CURSORES

UNAM. FACULTAD DE INGENIERÍA.

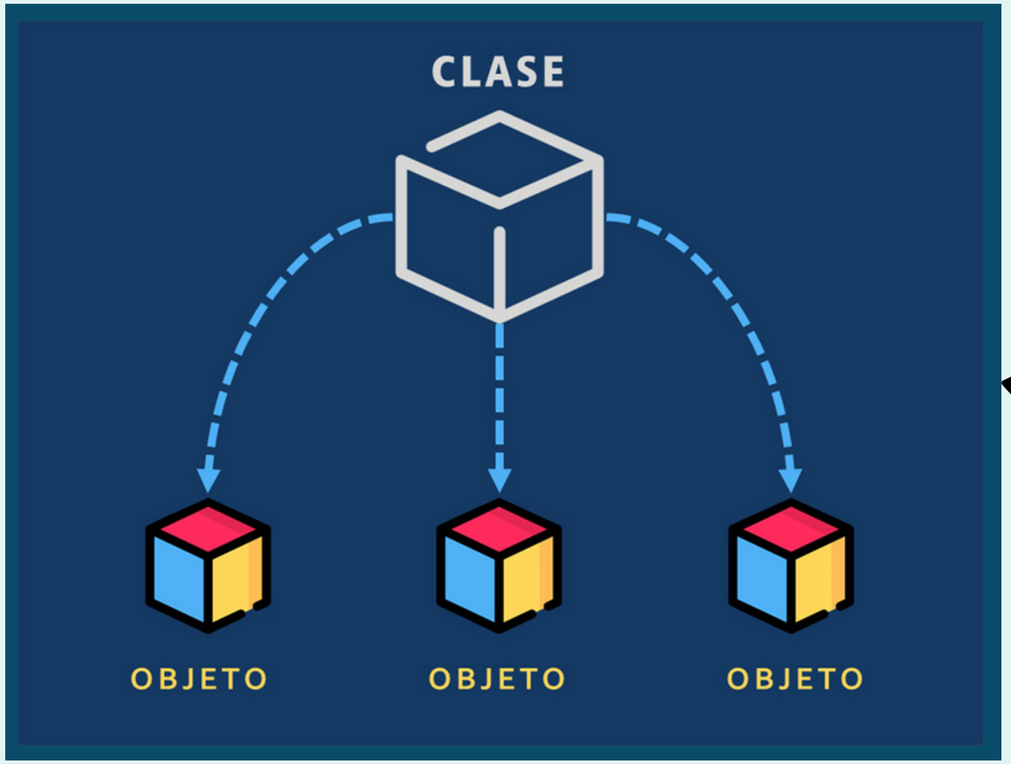
BASE DE DATOS.

