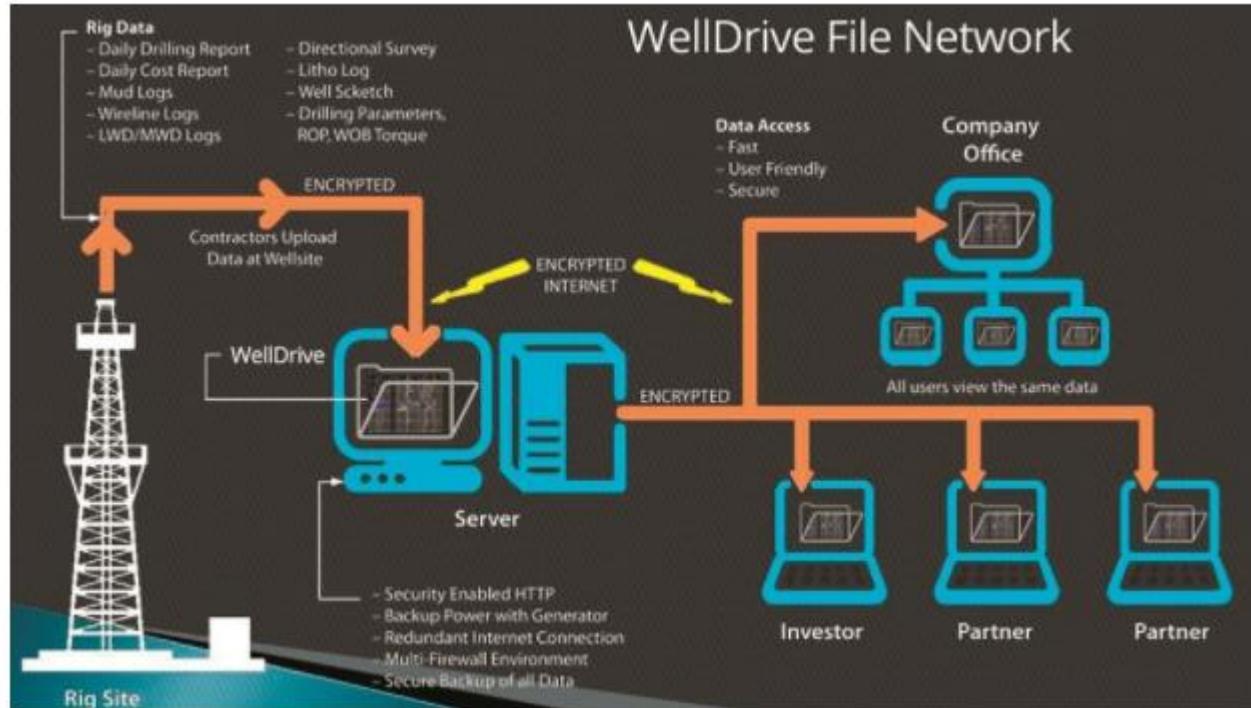


August 5th 2022

Bases de datos

Introducción y motivación

Las cuatro “V” del Big Data: volumen



“Research has determined that **a single well can produce about one terabyte of production data – every day.** That must be processed, organized and stored by someone, and can take hours of company time to do so.”

Fuente: [Production and operational data management made easier | Oil & Gas Product News \(oilandgasproductnews.com\)](http://oilandgasproductnews.com)

Las cuatro “V” del Big Data: volumen

London Stock Exchange completes acquisition of Refinitiv

29 January 2021



1



3



1

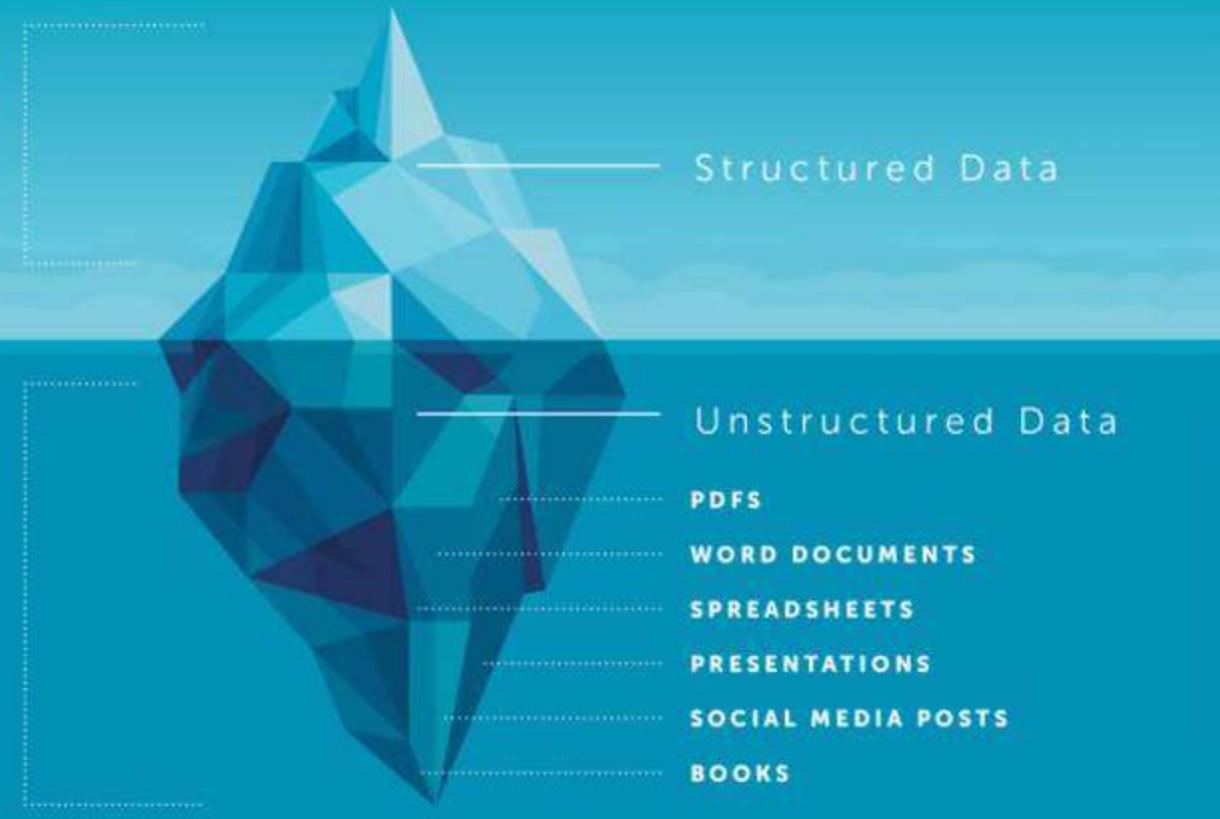


Source: London Stock Exchange

Further to the announcement on 28 January 2021, London Stock Exchange Group plc (“LSEG” or the “Company”) confirms that its all-share acquisition of Refinitiv (the “Transaction”) has now completed.

“It collects, aggregates, analyses, curates, stores and distributes huge volumes of data from a large number of sources. **It collects more than one petabyte of data every day** and processes more than a million unstructured documents. Its data platform provides 40 billion market updates every day.”

Fuente: <https://www.computerweekly.com/news/252467734/London-Stock-Exchange-agrees-to-acquire-Refinitiv-for-22bn>



<https://lawtomated.com/structured-data-vs-unstructured-data-what-are-they-and-why-care/>

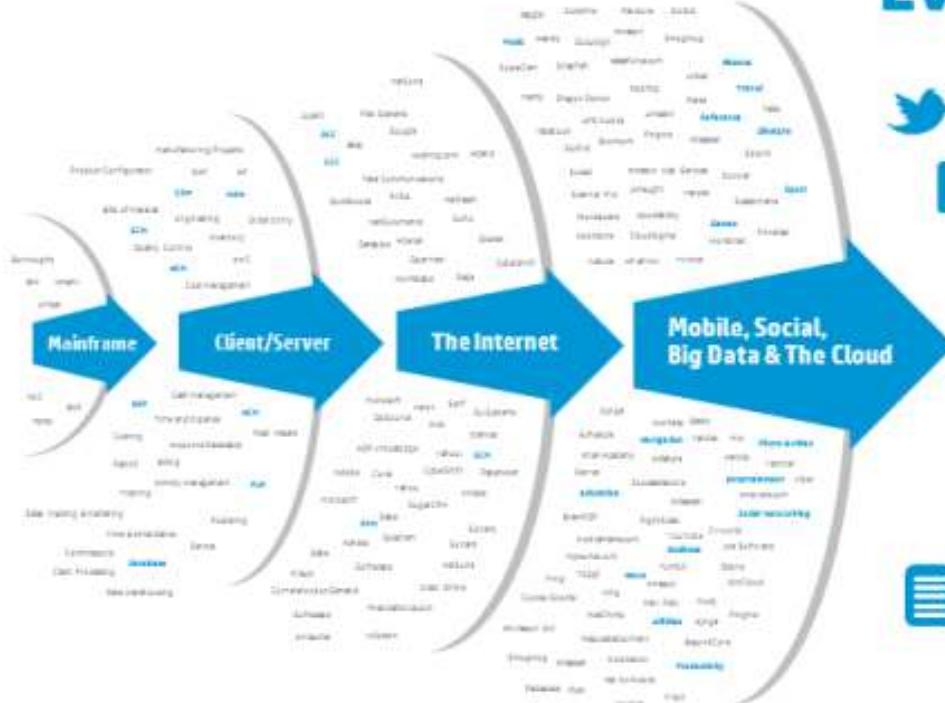
Las cuatro “V” del Big Data: variedad



- Base tecnológica de una ciudad inteligente, compuesta por sensores, comunicaciones y portales de datos abiertos.
- Procesamiento geográfico combinado con información de tráfico.
- Videos procesados en tiempo real.
- Sensores lumínicos, de energía, de concentración de NO₂.

Las cuatro “V” del Big Data: variedad (tipos de datos nuevos)

IDC predicts that by 2015 over 90 percent of that data will be unstructured (e.g., images, videos, MP3 music files, and other files based on social media and Web enabled workloads). While this data is full of rich information it is hard to understand and analyze². For many organizations, they haven't even begun to understand how to do it with these new data types.



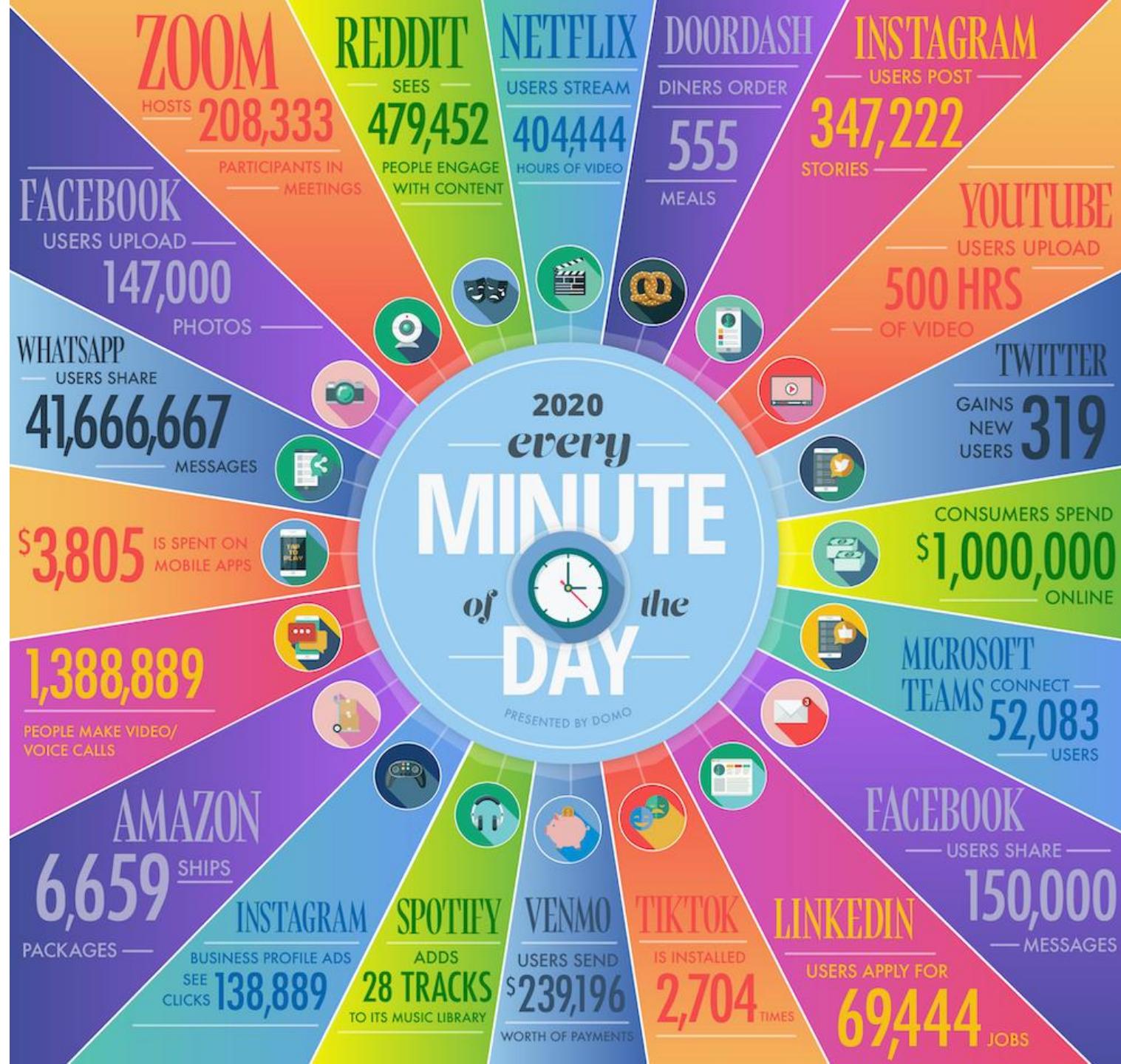
Every 60 seconds

- 98,000+ tweets
- 695,000 status updates
- 11million instant messages
- 698,445 Google searches
- 168 million+ emails sent
- 1,820TB of data created
- 217 new mobile web users

Fuente:

https://www.hp.com/hpinfo/newsroom/press_kits/2012/HPTDiscoverFrankfurt2012/IO_Whitepaper_Harness_the_Power_of_Big_Data.pdf

Fuente:
Data
Never
Sleeps 8.0
domo.com



Las cuatro “V” del Big Data: velocidad

THE WEEK

OPINION BRIEFINGS SPEED READS TALKING POINTS CARTOONS MORE ▾

IN-DEPTH BRIEFING

Wall Street's secret advantage: High-speed trading

They're unknown and invisible to most of us, but electronic trading programs now rule the stock markets

THE WEEK STAFF
JANUARY 11, 2015



What is high-speed trading? It's Wall Street's winning edge. By harnessing massive computer power to buy and sell stocks in the blink of an eye, high speed traders leverage tiny changes in value to make huge profits. The technique was pioneered in the

“Does co-location increase speed? Yes. **The physical proximity to the exchange server** reduces the time from when a firm's buy or sell order is entered and when it's executed. “By co-locating,” says Adam Afshar of Hyde Park Global, a high-speed trading firm, “we are able to take 21 milliseconds off our trades. In the past, 21 milliseconds was a trivial matter. Now it's a pivotal matter.” Several academic studies have found that shaving even one millisecond off every trade can be worth \$100 million a year to a large, high-speed trading firm.”

Fuente: <https://theweek.com/articles/493238/wall-streets-secret-advantage-highspeed-trading>

Las cuatro “V” del Big Data: veracidad



Semantic Web - Interoperability, Usability, Applicability *an IOS Press Journal*

Linked Data, Big Data, and the 4th Paradigm

Editorial

Pascal Hitzler,^a Krzysztof Janowicz,^b

^a Kno.e.sis Center, Wright State University, USA

^b University of California, Santa Barbara, USA

“Big Data, an increasing number of V’s has been used to characterize different dimensions and challenges of Big Data: volume, velocity, variety, value, and veracity. Interestingly, different (scientific) disciplines highlight certain dimensions and neglect others. For instance, super computing seems to be mostly interested in the volume dimension while researchers working on sensor webs and the internet of things seem to push on the velocity front. **The social sciences and humanities, in contrast, are more interested in value and veracity.**”

Fuente: <http://www.semantic-web-journal.net/system/files/swj488.pdf>

Las cuatro “V” del Big Data: veracidad

The Scientific
World Journal

[ScientificWorldJournal](#). 2015; 2015: 453597.

Published online 2015 Oct 1.

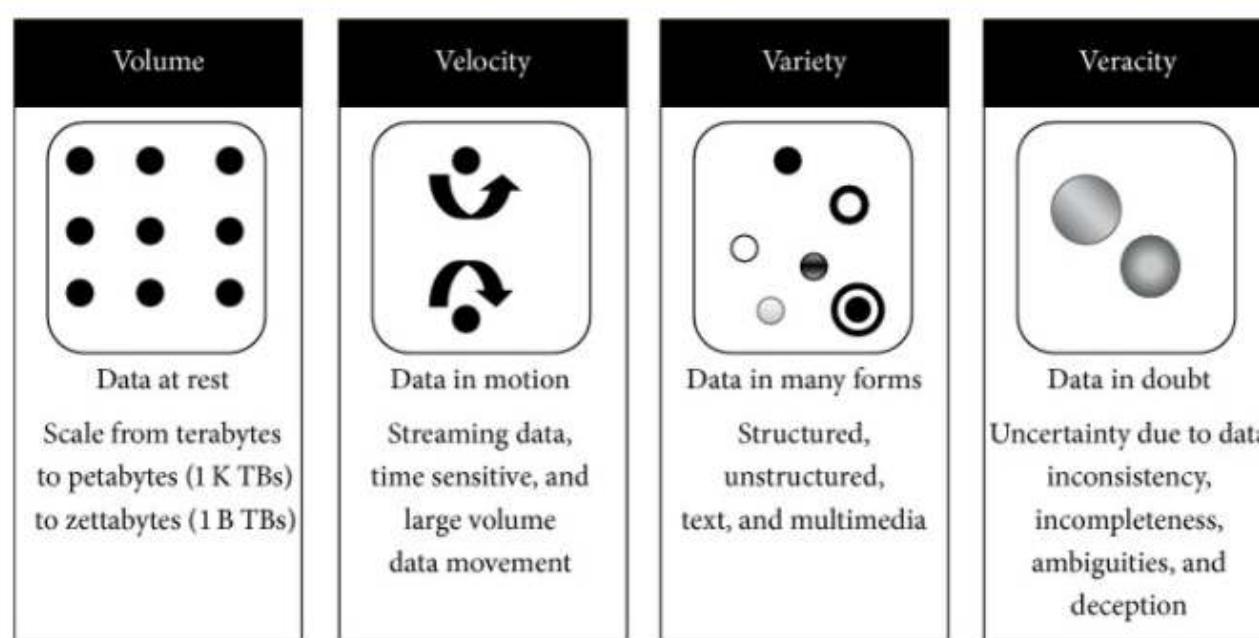
doi: [10.1155/2015/453597](https://doi.org/10.1155/2015/453597)

PMCID: F

PMI

Distilling Big Data: Refining Quality Information in the Era of Yottabytes

The data quality can generally be grouped under several categories like **accuracy, believability, reputation, objectivity, factuality, consistency, freedom from bias, correctness, and unambiguousness**.



The End of a DBMS Era (Might Be Upon Us)

By Michael Stonebraker

June 30, 2009

Comments (7)

VIEW AS:



SHARE:



Relational database management systems (DBMSs) have been remarkably successful in capturing the DBMS marketplace. To a first approximation they are “the only game in town,” and the major vendors (IBM, Oracle, and Microsoft) enjoy an overwhelming market share. They are selling “one size fits all”; i.e., a single relational engine appropriate for all DBMS needs. Moreover, the code line from all of the major vendors is quite elderly, in all cases dating from the 1980s. Hence, the major vendors sell software that is a quarter century old, and has been extended and morphed to meet today’s needs. In my opinion, these legacy systems are at the end of their useful life. They deserve to be sent to the “home for tired software.”

A Relational Model of Data for Large Shared Data Banks

E. F. CODD

IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. Ordering Dependence. Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

I. Relational Model and Normal Form

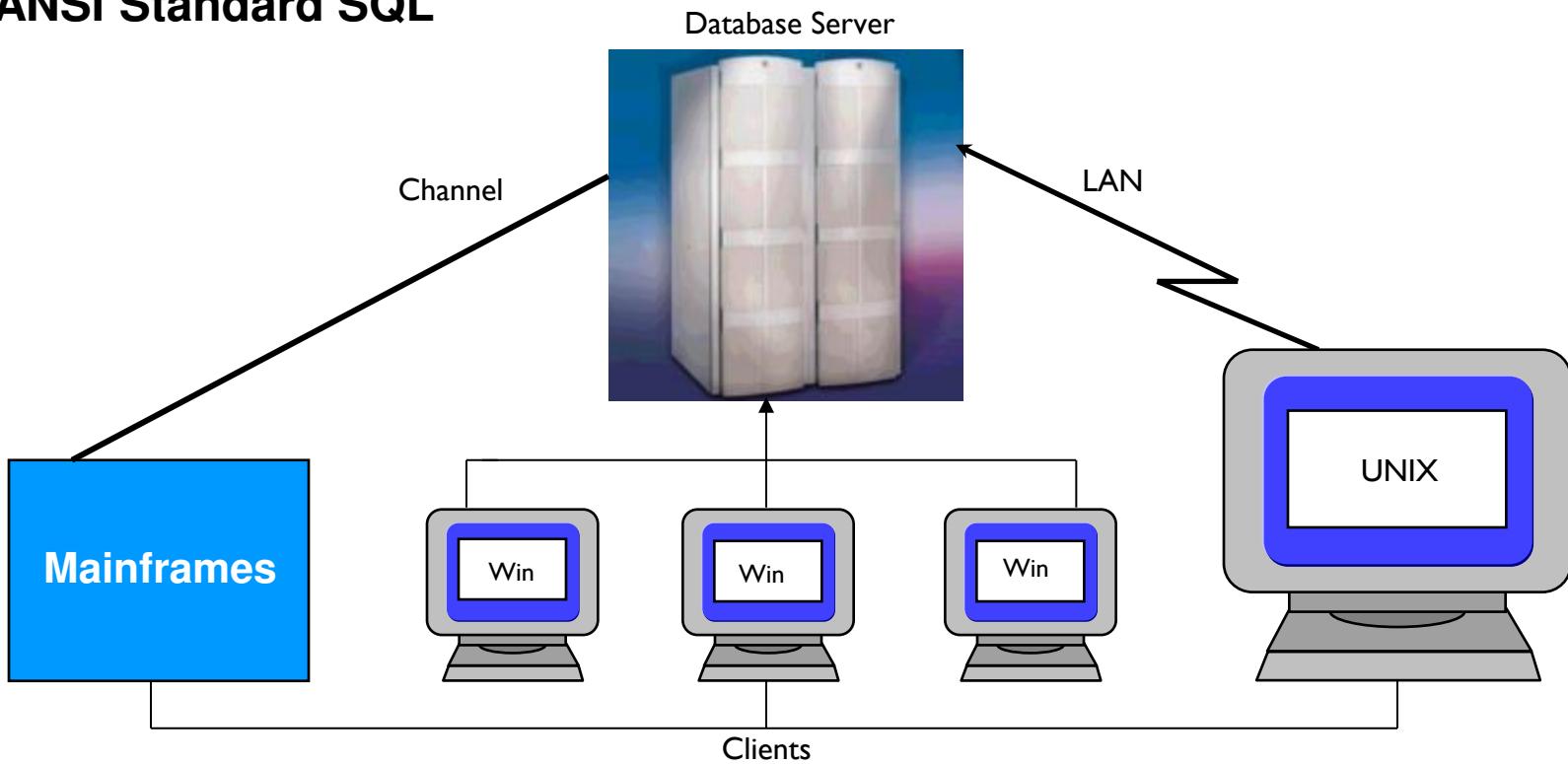
1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

What is a Database?

- **Relational Database Management System**
- **ANSI Standard SQL**



What is a Database?

A database is a collection of permanently stored data that is:

- **Logically related** - the data relates to other data (tables to tables).
- **Shared** - many users may access the data.
- **Protected** - access to data is controlled.
- **Managed** - the data has integrity and value.

Logical/Relational Modeling

The Logical Model

- Should be the same regardless of data volume
- Data is organized according to what it represents - real world business data in table (relational) form
- Is generic – the logical model is the template for physical implementation on any RDBMS platform

Normalization

- Process of reducing a complex data structure into a simple, stable one
- Involves removing redundant attributes, keys, and relationships from the conceptual data model

Transaction processing

is a unit of work that must be executed atomically and in apparent isolation

1. Logging in order to assure durability

2. Concurrency control transactions must appear to execute in isolation

3. Deadlock resolution transactions compete for resources

The ACID properties of transactions

"A" stands for "atomicity" the all-or-nothing execution of transactions.

Se tiene que ejecutar toda la query o nada

"I" stands for "isolation" the fact that each transaction must appear to be executed as if no other transaction is executing at the same time.

"D" stands for durability the condition that the effect on the database of a transaction must be never lost, once the transaction has completed.

The remaining letter "**C**" stands for consistency. Transactions are expected to preserve the consistency of the database.

Relational Databases

conjuntos: no estén repetidos y no estén ordenados

- Relational Databases are founded on Set Theory and based on the Relational Model.
- A Relational Database consists of a collection of logically related tables.
- A table is a two dimensional representation of data consisting of rows and columns.

| EMPLOYEE | | | | | | | | | Column (Attribute) | Relation |
|-----------------|-------------------------|-------------------|----------|-----------|------------|-----------|------------|---------------|--------------------|----------|
| EMPLOYEE NUMBER | MANAGER EMPLOYEE NUMBER | DEPARTMENT NUMBER | JOB CODE | LAST NAME | FIRST NAME | HIRE DATE | BIRTH DATE | SALARY AMOUNT | | |
| | | | | | | | | | | |
| 1006 | 1019 | 301 | 312101 | Stein | John | 861015 | 631015 | 3945000 | | |
| 1008 | 1019 | 301 | 312102 | Kanieski | Carol | 870201 | 680517 | 3925000 | | |
| 1005 | 0801 | 403 | 431100 | Ryan | Loretta | 861015 | 650910 | 4120000 | | |
| 1004 | 1003 | 401 | 412101 | Johnson | Darlene | 861015 | 560423 | 4630000 | | |
| 1007 | | | | Villegas | Arnando | 870102 | 470131 | 5970000 | | |
| 1003 | 0801 | 401 | 411100 | Trader | James | 860731 | 570619 | 4785000 | | |

The employee table has:

- Nine columns of data
- Six rows of data - one per employee
- Missing data values represented by nulls
- Column and row order are arbitrary

Primary Keys

Primary Key (PK) values uniquely identify each row in a table.

| EMPLOYEE | | | | | | | | |
|-----------------|-------------------------|-------------------|----------|-----------|------------|-----------|------------|---------------|
| EMPLOYEE NUMBER | MANAGER EMPLOYEE NUMBER | DEPARTMENT NUMBER | JOB CODE | LAST NAME | FIRST NAME | HIRE DATE | BIRTH DATE | SALARY AMOUNT |
| PK | | | | | | | | |
| 1006 | 1019 | 301 | 312101 | Stein | John | 861015 | 631015 | 3945000 |
| 1008 | 1019 | 301 | 312102 | Kanieski | Carol | 870201 | 680517 | 3925000 |
| 1005 | 0801 | 403 | 431100 | Ryan | Loretta | 861015 | 650910 | 4120000 |
| 1004 | 1003 | 401 | 412101 | Johnson | Darlene | 861015 | 560423 | 4630000 |
| 1007 | | | | Villegas | Arnando | 870102 | 470131 | 5970000 |
| 1003 | 0801 | 401 | 411100 | Trader | James | 860731 | 570619 | 4785000 |

Primary Key Rules

- A Primary Key is required for every table.
- Only one Primary Key is allowed in a table.
- Primary Keys may consist of one or more columns.
- Primary Keys cannot have duplicate values.
- Primary Keys cannot be null.
- Primary Keys are considered “non-changing” values.

Foreign Keys

EMPLOYEE (partial listing)

Job
Code
Table

| EMPLOYEE NUMBER | MANAGER EMPLOYEE NUMBER | DEPARTMENT NUMBER | JOB CODE | LAST NAME | FIRST NAME | HIRE DATE | BIRTH DATE | SALARY AMOUNT |
|-----------------|-------------------------|-------------------|----------|-----------|------------|-----------|------------|---------------|
| PK | FK | FK | FK | | | | | |
| 1006 | 1019 | 301 | 312101 | Stein | John | 861015 | 631015 | 3945000 |
| 1008 | 1019 | 301 | 312102 | Kanieski | Carol | 870201 | 680517 | 3925000 |
| 1005 | 0801 | 403 | 431100 | Ryan | Loretta | 861015 | 650910 | 4120000 |
| 1004 | 1003 | 401 | 412101 | Johnson | Darlene | 861015 | 560423 | 4630000 |
| 1007 | | | | Villegas | Arnando | 870102 | 470131 | 5970000 |
| 1003 | 0801 | 401 | 411100 | Trader | James | 860731 | 570619 | 4785000 |

Foreign Key (FK)
values model
relationships.

