

Universidad ORT Uruguay  
Facultad de Ingeniería

Obligatorio 2 de Base de Datos 2

Entregado como requisito para la obtención del puntaje para la  
materia Base de Datos 2

Joaquín Bonifacino – 243193

Rafael Cadenas – 269150

Tomás Caetano– 189213

Tutor: Danilo Fraque Bueri

2022

## Declaración de Autoría:

Nosotros, Joaquín Bonifacino, Rafael Cadenas y Tomás Caetano, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras trabajamos en el proyecto.
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra.
- En la obra, hemos acusado recibo de las ayudas recibidas
- Cuando la obra se basa en trabajo realizado juntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros.
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



---

Firma de autor

Joaquín Bonifacino

Aclaración de firma

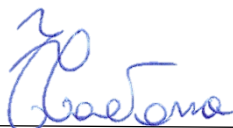


---

Firma de autor

Rafael Cadenas

Aclaración de firma



---

Firma de autor

Tomás Caetano

Aclaración de firma

## Contents

Declaración de Autoría:.....	2
1.....	4
1.1.    Análisis de la solución propuesta .....	4
1.2.    Identificación y clasificación de las restricciones de integridad para el esquema relacional resultante .....	5
1.3.    DDL completo que incluya tablas y restricciones estructurales.....	5
1.4.    Datos de Prueba .....	5
2.....	6
2.1.    Grafo de Asignación .....	6
2.2.    Views .....	6
3.....	7
3.1.    Consulta SQL.....	7
3.2.    Elección de tamaño de bloque.....	7
3.3.    Plan lógico y árbol inicial o canónico.....	8
3.3.1.    Árbol inicial.....	8
3.3.2.    Plan Lógico.....	8
3.4.    Posible plan físico .....	10
3.4.1.    Selecciones .....	10
3.4.2.    Proyecciones .....	10
3.4.3.    JOINS .....	10
3.5.    Posibles índices .....	12
3.5.1.    LANGUAGE .....	12
3.5.2.    TUSER .....	12
3.5.3.    INVOICE .....	12
3.5.4.    PRODUCTBELONGSTO .....	12
3.5.5.    CAR .....	12
3.6.    índices de DBMS.....	12
3.7.    Árbol B+ del índice de la tabla INVOICE .....	13
3.8.    Cálculos de ejecución con índice propuesto.....	13

# 1.

## 1.1. Análisis de la solución propuesta

Se define crear para la primera parte las 10 tablas a continuación:

-**Language**: Esta tabla es creada para poder relacionar un país con un idioma, y al crear un usuario que ese idioma pertenece al país adecuado (por ejemplo, para que no se pueda colocar un idioma que no es oficial de un país).

-**Payment method**: Tabla que almacena los payment method con la posibilidad de que sean tarjetas o crypto.

-**Tuser**: Contiene información de los usuarios con referencias a métodos de pago y lenguaje.

-**Telephone**: Tabla auxiliar para el campo multievaluado de teléfono de usuario.

-**Wardrobe**: Almacena productos de tipo vestimenta, con sus atributos como tamaño y precio. Cada línea sería un SKU.

