# Bases de datos columnares

**Bases de Datos No Relacionales**

Instituto de Computación, FING, UdelaR – 2016

CC-BY Lorena Etcheverry lorenae@fing.edu.uy

Las bases de datos columnares son bases de datos **relacionales!!**

No confundir con las *column-family stores* (variantes de *key-value stores*)

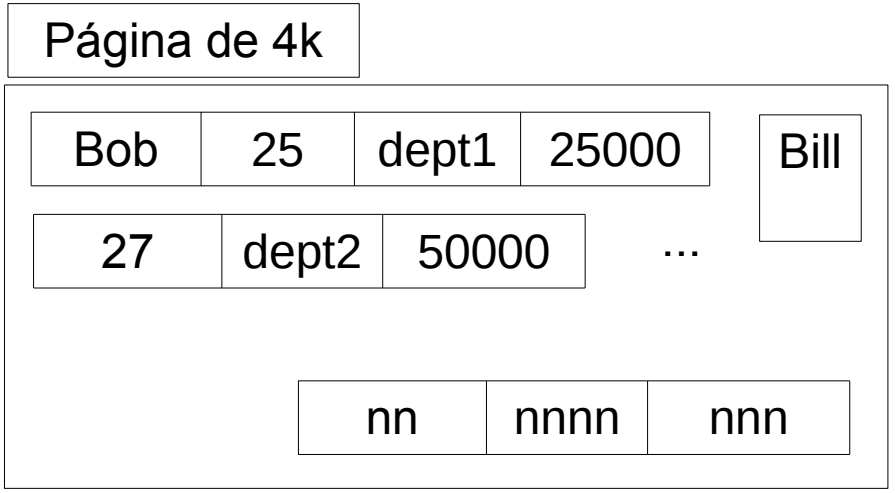
# Contexto

# Almacenamiento en RDBMS orientados a filas

Tabla a nivel lógico: conjunto de tuplas

| persona | edad | departamento | salario |
|---|---|---|---|
| Bob | 25 | dept1 | 25000 |
| Bill | 27 | dept2 | 50000 |
| Jill | 24 | dept3 | 40000 |

Tabla a nivel físico: Registros almacenados consecutivamente en páginas/bloques de disco



# Acceso a datos en RDBMS orientados a filas

- Los datos están ordenados físicamente por algunos atributos
  - Índices primarios (por clave primaria)
- Existen estructuras auxiliares para acceder en forma más eficiente a otros atributos
  - Índices secundarios (B-trees, hash index, etc)
- También puedo hacer acceso secuencial (*scan*)

# OLTP vs OLAP

## OnLine Transaction Processing

Clásico sistema operacional, basado en transacciones sobre registros.

Típicamente *write-intensive*

## OnLine Analytical Processing

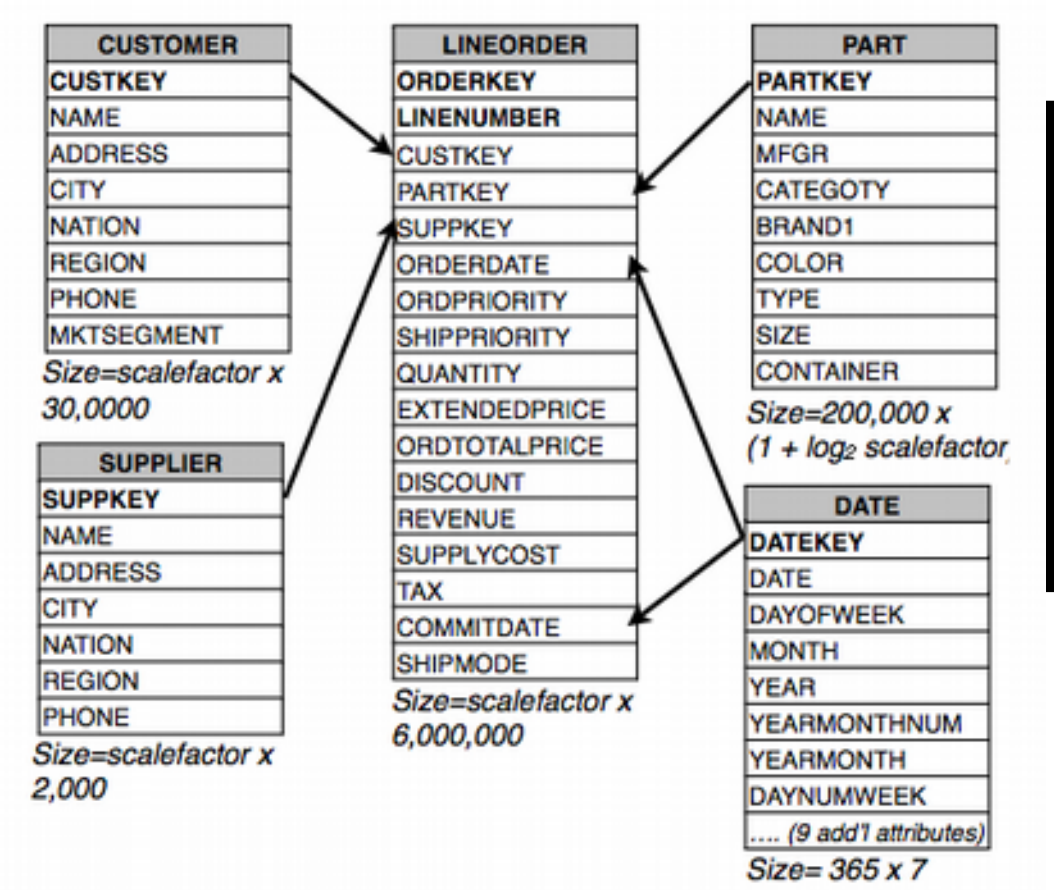
Consultas analíticas sobre los datos, típicamente involucran **muchos registros** en **muchas tablas**