- Foreign Keys (FK) are optional.
- A table may have more than one FK.
- A FK may consist of more than one column.
- FK values may be duplicated.
- FK values may be null.
- FK values may be changed.
- FK values must exist elsewhere as a PK (i.e. have referential integrity).

DEPARTMENT

| DEPARTMENT NUMBER | DEPARTMENT NAME | BUDGET AMOUNT | MANAGER EMPLOYEE NUMBER |
|-------------------|--------------------------|---------------|-------------------------|
| PK | | | FK |
| 501 | marketing sales | 80050000 | 1017 |
| 301 | research and development | 46560000 | 1019 |
| 302 | product planning | 22600000 | 1016 |
| 403 | education | 93200000 | 1005 |
| 402 | software support | 30800000 | 1011 |
| 401 | customer support | 98230000 | 1003 |
| 201 | technical operations | 29380000 | 1025 |

Answering Questions with a Relational Database

EMPLOYEE

| EMPLOYEE NUMBER | MANAGER EMPLOYEE NUMBER | DEPARTMENT NUMBER | JOB CODE | LAST NAME | FIRST NAME | HIRE DATE | BIRTH DATE | SALARY AMOUNT |
|-----------------|-------------------------|-------------------|----------|-----------|------------|-----------|------------|---------------|
| PK | FK | FK | FK | | | | | |
| I006 | I019 | 301 | 312101 | Stein | John | 861015 | 631015 | 3945000 |
| I008 | I019 | 301 | 312102 | Kanieski | Carol | 870201 | 680517 | 3925000 |
| I005 | 0801 | 403 | 431100 | Ryan | Loretta | 861015 | 650910 | 4120000 |
| I004 | I003 | 401 | 412101 | Johnson | Darlene | 861015 | 560423 | 4630000 |
| I007 | | | 432101 | Villegas | Arnando | 870102 | 470131 | 5970000 |
| I003 | 0801 | 401 | 411100 | Trader | James | 860731 | 570619 | 4785000 |

DEPARTMENT

| DEPARTMENT NUMBER | DEPARTMENT NAME | BUDGET AMOUNT | MANAGER EMPLOYEE NUMBER |
|-------------------|--------------------------|---------------|-------------------------|
| PK | | | FK |
| 501 | marketing sales | 80050000 | I017 |
| 301 | research and development | 46560000 | I019 |
| 302 | product planning | 22600000 | I016 |
| 403 | education | 93200000 | I005 |
| 402 | software support | 30800000 | I011 |
| 401 | customer support | 98230000 | I003 |
| 201 | technical operations | 29380000 | I025 |

1. Name the department in which James Trader works.
2. Who manages the Education Department?
3. Identify by name an employee who works for James Trader.
4. James Trader manages which department?

Relational Advantages

Advantages of a Relational Database compared to other database

methodologies include:

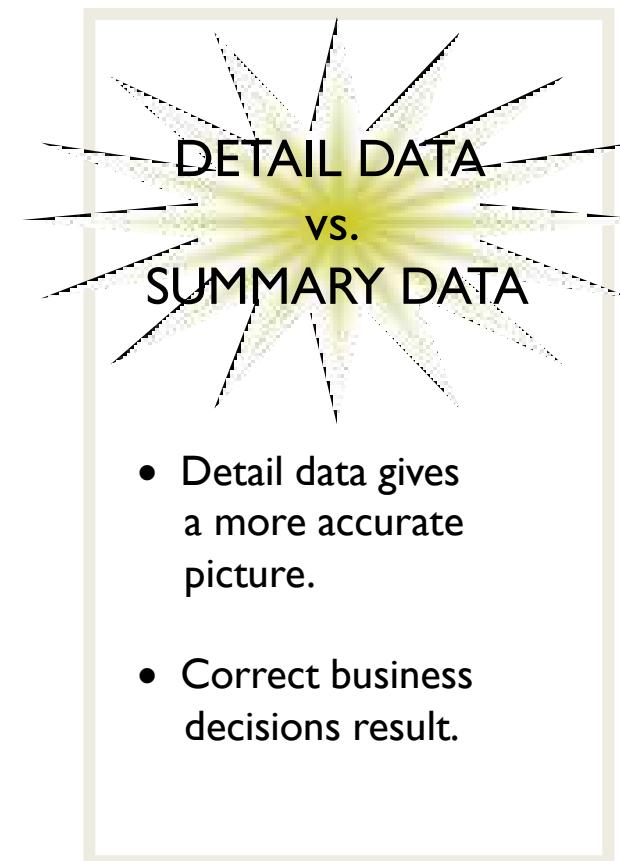
- More flexible than other types
- Allows businesses to quickly respond to changing conditions
- Being data-driven vs. application driven
- Modeling the business, not the processes
- Makes applications easier to build because the data does more of the work
- A single copy of the data may serve multiple purposes
- Supporting trend toward end-user computing
- Being easy to understand
- No need to know the access path
- Solidly founded in set theory

The Advantage of Using Detail Data

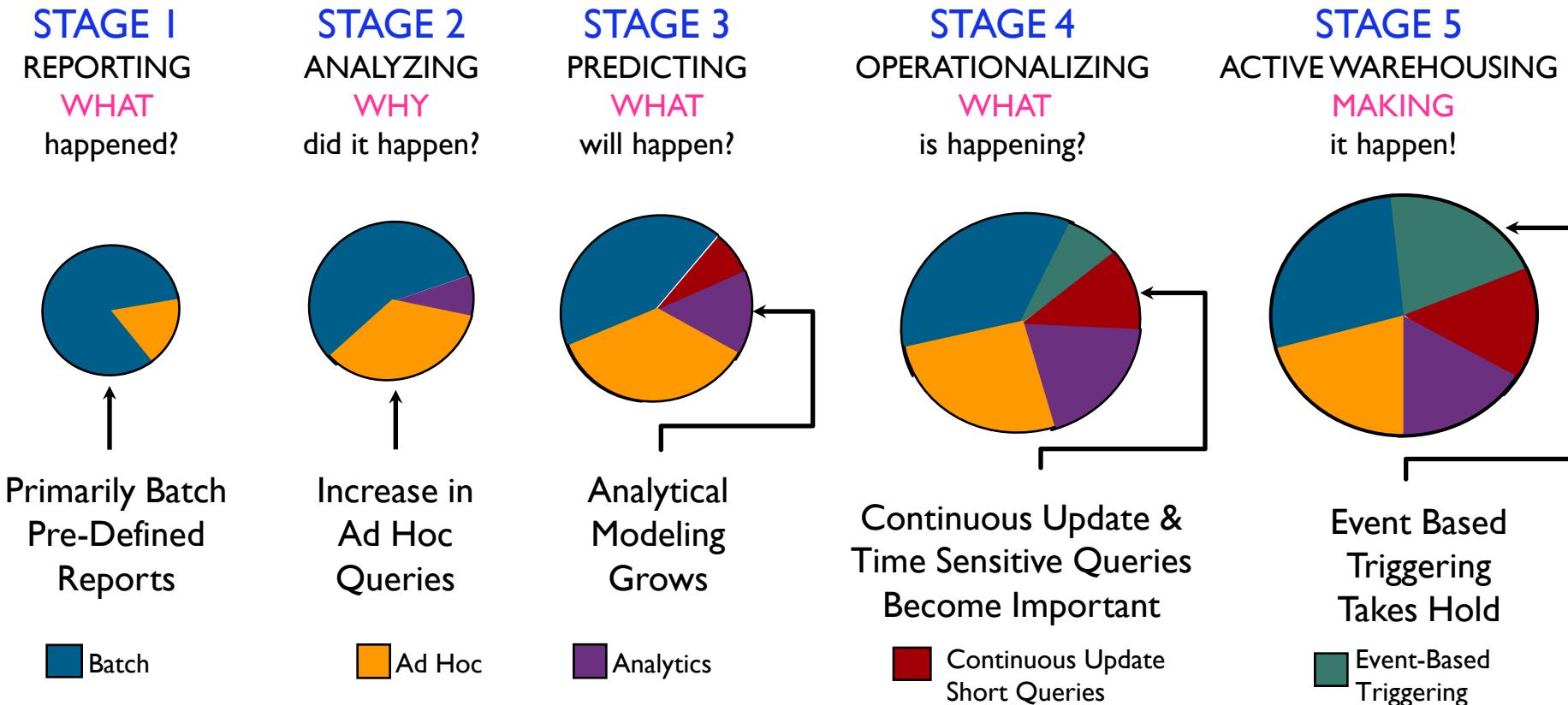
| STORE ITEM DAY | | | |
|----------------|-------------|--------|---------------|
| STORE NUMBER | ITEM NUMBER | DATE | QUANTITY SOLD |
| PK | | | |
| | | June01 | 110 |
| | | June02 | 126 |
| | | June03 | 127 |
| | | June04 | 144 |
| | | June05 | 102 |
| | 2 | June06 | 344 |
| | 2 | June07 | 410 |
| | 2 | June08 | 426 |
| | 2 | June09 | 297 |
| | 2 | June10 | 164 |
| | 2 | June11 | 167 |
| | 2 | June12 | 115 |
| | 2 | June13 | 108 |
| | 2 | June14 | 99 |
| ... | ... | ... | ... |
| 2 | 2 | June01 | 50 |
| 2 | 2 | June02 | 47 |
| 2 | 2 | June03 | 32 |
| 2 | 2 | June04 | 20 |
| 2 | 2 | June05 | 37 |
| 2 | 2 | June06 | 144 |
| 2 | 2 | June07 | 126 |
| 2 | 2 | June08 | 164 |
| 2 | 2 | June09 | 96 |
| 2 | 2 | June10 | 48 |
| 2 | 2 | June11 | 34 |
| 2 | 2 | June12 | 24 |
| 2 | 2 | June13 | 30 |
| 2 | 2 | June14 | 45 |
| ... | ... | ... | ... |
| 5 | 2 | June01 | 13 |
| 5 | 2 | June02 | 14 |
| 5 | 2 | June03 | 10 |
| 5 | 2 | June04 | 12 |
| 5 | 2 | June05 | 15 |
| 5 | 2 | June06 | 32 |
| 5 | 2 | June07 | 65 |
| 5 | 2 | June08 | 87 |
| 5 | 2 | June09 | 18 |
| 5 | 2 | June10 | 26 |
| 5 | 2 | June11 | 8 |
| 5 | 2 | June12 | 14 |
| 5 | 2 | June13 | 22 |
| 5 | 2 | June14 | 11 |

QUESTION: How effective was the national advertisement for jeans that ran June 6 through June 8?

| STORE ITEM DAY | | | |
|----------------|-------------|-------------|---------------|
| STORE NUMBER | ITEM NUMBER | WEEK ENDING | QUANTITY SOLD |
| PK | | | |
| ... | ... | ... | ... |
| 1 | 2 | June07 | 1363 |
| 2 | 2 | June14 | 1376 |
| ... | ... | ... | ... |
| 2 | 2 | June 07 | 456 |
| 2 | 2 | June14 | 441 |
| ... | ... | ... | ... |
| 5 | 2 | June07 | 161 |
| 5 | 2 | June14 | 186 |
| ... | ... | ... | ... |



Data Warehouse Usage Evolution



Data Analytics



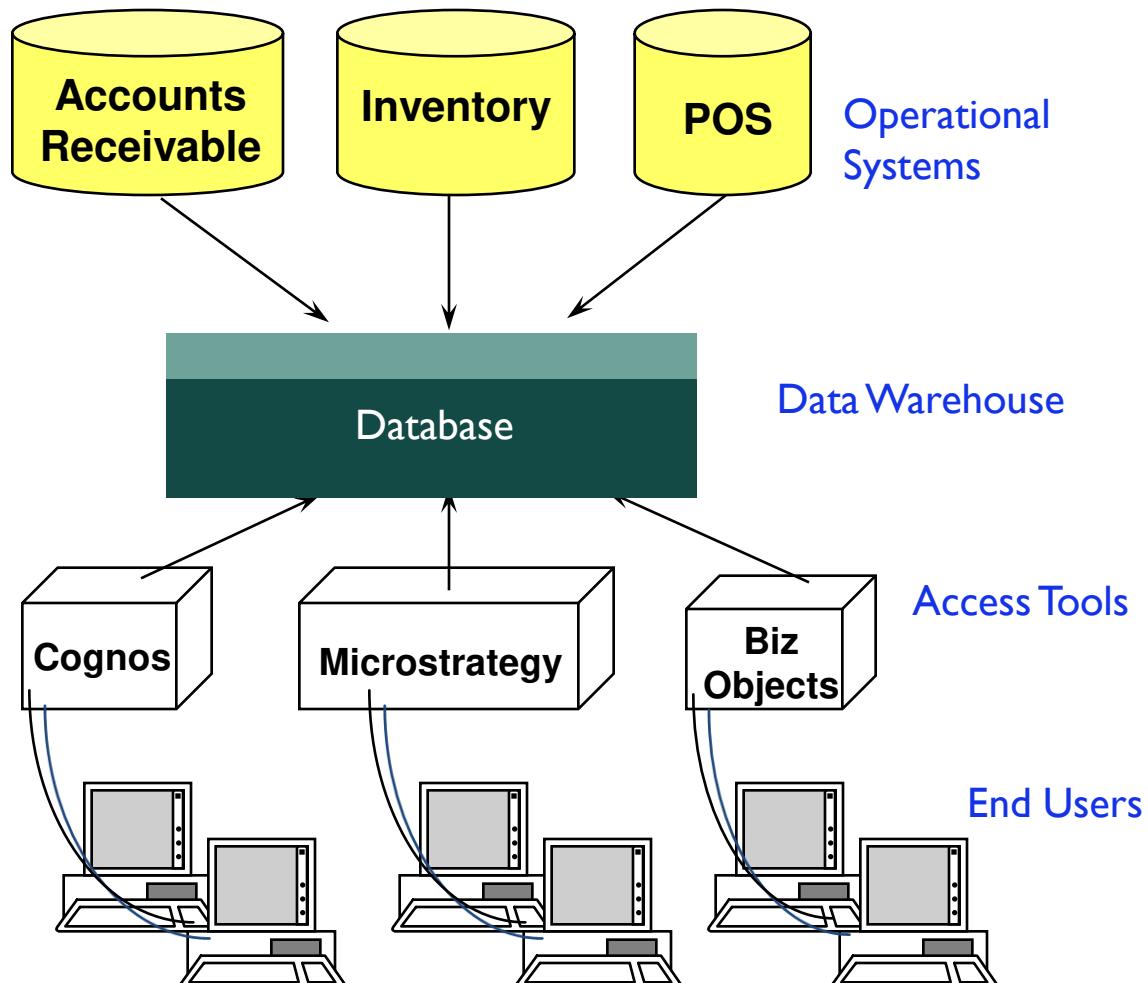
Data Analytics / Data Governance



The Data Warehouse

A central, enterprise-wide database that contains information extracted from operational systems.

- Based on enterprise-wide model
- Populated by extraction/loading of data from operational systems
- Provides a single view of the business

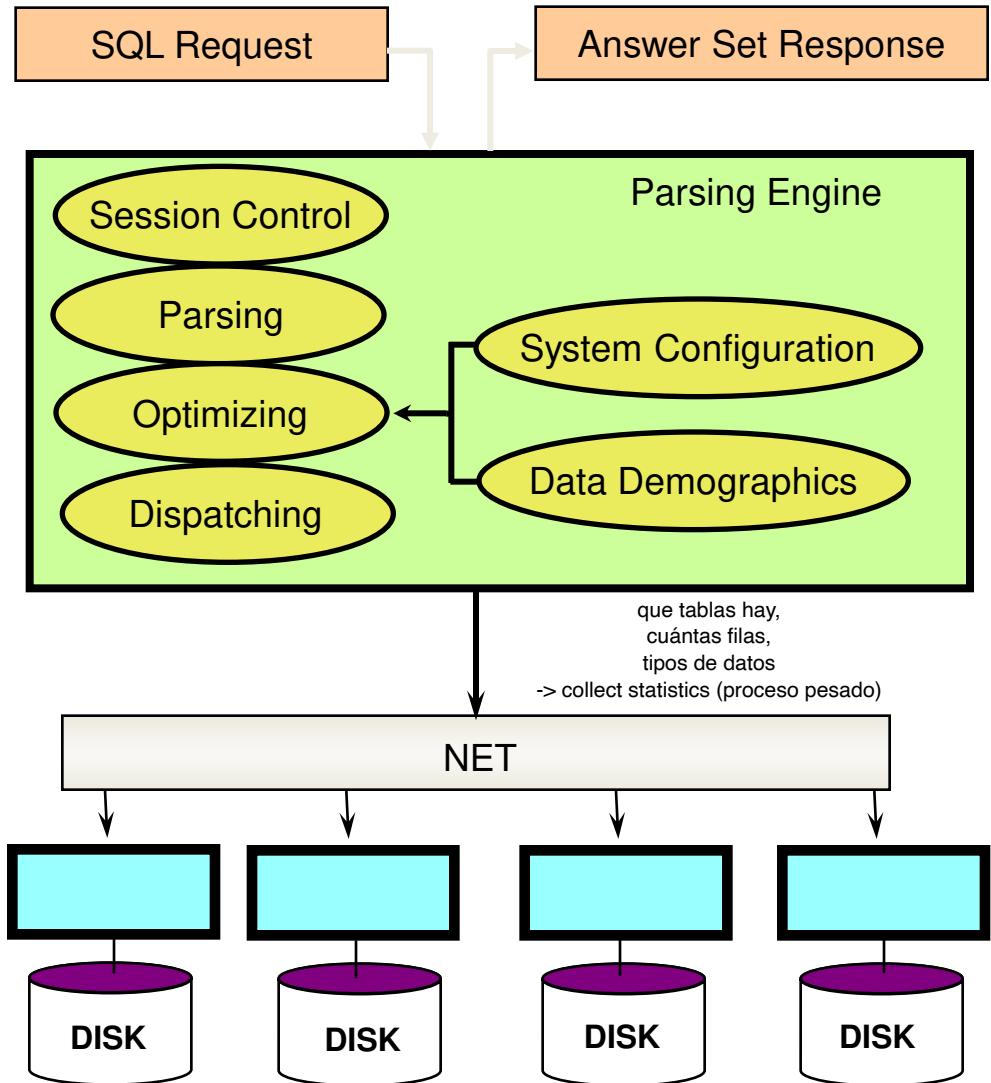


The Parsing Engine

el análisis sintáctico
desde el pto de vista del sistema

The Parsing Engine is responsible for:

- Managing individual sessions
- Parsing and optimizing your SQL requests
- Building query plans with the Optimizer
- Dispatching the optimized plan
- Sending the answer set response back to the requesting client



Database Objects

Databases are repositories for Database Objects:

- **Tables** – rows and columns of data
- **Views** – predefined subsets of existing tables
- **Triggers** – SQL statements associated with a table
- **Stored Procedures** – program stored
- **User Defined Function** –program that provides additional functionality

Database Space

Database objects require space in a Database or User as follows:

- **Tables** - require Perm Space
- **Views** - do not require Perm Space
- **Triggers** - do not require Perm Space
- **Stored Procedures** - require Perm Space
- **UDFs** – require Perm Space

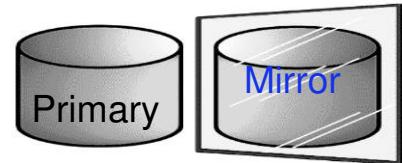
Space limits are specified for each database and for each user:

- **Perm Space** - maximum amount of space available for permanent tables and index subtables
- **Spool Space** - maximum amount of work space available for request processing
 - holds intermediate query result sets.

RAID Protection

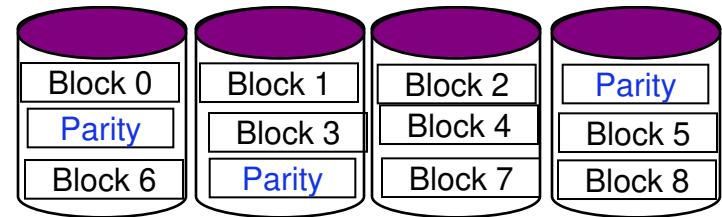
RAID 1 (Mirroring)

- Each physical disk has an exact.
- Mirroring reduces available disk space by 50%.



RAID 5 (Parity)

- Data striped across rank of 4 disks.
- If a disk fails, any missing block may be reconstructed using the other three disks.
- Parity reduces available disk space by 25% in a 4-disk rank.
- Reconstruction of failed disks takes longer than RAID 1.



Summary

RAID-1 - Good performance with disk failures
Higher cost in terms of disk space

RAID-5 - Reduced performance with disk failures
Lower cost in terms of disk space

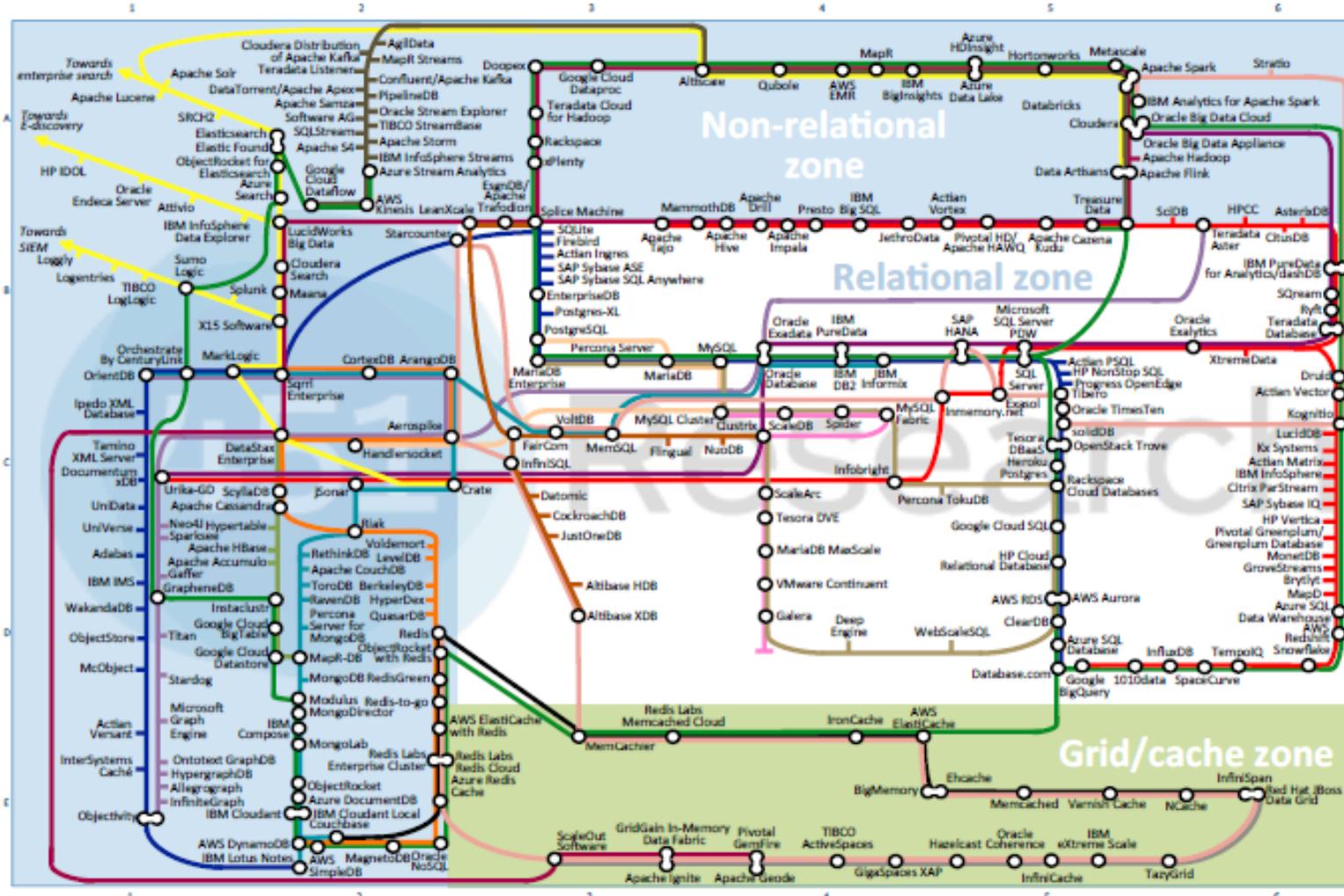
NoSQL

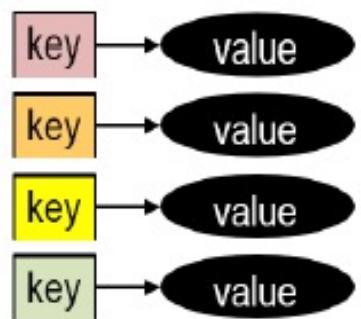
Data
Platforms
Map
January 2016

| Key | Description |
|-----|------------------------------|
| | General purpose |
| | Specialist analytic |
| | -in-a-Service |
| | BigTables |
| | Graph |
| | Document |
| | Key value stores |
| | Key value direct access |
| | Hadoop |
| | MySQL ecosystem |
| | Advanced clustering/sharding |
| | New SQL databases |
| | Data caching |
| | Data grid |
| | Search |
| | Appliances |
| | In-memory |
| | Stream processing |

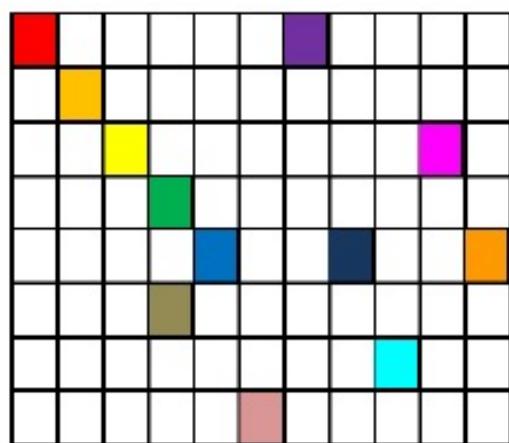
[https://
451research.com/
state-of-the-
database-landscape](https://451research.com/state-of-the-database-landscape)

© 2016 by 451 Research LLC.
All rights reserved.