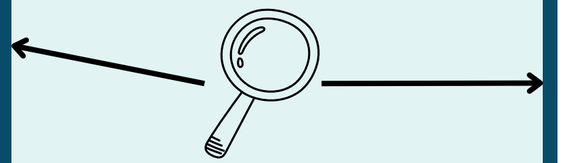
PÉREZ JIMÉNEZ ERANDI

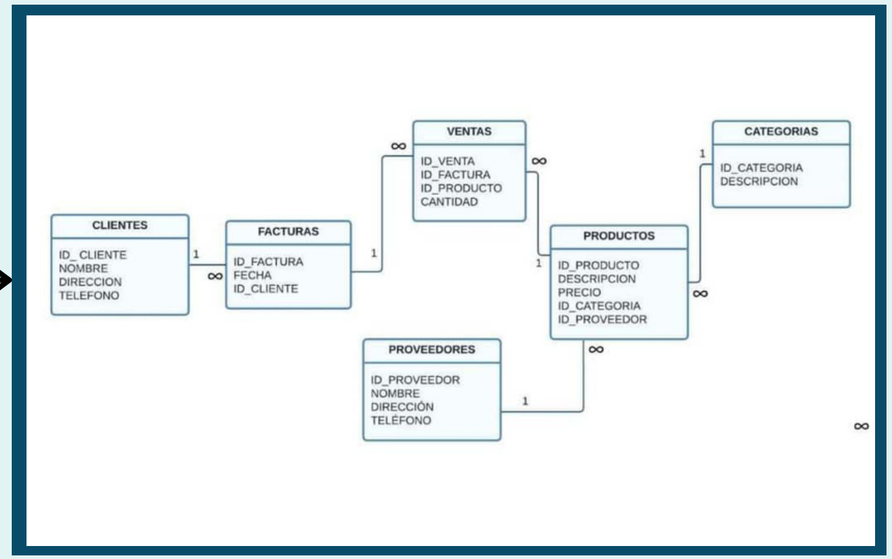
CRUZ VARGAS EMILIO

PLANTEAMIENTO DEL PROBLEMA

IMPEDANCE MISMATCH







Modelos lógicos basados en objetos

Modelos lógicos basados en registros

¿QUÉ ES UN CURSOR?

En base de datos un Cursor es un mecanismo el cual nos permite procesar fila por fila el resultado de una consulta.

columnas

filas

Nombre (text)	Apellido (text)	fechaAlta (date)	Nivel (numeric)	Salario (currency)
Alice	Mann	4/4/2009	4	75000
James	**Black**	**3/1/2010**	**4**	**75000**
Calista	Guerra	10/1/2006	6	80000
Fay	Fitzgerald	7/21/2002	7	100000
John	Bowen	11/11/2011	3	45000

“El cursor te permite trabajar con los datos uno por uno”

PASOS PARA PODER HACER USO DE UN CURSOR

- Crear un cursor a partir de una sentencia SQL
- Apertura del cursor
- Acceso a datos
- Cierre del cursor

SINTAXIS

```
DECLARE <nombre_cursor> CURSOR FOR <consulta>
```

```
OPEN<nombre_cursor>
```

```
FETCH <OPERATOR> FROM CURSOR  [INTO <variables>]
```

```
CLOSE <nombre_cursor>
```

```
DEALOCATE <nombre_cursor>
```

DECLARACION DE VARIABLES DE CURSOR

En PL/pgSQL, usamos variables de cursor para trabajar con conjuntos de filas en una base de datos. Hay dos formas de declarar estas variables:

Declaración directa

Declaración de una variable de cursor sin utilizar la sintaxis completa de la declaración de cursor

```
-- Declaración directa de un cursor
DECLARE mi_cursor CURSOR FOR
    SELECT columna1, columna2
    FROM tabla
    WHERE condicion;
```

```
declare<my_cursor>
refcursor
```

DECLARACION DE VARIABLES DE CURSOR

En PL/pgSQL, usamos variables de cursor para trabajar con conjuntos de filas en una base de datos. Hay dos formas de declarar estas variables:

Declaración con la sintaxis de cursor

Declaración de un cursor junto con la especificación de detalles adicionales, como si permite o no desplazarse hacia atrás (**NO SCROLL** o **SCROLL**) y si hay argumentos.

```
cursor_name [ [no] scroll ] cursor [( name datatype, name data type, ...)] for query;
```

CURSORES DE RECORRIDO

Non-scrollable (sin recorrido) ó Forward-only (unidireccional)

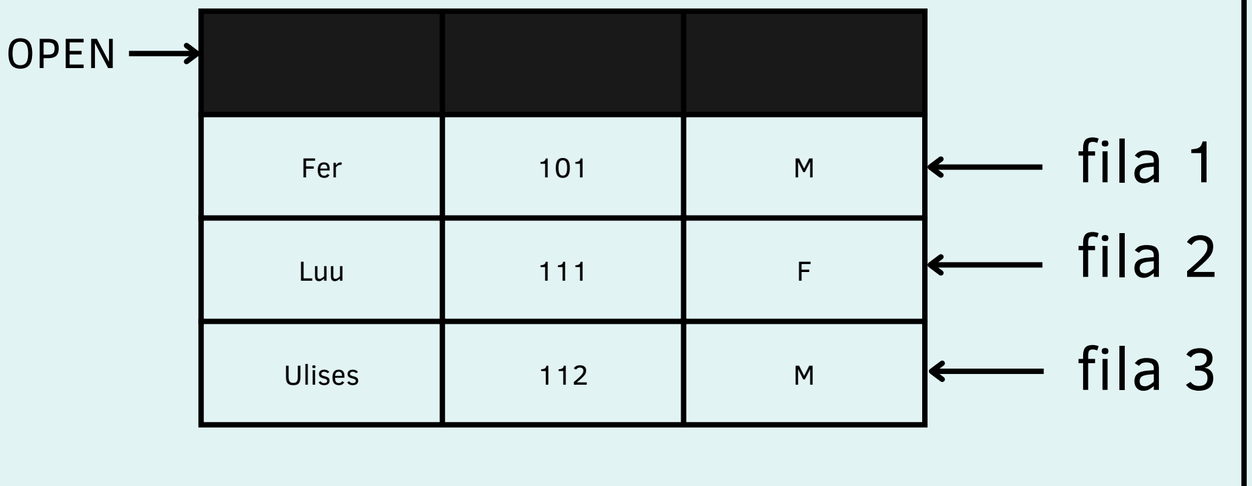
- Cada **fila** puede ser leída como mucho una vez.
- Si hace un UPDATE o DELETE el cursor permanecerá en la fila donde se quedó