Típicamente *read-intensive*

# OLAP y data warehouses

- Se proponen diseños lógicos sobre bases de datos relacionales para hacer OLAP en forma más eficiente.
  - Modelo estrella
  - Modelo *snowflake*
- Suelen desnormalizar tablas!!

# Un ejemplo: Star Schema Benchmark




```
select sum(lo_extendedprice*lo_discount)
from lineorder, dates
where
    lo_orderdate = d_datekey
    and d_year = 1993
    and 1 < lo_discount < 3
    and lo_quantity < 25;
```

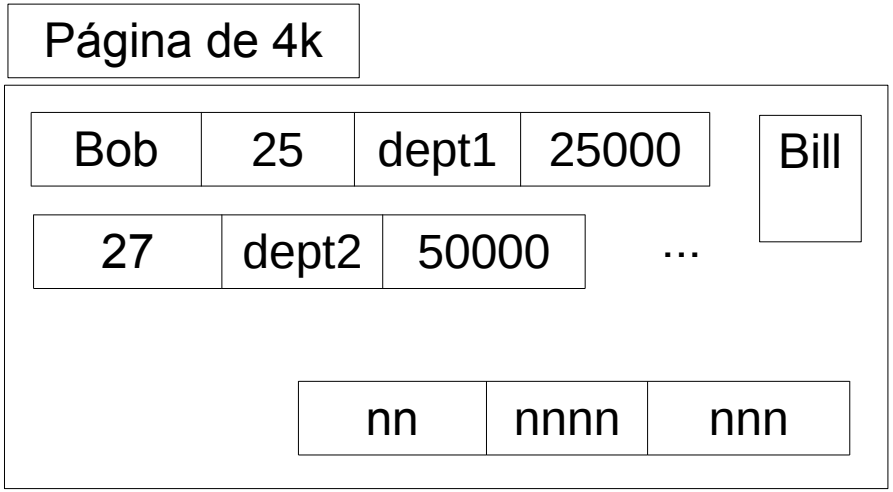
**Las consultas OLAP suelen involucrar "pocas" columnas y acceso via *scans***

Star Schema Benchmark, O´Neil et al. 2009

# Volviendo al ejemplo de juguete

| persona | edad | departamento | salario |
|---|---|---|---|
| Bob | 25 | dept1 | 25000 |
| Bill | 27 | dept2 | 50000 |
| Jill | 24 | dept3 | 40000 |

```
select sum(salario)
from personas
where edad >= 25
```




Para resolver esta consulta es necesario leer todos las páginas de disco en que está almacenada la tabla, y en cada página leer todos los registros