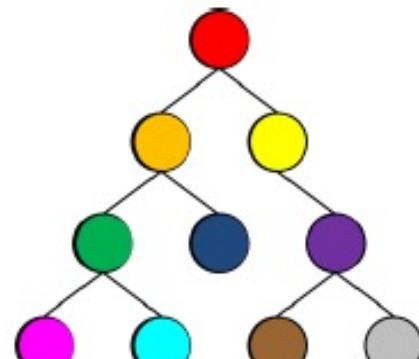




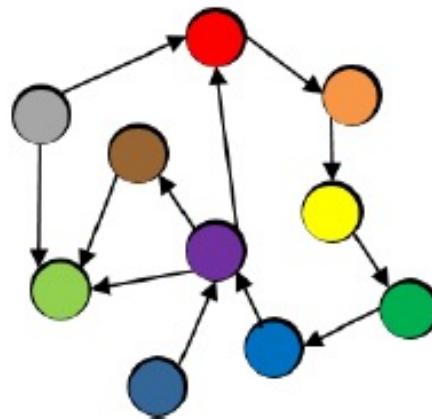
Key-Value



Column Family



Document



Graph Database

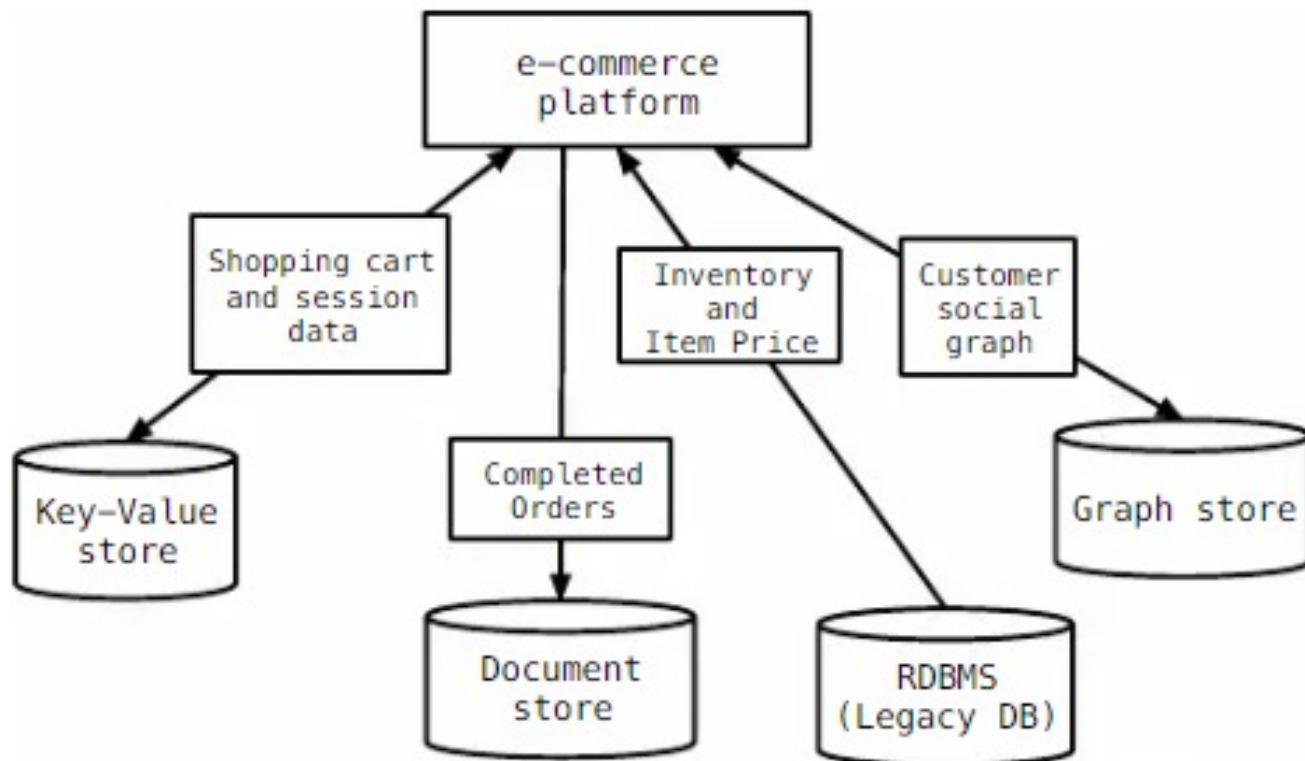


Figure 13.3 *Example implementation of polyglot persistence*

NoSQL Distilled

A Brief Guide to the Emerging
World of Polyglot Persistence

Pramod J. Sadalage
Martin Fowler

395 systems in ranking, August 2022

| Rank | | | DBMS | Database Model | Score | | |
|----------|----------|----------|--|--|----------|----------|----------|
| Aug 2022 | Jul 2022 | Aug 2021 | | | Aug 2022 | Jul 2022 | Aug 2021 |
| 1. | 1. | 1. | Oracle  | Relational, Multi-model  | 1260.80 | -19.50 | -8.46 |
| 2. | 2. | 2. | MySQL  | Relational, Multi-model  | 1202.85 | +7.98 | -35.37 |
| 3. | 3. | 3. | Microsoft SQL Server  | Relational, Multi-model  | 944.96 | +2.83 | -28.39 |
| 4. | 4. | 4. | PostgreSQL  | Relational, Multi-model  | 618.00 | +2.13 | +40.95 |
| 5. | 5. | 5. | MongoDB  | Document, Multi-model  | 477.66 | +4.68 | -18.88 |
| 6. | 6. | 6. | Redis  | Key-value, Multi-model  | 176.39 | +2.77 | +6.51 |
| 7. | 7. | 7. | IBM Db2 | Relational, Multi-model  | 157.23 | -3.99 | -8.24 |
| 8. | 8. | 8. | Elasticsearch | Search engine, Multi-model  | 155.08 | +0.75 | -2.01 |
| 9. | 9. | ↑ 10. | Microsoft Access | Relational | 146.50 | +1.41 | +31.66 |
| 10. | 10. | ↓ 9. | SQLite  | Relational | 138.87 | +2.20 | +9.06 |
| 11. | 11. | 11. | Cassandra  | Wide column | 118.15 | +3.74 | +4.49 |
| 12. | 12. | 12. | MariaDB  | Relational, Multi-model  | 113.89 | +1.37 | +14.92 |
| 13. | 13. | ↑ 23. | Snowflake  | Relational | 103.12 | +3.97 | +56.58 |
| 14. | 14. | ↓ 13. | Splunk | Search engine | 97.44 | -0.76 | +6.84 |
| 15. | ↑ 16. | ↑ 16. | Amazon DynamoDB  | Multi-model  | 87.26 | +3.32 | +12.36 |
| 16. | ↓ 15. | ↓ 15. | Microsoft Azure SQL Database | Relational, Multi-model  | 86.18 | +1.28 | +11.02 |
| 17. | 17. | ↓ 14. | Hive  | Relational | 78.66 | -0.82 | -5.27 |
| 18. | 18. | ↓ 17. | Teradata  | Relational, Multi-model  | 69.07 | -1.85 | +0.25 |
| 19. | 19. | ↓ 18. | Neo4j  | Graph | 59.35 | +0.94 | +2.40 |

DB-Engines

DB-Engines is an initiative to collect and present information on database management systems (DBMS). In addition to established relational DBMS, systems and concepts of the growing NoSQL area are emphasized.

The [DB-Engines Ranking](#) is a list of DBMS ranked by their current popularity. The list is updated monthly.

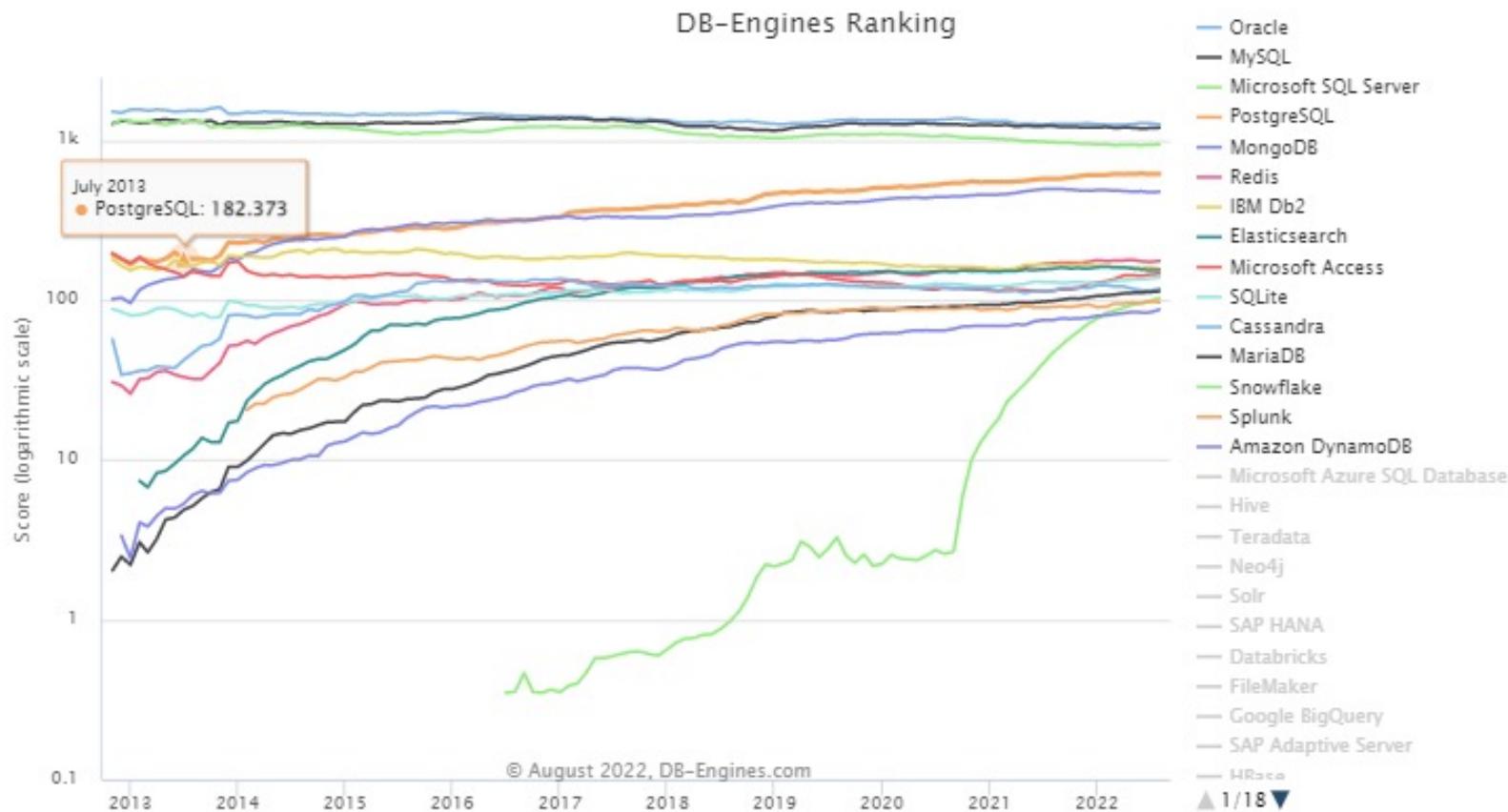
DB-Engines Ranking - Trend Popularity

The DB-Engines Ranking ranks database management systems according to their popularity.

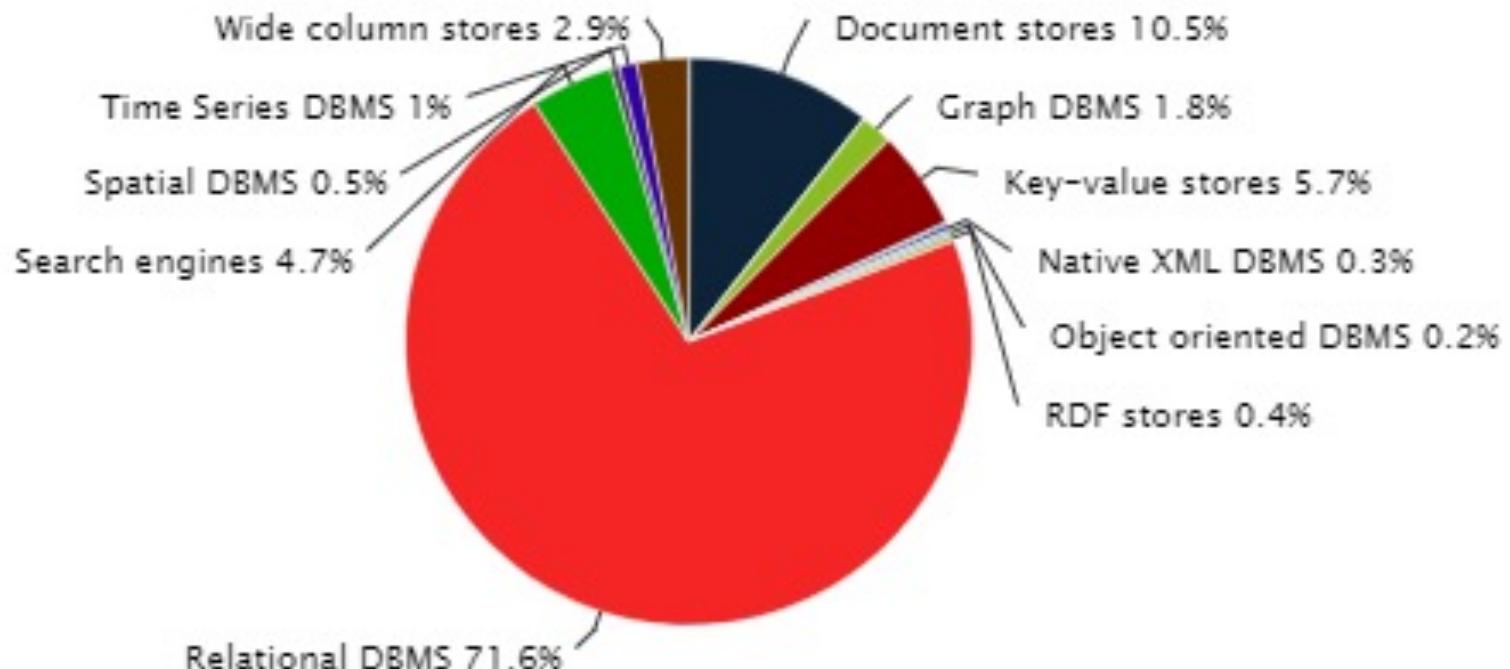
Read more about the [method](#) of calculating the scores.

| Rank | Trend | System | Score | Change |
|------|-------|------------|-------|--------|
| 1 | ▲ | Oracle | 1860 | + 27 |
| 2 | ▲ | MySQL | 1342 | + 47 |
| 3 | ▼ | SQL Server | 1278 | - 40 |
| 4 | | PostgreSQL | 174 | - 3 |
| 5 | | MS Access | 161 | - 8 |
| 6 | | DB2 | 158 | - 4 |

ranking table
August 2022

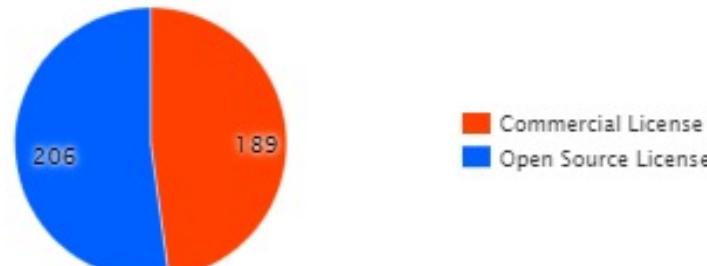


Ranking scores per category in percent, August 2022



© 2022, DB-Engines.com

Number of systems, August 2022



© 2022, DB-Engines.com

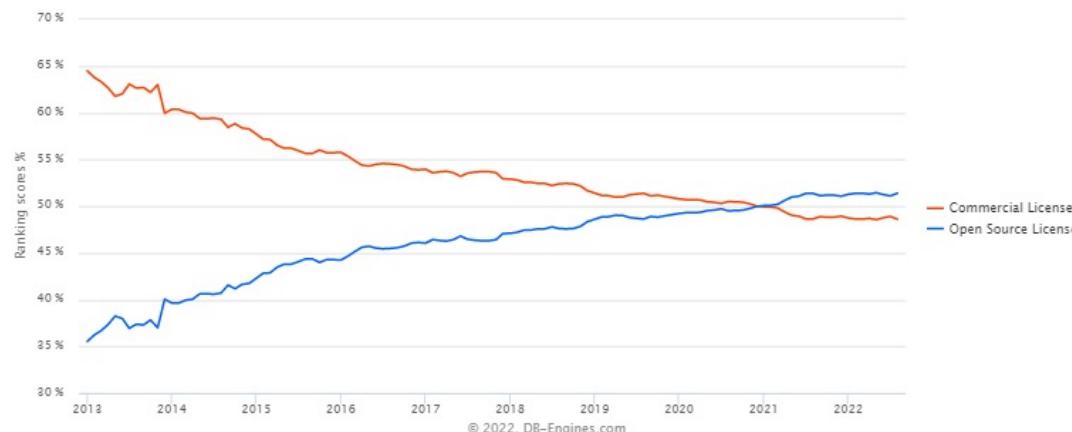
The top 5 commercial systems, August 2022

| Rank | System | Score | Overall Rank |
|------|----------------------|-------|--------------|
| 1. | Oracle | 1261 | 1. |
| 2. | Microsoft SQL Server | 945 | 3. |
| 3. | IBM Db2 | 157 | 7. |
| 4. | Microsoft Access | 147 | 9. |
| 5. | Snowflake | 103 | 13. |

The top 5 open source systems, August 2022

| Rank | System | Score | Overall Rank |
|------|---------------|-------|--------------|
| 1. | MySQL | 1203 | 2. |
| 2. | PostgreSQL | 618 | 4. |
| 3. | MongoDB | 478 | 5. |
| 4. | Redis | 176 | 6. |
| 5. | Elasticsearch | 155 | 8. |

Popularity trend



Catálogo de bases de datos

<https://dbdb.io/>

HOW TO WRITE A CV



Leverage the NoSQL boom

Scaling horizontally relational databases is a challenging task, if not impossible.

Escalamiento vertical: agregar cómputo a las computadoras existentes (ram, cpu)

Escalamiento horizontal: agregamos nodos o computadoras

One of the main problems that a **NoSQL databases** solves is scale, among others.

No to SQL? Anti-Database Movement Gains Steam

The meet-up in San Francisco last month had a whiff of revolution about it, like a latter-day techie version of the American Patriots planning the Boston Tea Party.



By Eric Lai

Computerworld | JUL 1, 2009 8:00 AM PT

The meet-up in San Francisco last month had a whiff of revolution about it, like a latter-day techie version of the American Patriots planning the Boston Tea Party.

The [inaugural get-together of the burgeoning NoSQL community](#) crammed 150 attendees into a meeting room at CBS Interactive.

Like the Patriots, who rebelled against Britain's heavy taxes, NoSQLers came to share how they had overthrown the tyranny of slow, expensive relational databases in favor of more efficient and cheaper ways of managing data.

MORE LIKE THIS ::

[Java in the Cloud: Google, Aptana, and Stax](#)

[Open Source CRM and ERP: Bending the Back Office](#)



[Review: Google Bigtable scales with ease](#)

NoSQL originally started off as a simple combination of two words—No and SQL—clearly and completely visible in the new term.

No acronym. What it literally means is, "I do not want to use SQL". To elaborate, "I want to access database without using any SQL syntax"

“...people are continually interpreting nosql to be *anti-RDBMS*, it's the only rational conclusion when the only thing some of these projects share in common is that they are *not relational databases*.”

Hitos

Bigtable, 2006

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,
Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall
and Werner Vogels

8

A Big Data Modeling Methodology for Apache Cassandra

1,

Artem Chebotko

DataStax Inc.

Email: achebotko@datastax.com

Andrey Kashlev

Wayne State University

Email: andrey.kashlev@wayne.edu

Shiyong Lu

Wayne State University

Email: shiyong@wayne.edu

Abstract—Apache Cassandra is a leading distributed database of choice when it comes to big data management with zero downtime, linear scalability, and seamless multiple data center deployment. With increasingly wider adoption of Cassandra

because of the expressivity of the Structured Query Language (SQL) that readily supports relational joins, nested queries, data aggregation, and numerous other features that help to retrieve a desired subset of stored data. As a result, traditional

Bigtable, 2006

Column family

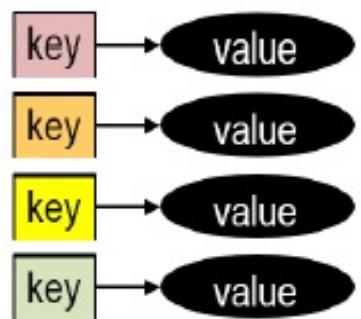
indexed by a row key, column key, and a timestamp

Amazon, Dynamo, 2007

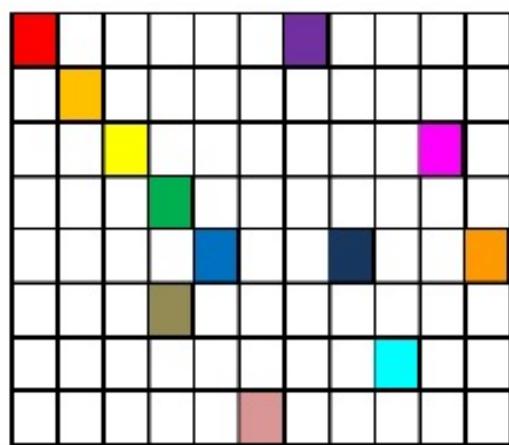
Key value

Facebook release Cassandra, 2008

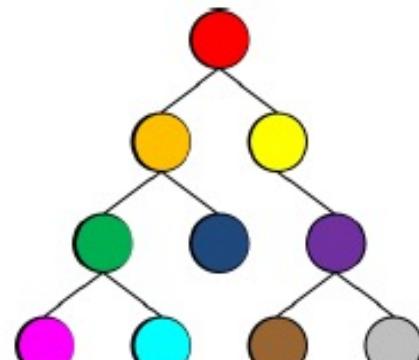
Column family



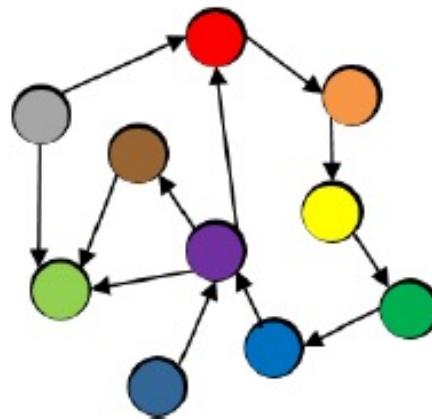
Key-Value



Column Family



Document



Graph Database

Key Value Store

En el tipo de almacén Key Value, se utiliza una tabla hash en la que una clave única apunta a un elemento.

Las claves pueden ser organizadas por grupos clave lógicos, requiriendo solamente estas claves para ser únicas dentro de su propio grupo. Esto permite tener claves idénticas en diferentes grupos lógicos. La siguiente tabla muestra un ejemplo de un almacén de valores clave, en el que la clave es el nombre de la ciudad y el valor es la dirección de Ulster University en esa ciudad.

| Key | Value |
|-------------|---|
| "Belfast" | {"University of Ulster, Belfast campus, York Street, Belfast, BT15 1ED"} |
| "Coleraine" | {"University of Ulster, Coleraine campus, Cromore Road, Co. Londonderry, BT52 1SA"} |

Document store

Most of the databases available under this category use XML or JSON. It is semi-structured as compared to rigid RDBMS.

For example, two records may have completely different set of fields or columns.

The records may or may not adhere to a specific schema (like the table definitions in RDBMS). For that matter, the database may not support a schema or validating a document against the schema at all.

(examples of document content using JSON)

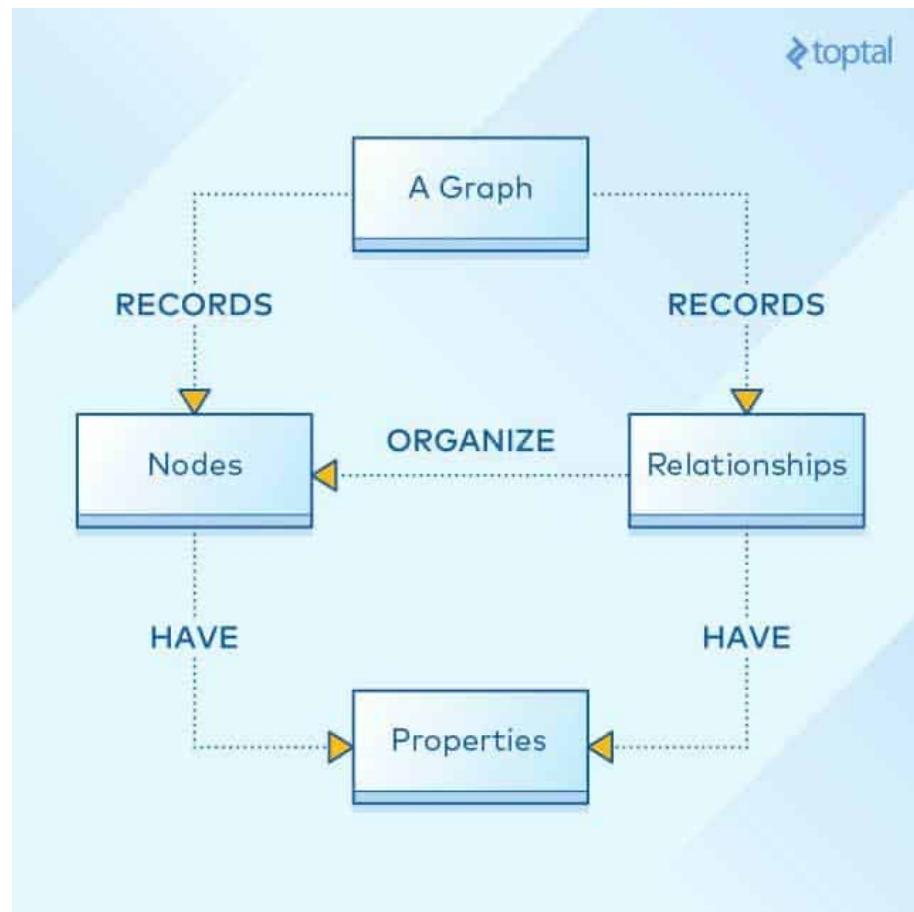
```
{  
    "EmployeeID": "SM1",  
    "FirstName" : "Anuj",  
    "LastName" : "Sharma",  
    "Age" : 45,  
    "Salary" : 10000000  
}  
  
{  
    "EmployeeID": "MM2",  
    "FirstName" : "Anand",  
    "Age" : 34,  
    "Salary" : 5000000,  
    "Address" : {  
  
        "Line1" : "123, 4th Street",  
        "City" : "Bangalore",  
        "State" : "Karnataka"  
    },  
    "Projects" : [  
        "nosql-migration",  
        "top-secret-007"  
    ]  
}  
  
{  
    "LocationID" : "Bangalore-SDC-BTP-CVRN",  
    "RegisteredName" : "ACME Software Development Ltd",  
    "RegisteredAddress" : {  
        "Line1" : "123, 4th Street",  
        "City" : "Bangalore",  
        "State" : "Karnataka"  
    },  
}
```

Column Family

| ActoresXSerie | |
|---------------|---|
| Titulo | K |
| Nombre | C |
| Edad | |
| Personaje | |

| ROW KEY | C.Cox:Edad | C.Cox:Personaje | J.Aniston:Edad | J.Aniston: Personaje | J.Galecki:Edad | J.Galecki:Personaje | J.Parsons:Edad | J.Parsons:Personaje |
|---------------------|------------|-----------------|----------------|----------------------|----------------|---------------------|----------------|---------------------|
| Friends | 52 | Monica Geller | | 48 Rachel Green | | | | |
| The Big Bang Theory | | | | | 40 | Leonard Hofstadter | | 38 Sheldon Cooper |

GRAFOS



List of some of the mature, popular and powerful NoSQL databases

| Document | Key-Value | XML | Column | Graph |
|------------|------------|-------|----------------|---------------|
| MongoDB | Redis | BaseX | BigTable | Neo4J |
| CouchDB | Membase | eXist | Hadoop / HBase | FlockDB |
| RavenDB | Voldemort | | Cassandra | InfiniteGraph |
| Terrastore | MemcacheDB | | SimpleDB | Cloudera |

Replacing Datacenter Oracle with Global Apache Cassandra on AWS

ROOPA TANGIRALA
Engineering Manager
Netflix

#StrataHadoop



Strata + Hadoop
WORLD

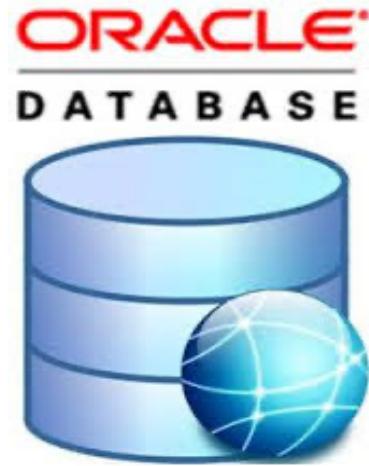
1997

Netflix was originally nothing more than an online version of a traditional video rental store. Customers could visit the Netflix website and choose which items they wished to rent. Each rental cost \$4, along with a \$2 shipping fee.



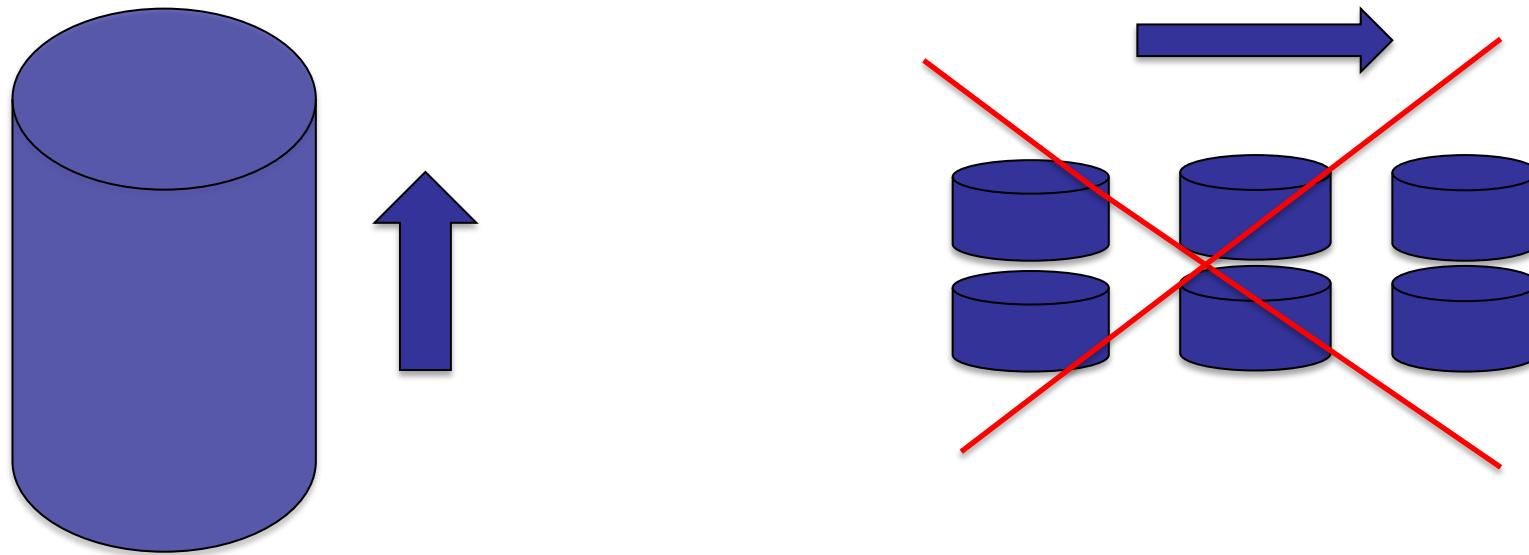
2007 Netflix shipped its billionth DVD!

BACKEND



RDBMS not flexible enough to handle modern day applications, which are a mixture of structured, semi-structured and unstructured data.
WHILE Oracle handles structured data well it lacks dynamic datamodel which is important for high traffic modern applications.