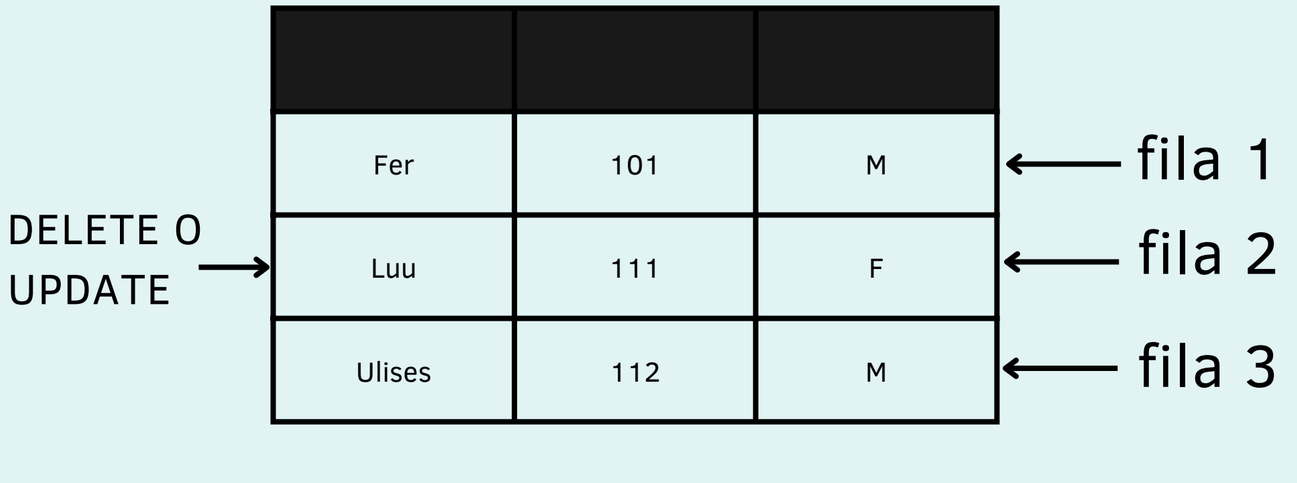
```
DECLARE
   nombre_cursor CURSOR FOR SELECT columna1, columna2 FROM nombre_tabla;
   registro nombre_tabla%ROWTYPE;
BEGIN
   OPEN nombre_cursor;

   FETCH nombre_cursor INTO registro;
   WHILE FOUND LOOP
      -- Procesar la fila actual almacenada en 'registro'

      -- Obtener la siguiente fila
      FETCH nombre_cursor INTO registro;
   END WHILE;

   CLOSE nombre_cursor;
END;
```





Scrollable cursor (desplazable)

- Se puede colocar en una fila o en un conjunto de filas
- Se puede mover directamente a las filas sin tener que hacer FETCH para todas las filas

SENSITIVE

Si necesitas ver los datos más recientes.

INSENSITIVE

Si no necesita ver las actualizaciones realizar por otros cursores.

SENSITIVE STATIC

No necesita ver los resultados de las operaciones de inserción.

SENSITIVE DINAMIC

Necesita ver todas las actualizaciones e inserciones.