# ¿Cómo reducir las operaciones de acceso a disco y el tiempo de los *scans*?

**¡Almacenando los datos por columnas!**

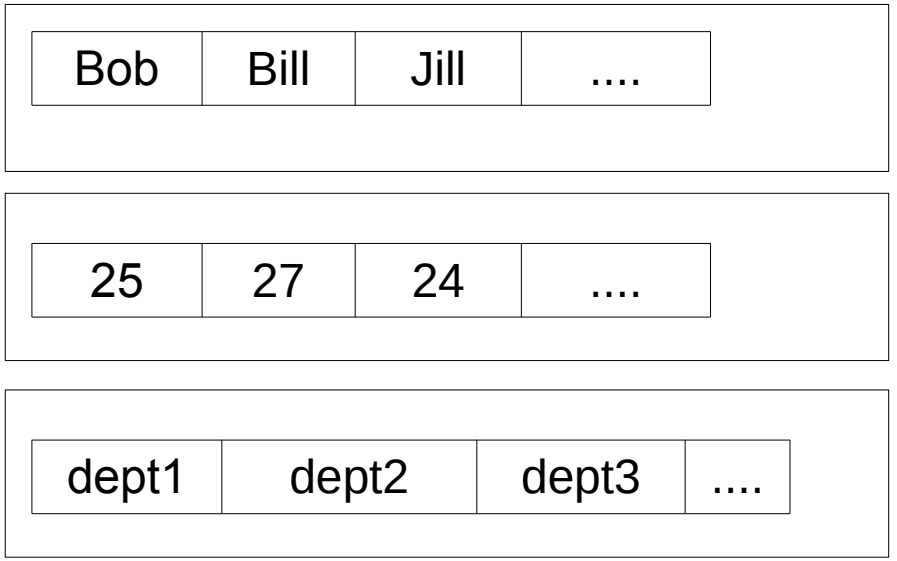
# Almacenamiento en RDBMS orientados a columnas

Tabla a nivel lógico: conjunto de tuplas

| persona | edad | departame | |
|---|---|---|---|
| Bob | 25 | dept1 | |
| Bill | 27 | dept2 | 50000 |
| Jill | 24 | dept3 | 40000 |

```
select sum(salario)
from personas
where edad >= 25
```

Tabla a nivel físico: Columnas almacenadas consecutivamente en páginas/bloques de disco diferentes




Ahora sólo leo las páginas donde se almacenan las columnas que necesito

# Filas vs Columnas

+ es fácil agregar o modificar una tupla

- puedo leer datos innecesarios

+ leo sólo los datos que preciso

- modificar una tupla implica tocar muchas páginas

# Compresión de columnas

- Es probable que las columnas tengan repetidos
- Puedo buscar representaciones compactas de las columnas
- Ejemplo: si las columnas están ordenadas por ese valor puedo usar run-length encoding (RLE)
  - Ejemplo: AAAAAABBBB ->Ax6,Bx4

# Algunos sistemas orientados a columnas

Sybase IQ™

monetdb

INFOBRIGHT

VERTICA
An HP Company

# C-Store : A Column-oriented DBMS

## Stonebraker et al. VLDB, 2005

# ¿Qué es C-Store?

Un prototipo desarrollado en MIT.

Almacenamiento columnar.

Compresión de datos.

Procesamiento en múltiples nodos.

Mecanismos para **escrituras** eficientes

Optimizar consultas *read-only* en presencia de *updates*

# C-Store: modelo de datos

- Soporta el modelo relacional y consultas en SQL
- Introduce la noción de **proyección**
  - no confundir con el operador de álgebra!!
- Una proyección es como una vista de la tabla lógica, pero en realidad es lo que se almacena!!

**EMPLEADOS**

| nombre | edad | departamento | salario |
|---|---|---|---|
| Bob | 25 | dept1 | 25000 |
| Bill | 27 | dept2 | 50000 |
| Jill | 24 | dept2 | 40000 |