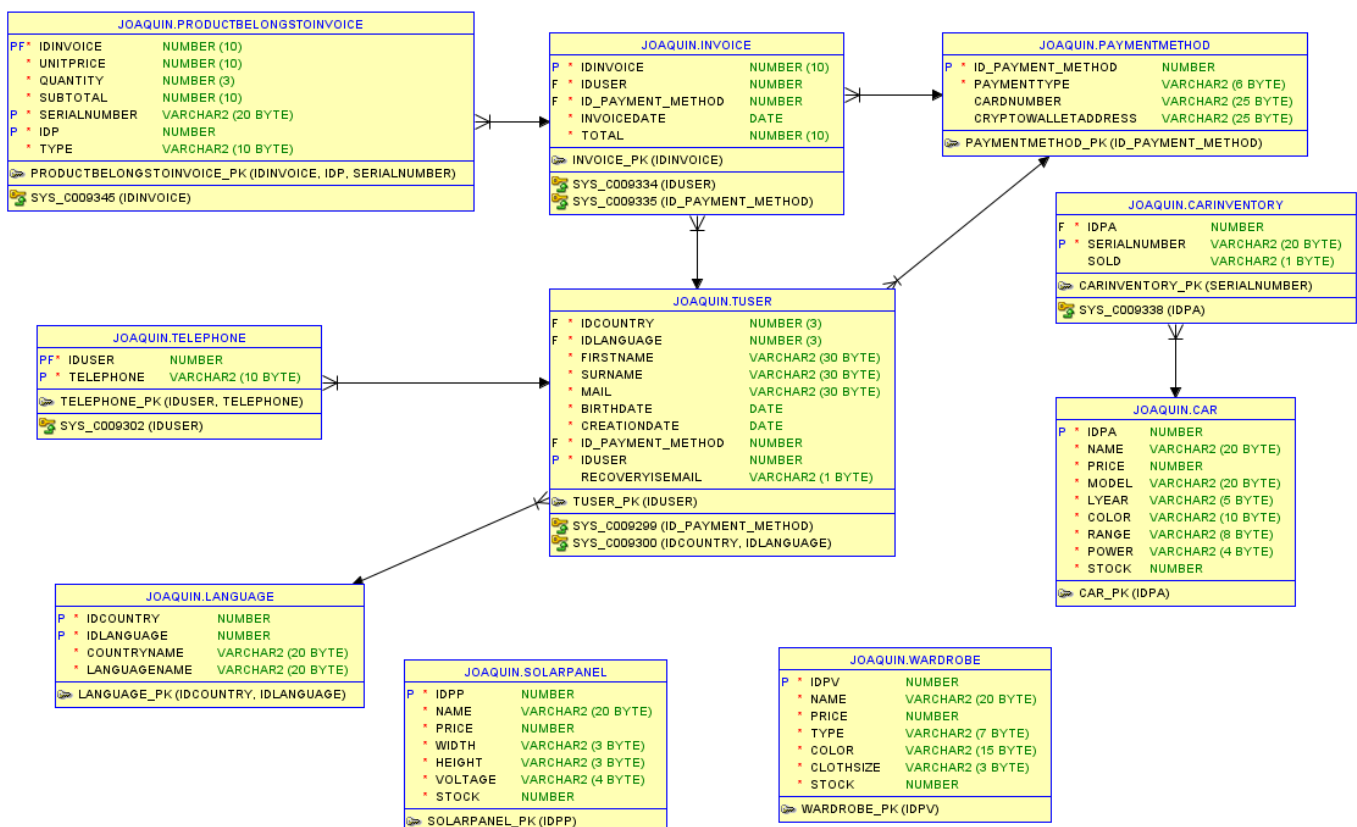
-**Car**: Almacena los distintos modelos o variantes de modelo de auto.

-**Car Inventory**: Almacena las distintas instancias de un modelo, diferenciándose con números seriales y si esa instancia ya fue vendida o no.

-**Solar Panel**: Almacena los diferentes tipos de panel.

-**Invoice**: Tabla que incluye datos generales de cada factura, relacionándose con el usuario y el payment Method.

-**Product belongs to invoice**: Contiene las líneas de factura de producto, es decir, tiene los detalles de cada factura. El manejo de tipo de producto e ID queda delegado a varios triggers encargados de permitir la referencia al objeto correcto, ya que así evitamos campos en nulo innecesarios.