NO HORIZONTAL SCALING



Oracle scales up , master slave design limits both its scalability and performance for online applications, the failure of Oracle to add more capacity online in an elastic fashion without impacting the applications is a major drawback.

EVERY TWO WEEKS!!



DVD rentals delivered to your home - plans from only \$4.99 a month! No late fees - ever! Fast and free shipping both ways. FREE Trial.

The Netflix web site is temporarily unavailable.

We apologize for any inconvenience this causes you.

Please visit us again soon.

You can contact Netflix Customer Service at 1-888-638-3549.

Website went down every two weeks to include changes to PLSQL procedures, packages, schema changes and other backend database changes. Was ok for DVD since the shipping could happen after the site was back up but not acceptable for streaming service.

2008



Home / News & Analysis / Netflix Outage Blamed on Hardware

Netflix Outage Blamed on Hardware

BY CHLOE ALBANESIUS

AUGUST 25, 2008 10:03AM EST

0 COMMENTS

Faulty hardware is to blame for a glitch that took Netflix shipping systems offline for several days earlier this month, according to a blog post from the company.

0 SHARES 

Faulty hardware is to blame for a glitch that took Netflix shipping systems offline for several days earlier this month, according to a [blog post](#) from the company.

"With some great forensic help from our vendors, root cause was identified as a key faulty hardware component," according to Mike Osier, head of IT operations at Netflix. "It definitively caused the problem yet reported no detectable errors."

Problems started on Monday, August 11 when Netflix monitors flagged a database corruption in its shipping system. "Over the course of the day, we began experiencing similar problems in peripheral databases until our shipping system went down," Osier wrote.

This was the ugly outage and changed the way they think about infrastructure , reliability and availability.

#StrataHadoop

NETFLIX

Strata + Hadoop
WORLD

MOVE TO CLOUD



#StrataHadoop

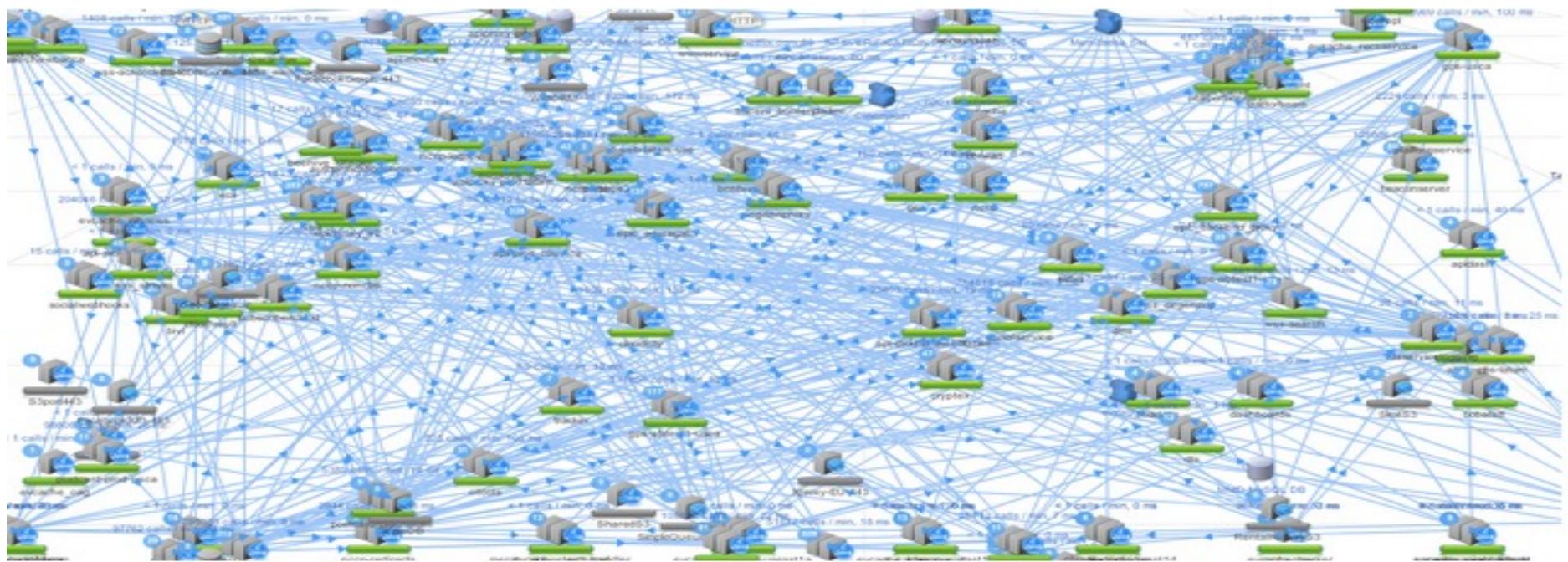
NETFLIX

Strata + Hadoop
WORLD

REQUIREMENTS

- HIGHLY AVAILABLE
- MULTI DATACENTER SUPPORT
- PREDICTABLE PERFORMANCE AT SCALE





- Horizontal, Homogenous, Commoditized

CHALLENGES

SECURITY



DENORMALIZE

DENORMALIZE

DENORMALIZE

DENORMALIZE

DENORMALIZE

O'REILLY®

2nd Edition



Cassandra

The Definitive Guide

DISTRIBUTED DATA AT WEB SCALE

Jeff Carpenter & Eben Hewitt

Row-Oriented

Cassandra's data model can be described as a partitioned row store, in which data is stored in sparse multidimensional hashtables. "Sparse" means that for any given row you can have one or more columns, but each row doesn't need to have all the same columns as other rows like it (as in a relational model). "Partitioned" means that each

row has a unique key which makes its data accessible, and the keys are used to distribute the rows across multiple data stores.

Cassandra's Data Model

In this section, we'll take a bottom-up approach to understanding Cassandra's data model.

The simplest data store you would conceivably want to work with might be an array or list. It would look like [Figure 4-1](#).



Figure 4-1. A list of values

So we'd like to add a second dimension to this list: names to match the values. We'll give names to each cell, and now we have a map structure, as shown in [Figure 4-2](#).

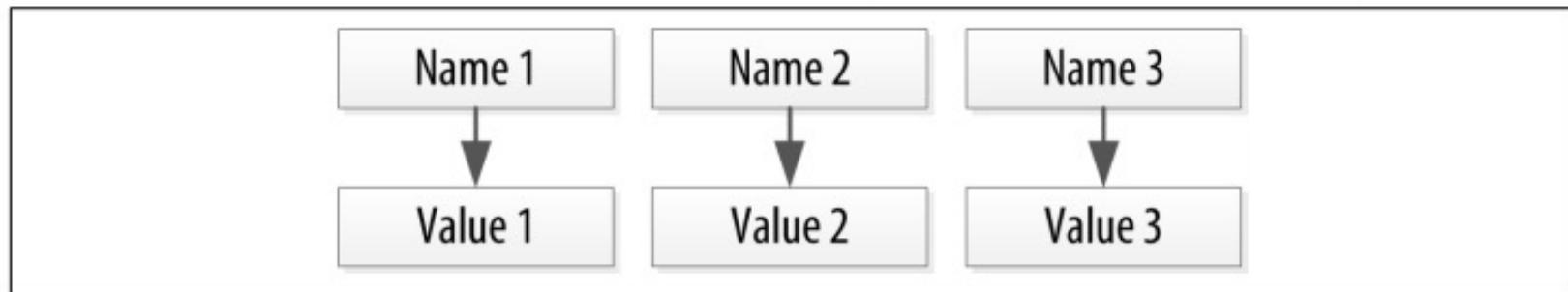


Figure 4-2. A map of name/value pairs

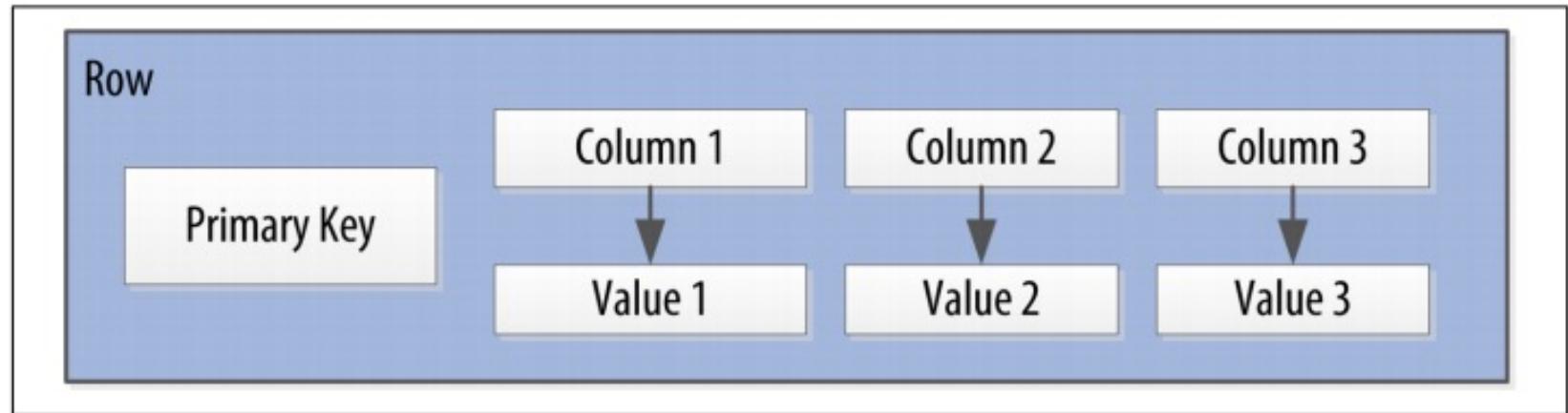


Figure 4-3. A Cassandra row

Cassandra defines a *table* to be a logical division that associates similar data. For example, we might have a `user` table, a `hotel` table, an `address book` table, and so on. In this way, a Cassandra table is analogous to a table in the relational world.

Now we don't need to store a value for every column every time we store a new entity. Maybe we don't know the values for every column for a given entity. For example, some people have a second phone number and some don't, and in an online form backed by Cassandra, there may be some fields that are optional and some that are required. That's OK. Instead of storing null for those values we don't know, which would waste space, we just won't store that column at all for that row. So now we have a sparse, multidimensional array structure that looks like Figure 4-4.

Table

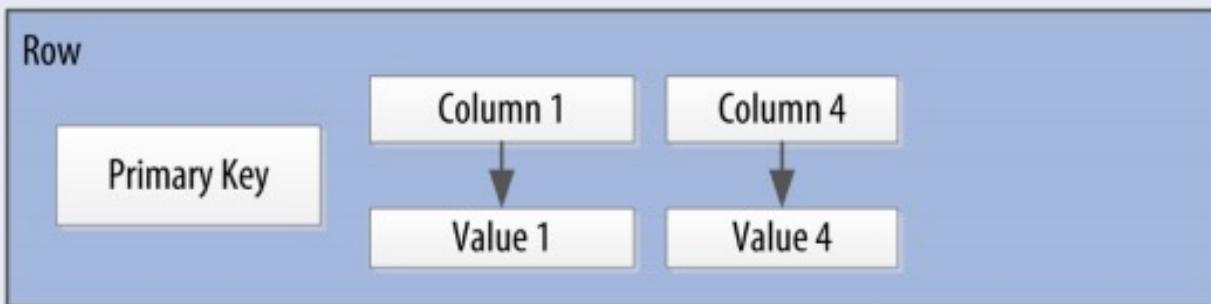
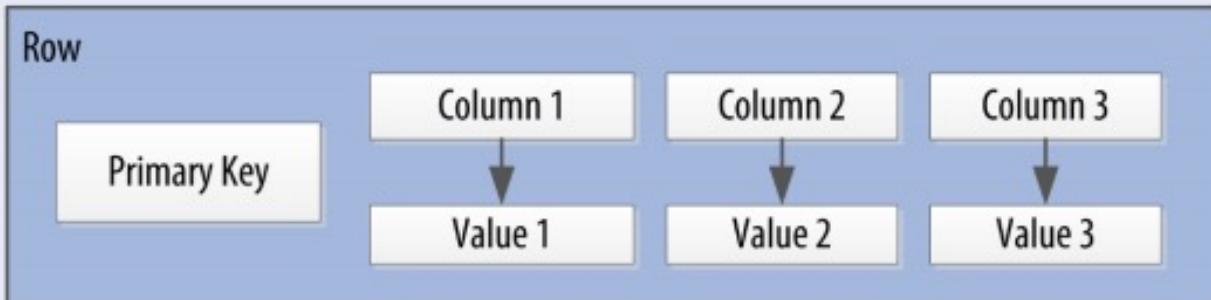


Figure 4-4. A Cassandra table

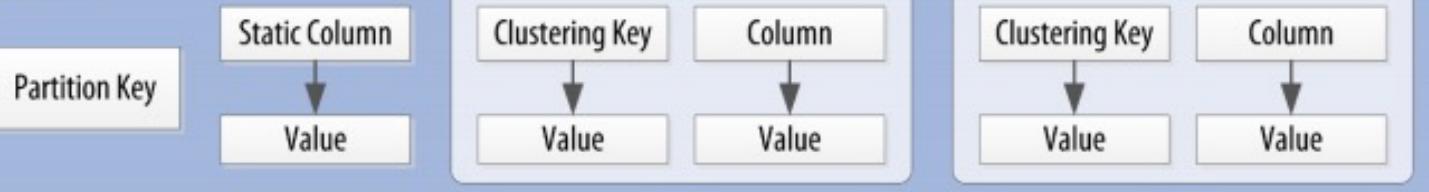
| | Column 1 | Column 2 | Column 3 | Column 4 |
|-------|----------|----------|----------|----------|
| Row 1 | Value 1 | Value 2 | Value 3 | |
| Row 2 | Value 1 | | | Value 4 |

When designing a table in a traditional relational database, you're typically dealing with "entities," or the set of attributes that describe a particular noun (`hotel`, `user`, `product`, etc.). Not much thought is given to the size of the rows themselves, because row size isn't negotiable once you've decided what noun your table represents. However, when you're working with Cassandra, you actually have a decision to make about the size of your rows: they can be wide or skinny, depending on the number of columns the row contains.

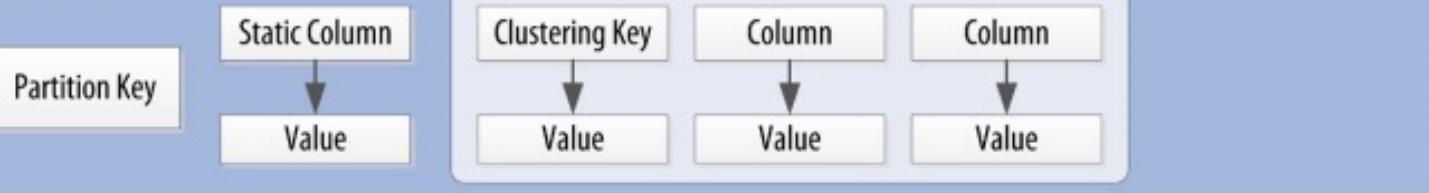
A wide row means a row that has lots and lots (perhaps tens of thousands or even millions) of columns. Typically there is a smaller number of rows that go along with so many columns. Conversely, you could have something closer to a relational model, where you define a smaller number of columns and use many different rows—that's the skinny model. We've already seen a skinny model in [Figure 4-4](#).

Table

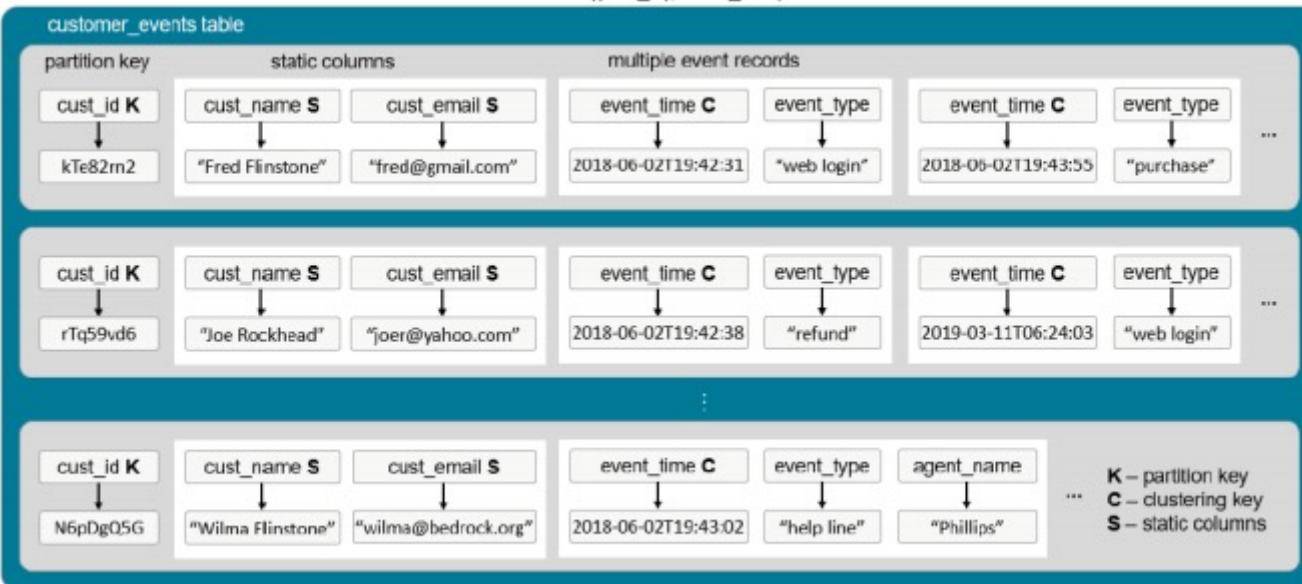
Wide Row



Wide Row



PRIMARY KEY: ((cust_id), event_time)



CONSISTENCIA

Modelos de consistencia: determina la visibilidad y el orden aparente de las actualizaciones.

Es básicamente un contrato entre los procesos y el almacén de datos. Este contrato dice que, si los procesos aceptan obedecer ciertas reglas, el almacén promete funcionar correctamente.

Consistencia fuerte: todas las operaciones de lectura deben devolver datos de la última operación de escritura completada.

Consistencia eventual: las operaciones de lectura verán, conforme pasa el tiempo, las escrituras. En un estado de equilibrio, el estado devolvería el último valor escrito.

A medida que

$$t \longrightarrow \infty$$

los clientes verán las escrituras

Consistencia no estricta

Read Your Own Writes (RYOW) consistency

El cliente lee sus actualizaciones inmediatamente después de que hayan sido completadas, independientemente si escribe en un server y lee de otro. Las actualizaciones de otros clientes no tienen por qué ser visibles instantáneamente.

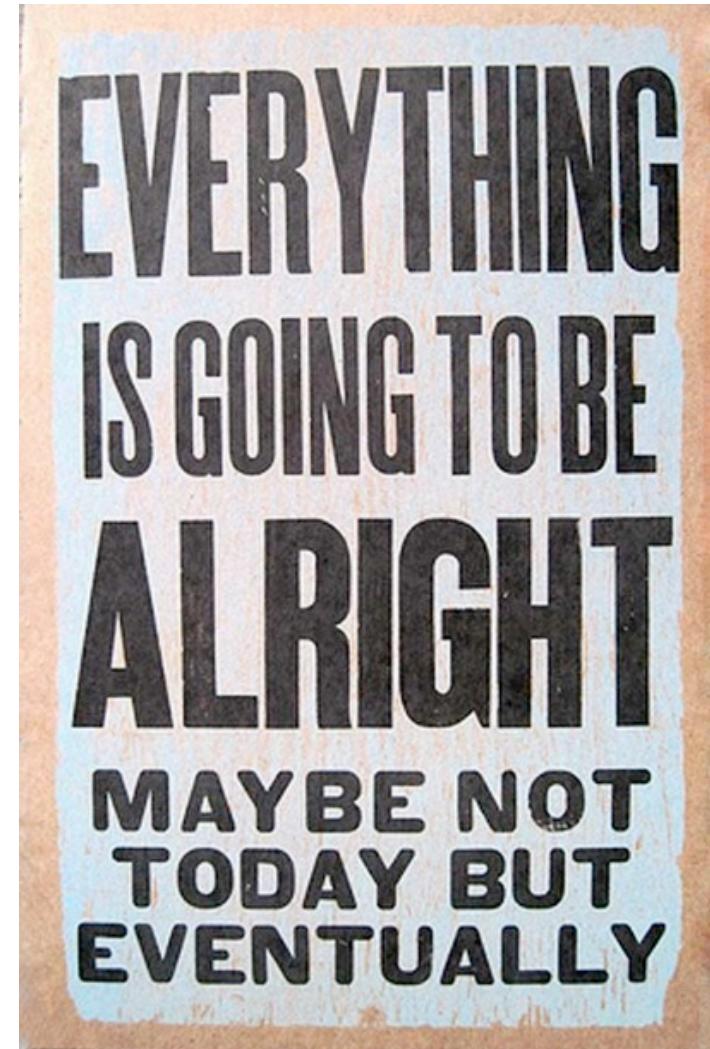
Consistencia no estricta

Consistencia de sesión

Un poco más débil que RYW. Provee ésta consistencia solo si el cliente está dentro de la misma sesión. Usualmente sobre el mismo server.

Consistencia no estricta

Consistencia eventual



Teorema CAP

Consistency → todos ven los mismos datos al mismo tiempo

Availability → si se puede comunicar con un nodo en el cluster, entonces se pueden leer y escribir datos

Partition tolerance → el cluster puede sobrevivir a roturas de comunicación que lo dividan en particiones que no pueden comunicarse entre ellas

Es imposible para cualquier sistema de computación distribuida proveer las tres propiedades simultáneamente.

Las 8 Falacias de la Computación Distribuida

1. La red es confiable
2. La latencia es cero; la latencia no es problema
3. El ancho de banda es infinito
4. La red es segura
5. La topología no cambia
6. Hay uno y solo un administrador
7. El coste de transporte es cero
8. La red es homogénea

Teorema CAP

Las 8 Falacias de la Computación Distribuida

Consistency

Availability

Partition tolerance

Teorema CAP

Las 8 Falacias de la Computación Distribuida

¡Hay que tolerar particiones!

CP - Consistency/Partition Tolerance

AP - Availability/Partition Tolerance

Ejemplo

Las transacciones bancarias son inconsistentes, particularmente para ATMs, que están diseñados para tener un comportamiento en modo normal y otro en modo partición. En modo partición la *Availability* es elegida por sobre la *Consistencia*

Históricamente la industria financiera ha sido inconsistente a causa de la imperfección de las comunicaciones.

ATMs realizan operaciones conmutativas de manera que el orden en el cuales se aplican no importa.

Si hay una partición el cajero puede seguir funcionando; luego al volver a recuperar la partición se envían las operaciones y el saldo final sigue siendo correcto.

Lo que se hace es delimitar y administrar el riesgo. Los ATMs son rentables aún a costa de alguna pérdida.

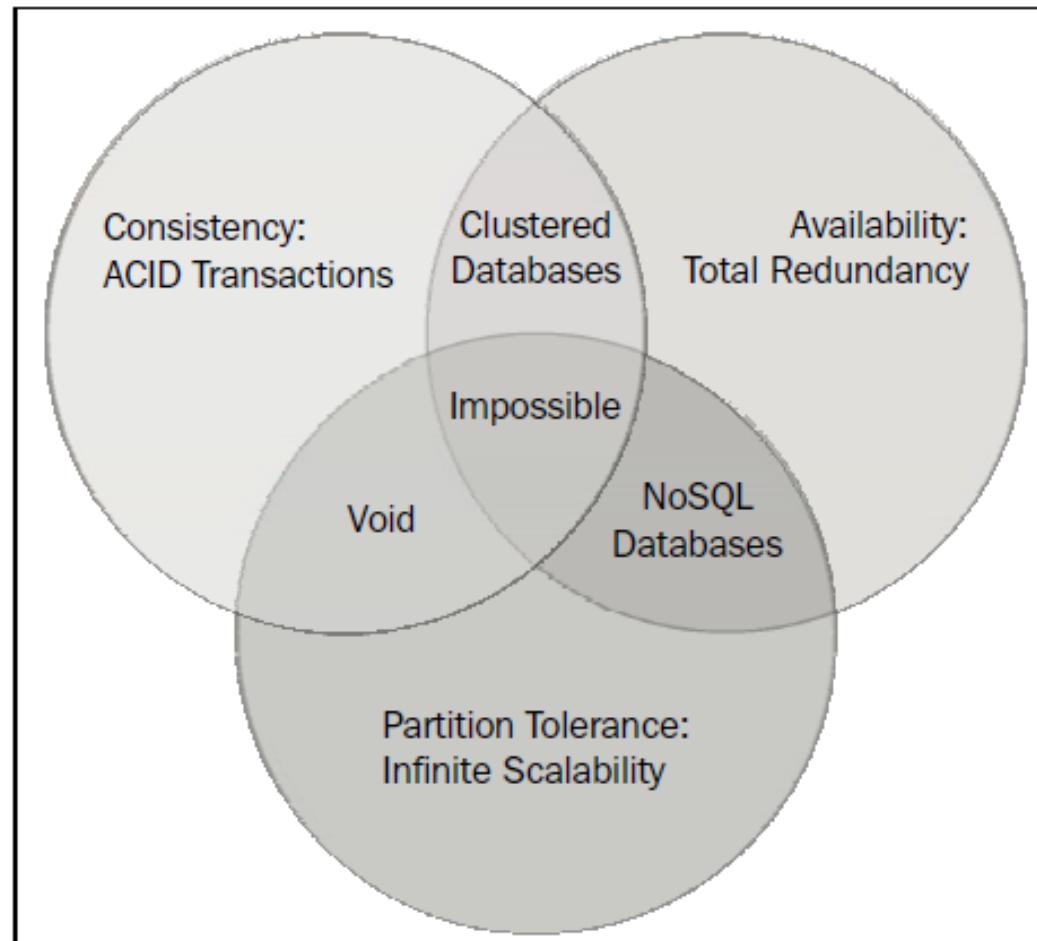
Traditional RDBMS applications have focused on ACID transactions:

- **Atomicity:** Everything in a transaction succeeds lest it is rolled back.
- **Consistency:** A transaction cannot leave the database in an inconsistent state.
- **Isolation:** One transaction cannot interfere with another.
- **Durability:** A completed transaction persists, even after applications restart.

BASE, coined by Eric Brewer:

- **Basic availability**: Each request is guaranteed a response—successful or failed execution.
- **Soft state**: The state of the system may change over time, at times without any input (for eventual consistency).
- **Eventual consistency**: The database may be momentarily inconsistent but will be consistent eventually.

Relationship between CAP, ACID, and NoSQL



| Feature | NoSQL Databases | Relational Databases |
|--------------|-------------------------|---------------------------|
| Performance | High | Low |
| Reliability | Poor | Good |
| Availability | Good | Good |
| Consistency | Poor | Good |
| Data Storage | Optimized for huge data | Medium sized to large |
| Scalability | High | High (but more expensive) |

SQL vs NOSQL vs NEWSQL: FINDING THE RIGHT SOLUTION

OLDSQL

Traditional SQL databases have been around for decades and serve as the core foundation of nearly every application we use today.

THE OLDSQL ADVANTAGES

An established market and ecosystem with vast amounts of standards-based tooling.

THE OLDSQL DISADVANTAGES

They are good for general purpose applications with modest performance requirements, but struggle as needs grow.

NOSQL

SOME NOSQL SYSTEMS PUT AVAILABILITY FIRST

SOME NOSQL SYSTEMS FOCUS ON FLEXIBILITY

SOME NOSQL SYSTEMS FOCUS ON ALTERNATIVE
DATA MODELS OR SPECIAL USE CASES

THE NOSQL ADVANTAGES

Highest availability across multiple data centers

THE NOSQL DISADVANTAGES

These systems are fundamentally not transactional (ACID)

NEWSQL

NewSQL systems all start with the relational data model and the SQL query language, and they all try to address some of the same sorts of scalability, inflexibility or lack-of-focus that has driven the NoSQL movement. Many offer stronger consistency guarantees.

