibir algo desde la aplicación (con el departamento no tuvimos problema porque es una cadena y acepta vacíos), entonces al verificar los datos ingresados, si el piso está vacío, nos pareció lógico que se ponga automáticamente en 0, similar a una planta baja.

GESTIÓN DE RESERVAS