## 1.2. Identificación y clasificación de las restricciones de integridad para el esquema relacional resultante

Entregado como archivo excel llamado "Restricciones V1.0.xlsx"

## 1.3. DDL completo que incluya tablas y restricciones estructurales.

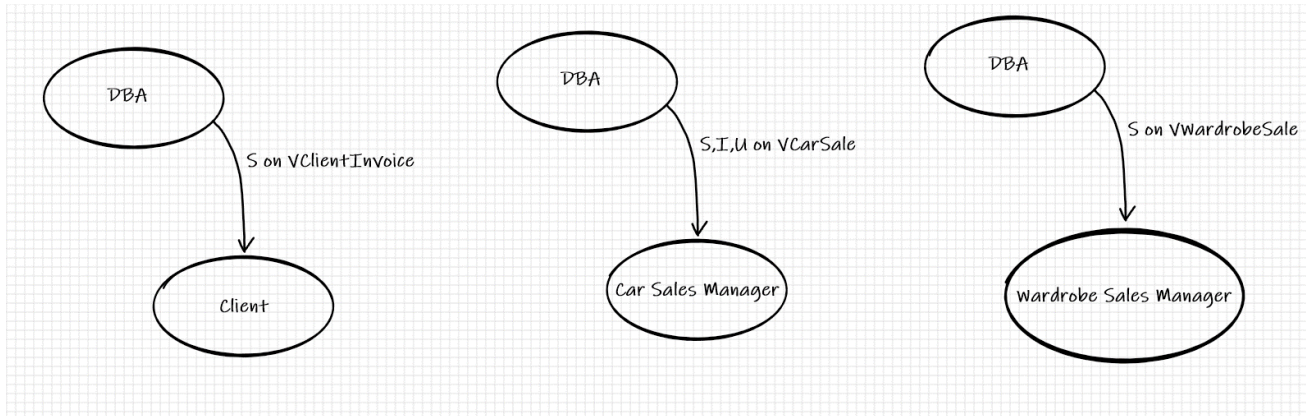
Entregado en un archivo llamado "ObligatorioBD2.sql"

## 1.4. Datos de Prueba

Entregado en un archivo SQL llamado "Inserts.sql"

## 2.

### 2.1. Grafo de Asignación



### 2.2. Views

#### VWARDROBESALE:

```
CREATE VIEW VWARDROBESALE AS (  
SELECT * FROM PRODUCTBELONGSTO WHERE IDP IN(SELECT IDPV FROM WARDROBE)  
);  
GRANT SELECT ON VWARDROBESALE TO CLIENT;
```

#### VCARSALE:

```
CREATE VIEW VCARSALE AS (  
SELECT * FROM PRODUCTBELONGSTO WHERE SERIALNUMBER <> '00000000000000000000';  
GRANT SELECT, UPDATE, INSERT ON VCARSALE TO CARSALESMANAGER;
```

#### VCLIENTINVOICE:

```
CREATE VIEW VCLIENTEINVOICE AS (  
SELECT I.*, P.TYPE FROM INVOICE I, PRODUCTBELONGSTOINVOICE P  
WHERE I.IDUSER=IDUSER()  
AND P.IDINVOICE=I.IDINVOICE);  
GRANT SELECT ON VCLIENTEINVOICE TO CLIENT;
```

Asumimos a "IDUSER()" como la función que retorna el id del usuario conectado

### 3.

#### 3.1. Consulta SQL

```
SELECT U.FIRSTNAME, U.SURNAME, U.MAIL,p.unitprice, p.subtotal FROM
TUSER U, LANGUAGE L, INVOICE I , PRODUCTBELONGSTOINVOICE P, CAR C
WHERE u.idcountry = l.idcountry
AND l.countryname='Mexico'
AND U.iduser = l.iduser
AND P.IDINVOICE = I.IDINVOICE
AND p.idp = c.idpa
AND c.model='S'
AND c.color='BLUE'
AND extract(YEAR FROM i.invoicedate) IN (2021);
```

#### 3.2. Elección de tamaño de bloque

Cálculos de tuplas suponiendo que se vendió la misma cantidad de autos en los últimos 3 años:

El cálculo de cuanto ocupa cada tupla según su tabla, verificar el archivo excel adjunto.