THE NEED FOR SPEED: FAST IN-MEMORY SQL

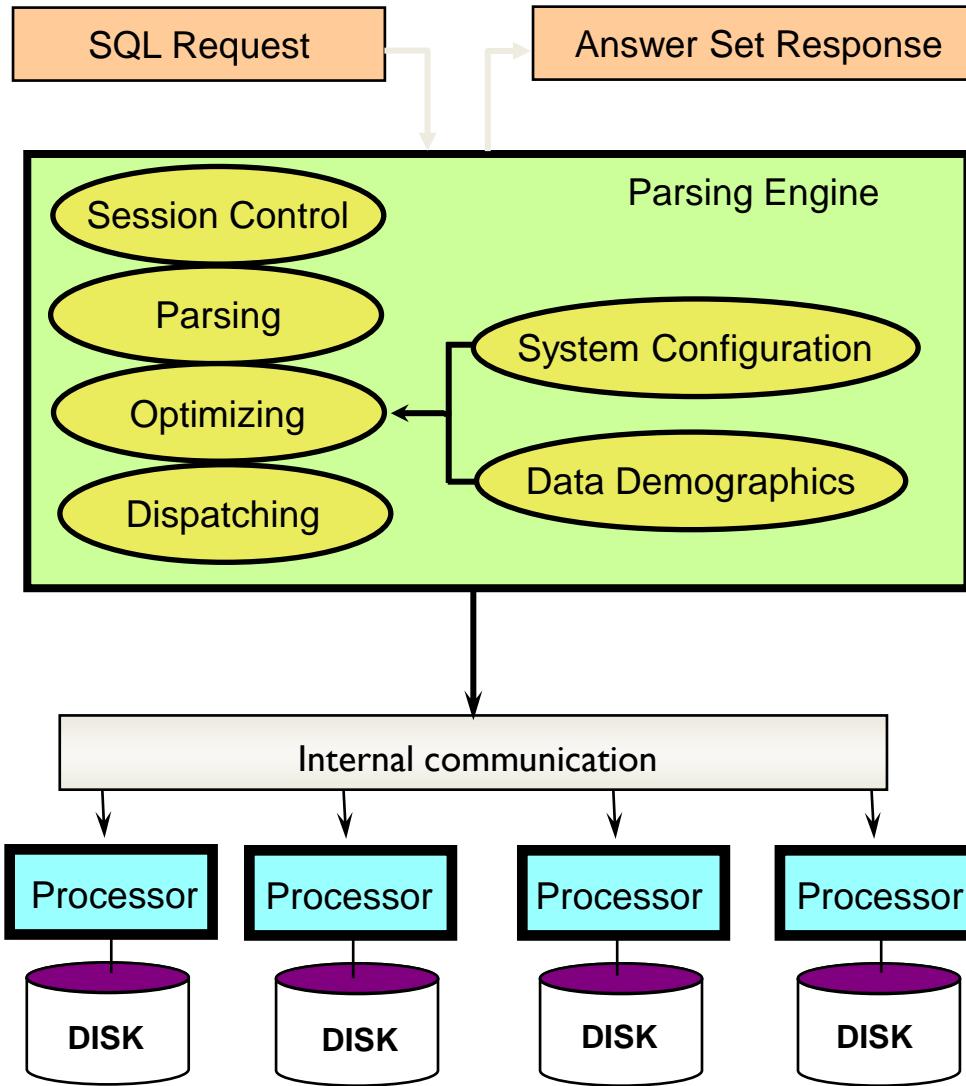
THE NEWSQL ADVANTAGES:

Many systems offer NoSQL-style clustering with more traditional data and query models

THE NEWSQL DISADVANTAGES

Offers only partial access to the rich tooling of traditional SQL systems

Optimización





Chinook2/postgres@BD

Query Editor Query History

```
1 ANALYZE VERBOSE album;
2
3
```

Data Output Explain **Messages** Notifications

INFO: analyzing "public.album"
INFO: "album": scanned 13 of 13 pages, containing 347 live rows and 0 dead rows; 347 rows in sample, 347 estimated total rows
ANALYZE

Query returned successfully in 62 msec.



Chinook2/postgres@BD

Query Editor Query History

```
1 VACUUM VERBOSE album;
2
3
```

Data Output Explain **Messages** Notifications

INFO: vacuuming "public.album"
INFO: "album": found 0 removable, 347 nonremovable row versions in 13 out of 13 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 584
There were 1284 unused item identifiers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
VACUUM

Query returned successfully in 67 msec.



Chinook2/postgres@BD

Query Editor Query History

```
1 VACUUM FULL VERBOSE ANALYZE album;
```

- Chinook2
 - Casts
 - Catalogs (2)
 - ANSI (information_schema)
 - PostgreSQL Catalog (pg_catalog)
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (63)
 - pg_aggregate
 - pg_am
 - pg_amop
 - pg_amproc
 - pg_attrdef
 - pg_attribute

- ▼ Catalogs (2)
 - > ANSI (information_schema)
 - ▼ PostgreSQL Catalog (pg_catalog)
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries (22)
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > Sequences
 - > Tables (63)
 - > Trigger Functions
 - > Types
 - ▼ Views (63)
 - > pg_available_extension_versions
 - > pg_available_extensions
 - > pg_config
 - > pg_cursors
 - > pg_file_settings



Query Editor Query History

```
1 SELECT
2     attname
3     ,array_to_string
4         (most_common_vals, E'\n')
5     as most_common_vals
6 FROM pg_stats
7 WHERE tablename = 'customer';
```

Data Output Explain Messages Notifications

| | attname name | most_common_vals | |
|---|-----------------|------------------|--|
| 1 | customer_id | [null] | |
| 2 | first_name | Frank | |
| 3 | last_name | [null] | |
| 4 | company | [null] | |
| 5 | address | [null] | |
| 6 | city | Berlin | |
| 7 | state | CA | |
| 8 | country | USA | |

Query Query History

```
1 Select
2     relname
3     ,n_live_tup
4     ,last_analyze
5     from pg_stat_user_tables
6     group by 1,2,3
7     order by 2 desc
8
```

Data output Messages Notifications

| | relname name | n_live_tup bigint | last_analyze timestamp with time zone |
|----|-----------------|----------------------|--|
| 1 | playlist_tra... | 8715 | 2023-09-03 23:50:04.4749... |
| 2 | track | 3503 | 2023-09-03 23:50:11.9954... |
| 3 | invoice_line | 2240 | 2023-09-03 23:49:25.1731... |
| 4 | invoice | 412 | 2023-09-03 23:49:16.0912... |
| 5 | album | 347 | 2023-09-03 23:47:53.3312-... |
| 6 | artist | 275 | 2023-09-03 23:48:21.0290... |
| 7 | customer | 59 | 2023-09-03 23:48:42.9285... |
| 8 | genre | 25 | 2023-09-03 23:49:03.9523... |
| 9 | playlist | 18 | 2023-09-03 23:49:54.4686... |
| 10 | employee | 8 | 2023-09-03 23:48:55.0029... |
| 11 | media_type | 5 | 2023-09-03 23:49:43.1133... |



Chinook2/postgres@BD

Query Editor Query History

```
1 SELECT pg_size.pretty(pg_database_size('Chinook2')) AS Chinook2_size;
```

Data Output Explain Messages Notifications

| | chinook2_size | 🔒 |
|---|---------------|---|
| ▲ | text | |

| | |
|---|-------|
| 1 | 11 MB |
|---|-------|



Chinook2/postgres@BD

Query Editor Query History

```
1 SELECT
2     relname AS "table_name",
3     pg_size.pretty(pg_table_size(C.oid)) AS "table_size"
4 FROM
5     pg_class C
6 LEFT JOIN pg_namespace N ON (N.oid = C.relnamespace)
7 WHERE nspname NOT IN ('pg_catalog', 'information_schema')
8         AND relkind IN ('r')
9 ORDER BY pg_table_size(C.oid)
10 DESC LIMIT 5;
```

Data Output Explain Messages Notifications

| | table_name | 🔒 | table_size | 🔒 |
|---|------------|---|------------|---|
| ▲ | name | | text | |

| | | |
|---|-------|--------|
| 1 | track | 384 kB |
|---|-------|--------|

| | | |
|---|----------------|--------|
| 2 | playlist_track | 336 kB |
|---|----------------|--------|

| | | |
|---|--------------|--------|
| 3 | invoice_line | 144 kB |
|---|--------------|--------|

| | | |
|---|---------|-------|
| 4 | invoice | 72 kB |
|---|---------|-------|

| | | |
|---|----------|-------|
| 5 | customer | 40 kB |
|---|----------|-------|

✓ Successfully run.

Organización de Archivos

Una BD se almacena como una colección de archivos

Cada archivo contiene **registros del mismo tipo** y se divide en **bloques de igual tamaño**

Organización de Archivos se refiere a la forma en que los datos son almacenados dentro de un archivo y las formas en que pueden accederse

Dos tipos de archivos clásicos:

HeapFile

SortedFile

M

Open in app

Get started

Sequential search

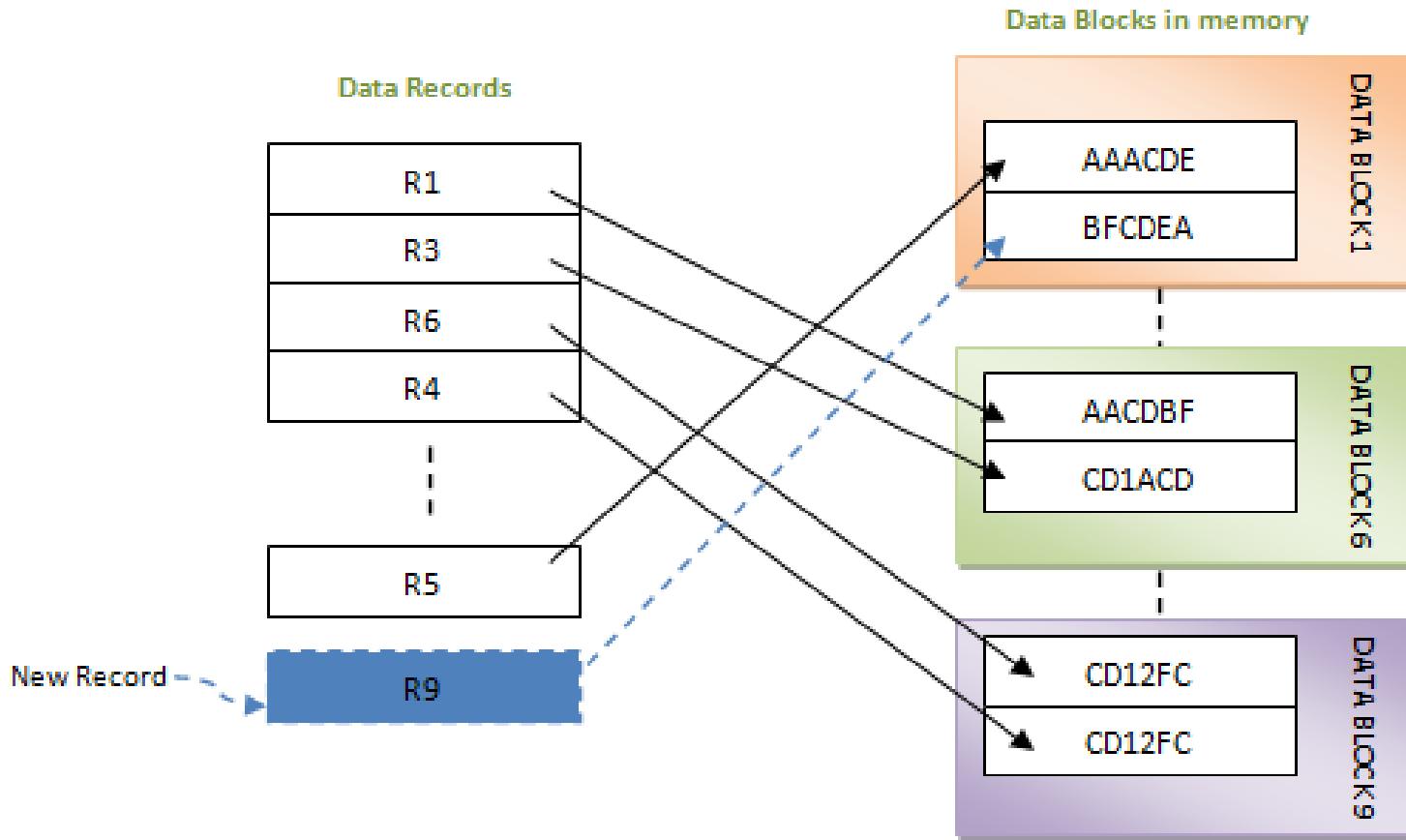
steps: 0



www.penjee.com

Búsqueda lineal

HeapFile



HeapFile

Registros sin orden

Al insertarse un registro, se lo agrega al final del archivo o en alguno de los bloques con espacio libre

Las operaciones de búsqueda requieren búsqueda lineal por **todos los bloques** del archivo

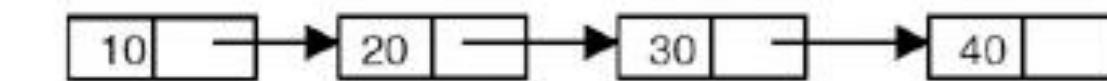
Ventajas:

inserción

Desventajas:

búsqueda y modificación

SortedFile



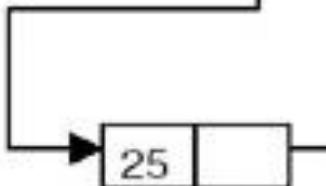
Sorted linked list



node to be inserted

Before insertion

pointer prev



node inserted

After insertion

SortedFile

Registros ordenados a partir de una clave de búsqueda A

Al insertarse un registro se lo agrega ordenadamente, lo que puede provocar una reorganización en los bloques del archivo

Mejoran las búsquedas por A, pero el resto de las operaciones suelen requerir una búsqueda lineal

Índices

Estructuras adicionales: aceleran ciertas operaciones de búsqueda sobre tablas

Mayor costo en operaciones de escritura, actualización y borrado

Mayor costo en espacio ocupado

Dos tipos de índices clásicos son:

B+

Hash

Índices B+

Los índices B+ son árboles balanceados

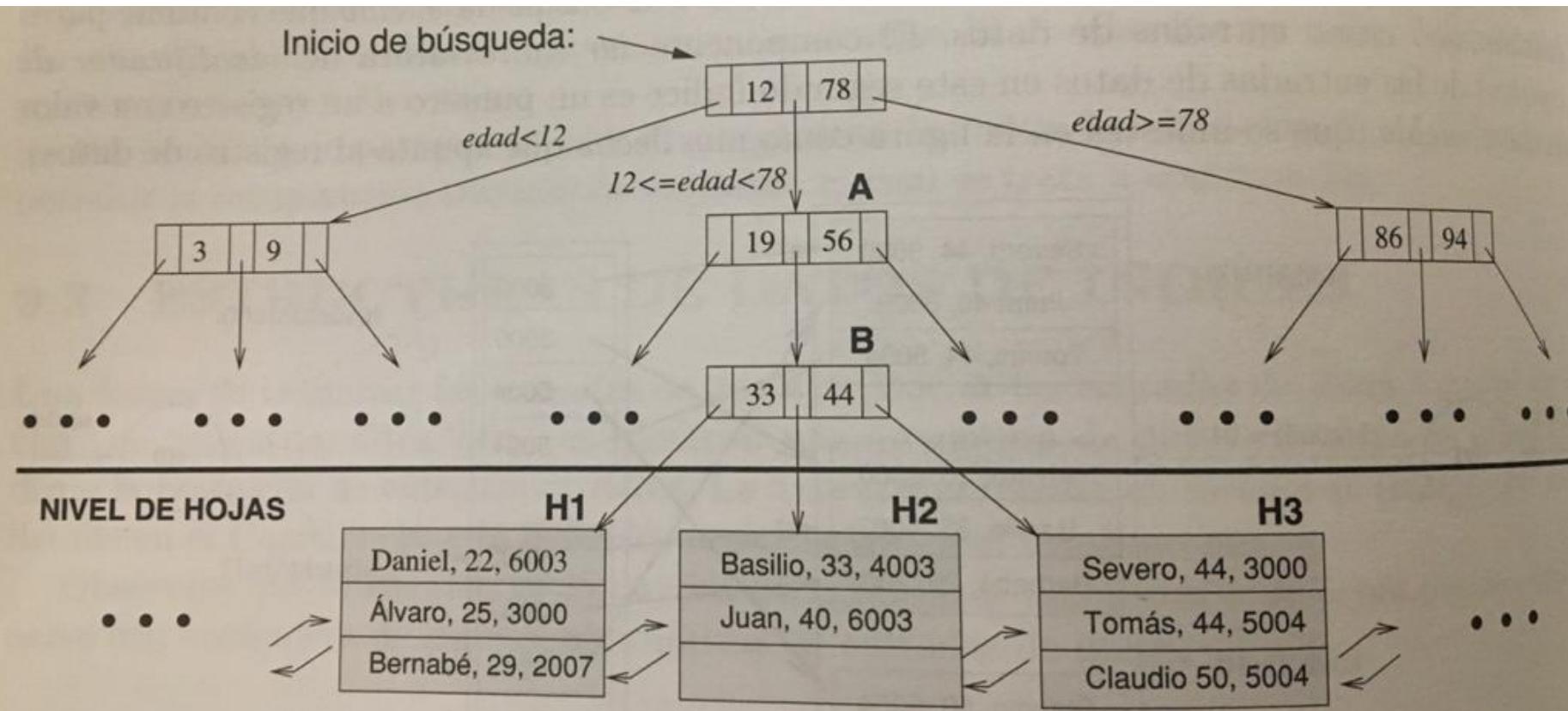


Figura 9.3 Índice estructurado en árbol

Índices B+

Los índices B+ son árboles balanceados

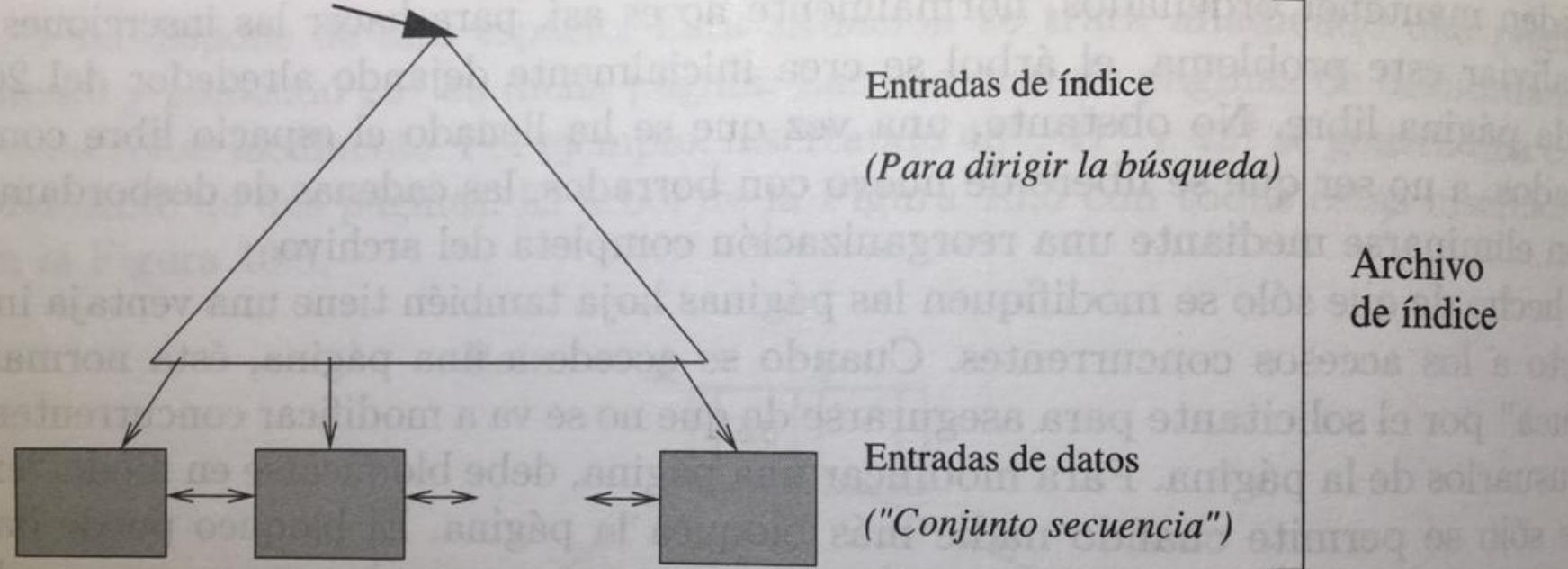
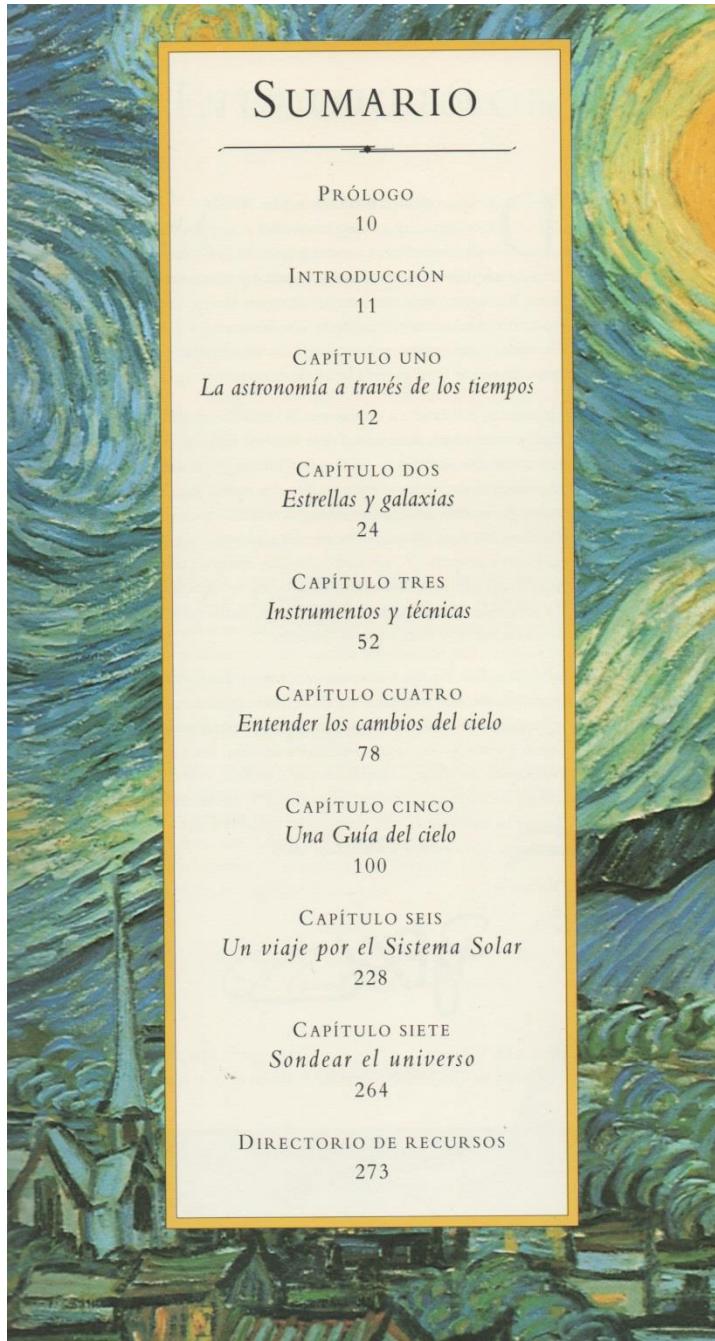


Figura 10.7 Estructura de un árbol B+



SUMARIO

| | |
|--|-----|
| PRÓLOGO | 10 |
| INTRODUCCIÓN | 11 |
| CAPÍTULO UNO <i>La astronomía a través de los tiempos</i> | |
| 12 | |
| CAPÍTULO DOS <i>Estrellas y galaxias</i> | |
| 24 | |
| CAPÍTULO TRES <i>Instrumentos y técnicas</i> | |
| 52 | |
| CAPÍTULO CUATRO <i>Entender los cambios del cielo</i> | |
| 78 | |
| CAPÍTULO CINCO <i>Una Guía del cielo</i> | |
| 100 | |
| CAPÍTULO SEIS <i>Un viaje por el Sistema Solar</i> | |
| 228 | |
| CAPÍTULO SIETE <i>Sondear el universo</i> | |
| 264 | |
| DIRECTORIO DE RECURSOS | 273 |

Directorio de recursos

ÍNDICE y GLOSARIO

En esta combinación de índice y glosario, los números de página en negrita indican la referencia principal y las cursivas señalan las ilustraciones y las fotografías.

A

- AAVSO 74
Abell, George 51
Achernar 103, 106, 121, 127, 129
Acrux 106, 163
Adams, John Couch 255, 255
adaptación a la oscuridad Proceso por el cual el ojo humano aumenta la sensibilidad en condiciones de poca (o ninguna) iluminación. 58
adaptadores de cámara 67
Adhara 106
Águila, Nebulosa del 212, 212
agujero negro Objeto macizo tan denso que no permite salir luz ni radiación alguna. 35, 35, 47, 267

Alpha (α) Tauri véase Aldebaran

Alpha (α) Ursae Minoris véase Polaris

Alpha (α) Virginis véase Spica

Altair 106, 112, 114, 115, 124, 126, 137, 137, 164

altitud 63, 63, 81, 81

Ames, Adelaide 51

Andrómeda 29, 51, 108, 116, 132-3, 132, 133

Andrómeda, Galaxia 28, 46, 48, 49, 51, 133, 133

Anillo, Nebulosa del 43, 45, 186, 186

Anillo, Cola de, Galaxia 162

anillos véase planetas

Antares 30, 91, 106, 183, 208, 209

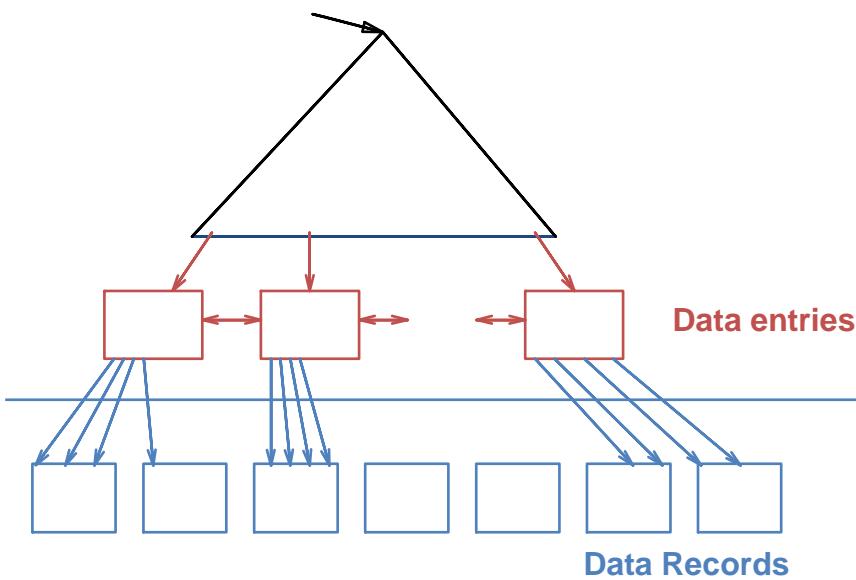
golpean la atmósfera de la Tierra y hacen brillar algunos de sus gases. 54, 98, 98

azimut 63, 63, 81, 81

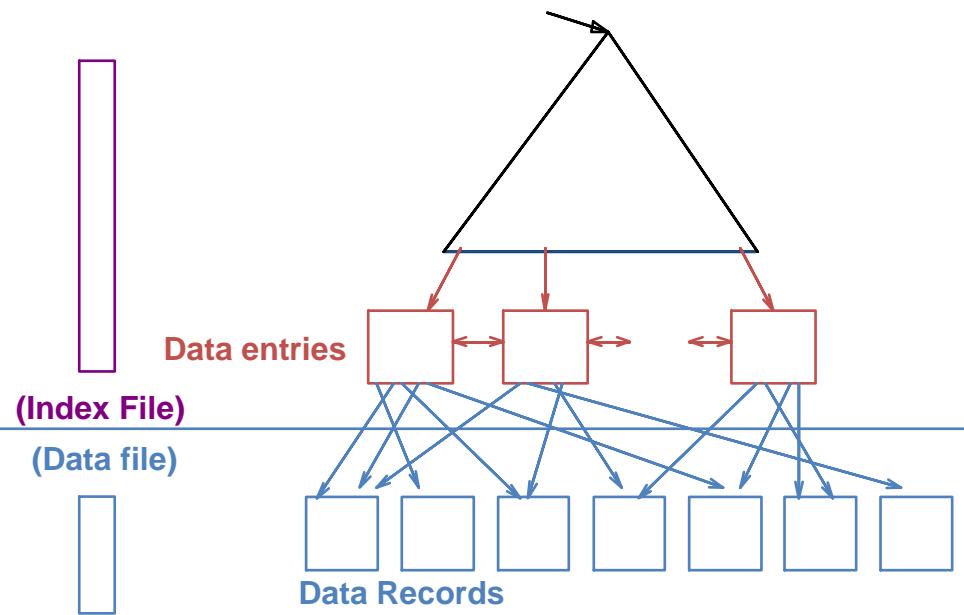
B

- Baade, Walter** 23
Barnard, E. E. 193
Barnard 33 véase Cabeza de Caballo, Nebulosa
Barnard, Estrella de 27, 30, 91, 193
Bartsch, Jakob 189
Bayer, Johann 88-89, 153, 156, 167, 173, 178, 179, 190, 196, 218, 219, 226
Bellatrix 106
Berenice, Cabellera de véase Coma Berenices
Bessel, Friedrich 147
Beta (β) Arae 138

Clustered vs. Unclustered Index

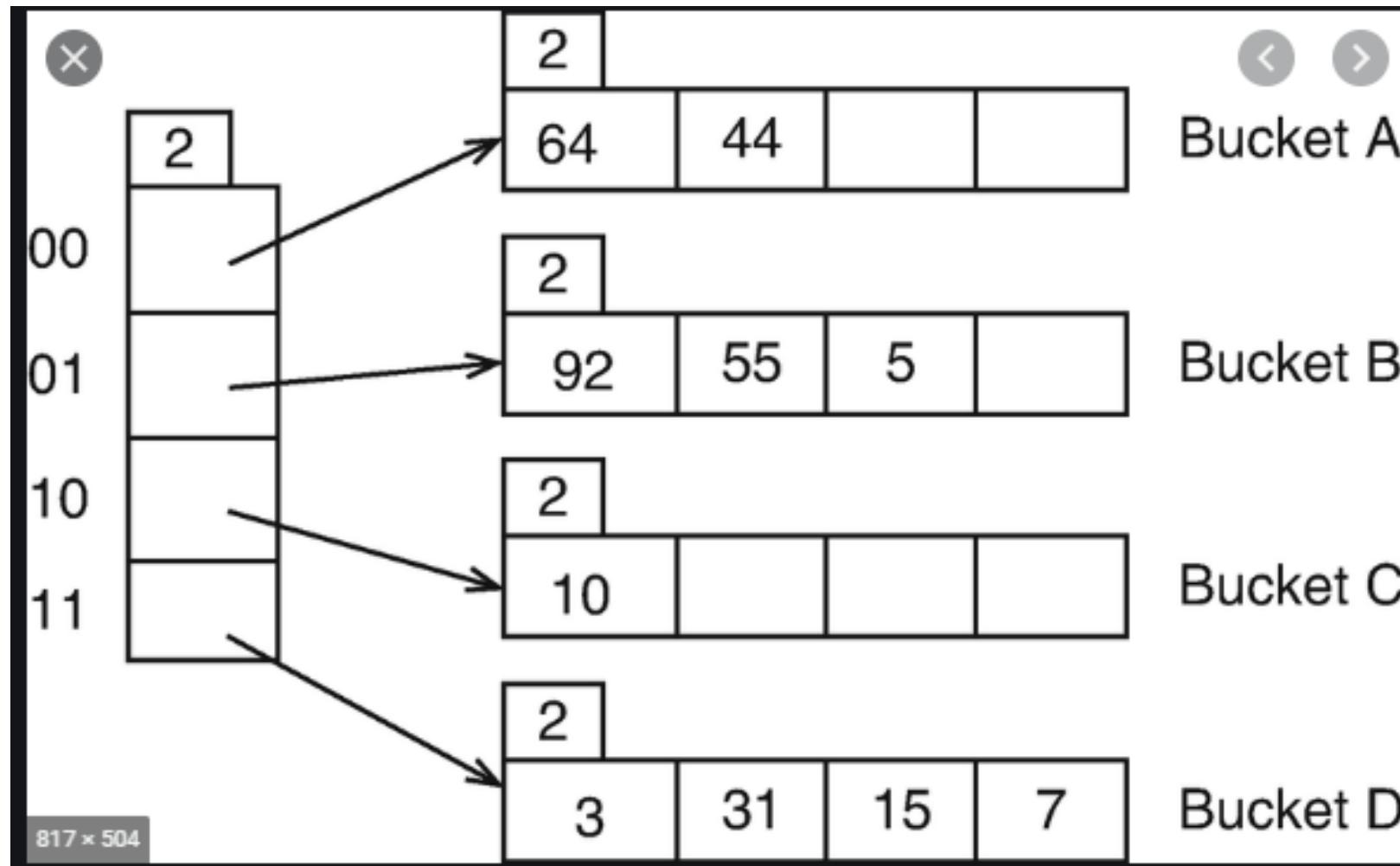


CLUSTERED



UNCLUSTERED

Índices Hash



Índices Hash

The screenshot shows the Correo Argentino website's CPA search interface. The search form has been filled with 'Ciudad Autónoma de Buenos Aires' as the province and 'Ciudad Autónoma Buenos Aires' as the locality. In the address fields, 'Calle Tinogasta' is entered in the 'Calle' field and '2950' in the 'Altura' field. A CAPTCHA 'KK2ADA' is present. The search button is labeled 'Consultar'. Below the form, the results are displayed as 'CALLE TINOGASTA 2950 C1417EHL'.