Sintaxis

```
DECLARE
   nombre_cursor SCROLL CURSOR FOR SELECT columna1, columna2 FROM nombre_tabla;
   registro nombre_tabla%ROWTYPE;
BEGIN
   OPEN nombre_cursor;

   MOVE ABSOLUTE 3 FROM nombre_cursor;

   -- Obtener la fila actual
   FETCH nombre_cursor INTO registro;

   -- Realizar operaciones con la fila actual

   CLOSE nombre_cursor;
END;
```

La posición de un cursor de recorrido puede especificarse de forma relativa a la posición actual del cursor o de forma absoluta a partir del principio del set de resultados.

```
FETCH [ NEXT | PRIOR | FIRST | LAST ] FROM cursor_name
```

```
FETCH ABSOLUTE n FROM cursor_name
```

```
FETCH RELATIVE n FROM cursor_name
```

UTILIZACION DE CURSORES

Después de abrir un cursor, podemos manipularlo utilizando las sentencias FETCH, MOVE, UPDATE o DELETE.
Obtencion de la siguente fila:

```
fetch [ direction { from | in } ] cursor_variable into target_variable;
```

La sentencia FETCH obtiene la siguiente fila del cursor y le asigna una variable_objetivo, que puede ser un registro, una variable de fila o una lista de variables separadas por comas. Si no se encuentra ninguna fila más, la variable_objetivo se asigna a NULL(s).

VARIABLES DEL CURSOR

<OPERATOR> :
NEXT --- Se utiliza para avanzar al siguiente elemento en el conjunto de resultados
PRIOR--- Mueve el cursor hacia atrás, al elemento anterior en el conjunto de resultados
FIRST --- Coloca el cursor en el primer elemento del conjunto de resultados
LAST ---Coloca el cursor en el último elemento del conjunto de resultados.
ABSOLUTE numero --- Posiciona el cursor en una posición absoluta específica en el conjunto de resultados, según el número proporcionado.
RELATIVE numero --- Mueve el cursor a una posición relativa específica en el conjunto de resultados, según el número proporcionado.

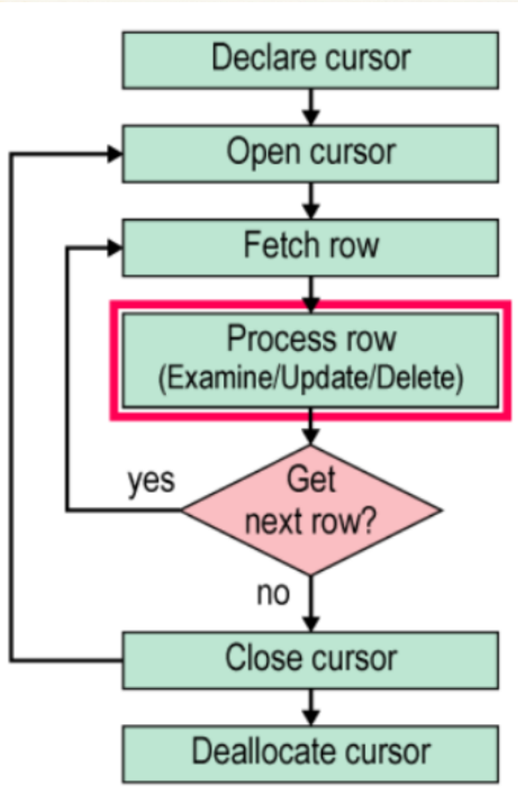
NOTA:
FORWARK
BACKWARK
son sólo para cursores declarados con la opción SCROLL.

BORRAR O ACTUALIZAR FILA

Una vez posicionado un cursor, podemos borrar o actualizar la fila identificada por el cursor utilizando las sentencias DELETE WHERE CURRENT OF o UPDATE WHERE CURRENT OF como se indica a continuación:

```
update table_name
set column = value, ...
where current of cursor_variable;
delete from table_name
where current of cursor_variable;
```

DECLARACIÓN DE UN CURSOR




```
-- Declarar un cursor
DECLARE my_cursor CURSOR FOR
SELECT * FROM my_table;

-- Abrir el cursor
OPEN my_cursor;

-- Recorrer los resultados
FETCH NEXT FROM my_cursor INTO my_record;

-- Procesar la fila en la variable
my_record

-- Cerrar el cursor
CLOSE my_cursor;
```

¿NETWORK ROUND TRIP?