esta tabla no está materializada, lo que se almacena son sus particiones

**EMPLEADOS1(nombre, edad | edad)**

| nombre | edad |
|---|---|
| Jill | 24 |
| Bob | 25 |
| Bill | 27 |

**Sort key**

| key |
|---|
| 3 |
| 1 |
| 2 |

**EMPLEADOS2(nombre, salario | salario)**

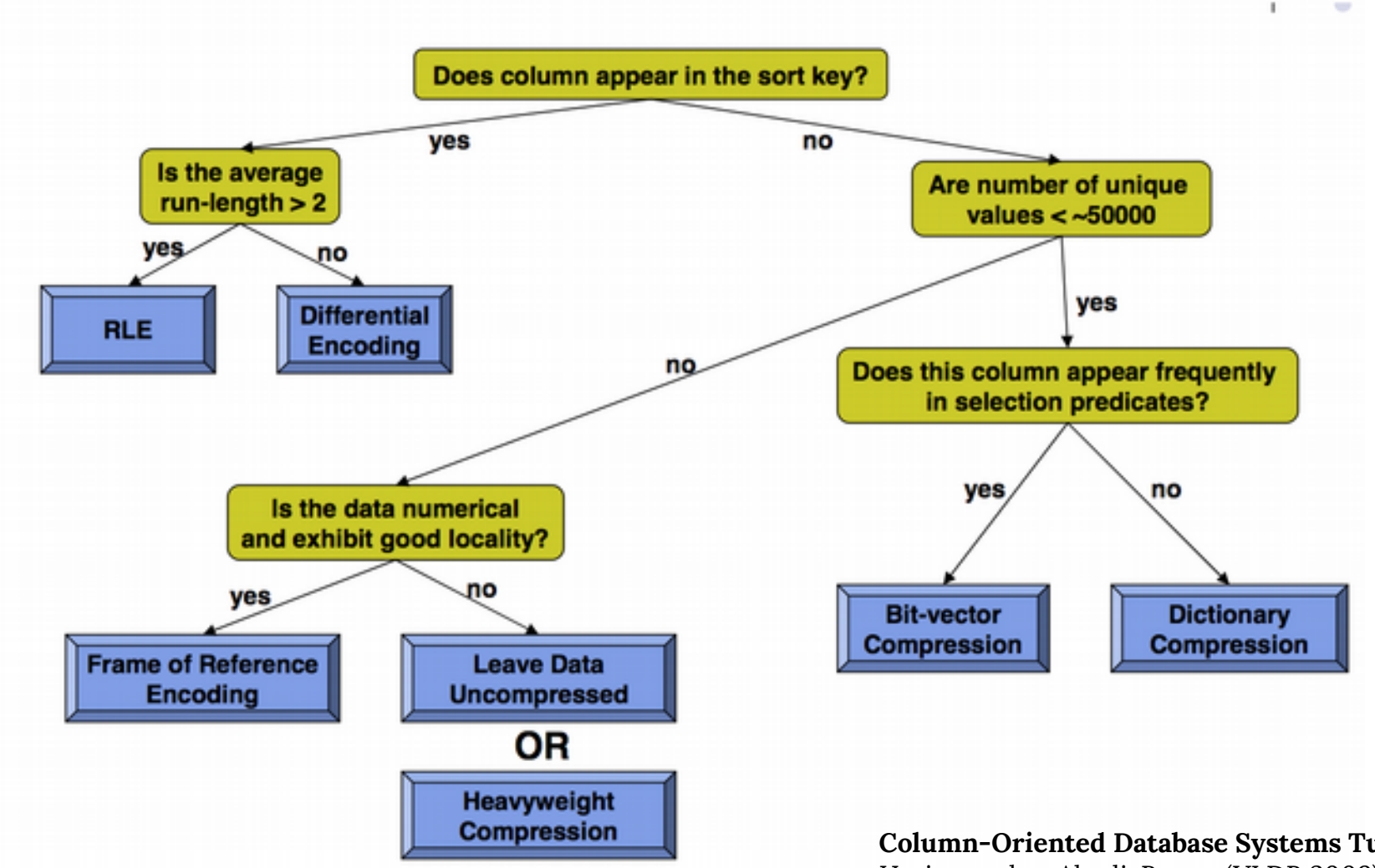
| nombre | salario |
|---|---|
| Bob | 25000 |
| Bill | 50000 |
| Jill | 40000 |

¿cómo reconstruir las tuplas de empleados a partir de las tuplas de las particiones?