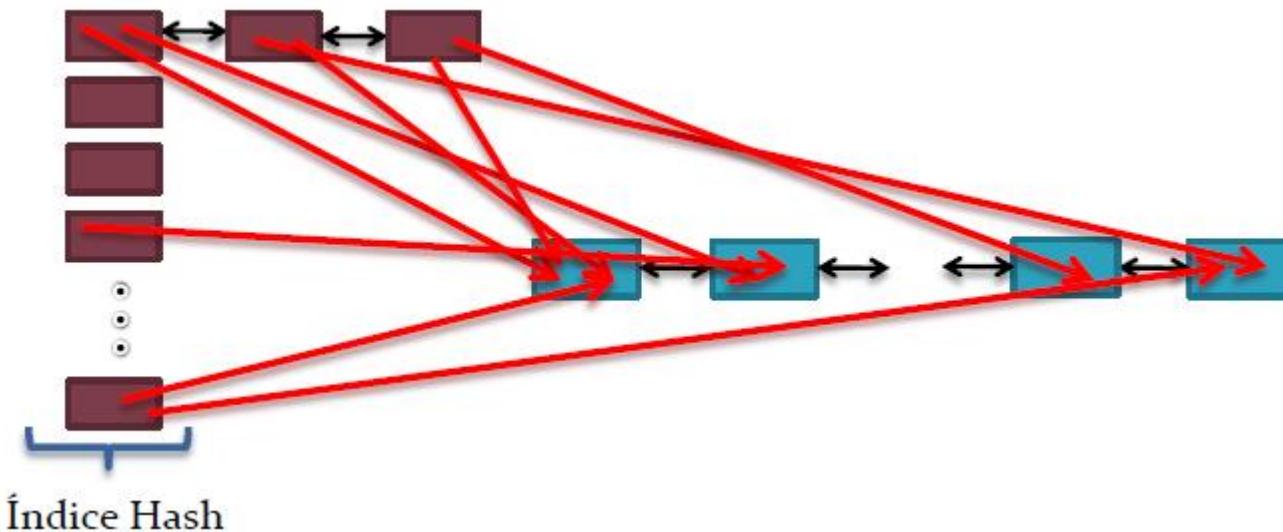
This screenshot shows the same CPA search interface as the previous one, but with an incorrect CAPTCHA entry ('BOGS4Y'). The search results are now displayed as 'CALLE TINOGASTA 3000 C1417EHL', indicating a mismatch between the input address and the search results.

Índices Hash

Una tabla de hash almacena las claves de búsqueda

Cada posición de una tabla hash se asocia con un conjunto de registros. Por esta razón cada posición suele llamarse “un bucket”, y los valores de hash, “índices bucket”.

En cada bucket → **cantidad variable de bloques**





Chinook2/postgres@BD

Query Editor Query History

```
1 -- Table: public.album
2
3 CREATE INDEX ifk_album_artist_id           ← nombre del índice
4
5   ON public.album                         ← para qué tabla se crea
6
7   USING btree                           ← tipo de índice
8
9   (artist_id ASC NULLS LAST)            ← campo; orden asc; nulls al final
10
11 TABLESPACE pg_default;                ← en qué tabla se crea
```



24th June 2021: PostgreSQL 14 Beta 2 Released!

[Documentation → PostgreSQL 8.3](#)Supported Versions: [Current \(13\)](#) / [12](#) / [11](#) / [10](#) / [9.6](#)Development Versions: [14](#) / [devel](#)Unsupported versions: [9.5](#) / [9.4](#) / [9.3](#) / [9.2](#) / [9.1](#) / [9.0](#) / [8.4](#) / [8.3](#) / [8.2](#) / [8.1](#) / [8.0](#) / [7.4](#) / [7.3](#) / [7.2](#)

Search the documentation for...



This documentation is for an unsupported version of PostgreSQL.

You may want to view the same page for the [current](#) version, or one of the other supported versions listed above instead.

PostgreSQL 8.3.23 Documentation

[Prev](#)[Fast Backward](#)

Chapter 11. Indexes

[Fast Forward](#)[Next](#)

11.2. Index Types

PostgreSQL provides several index types: B-tree, Hash, GiST and GIN. Each index type uses a different algorithm that is best suited to different types of queries. By default, the `CREATE INDEX` command will create a B-tree index, which fits the most common situations.

B-trees can handle equality and range queries on data that can be sorted into some ordering. In particular, the PostgreSQL query planner will consider using a B-tree index whenever an indexed column is involved in a comparison using one of these operators:

<
<=
=
>=
>

Hash indexes can only handle simple equality comparisons.
The [query planner](#) will consider using a hash index whenever an indexed column is involved in a comparison using the = operator.

B+

vs

Hash

Un BTree es un árbol ordenado. Un hash es un mapeo de un conjunto a otro.

Ejemplo de función de hash:

tomar un número natural y hacer una suma iterativa de dígitos

hash (179) -> 17 -> 8

necesito 9 posiciones para mi índice

¿Dónde está el 179? -> buscar el índice 8, y recorrer todos los buckets

¿si quiero los números ordenados? Esta función de hash no ayuda.

179 278 197 ocupan buckets en el mismo índice

Con Btree puedo recorrer el árbol en orden de prefijo y obtener una clasificación, en orden de postfijo y obtener la clasificación inversa.

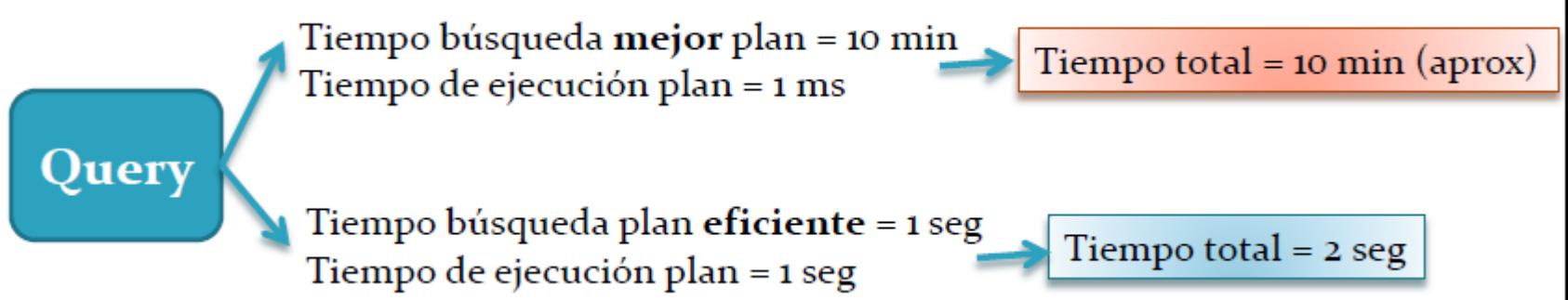
La única ventaja de Hash es la velocidad de acceso. Pero eso depende de las colisiones.

Optimizador de consultas

Dada una consulta, encontrar un plan de ejecución eficiente

Cuando un usuario formula una consulta, se analiza y envía esta consulta a un **optimizador de consultas**, que utiliza información sobre el modo que se guardan los datos para producir un **plan de ejecución** eficiente para la evaluación de esa consulta. Un **plan de ejecución** es un detalle de las acciones que debe realizar el motor para la evaluación de la consulta, representado habitualmente como un árbol de operadores con anotaciones que contiene información detallada adicional sobre los métodos de acceso que se deben emplear, etc.

Plan de ejecución → algoritmos y estructuras



¿Plan eficiente o mejor plan?

Analizar todo el espacio de búsqueda → costoso

Evitar analizarlo completo, pero obtener planes eficientes

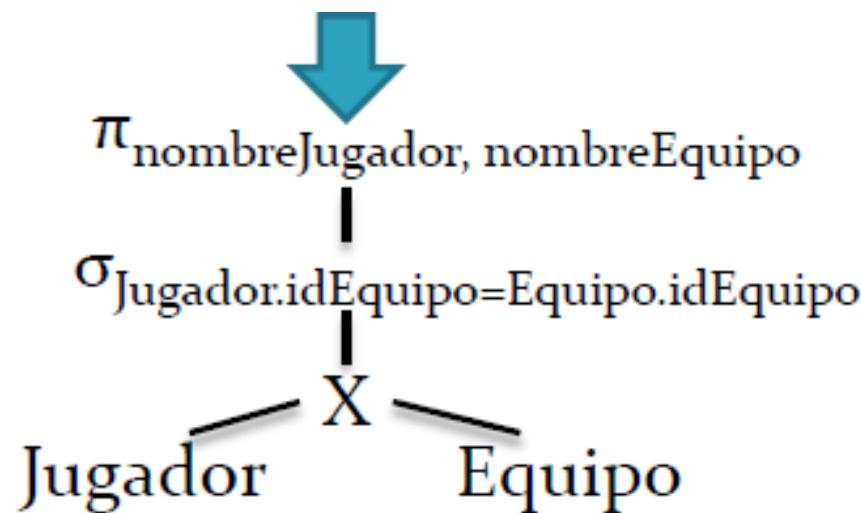
Primer paso

SQL



Árbol Canónico

```
SELECT nombreJugador, nombreEquipo  
FROM Jugador J, Equipo E  
WHERE J.idEquipo = E.idEquipo
```



Segundo paso

Modificaciones sobre el árbol

Buscan mejorar la performance de la consulta **independientemente** de la organización física.

Involucran propiedades que permiten construir una consulta **equivalente** a la original.

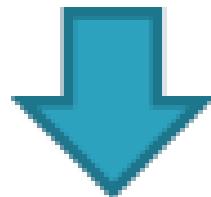
Técnicas Algebraicas

Heurísticas

Basadas en propiedades **algebraicas**

Árboles **equivalentes**

Por lo general, mejoran la performance de las consultas

 $\pi_{\text{nombreJugador}, \text{nombreEquipo}}$

|

 $\sigma_{\text{Jugador.idEquipo} = \text{Equipo.idEquipo}}$

|

X

Jugador Equipo

A diagram showing two entities, Jugador and Equipo, connected by a relationship labeled X. The Jugador entity is on the left, and the Equipo entity is on the right. A large blue downward-pointing arrow is positioned above the entities, indicating a projection operation.

Técnicas Físicas

Seleccionar implementaciones para los operadores basándose en **cómo** están organizados los archivos y las estructuras adicionales que existen

Utilizan el **Catálogo**

Información estadística de los datos

Se actualiza periódicamente y no está siempre sincronizado con los datos reales

Permite estimar la selectividad de los diferentes operadores

¿Cómo comparar planes?

- Se define un **modelo de costos**
- El costo será expresado en **cantidad de accesos a disco** (lecturas + escrituras)
 - Predomina sobre tiempo de CPU
- El costo de un plan será una estimación
- Se elegirá al plan con menor costo estimado

- Detalles:
 - No asumiremos nada sobre el *Buffer Manager*, por lo que siempre consideraremos que un pedido de lectura o escritura significa acceder a disco
 - En un bloque de R, hay sólo tuplas de R (y no de otras relaciones)
 - Las tuplas de R se guardan enteras en un bloque (por ejemplo, no puede haber mitad en el bloque i y la otra mitad en el bloque i+1)
 - Siempre asumiremos *peor caso*

¿Cómo se pasan los resultados entre nodos?

Materialización

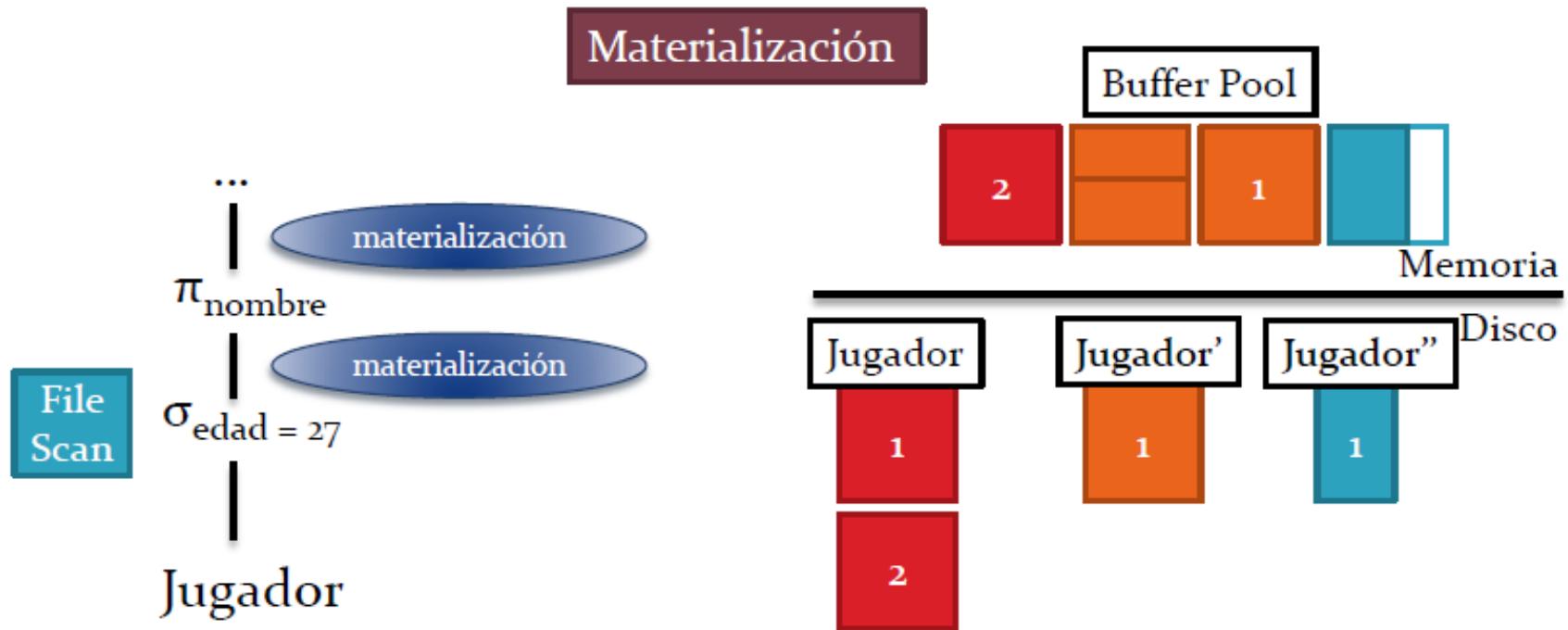
Los resultados intermedios se guardan **directamente en disco**.

El siguiente paso de la consulta deberá levantarlos de disco **nuevamente**.

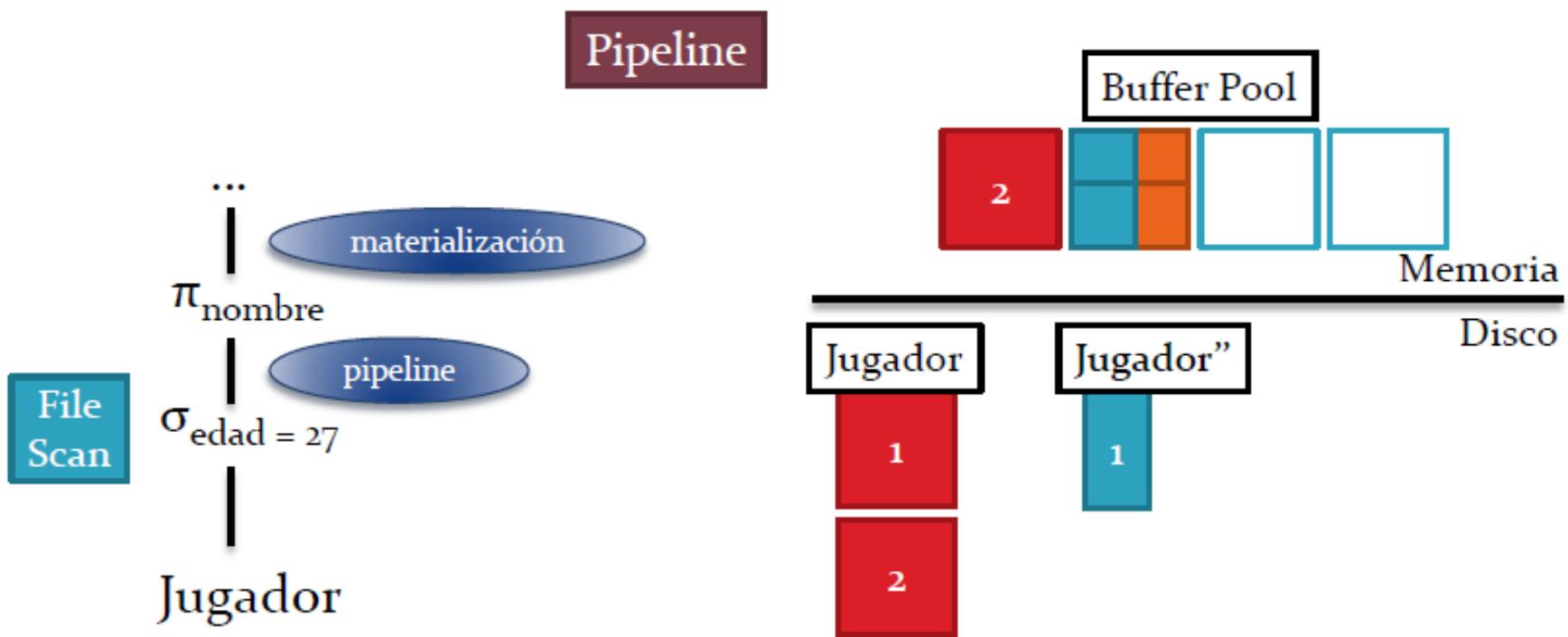
Pipeline

Las tuplas se van pasando al nodo superior del árbol **mientras** se continúa ejecutando la operación

Materialización



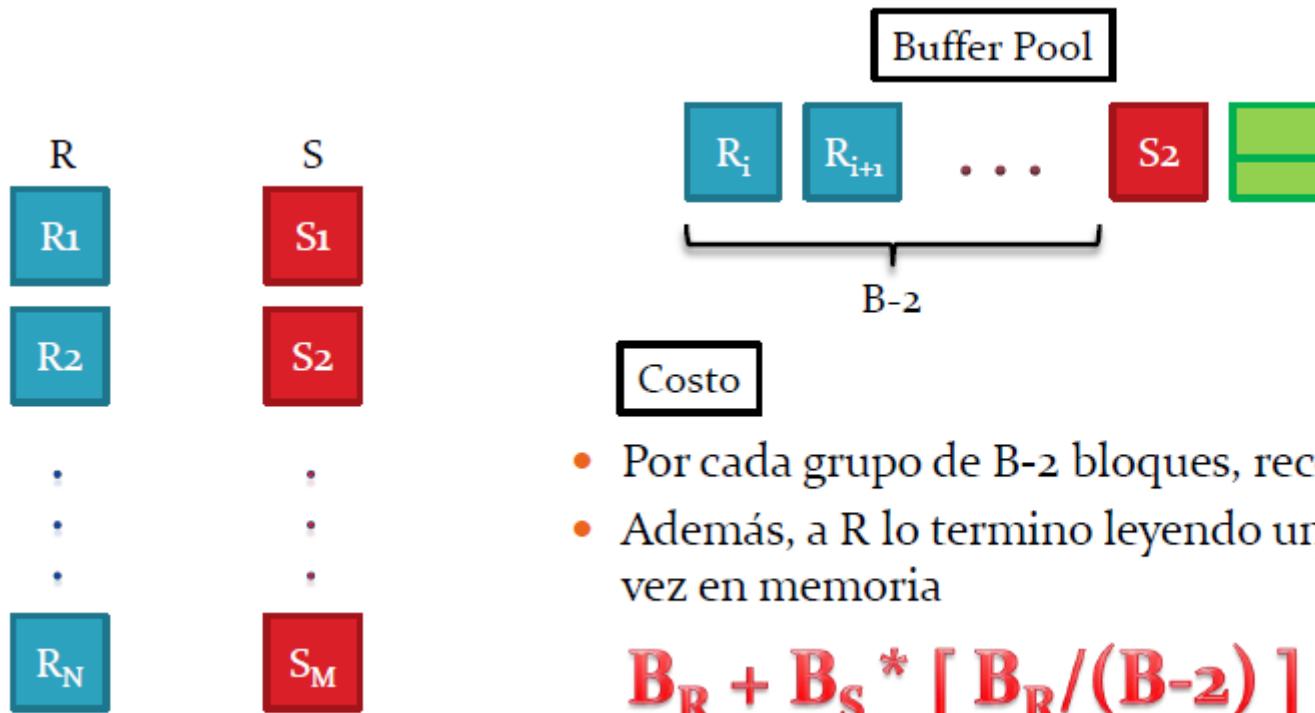
Pipeline



R \bowtie S

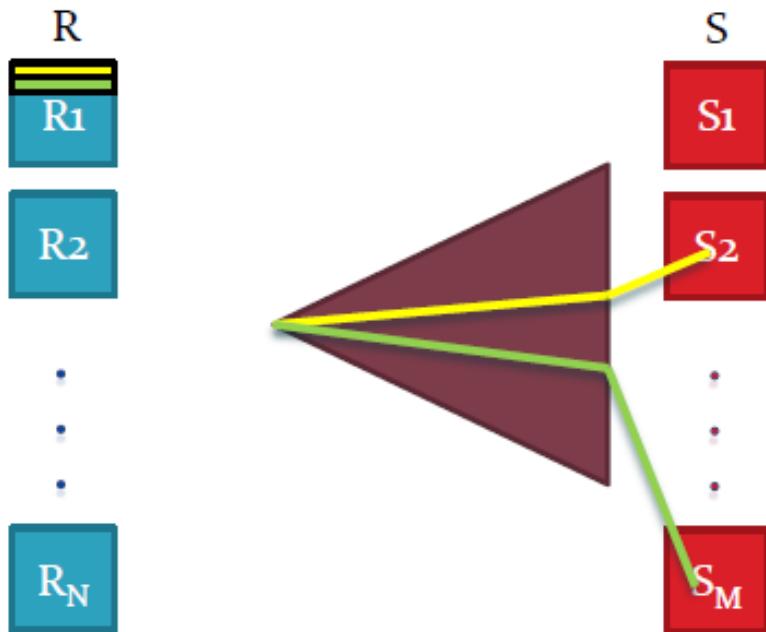
Block Nested Loops Join (BNLJ)

B bloques de memoria



Index Nested Loops Join (INLJ)

Índice I sobre S



Costo

- Por cada tupla de R, hago una búsqueda en el índice I
- Además, a R lo termino recorriendo una sola vez en memoria

$$B_R + T_R * \text{("costo índice")}$$



Dependerá del tipo de índice que exista

Sort Merge Join (SMJ)

B bloques de memoria

| R | | S | |
|------|-----|-----|-----|
| 1 | A | 1 | x |
| 2 | B | 1 | y |
| ... | ... | ... | ... |
| 1000 | ABC | 100 | xyz |

Costo

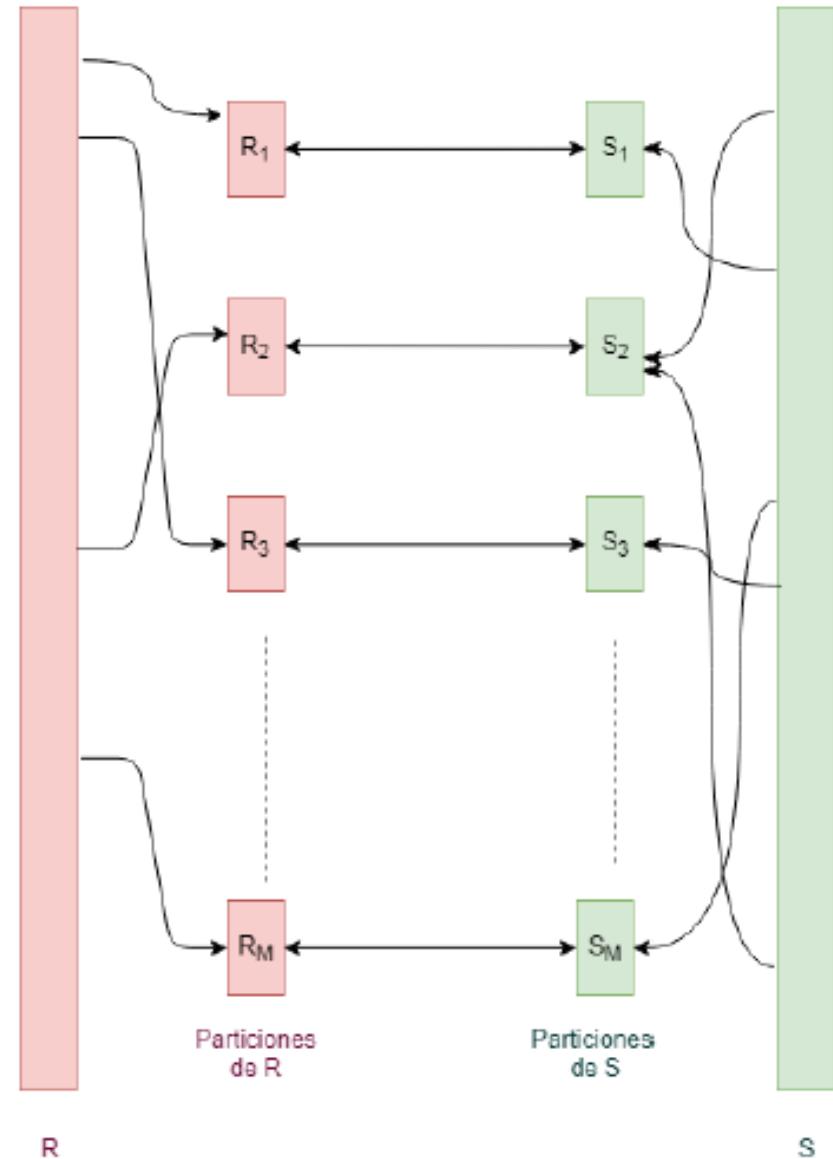
- Se ordenan R y S (si alguno ya está ordenado, este costo no se considera)
- Se hace el *merge* entre R y S ordenados

$$(\lceil \log_{B-1} [B_R/B] \rceil + 1) * 2B_R + (\lceil \log_{B-1} [B_S/B] \rceil + 1) * 2B_S + B_R + B_S$$