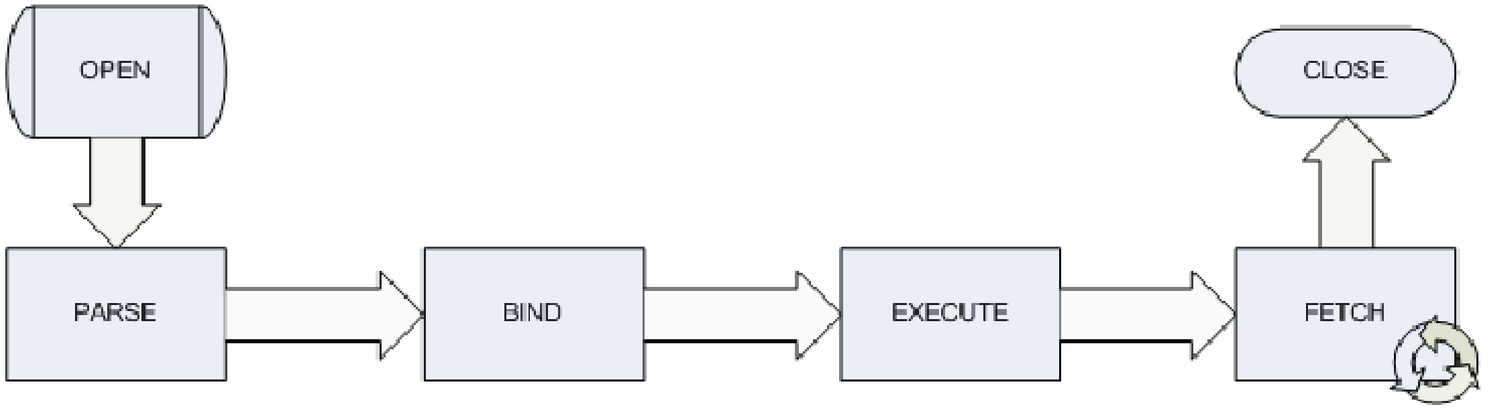
→ ¿Cuántos puedo abrir?

→ ¿Cuántas formas diferentes existen para procesar una fila?

→ Y si... NO CIERRO UN CURSOR

- Locks (Shared, Exclusive, Read and Write Lock)
- Recursos no liberador

Atributo	Tipo	Descripción
%ISOPEN	Booleano	TRUE si el cursor está abierto.
%NOTFOUND	Booleano	TRUE si la recuperación más reciente no devuelve ninguna fila.
%FOUND	Booleano	TRUE si la recuperación más reciente devuelve una fila.
%ROWCOUNT	Númerico	Proporciona el número total de filas devueltas hasta ese momento.



Fases para procesar una instrucción SQL

CASO DE USO

Creación de la tabla

Query | Query History

```sql
CREATE TABLE proveedor (
    id_proveedor SERIAL PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    direccion VARCHAR(255),
    telefono VARCHAR(15),
    email VARCHAR(50)
);

INSERT INTO proveedor (nombre, direccion, telefono, email) VALUES
('Proveedor A', 'Calle 123, Ciudad A', '123-456-7890', 'proveedorA@example.com'),
('Proveedor B', 'Avenida 456, Ciudad B', '987-654-3210', 'proveedorB@example.com'),
('Proveedor C', 'Calle Principal, Ciudad C', '555-123-4567', 'proveedorC@example.com'),
('Proveedor D', 'Avenida Secundaria, Ciudad D', '111-222-3333', 'proveedorD@example.com'),
('Proveedor E', 'Calle 789, Ciudad E', '444-555-6666', 'proveedorE@example.com'),
('Proveedor F', 'Avenida 012, Ciudad F', '777-888-9999', 'proveedorF@example.com'),
('Proveedor G', 'Calle Secundaria, Ciudad G', '333-222-1111', 'proveedorG@example.com'),
('Proveedor H', 'Avenida 456, Ciudad H', '999-888-7777', 'proveedorH@example.com'),
('Proveedor I', 'Calle Principal, Ciudad I', '666-555-4444', 'proveedorI@example.com'),
('Proveedor J', 'Avenida 789, Ciudad J', '222-111-0000', 'proveedorJ@example.com');
```