Se mantienen *join indexes* para vincular particiones

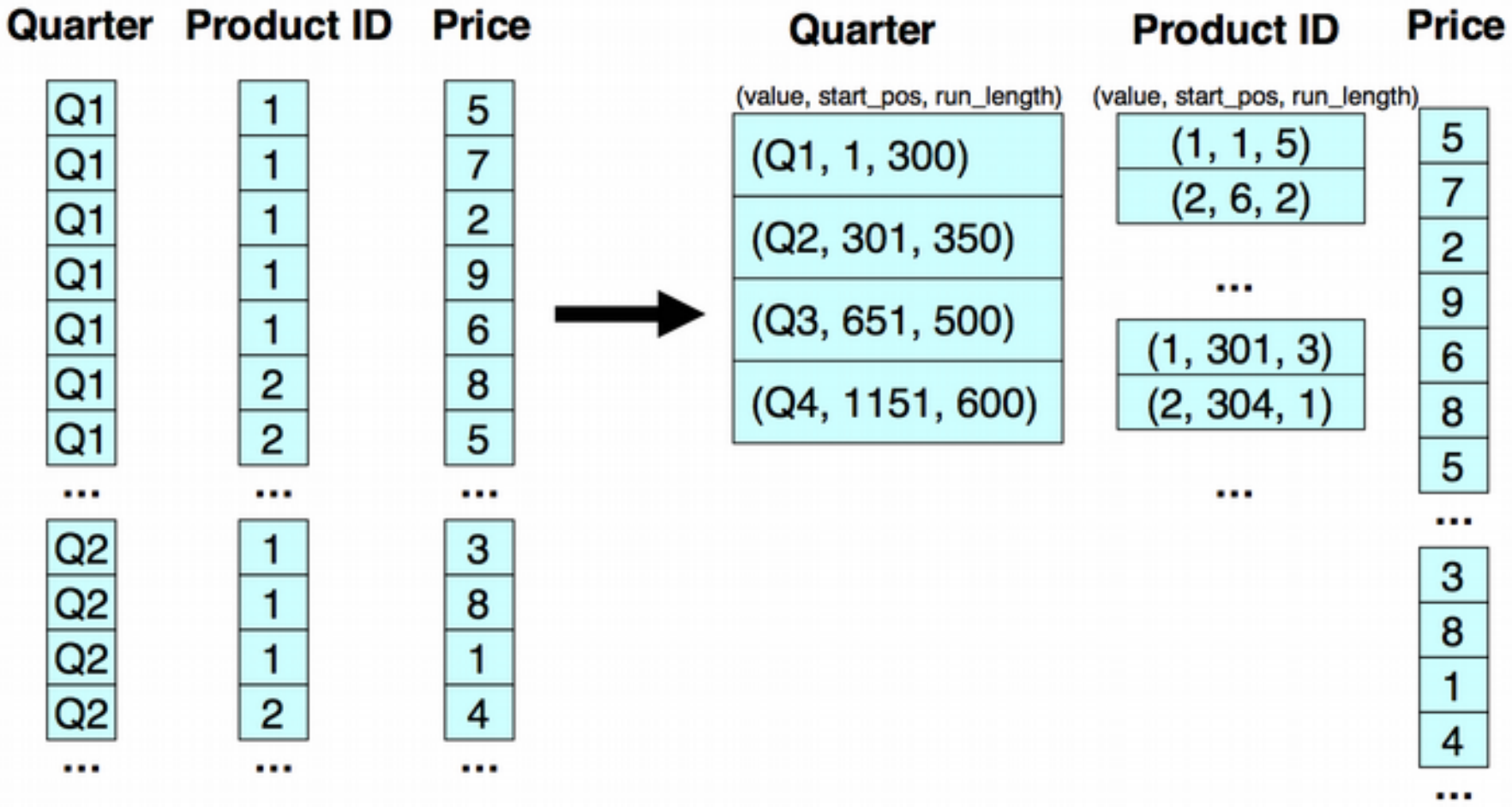
# Compresión

Sugieren diferentes tipos dependiendo de la naturaleza de los datos




**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

# Compresión: run-length encoding