Tabla	Cant Tuplas	Tamaño	Factor de bloqueo	Cant Bloques
TUSER	2000000	115	35	57143
PRODUCTBELONGSTO	2359296	56	73	32320
CAR	500	85	48	11
INVOICE	786432	20	204	3856
LANGUAGE	48	48	85	1

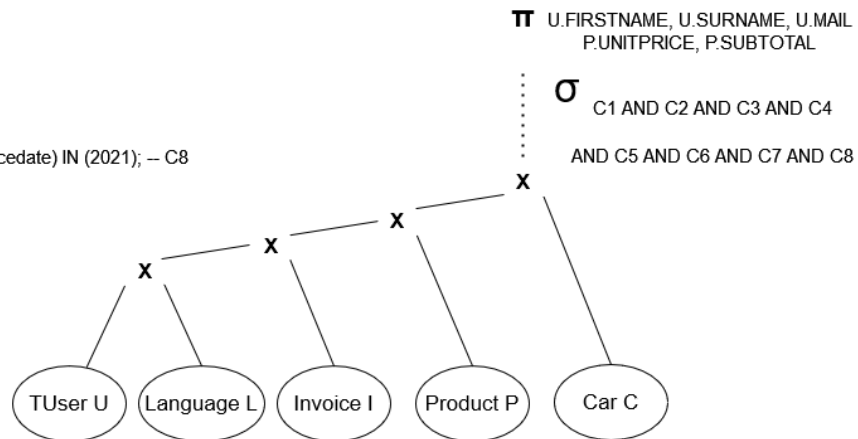
Tamaño bloque	4096
---------------	------

Como el tamaño de la tupla de TUSER es muy grande (como mostrado en los cálculos), se decidió ir por el tamaño de bloque más grande posible siendo el mismo 4096, también, después de investigar descubrimos que la base de datos promedio utiliza tamaños entre 4096 y 8192.

### 3.3. Plan lógico y árbol inicial o canónico

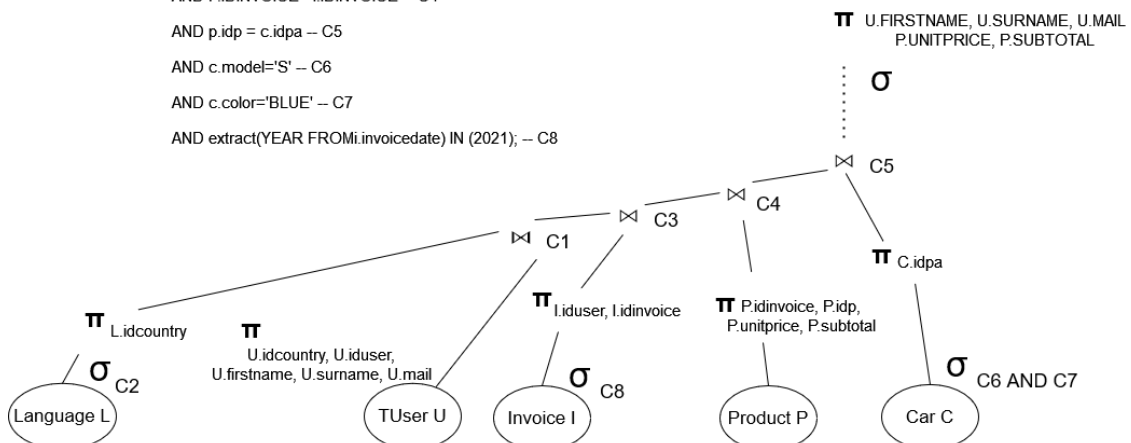
#### 3.3.1. Árbol inicial

```
SELECT U.FIRSTNAME,U.SURNAME, U.MAIL,p.unitprice, p.subtotal FROM
TUSER U, LANGUAGE L,INVOICE I, PRODUCTBELONGSTOINVOICE P,CAR C
WHERE u.idcountry =l.idcountry -- C1
AND l.countryname='Mexico'-- C2
AND U.iduser =l.iduser -- C3
AND P.IDINVOICE =I.IDINVOICE -- C4
AND p.idp = c.idpa -- C5
AND c.model='S' -- C6
AND c.color='BLUE' -- C7
AND extract(YEAR FROM i.invoicedate) IN (2021); -- C8
```



