1. Diseño de la aplicación:

SELECT S.NombreTabla, S.NombreConstraint, X.NombreCol,w.NombreColPK,w.tipoDatoColPK

FROM (

(SELECT TABLE\_NAME AS NombreTabla, CONSTRAINT\_NAME AS NombreConstraint FROM SYS.ALL\_CONSTRAINTS WHERE OWNER = 'ISIS2304A231720') s

INNER JOIN

(SELECT TABLE\_NAME as NombreTabla, COLUMN\_NAME AS NombreCol , POSITION as

OrdenColPK, CONSTRAINT\_NAME AS NombrePK

FROM SYS.ALL\_CONS\_COLUMNS WHERE OWNER = 'ISIS2304A231720') X

ON S.NombreTabla=X.NombreTabla AND S.NombreCONSTRAINT=X.NombrePK

INNER JOIN

(SELECT DISTINCT CONSTRAINT\_NAME as NombrePK

FROM SYS.ALL\_CONS\_COLUMNS WHERE OWNER = 'ISIS2304A231720')Z

ON S.NombreCONSTRAINT=Z.NombrePK AND X.NombrePK=Z.NombrePK

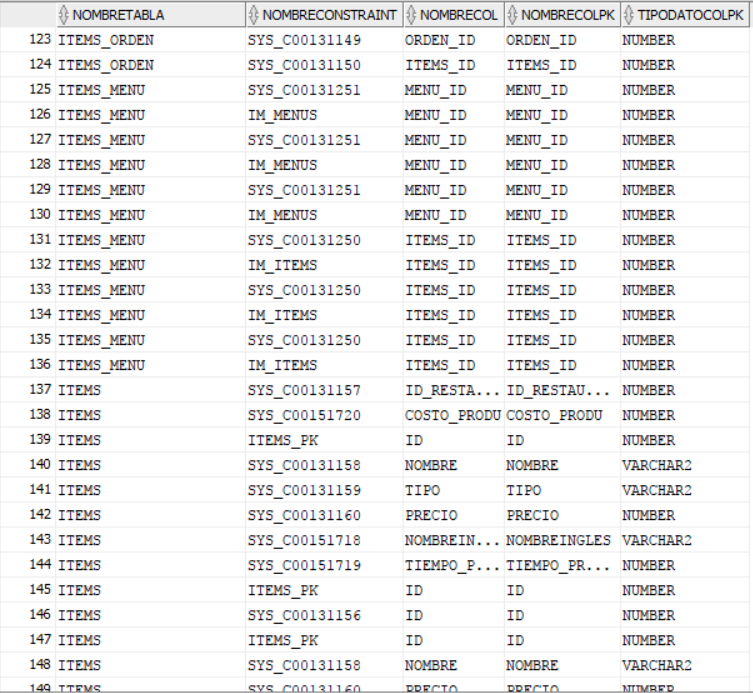
INNER JOIN

(SELECT COLUMN\_NAME AS NombreColPK,DATA\_TYPE as tipoDatoColPK

FROM SYS.ALL\_TAB\_COLUMNS WHERE OWNER = 'ISIS2304A231720')W

ON X.NombreCol=W.NombreColPK )

ORDER BY s.NombreTabla desc;



Cambios relevantes:

* Se crearon nuevas tablas para poder cumplir con los nuevos requerimientos funcionales, en este caso EQUIV\_INGREDIENTe y EQUIV\_PRODUCTO, al igual que un atributo nuevo que representara la mesa en la que se encuentra el cliente.

1. Requerimientos funcionales:

11: Para registrar los ingredientes equivalentes se tiene encuenta que restaurante es el que esta realizando esa decisión ya que para otros restaurantes puede que esa equivalencia no se tenga encuenta, asi manejamos la consistencia de estos datos, las otras propiedades ACID se cumplen por medio de la lógica y los niveles de aislamiento de Oracle.

12: al igual que en el requerimiento 11 se tiene en cuenta el restaurante que realiza la tarea, ya que se tienen que verificar otros factores para la consistencia de datos, por ejemplo a equivalencia de los dos ítems debe ser del mismo restaurante, no se deben poder hacer equivalencias con productos que no ofrecen.

13: en este requerimiento era sencillo crear problemas de consistencia, toca tener en cuenta que restaurante realiza la operación a que producto, cuanta cantidad desea agregar y la cantidad de los productos actual que puede variar dependiendo de las ventas del día anterior.

14:

En este requerimiento se usan las tablas de equivalencia de productos cuando el usuario pide un cambio de equivalencia la sentencia busca el producto y su equivalencia valida que al equivalencia sea correcta y realiza un update a la orden con el id de la equivalencia.

15:

Para esto se usa la tabla mesa y mesa \_orden donde al registrar un pedido a la mesa el usuario que registro asocia un pedido nuevo a una mesa para luego añadirle ítems.

16:

En este el restaurante comprueba si todo los pedidos ya se entregaron para esto el restaurante puede verificar si todo el pedido ya esta completo y se descuenta de su inventario , se añadió un estado a ítem orden que dice si fue entregado o no.

17: al cancelar un pedido se debe eliminar y actualizar información de muchas tablas. Es por eso que al momento de cancelar la orden se mira todos ls ítems que se ordenaron, en la tabla ITEMS se tiene que aumentar la cantidad de cada ítem correspondiente a la orden, luego se debe eliminar de la tabla ítems\_orden y orden la información respecto a la orden específica a cancelar, cualquier usuario restaurante puede cancelar una orden sin importar los ítems en esta.

3: Requerimientos de consulta:

7: este requerimiento no es muy complicado de realizar debido al diseño de la base de datos, entonces así el cliente solicitara solo menus siempre descomponemos la oden en ítems individuales para facilitar su consulta. Solo se realiza un get con el id del usuario (ya que tiene que estar registrado ) el cual nos devuelve una lista de todos los ítems solicitados por ese usuario.

8: en este requerimiento se pedía que se consultara la totalidad de pedidos de la rotonda, para el admin cuando realiza la consulta le mostrara la lista de restaurantes discriminados por ítem y ventas totales del ítem, si consulta como restaurante solo saldrán esta lista con los ítems del restaurantes