**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

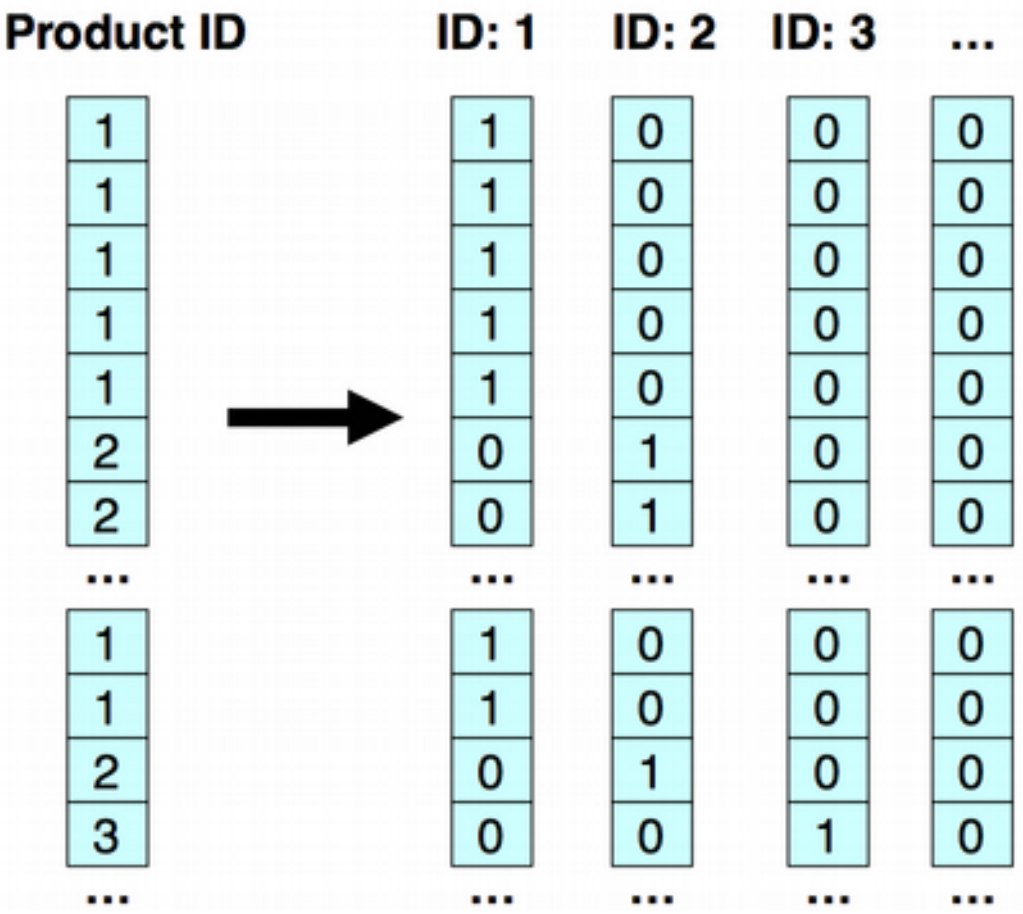
# Compresión: bit-vector encoding

**For each unique value, v, in column c, create bit-vector b**

1. b[i] = 1 if c[i] = v

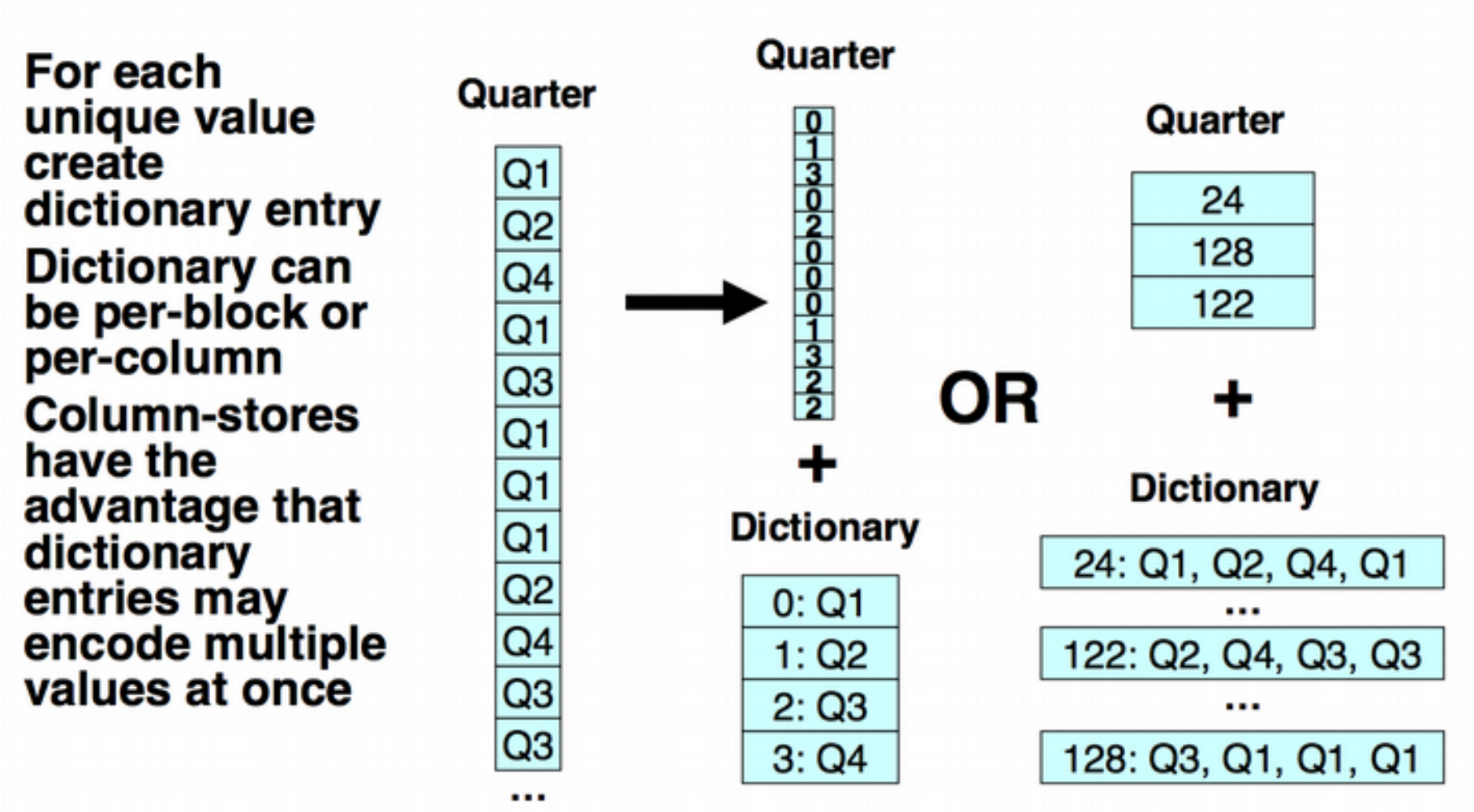
**Good for columns with few unique values**