#### 3.3.2. Plan Lógico

```
SELECT U.FIRSTNAME,U.SURNAME, U.MAIL,p.unitprice, p.subtotal FROM
TUSER U, LANGUAGE L,INVOICE I, PRODUCTBELONGSTOINVOICE P,CAR C
WHERE u.idcountry =l.idcountry -- C1
AND l.countryname='Mexico'-- C2
AND U.iduser =l.iduser -- C3
AND P.IDINVOICE =I.IDINVOICE -- C4
AND p.idp = c.idpa -- C5
AND c.model='S' -- C6
AND c.color='BLUE' -- C7
AND extract(YEAR FROM i.invoicedate) IN (2021); -- C8
```



En base a las heurísticas, después de bajar las selecciones y proyectar al menor nivel los campos necesarios para los JOINS y para la proyección final nos dispusimos a hacer el cálculo de aproximadamente cuantas tuplas habrían después de cada selección de cada tabla y si poder colocar (dentro de lo posible) a la izquierda las que tenían la menor cantidad de tuplas.

A través de la información dada en la letra se pudieron determinar los siguientes valores:



Tabla	Tuplas	%	Resultado
TUSER	2000000	100%	2000000
PRODUCTBELONGSTO	2359296	100%	2359296
CAR	500	3%	15
INVOICE	786432	33%	262144
LANGUAGE	48	2%	1

En la tabla language estimamos 48 entradas ya que en la página actual de tesla existen 24 países, dando como promedio 2 idiomas por país como caso extremo.

En la tabla Tuser, la cual no puede ser reducida ya que no cuenta con una selección a bajar, por lo tanto, se ubica en segundo lugar de izquierda a derecha ya que debe de compartir un JOIN con Language que ocupa el lugar más a la izquierda ya que retorna una sola tupla ya que en este caso México tiene un solo lenguaje. Al no tener selección se estima que retorna 2 millones de tuplas.

después se encuentra invoice, estimamos esa cantidad de tuplas ya que se estima que cada invoice tiene al menos 1 auto, por un lado, porque se necesita tener un auto para comprar otro producto dando como resultado 786432 tuplas preselección, la letra especifica que la cantidad de autos vendidos en 2021 es 262144 por lo tanto esa es la cantidad de tuplas aproximada post selección.

En productbelongto, estimamos lo mismo que con invoice ya que al menos 1 línea de cada 3 es de un auto, la base de datos actual cuenta con un historial de venta de los últimos 3 años, por lo que podemos inducir nueve veces más la cantidad de invoices, asumiendo que vendieron lo mismo los últimos 3 años y tuvieron la misma cantidad de líneas por invoice en promedio. Ya que no utilizamos selecciones en esta tabla la salida final es 2359296 tuplas.

Finalmente, en la tabla car sucede que la letra menciona que tienen 500 modelos posibles, solo siendo 10% los modelos del tesla S, y a su vez de ese 10% solo 1/3 son azules. Por ende, podemos estimar unos 17 modelos a partir de esa selección.

### 3.4. Posible plan físico

#### 3.4.1. Selecciones

##### **En el caso de la selección C2:**

1. Full scan: Siempre se puede hacer, en este caso hay un solo bloque por lo que utilizaremos este método  
Costo = 1 Acceso a bloque, Costo materializar Resultado intermedio = 1 Bloque
2. Búsqueda binaria: Al ser una tabla de pocas tuplas es innecesario utilizarla
3. Con índices: No se puede utilizar un índice ya que el único que nos da la letra es secundario por clave.

##### **En el caso de la selección C6 y C7:**

1. Full scan: Siempre se puede hacer, son 11 accesos a bloque  
Costo = 11 accesos a bloque, Costo materializar Resultado intermedio = 15 Tuplas \* 2 bytes (proyección de C.IDPA) / 4096bytes = 1 Bloque.
2. Búsqueda binaria: No está ordenado por el atributo por lo tanto no se puede utilizar
3. Con índices: El índice secundario por clave que nos proporciona la letra no nos sirve ya que los atributos por los cuales se busca seleccionar no son claves primarias.