Tabla

	id_proveedor [PK] integer	nombre character varying (255)	direccion character varying (255)	telefono character varying (15)	email character varying (50)
1	1	Proveedor A	Calle 123, Ciudad A	123-456-7890	proveedorA@example.com
2	2	Proveedor B	Avenida 456, Ciudad B	987-654-3210	proveedorB@example.com
3	3	Proveedor C	Calle Principal, Ciudad C	555-123-4567	proveedorC@example.com
4	4	Proveedor D	Avenida Secundaria, Ciudad D	111-222-3333	proveedorD@example.com
5	5	Proveedor E	Calle 789, Ciudad E	444-555-6666	proveedorE@example.com
6	6	Proveedor F	Avenida 012, Ciudad F	777-888-9999	proveedorF@example.com
7	7	Proveedor G	Calle Secundaria, Ciudad G	333-222-1111	proveedorG@example.com
8	8	Proveedor H	Avenida 456, Ciudad H	999-888-7777	proveedorH@example.com
9	9	Proveedor I	Calle Principal, Ciudad I	666-555-4444	proveedorI@example.com
10	10	Proveedor J	Avenida 789, Ciudad J	222-111-0000	proveedorJ@example.com

```sql
CREATE OR REPLACE FUNCTION obtener_datos_proveedor(n INTEGER)
RETURNS SETOF proveedor AS $$
DECLARE
    registro proveedor%ROWTYPE;
    cur_precios CURSOR FOR SELECT * FROM proveedor;
BEGIN
    OPEN cur_precios;

    FOR i IN 1..n
    LOOP
        FETCH cur_precios INTO registro;

        EXIT WHEN NOT FOUND;

        RETURN NEXT registro;
    END LOOP;

    CLOSE cur_precios;

    RETURN;
END $$ LANGUAGE plpgsql;

```

Creación de función y de cursor

```sql
SELECT * FROM obtener_datos_proveedor(5);

```

Llamada a la función

Resultado

	id_proveedor integer	nombre character varying (255)	direccion character varying (255)	telefono character varying (15)	email character varying (50)
1	1	Proveedor A	Calle 123, Ciudad A	123-456-7890	proveedorA@example.com
2	2	Proveedor B	Avenida 456, Ciudad B	987-654-3210	proveedorB@example.com
3	3	Proveedor C	Calle Principal, Ciudad C	555-123-4567	proveedorC@example.com
4	4	Proveedor D	Avenida Secundaria, Ciudad D	111-222-3333	proveedorD@example.com
5	5	Proveedor E	Calle 789, Ciudad E	444-555-6666	proveedorE@example.com

```
cruze@Emi MINGW64 ~
$ C:/Users/cruze/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/cruze/Downloads/import requests.py"
Cuántos registros quieres: 2
Respuesta: [{"id_proveedor":1,"0":1,"nombre":"Proveedor A","1":"Proveedor A","direccion":"Calle 123, Ciudad A","2":"Calle 123, Ciudad A","telefono":"123-456-7890","3":"123-456-7890","email":"proveedorA@example.com","4":"proveedorA@example.com"},{"id_proveedor":2,"0":2,"nombre":"Proveedor B","1":"Proveedor B","direccion":"Avenida 456, Ciudad B","2":"Avenida 456, Ciudad B","telefono":"987-654-3210","3":"987-654-3210","email":"proveedorB@example.com","4":"proveedorB@example.com"}]
Valores en formato JSON:
[
  {
    "id_proveedor": 1,
    "0": 1,
    "nombre": "Proveedor A",
    "1": "Proveedor A",
    "direccion": "Calle 123, Ciudad A",
    "2": "Calle 123, Ciudad A",
    "telefono": "123-456-7890",
    "3": "123-456-7890",
    "email": "proveedorA@example.com",
    "4": "proveedorA@example.com"
  },
  {
    "id_proveedor": 2,
    "0": 2,
    "nombre": "Proveedor B",
    "1": "Proveedor B",
    "direccion": "Avenida 456, Ciudad B",
    "2": "Avenida 456, Ciudad B",
    "telefono": "987-654-3210",
    "3": "987-654-3210",
    "email": "proveedorB@example.com",
    "4": "proveedorB@example.com"
  }
]
```

Thank you

REFERENCIAS

- GeeksforGeeks. (2023, 5 abril). Impedance mismatch in DBMS. https://www.geeksforgeeks.org/impedance-mismatch-in-dbms/