**Each bit-vector can be further compressed if sparse**



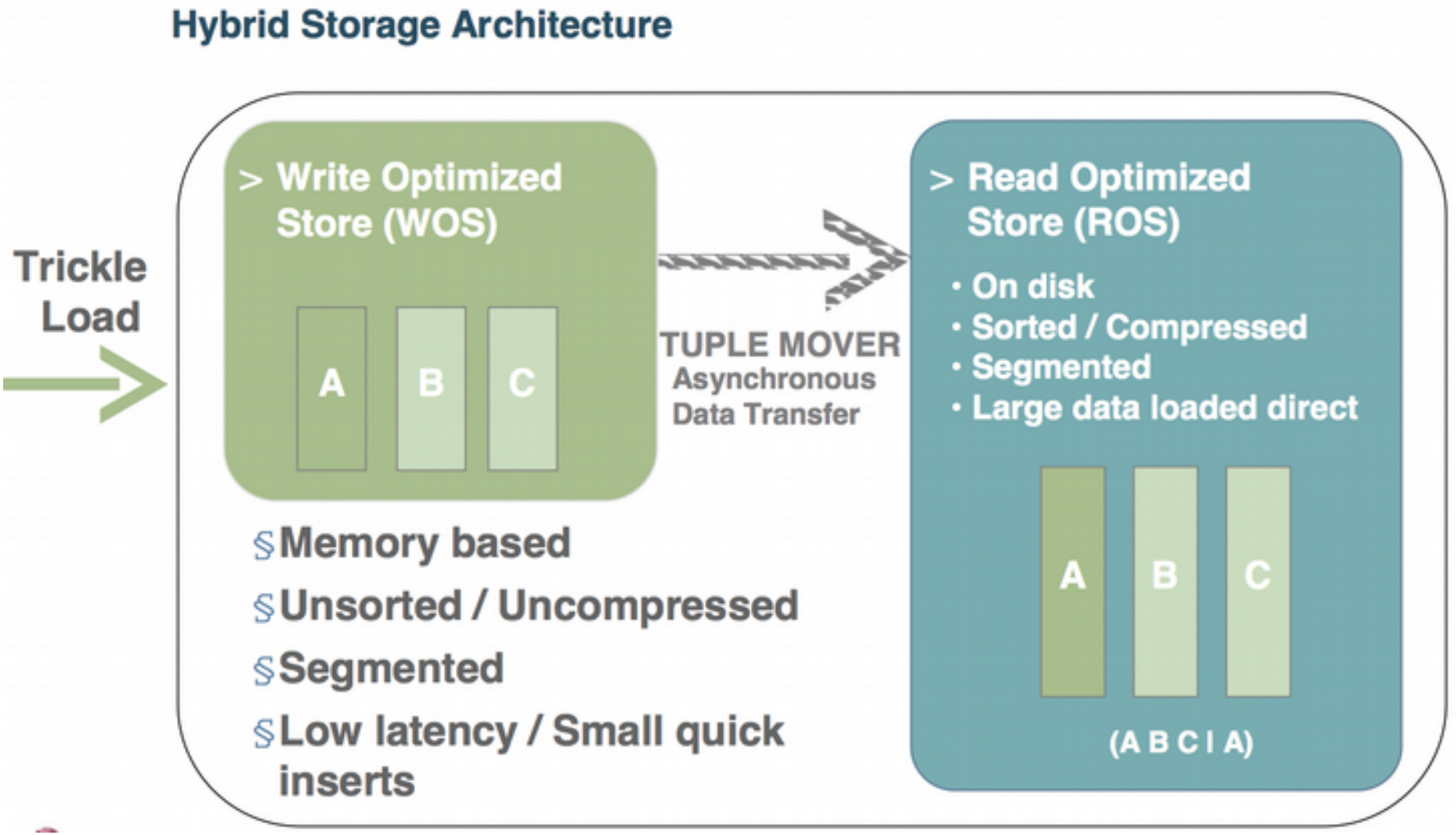

**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

# Compresión: dictionary encoding




**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

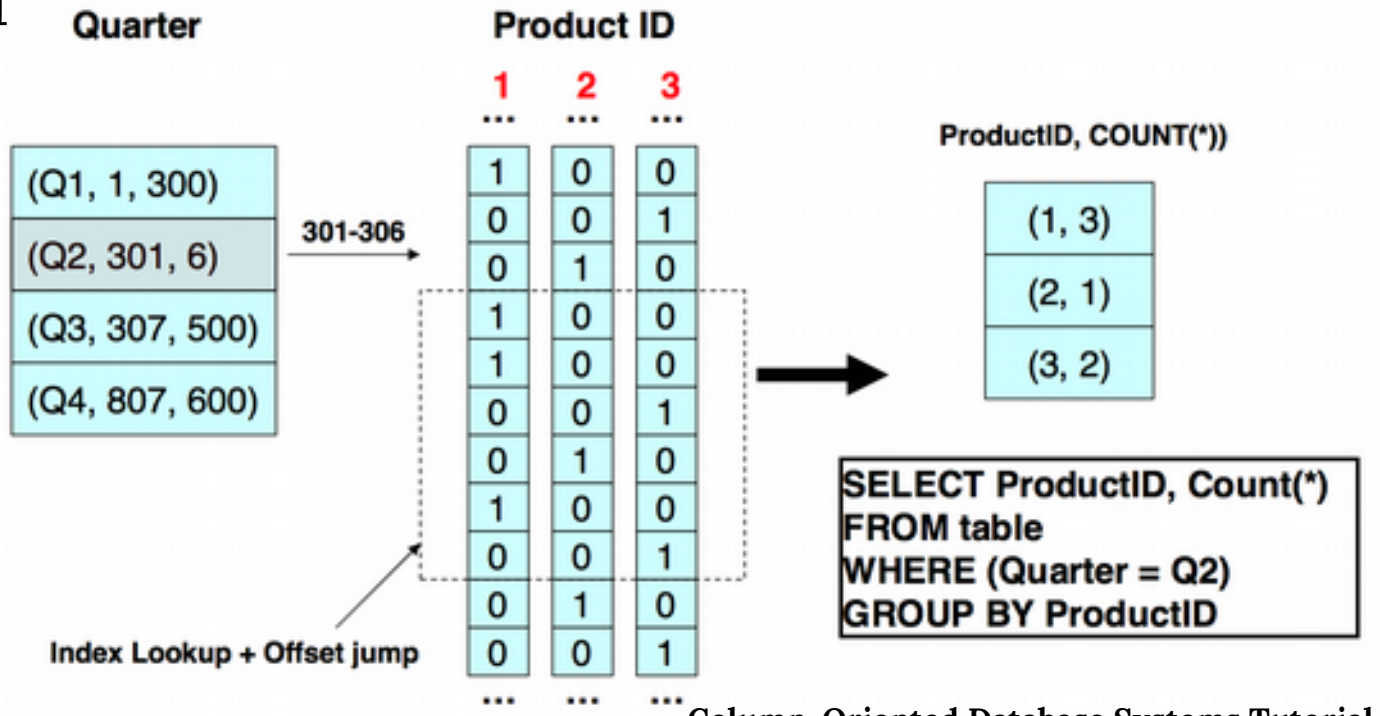
# C-Store: escrituras eficientes




**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

# C-Store: Procesamiento de consultas

- Estrategia:
  - Reconstruir las tuplas a último momento
  - Operar en lo posible sobre la representación compacta




**Column-Oriented Database Systems Tutorial**
Harizopoulos, Abadi, Boncz (VLDB 2009)

# Resumiendo

- El modelo de almacenamiento en columnas mejora la eficiencia de las consultas analíticas
- La idea clave es lograr leer sólo los datos que preciso, y reducir las lecturas a disco
- Requiere de algoritmos complejos para manejar los datos y operar sobre ellos!