##### **En el caso de la selección C8:**

1. Full scan: En este caso es la única opción  
Costo = 3856 accesos a bloques, Costo materializar Resultado intermedio = 4 Bytes \* 262144 Tuplas (1/3 de los invoices) / 4096 = 86 Bloques
2. Búsqueda binaria: No se puede ya que no están ordenados por fecha
3. Con índices: No se puede utilizar ya que estamos seleccionando por fecha que no es la clave primaria y el índice actual es secundario por clave.

#### 3.4.2. Proyecciones

Solo hay un algoritmo para las proyecciones que recorre todo, así que se utilizara ese.

#### 3.4.3. JOINS

##### **En el caso del JOIN C1:**

1. Nested Loops: Ya que sale una sola tupla de Language utilizaremos nested loops.  
Costo De acceso = 1(Blang)+ (1 \* 57143 (bloques de TUSER)) = 57144,  
Costo de Materialización = 40000 tuplas (usuarios de mexico) \* 100 Bytes / 4096 = 977 Bloques.
2. Index JOIN: No se puede utilizar ya que existe un índice para la tabla de la derecha (TUSER) pero no es del atributo por el cual sucede el JOIN
3. Merge sort: Ninguno de los 2 están ordenados y no es necesario.
4. Hash JOIN: No tiene sentido ya que tenemos una sola tupla del lado izquierdo

##### **En el caso del JOIN C3:**

1. Nested Loops: No es viable ya que ambas tablas tienen un tamaño masivo
2. Index JOIN: No se puede ya que el índice de la tabla derecha no es del atributo por el cual estamos uniendo.

3. Merge sort: Ninguno de los 2 están ordenados y su costo de hacerlo es muy alto
4. Hash JOIN: Nos parece el más adecuado ya que por la masividad de los datos de ambos lados, y que no podemos utilizar índices ni esta ordenado, estimamos que es el de menor costo.  
 Costo de acceso =  $3 * 977 + 3 * 86 = 3189$  Bloques  
 Costo de Materialización =  $102 \text{ Bytes} * 40000/3$  (Promedio de usuarios que tienen 1 compra en el año 2021) /  $4096 = 333$  Bloques

#### **En el caso del JOIN C4:**

1. Nested Loops: No es viable ya que ambas tablas tienen un tamaño masivo
2. Index JOIN: No se puede utilizar ya que el índice por clave es de una clave compuesta y el atributo por el cual hacemos join es una superclave.
3. Merge sort: Ninguno de los 2 están ordenados y su costo de hacerlo es muy alto
4. Hash JOIN: Es posible y utilizaremos este.  
 Costo de acceder =  $3 * 333 + 3 * 576 = 2727$  Bloques  
 Costo de Materialización =  $122 \text{ Bytes} * (40000/3 \text{ tuplas}) / 4096 = 398$  Bloques

Nota: Se sabe por C3 que la cantidad de invoices son aproximadamente 40000 y en promedio cada uno tiene 1 línea de factura con el vehículo, por lo que se estiman la misma cantidad de tuplas solo que en este caso al agregar los datos del producto se suman 20 bytes.

#### **En el caso del JOIN C5:**

1. Nested Loops: Es viable si se invierten los lugares (quedando 1 bloque en el lado izquierdo).  
 Costo de acceso =  $1 + (1 * 398) = 399$  Bloques.  
 Costo de Materialización:  $124 * 175 \text{ Tuplas (Ver nota)} / 4096 = 6$  Bloques
2. Index JOIN: Se puede, pero después de invertir ambos lados como en nested loops, nos queda 1 solo bloque del lado izquierdo, por lo tanto, es innecesario utilizar un índice
3. Merge sort: Ninguno de los 2 están ordenados y el costo por izquierda es muy alto.
4. Hash JOIN: Es posible, pero estimamos que sería menos eficiente que el Nested Loops (si se invierte).