Hash Join (HJ)

- Las tuplas de S1 se comparan solamente con las de R1
- Se necesitan, para M particiones, M + 1 bloques en memoria

$$CI = 3 * (B_R + B_S)$$

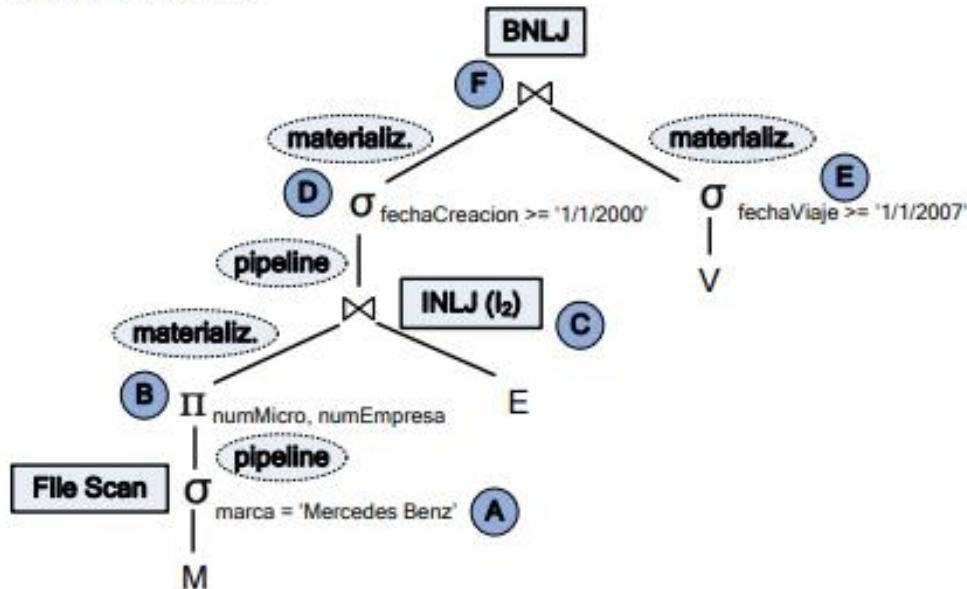


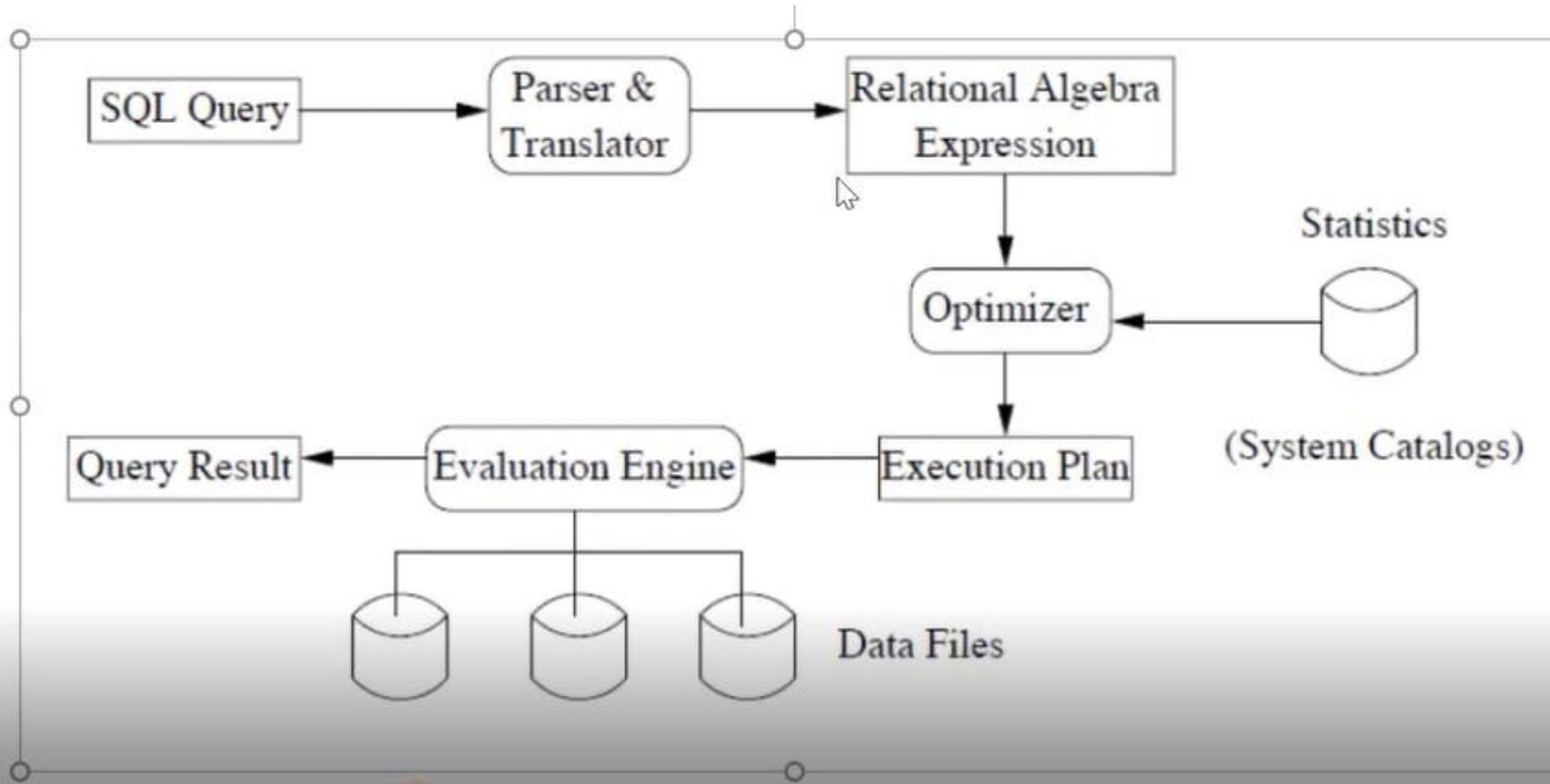
```

1  SELECT
2      numMicro
3          , numEmpresa
4          , nombreEmpresa
5          , fechaCreacion
6          , direccion
7          , destino
8          , fechaViaje
9
10     FROM
11         Viaje V
12         , Empresa E
13         , Micro M
14
15     WHERE
16         M.numMicro = V.numMicro
17         AND M.numEmpresa = E.numEmpresa
18         AND E.fechaCreacion >= '1/1/2000'
19         AND V.fechaViaje >= '1/1/2007'
20         AND V.fechaViaje < '1/2/2007'
21         AND M.marca = 'Mercedes Benz'

```

Calcular los costos:





Document Databases

Definición

Document Database

Es una base no-relacional que almacena los datos como documentos estructurados.

El concepto principal es el **documento**

- Las BD almacena y recupera documentos.
- Los documentos pueden ser XML, **JSON**, BSON, etc

Definición

Document Database

Es una base no-relacional que almacena los datos como documentos estructurados.

El concepto principal es el **documento**

- Las BD almacena y recupera documentos.
- Los documentos pueden ser XML, **JSON**, BSON, etc

Documento

Es una colección de pares: nombre de campo y valor. Los valores pueden ser un valor simple o una estructura compleja como listas, otro documento o listas de documentos hijos

Definición

Document Database

Es una base no-relacional que almacena los datos como documentos estructurados.

El concepto principal es el **documento**

- Las BD almacena y recupera documentos.
- Los documentos pueden ser XML, **JSON**, BSON, etc

Documento

Es una colección de pares: nombre de campo y valor. Los valores pueden ser un valor simple o una estructura compleja como listas, otro documento o listas de documentos hijos

Ejemplos

MongoDB, RavenDB, eXist, CouchDB, CouchBase, ArangoDB

XML vs JSON

```
<order id="1234">
<customer id="52">Adam Fowler</customer>
<items>
<item qty="2" id="456" unit_price="2.00" price="4.00">Hammer</item>
<item qty="1" id="111" unit_price="0.79" price="0.79">Hammer Time</item>
</items>
<delivery_address lon="-43.24" lat="54.12">
<street>Some Place</street>
<town>My City</town>
...
</delivery_address>
</order>
```

```
{
"orderid": 1234,
"Customer": {"id":52, "Nombre": "Jhon Doe"},
"items": [ {"qty": 2, "id":456, "unit_price": 2, "price":4},
          {"qty": 1, "id":111, "unit_price": 0.79, "price":0.79}],
"delivery_address": {"lon": -43.24, "lat":54.12, "street": "Some Place", "ciudad": "My City"}
}
```

Metodología

A Big Data Modeling Methodology for NoSQL Document Databases

Gerardo ROSSEL, Andrea MANNA

Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

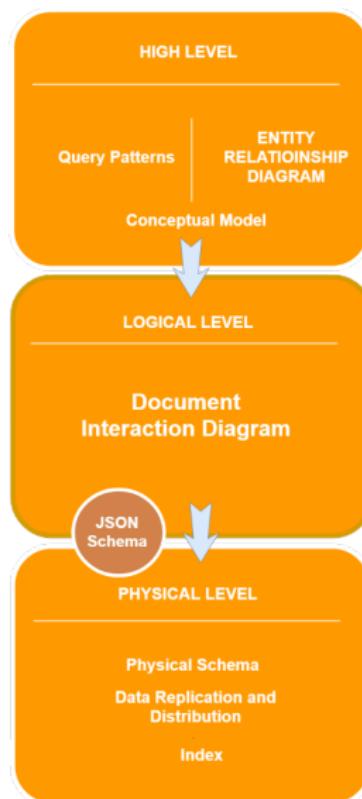
Departamento de Computación. Buenos Aires, Argentina

grossel@dc.uba.ar, amanna@dc.uba.ar

In recent years, there has been an increasing interest in the field of non-relational databases. However, far too little attention has been paid to design methodology. Key-value datastores are an important component of a class of non-relational technologies that are grouped under the name of NoSQL databases. The aim of this paper is to propose a design methodology for this type of database that allows overcoming the limitations of the traditional techniques. The proposed methodology leads to a clean design that also allows for better data management and consistency.

Keywords: NoSQL, Document Databases, Conceptual Modeling, Data Modeling, NoSQL Database developing.

Modelización



Consideraciones de Diseño

Desnormalización

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product_ID: 3648  
}  
  
{  
    product_ID: 3648,  
    product_description: "1 package laser printer paper.  
    100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : "office supplies",  
    list_price : 9.00  
}
```

Desnormalización

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product_ID: 3648  
}  
  
{  
    product_ID: 3648,  
    product_description: "1 package laser printer paper.  
        100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : "office supplies",  
    list_price : 9.00  
}
```

Desnormalizacion

Desnormalización

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product_ID: 3648  
}
```

```
{  
    product_ID: 3648,  
    product_description: "1 package laser printer paper.  
        100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : "office supplies",  
    list_price : 9.00  
}
```

Desnormalizacion



Desnormalización

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product_ID: 3648  
}
```

```
{  
    product_ID: 3648,  
    product_description: "1 package laser printer paper.  
        100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : "office supplies",  
    list_price : 9.00  
}
```

Desnormalizacion

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product :  
    {  
        product_description: "1 package laser printer  
            paper. 100% recycled.",  
        product_name : "Eco-friendly Printer Paper",  
        product_category : "office supplies",  
        list_price : 9.00  
    }  
}
```

Desnormalización

¿Cuanta desnormalización es demasiada?

- Generar facturas y remitos para los clientes (95 %)
- Generar reportes para la gerencia (5 %)

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product :  
        {  
            product_description: "1 package laser printer  
                paper. 100% recycled.",  
            product_name : "Eco-friendly Printer Paper",  
            product_category : "office supplies",  
            list_price : 9.00  
        }  
}
```

```
{  
    product_description: "1 package laser printer paper.  
        100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : 'office supplies',  
    list_price : 9.00  
}
```

Desnormalización

¿Cuanta desnormalización es demasiada?

- Generar facturas y remitos para los clientes (95 %)
- Generar reportes para la gerencia (5 %)

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product :  
        {  
            product_description: "1 package laser printer  
                paper. 100% recycled.",  
            product_name : "Eco-friendly Printer Paper",  
            product_category : "office supplies",  
            list_price : 9.00  
        }  
}
```

```
{  
    product_description: "1 package laser printer paper.  
        100% recycled.",  
    product_name : "Eco-friendly Printer Paper",  
    product_category : 'office supplies',  
    list_price : 9.00  
}
```

```
{  
    order_item_ID : 834838,  
    order_ID: 8827,  
    quantity: 3,  
    cost_per_unit: 8.50,  
    product_name : "Eco-friendly Printer Paper"  
}
```

Diseño Físico

Documentos mutables

```
{  
    truck_id: 'T87V12',  
    time: '08:10:00',  
    date : '27-May-2015',  
    driver_name: 'Jane Washington',  
    fuel_consumption_rate: '14.8 mpg',  
    ...  
}
```

Diseño Físico

Documentos mutables

```
{  
  truck_id: 'T87V12',  
  time: '08:10:00',  
  date : '27-May-2015',  
  driver_name: 'Jane Washington',  
  fuel_consumption_rate: '14.8 mpg',  
  ...  
}
```



```
{  
  truck_id: 'T87V12',  
  date : '27-May-2015',  
  driver_name: 'Jane Washington',  
  operational_data:  
  [  
    {time : '00:01',  
     fuel_consumption_rate: '14.8 mpg',  
     ...},  
    {time : '00:04',  
     fuel_consumption_rate: '12.2 mpg',  
     ...},  
    {time : '00:07',  
     fuel_consumption_rate: '15.1 mpg',  
     ...},  
    ...]  
}
```

Diseño Físico

Documentos mutables

```
{  
    truck_id: 'T87V12',  
    time: '08:10:00',  
    date : '27-May-2015',  
    driver_name: 'Jane Washington',  
    fuel_consumption_rate: '14.8 mpg',  
    ...  
}
```

```
{  
    truck_id: 'T87V12',  
    date : '27-May-2015',  
    driver_name: 'Jane Washington',  
    operational_data:  
        [  
            {time : '00:01',  
             fuel_consumption_rate: '14.8 mpg',  
             ...},  
            {time : '00:04',  
             fuel_consumption_rate: '12.2 mpg',  
             ...},  
            {time : '00:07',  
             fuel_consumption_rate: '15.1 mpg',  
             ...},  
            ...  
        ]  
}
```

```
{truck_id: 'T8V12'  
date: '27-May-2015'  
operational_data:  
    [(time: '00 : 00',  
     fuel_consumption_rate: 0.0)  
     {time: '00 : 00',  
      fuel_consumption_rate: 0.0}  
     .  
     .  
     .  
     {time: '00 : 00',  
      fuel_consumption_rate: 0.0}]
```

200 Embedded
Documents with
Default Values

Considerar el ciclo de vida

Modelo Conceptual -> DID -> Documentos

- DER - Modelo conceptual de alto nivel.
- DID (Modelo/Diagrama de Interrelación de Documentos).
- JSON Schema: especificación de la estructura de los documentos.

¿Cómo resolvemos la interrelación entre documentos?

- DER - Modelo conceptual de alto nivel.
- DID (Modelo/Diagrama de Interrelación de Documentos).
- JSON Schema: especificación de la estructura de los documentos.

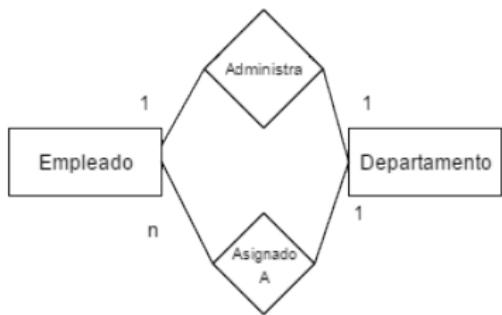
¿Cómo resolvemos la interrelación entre documentos?

Incrustar o Referenciar

La decisión más importante es si incrustar o referenciar, lo que determinará el grado de desnormalización de los documentos

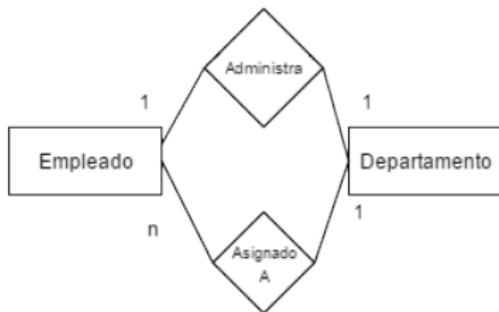
Cardinalidad 1 a N / 1 a 1

Se omiten los atributos por razones didácticas



Cardinalidad 1 a N / 1 a 1

Se omiten los atributos por razones didácticas



- Incrustar el departamento en el empleado
- Incrustar los empleados en el departamento
- Referenciar los empleados e incrustar el departamento en empleado.
- Referenciar de ambos lados
- Incrustar de ambos lados
- etc, etc...

Cardinalidad 1 a N / 1 a 1

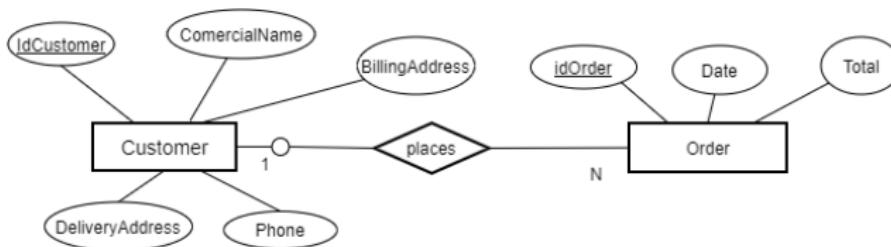
¿Que es Referenciar?

En un documento se hace referencia a un ID o una lista de ID de otro documento

Cardinalidad 1 a N / 1 a 1

¿Qué es Referenciar?

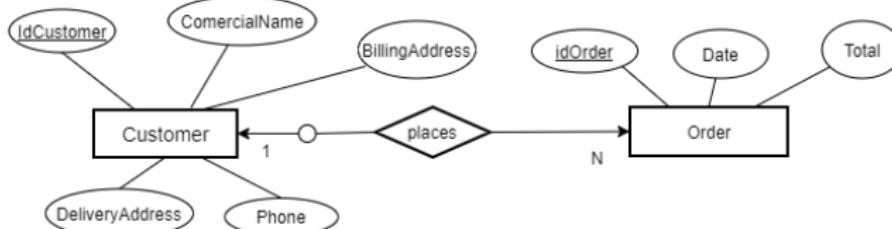
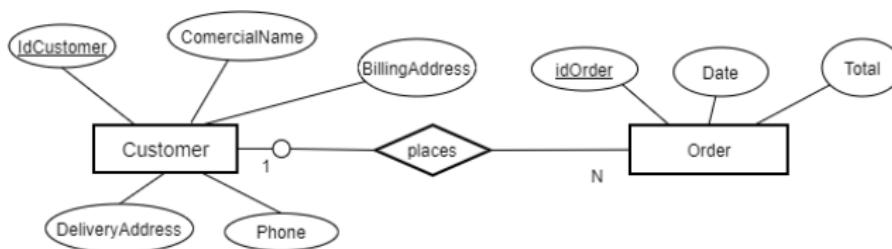
En un documento se hace referencia a un ID o una lista de ID de otro documento



Cardinalidad 1 a N / 1 a 1

¿Qué es Referenciar?

En un documento se hace referencia a un ID o una lista de ID de otro documento



Cardinalidad 1 a N / 1 a 1

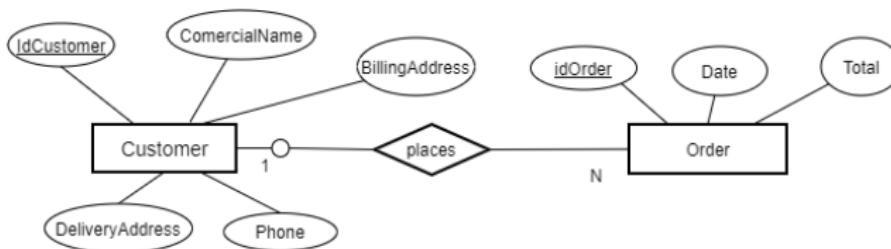
¿Que es Incrustar?

En un documento se incluyen todos los datos (en principio) de otro documento

Cardinalidad 1 a N / 1 a 1

¿Qué es Incrustar?

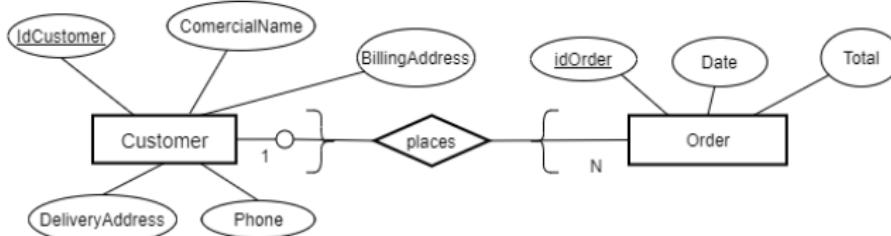
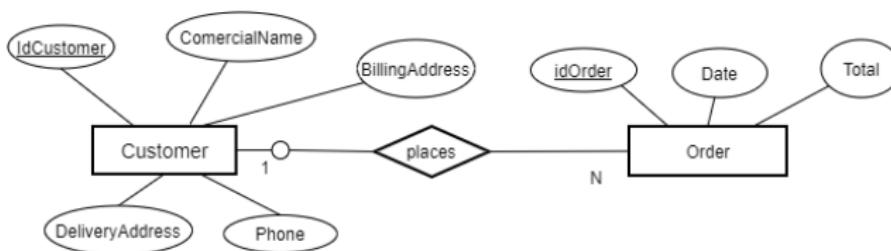
En un documento se incluyen todos los datos (en principio) de otro documento



Cardinalidad 1 a N / 1 a 1

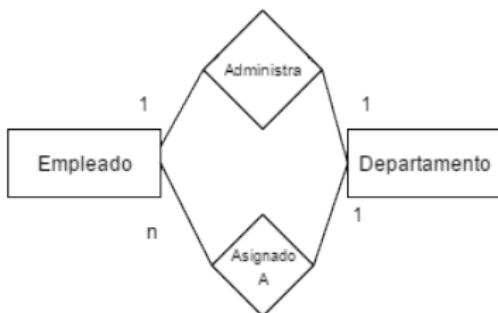
¿Qué es Incrustar?

En un documento se incluyen todos los datos (en principio) de otro documento



Cardinalidad 1 a N / 1 a 1

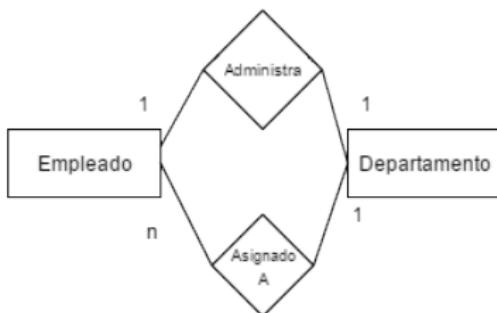
Se omiten los atributos por razones didácticas



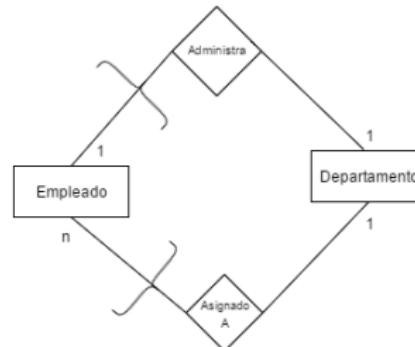
Modelo conceptual: DER

Cardinalidad 1 a N / 1 a 1

Se omiten los atributos por razones didácticas



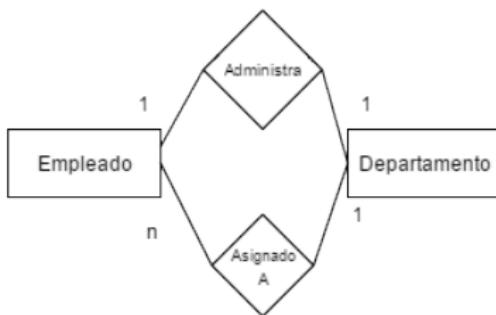
Modelo conceptual: DER



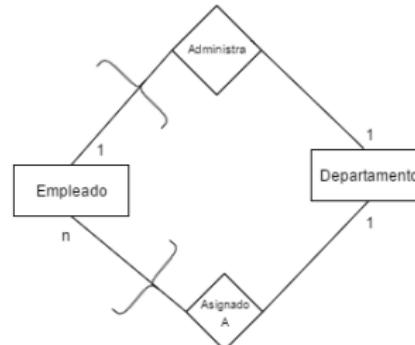
DID: Alternativa 1 Todo Incrustado en Depto

Cardinalidad 1 a N / 1 a 1

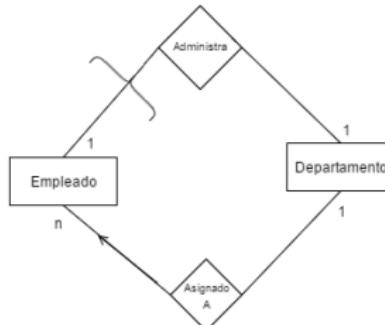
Se omiten los atributos por razones didácticas



Modelo conceptual: DER



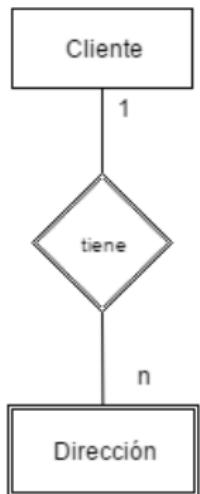
DID: Alternativa 1 Todo Incrustado en Depto



DID: Alternativa 2 - Incrustar sólo Gerente.

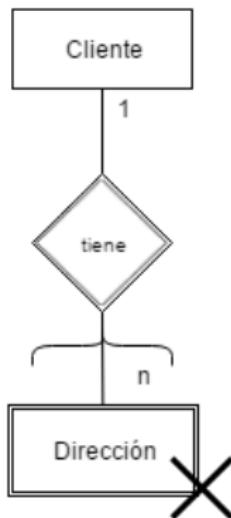
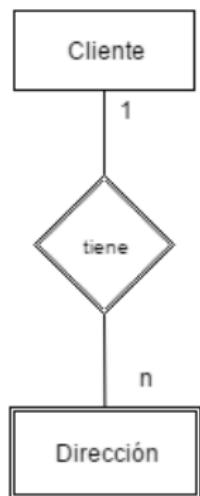
Entidades débiles

Se omiten los atributos por razones didácticas



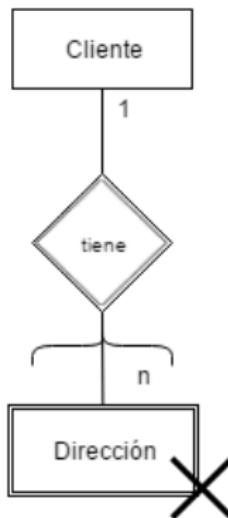
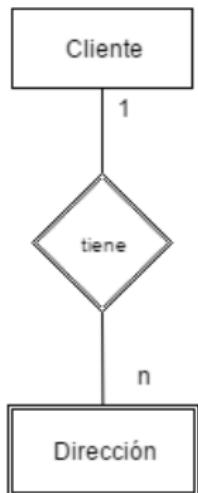
Entidades débiles

Se omiten los atributos por razones didacticas



Entidades débiles

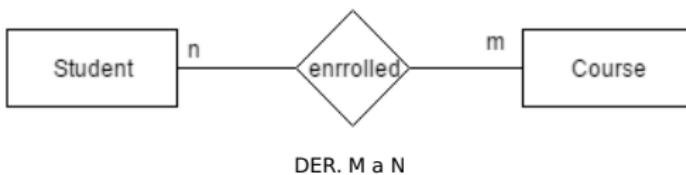
Se omiten los atributos por razones didácticas