**Proyección Final**: luego de C5 se proyectan los campos deseados según el árbol y nos dan 108 Bytes y 175 Tuplas, por lo que la cantidad de bloques resultantes es 5 Bloques.

### 3.5. Posibles índices

#### 3.5.1. LANGUAGE

Se podría realizar un índice por Nombre de país, aunque en este caso no aportaría nada ya que la tabla se trata de un solo bloque.

#### 3.5.2. TUSER

Se podría utilizar una Índice secundario denso, el mismo sería por el campo IDCOUNTRY, teniendo la misma cantidad de entradas que TUSER. Esto permite que el índice ocupe menos bloques que la tabla a la que hace referencia y no tenga que ir a buscar todos los bloques sino solo los que contienen tuplas con IDCOUNTRY correspondiente.

#### 3.5.3. INVOICE

Para esta tabla se puede utilizar un índice por agrupamiento por el atributo fecha, ya que el sistema no debería crear una factura con una fecha anterior, por lo que la tabla quedaría ordenada indirectamente (se puede incluso forzar con un trigger que verifique con la fecha de la BD)

Se podría haber utilizado un índice por IDUSER para poder utilizarlo en el JOIN de C3

#### 3.5.4. PRODUCTBELONGSTO

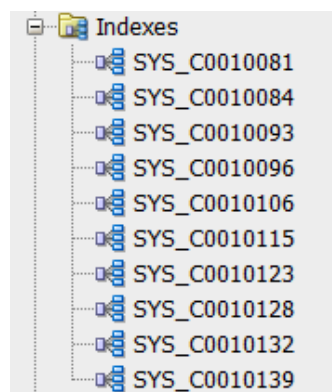
Para esta tabla se puede crear un índice secundario por no clave (IDINVOICE), Este índice puede ser sin indirección ya que se sabe que en promedio una factura tiene 3 productos.

#### 3.5.5. CAR

Se puede utilizar un índice secundario por no clave con el atributo “modelo”, ya que los modelos tienden a discontinuarse, pero sus colores no, por lo que los punteros no deberían aumentar mientras que al hacerlo por colores aumentara a medida que existan más modelos (probablemente con color azul).

### 3.6. índices de DBMS

Descubrimos que el motor DBMS de Oracle autogenera índices por clave primaria cuando se crea una tabla automáticamente.



### 3.7. Árbol B+ del índice de la tabla INVOICE

Para el índice existen por año 365 entradas (1 por día en el periodo de 3 años estipulado), dando un total de 1095 punteros totales en el índice.

Sabiendo que el tamaño de bloque es 4096Bytes y 6 bytes están reservados para un puntero específico (en los nodos), nos quedarían 4090 Bytes para almacenar los punteros. Podemos calcular entonces  $4090/6 = 681$  punteros. De estos punteros se puede utilizar inicialmente el 70% (477). Quedando entonces la siguiente tabla armada.

Nivel	#Nodos	# Entradas	#Hijos
1	1	477	478
2 = Nivel Hoja	478	478 nodos con 477 Entradas cada uno. 228006 entradas posibles.	

Sabemos que como máximo el índice puede ocupar 479 Bloques. Para el acceso sabemos que el costo sería  $1289 = 3 + 1286$  (1/3 de la cantidad de bloques de Invoice)

### 3.8. Cálculos de ejecución con índice propuesto.

El costo de C8 sería menor con el nuevo índice por agrupamiento por el atributo fecha:

Costo = 1289 accesos a bloques

Costo materializar Resultado intermedio = 4 Bytes \* 262144 Tuplas (1/3 de los invoices) / 4096 = 86 Bloques

Como se puede ver el costo de la selección baja un 66% aproximadamente mientras que el costo de materializar el resultado es el mismo ya que las tuplas que se deben devolver son las mismas. Esto indica que no modifica los resultados del resto del árbol.