```
{
  "cliente_id": 76123,
  "nombre": "Acme Data Modeling Services",
  "tipo_de_cliente": "business",
  "direcciones" :
  [
    {
      "calle: "San Martin 2222",
      ciudad: "Caseros",
      provincia: "Buenos Aires",
      codigo_postal: 99076},
    {
      calle: "9 de Julio 2223",
      ciudad: "CABA",
      codigo_postal: 01097}
  ]
}
```

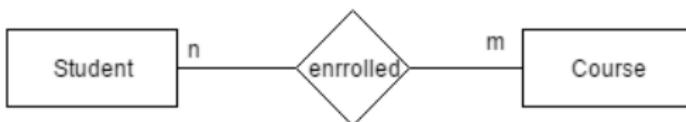
Cardinalidad M a N

Se omiten los atributos por razones didácticas

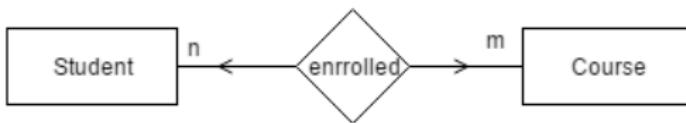


Cardinalidad M a N

Se omiten los atributos por razones didácticas



DER. M a N



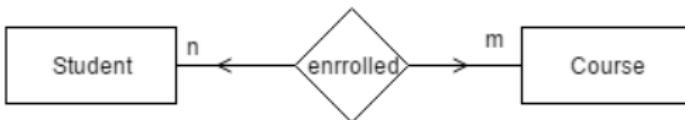
DID. M a N con referencias

Cardinalidad M a N

Se omiten los atributos por razones didácticas



DER. M a N

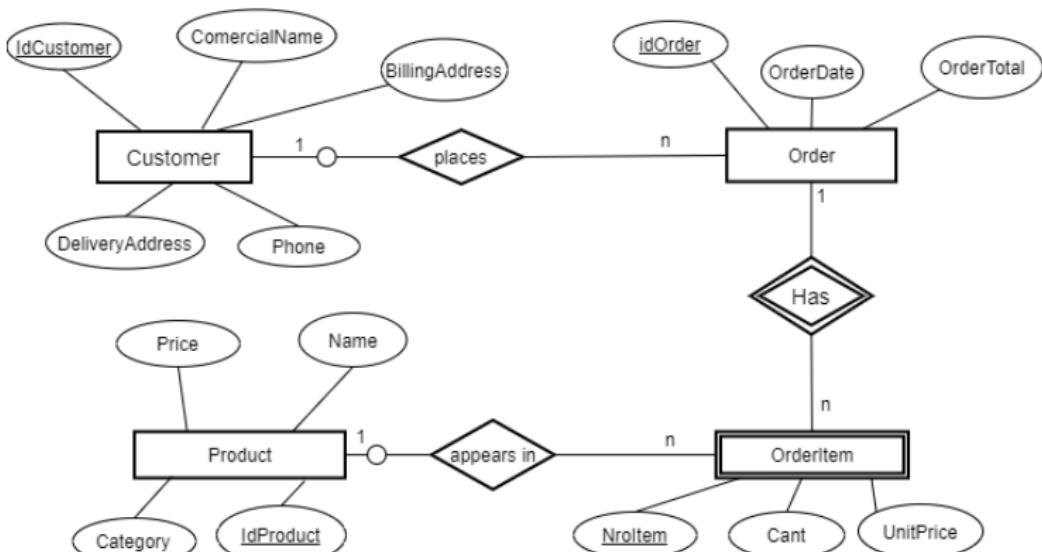


DID. M a N con referencias

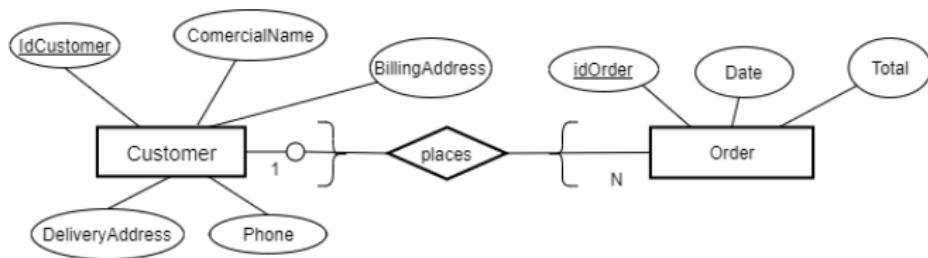
```
{
  { courseID: 'C1667',
    title: 'Introduction to Anthropology',
    instructor: 'Dr. Margret Austin',
    credits: 3,
    enrolledStudents: ['S1837', 'S3737', 'S9825' ...
      'S1847' ],
    { courseID: 'C2873',
      title: 'Algorithms and Data Structures',
      instructor: 'Dr. Susan Johnson',
      credits: 3,
      enrolledStudents: ['S1837', 'S3737', 'S4321', 'S9825'
        ... 'S1847' ],
      { courseID: C3876,
        title: 'Macroeconomics',
        instructor: 'Dr. James Schulen',
        credits: 3,
        enrolledStudents: ['S1837', 'S4321', 'S1470', 'S9825'
          ... 'S1847' ],
        ...
      }
    }
  }
}
```

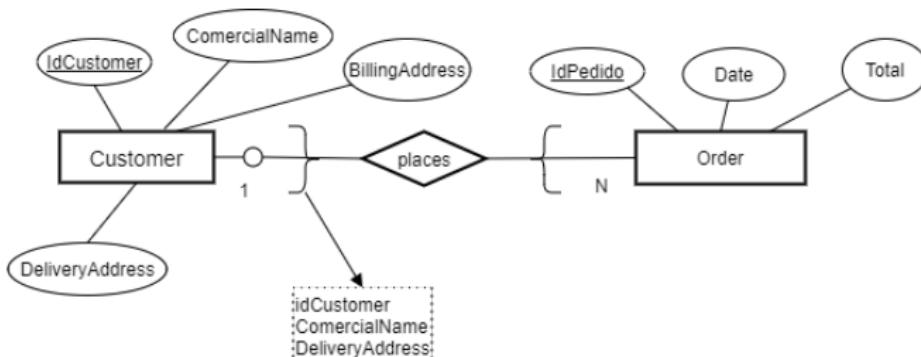
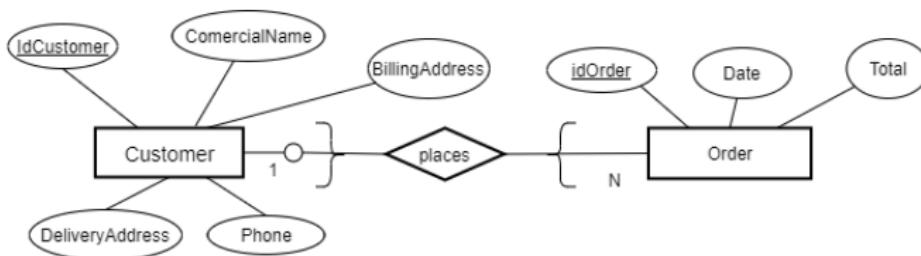
```
{
  { studentID:'S1837',
    name: 'Brian Nelson',
    gradYear: 2018,
    courses: ['C1667', C2873,'C3876'],
    { studentID: 'S3737',
      name: 'Yolanda Deltor',
      gradYear: 2017,
      courses: [ 'C1667','C2873'],
      ...
    }
  }
}
```

Desnormalización parcial



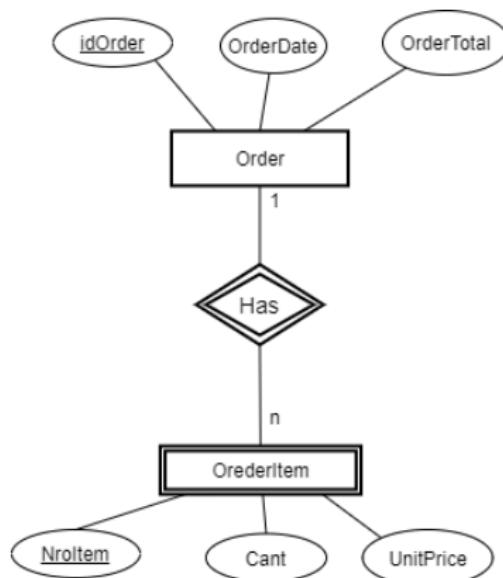
DER





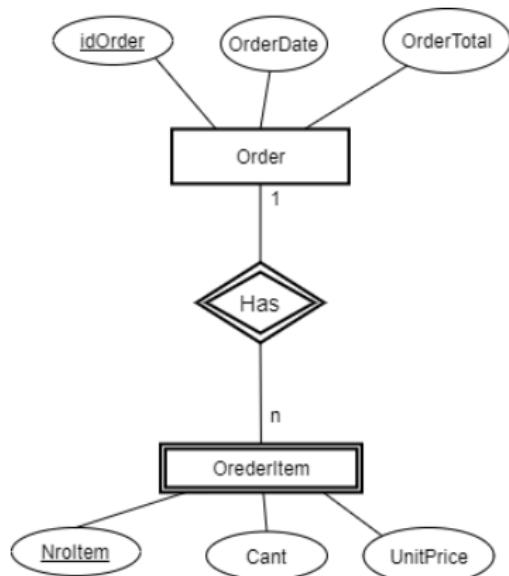
DID con desnormalización parcial

Desnormalización parcial

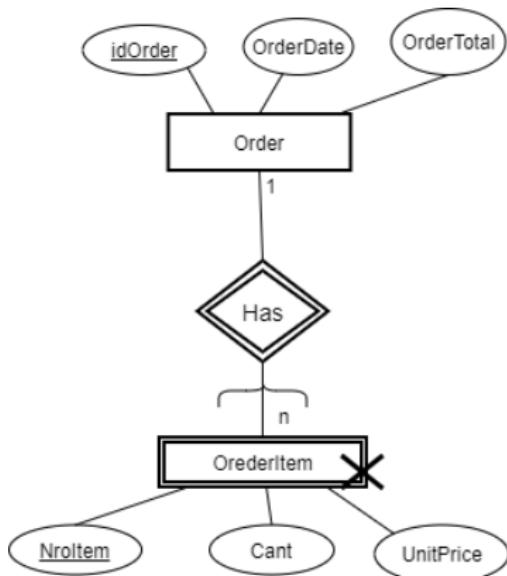


DID con desnormalización parcial

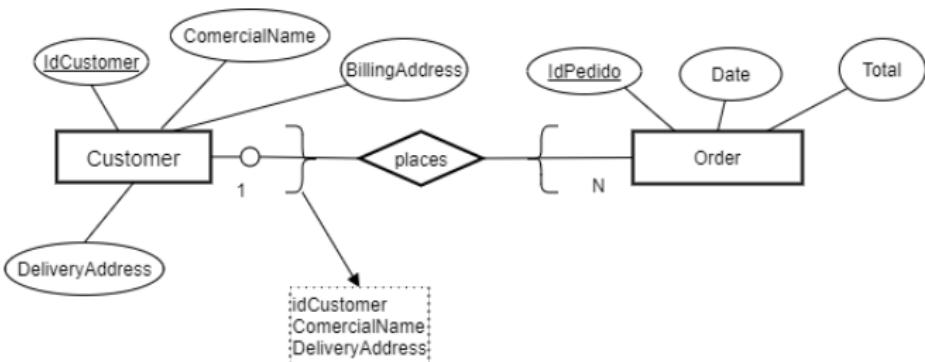
Desnormalización parcial

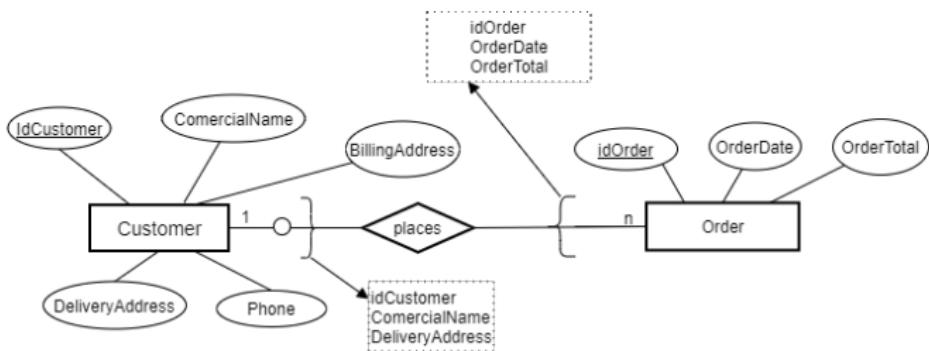
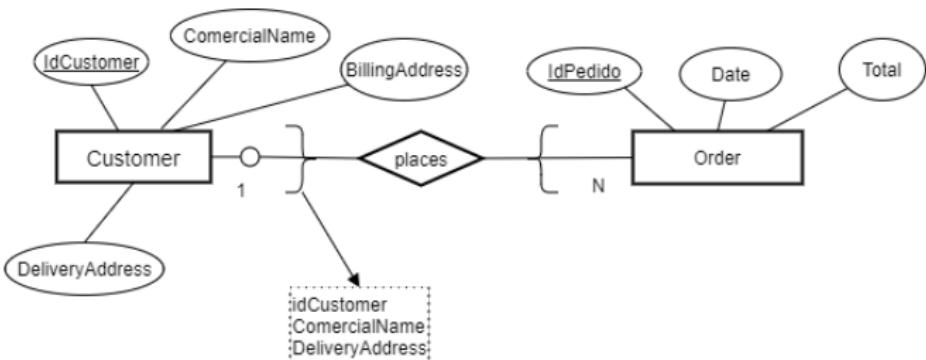


DID con desnormalización parcial

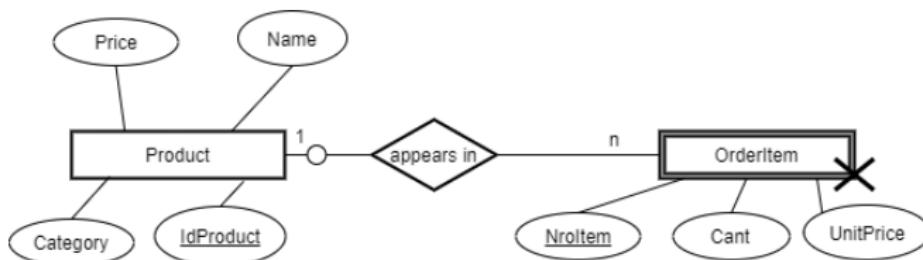


DID con desnormalización parcial

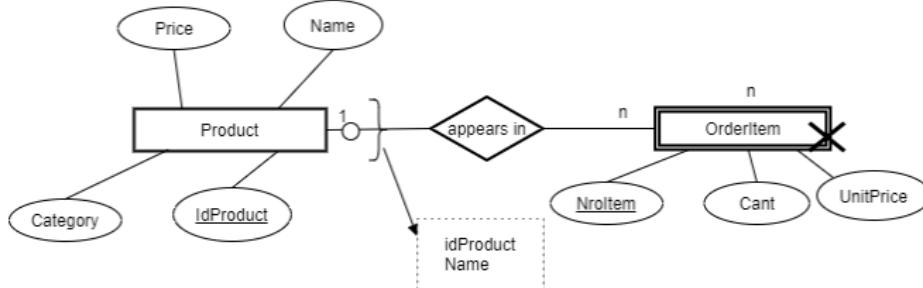
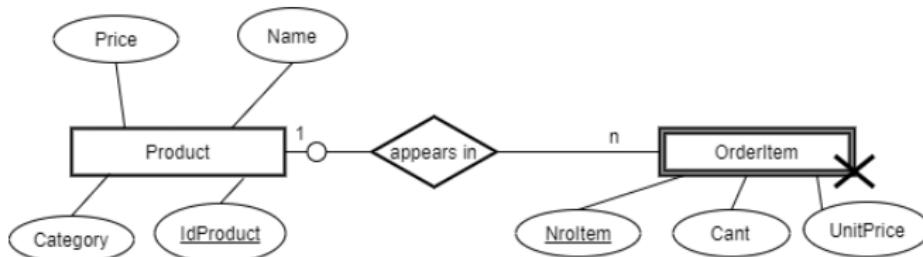




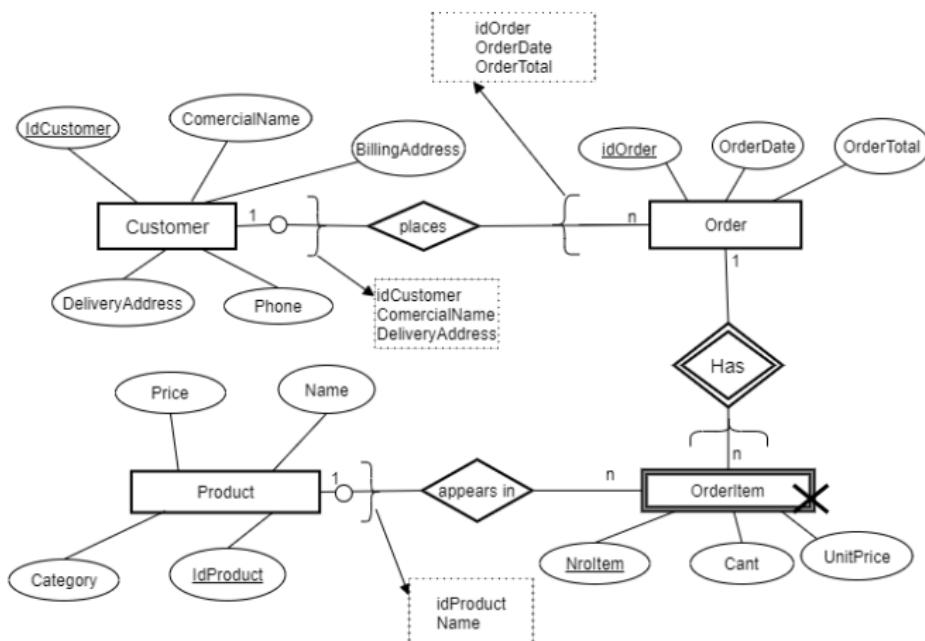
Producto-OrderItem



Producto-OrderItem



DID COMPLETO

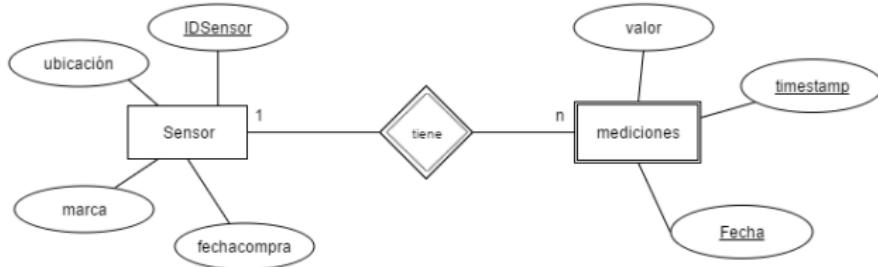


DID con desnormalización parcial

JSON Scheme para Documento Orden

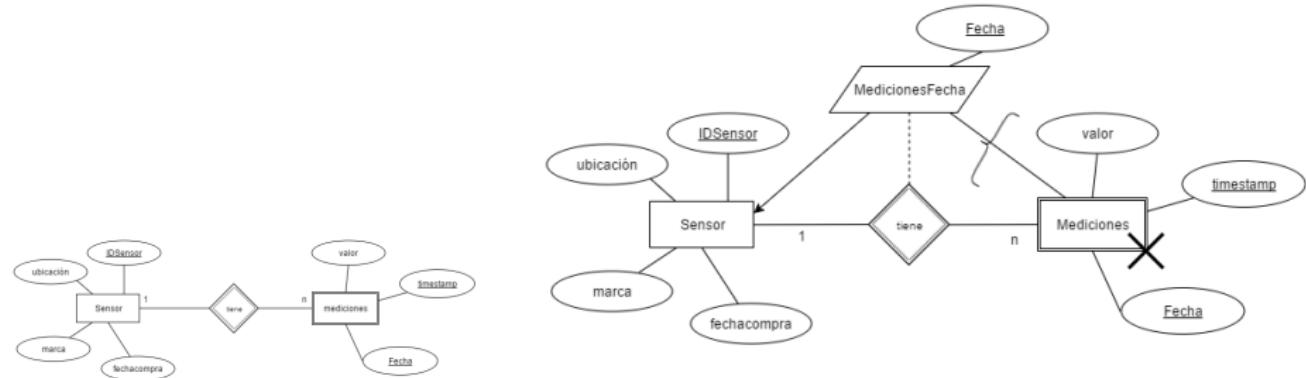
```
"Order": {"type": "object",
  "properties": {
    "idOrder": {"type": "integer" },
    "Date": {"type": "string", "format": "date-time" },
    "Total": {"type": "integer" },
    "Customer": {
      "type": "object",
      "properties": {
        "idCustomer": {"type": "integer" },
        "ComercialName": {"type": "string" },
        "DeliveryAddress": {"type": "string" }
      }
    },
    "OrderItem": {
      "type": "Array" ,
      "items": {
        "type": "object",
        "properties": {
          "Cant": {"type": "integer" },
          "IdProduct": {"type": "integer" },
          "Name": {"type": "string" },
          "UnitPrice": {"type": "string" }
        }
      }
    }
  }
}
```

Uso de Documentos Auxiliares



¿Qué pasa cuando la cantidad de mediciones es muy grande y además se actualiza permanentemente?

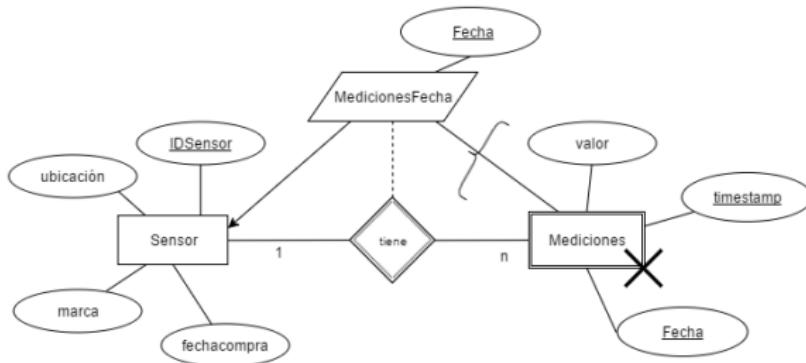
Uso de Documentos Auxiliares



Se necesita crear un tipo de documento auxiliar que permita particionar las mediciones

Uso de Documentos Auxiliares

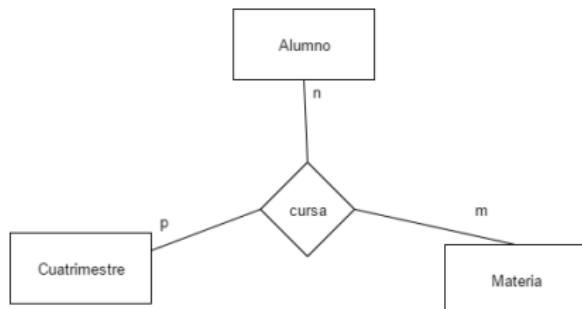
JSON Schema



```
"MedicionesFecha": {"type": "object",
  "properties": {
    "IDSensor": {"type": "integer" },
    "Fecha": {"type": "format": "date-time" },
    "Mediciones": {"type": "array",
      "items": {"type": "object",
        "properties": {"timestamp": {"type": "string", "format": "date-time"}, "valor": {"type": "decimal"}}}}
  }
}
```

Interrelaciones Ternarias

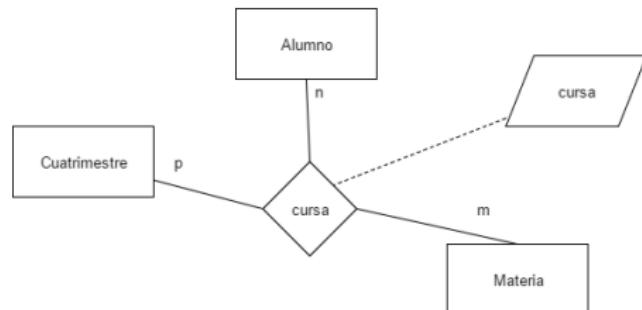
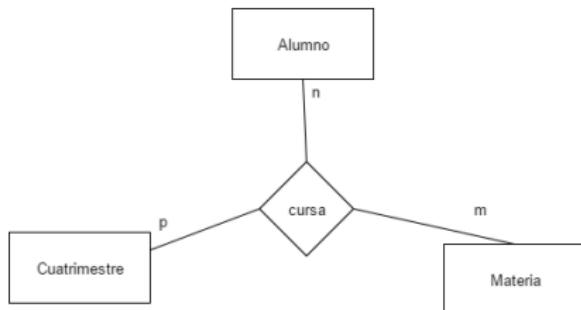
Se omiten los atributos por razones didácticas



¿Como resolvemos la interrelación *cursa*?

Interrelaciones Ternarias

Se omiten los atributos por razones didácticas

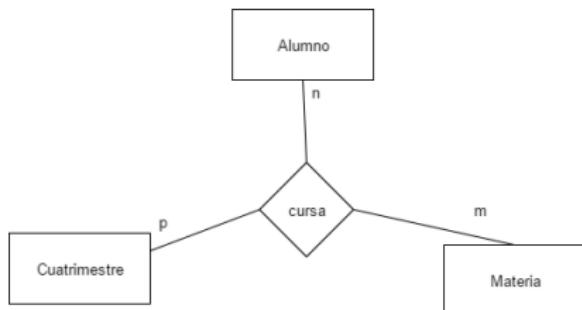


Opción básica:

Se genera un documento con las claves de cada uno

Interrelaciones Ternarias

Se omiten los atributos por razones didácticas

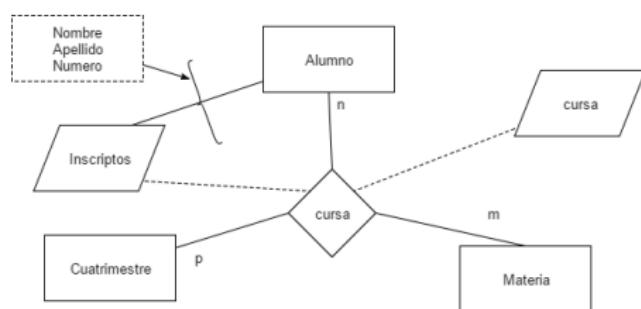
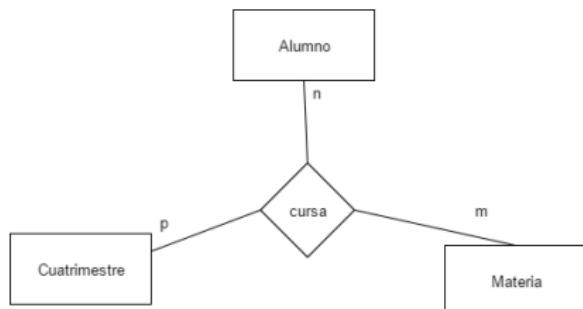


Supongamos

Una consulta muy común es saber cuales son los alumnos anotados en una materia en un cuatrimestre

Interrelaciones Ternarias

Se omiten los atributos por razones didácticas

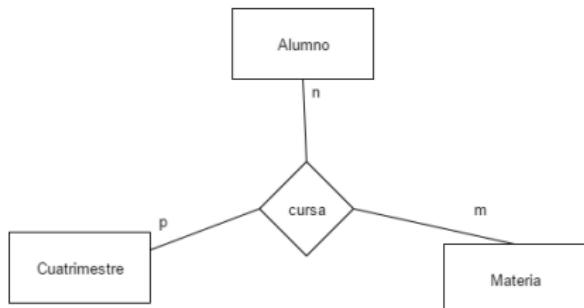


Supongamos

Una consulta muy común es saber cuales son los alumnos anotados en una materia en un cuatrimestre

Interrelaciones Ternarias

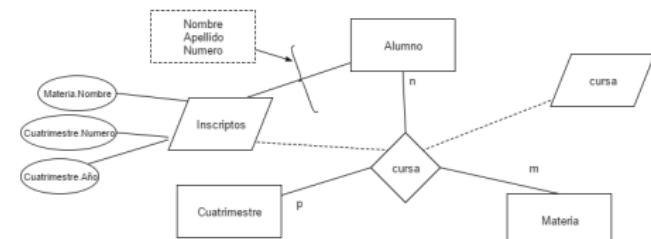
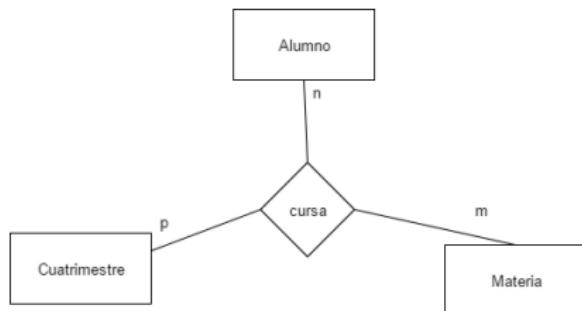
Se omiten los atributos por razones didácticas



¿Y si queremos además el nombre de la materia y el numero y año del cuatrimestre?

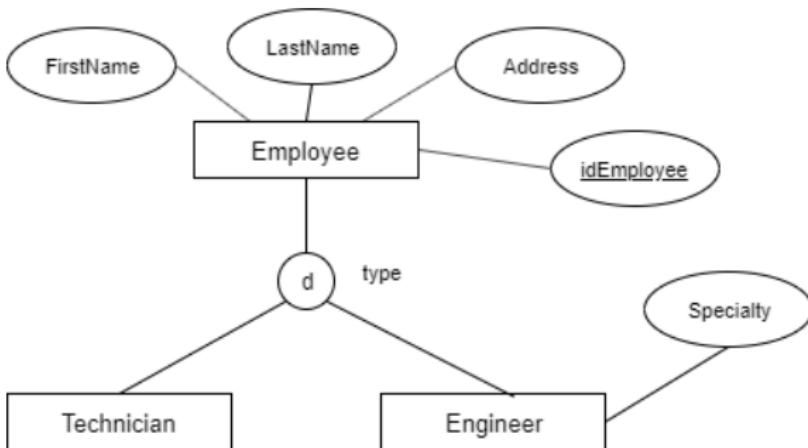
Interrelaciones Ternarias

Se omiten los atributos por razones didácticas

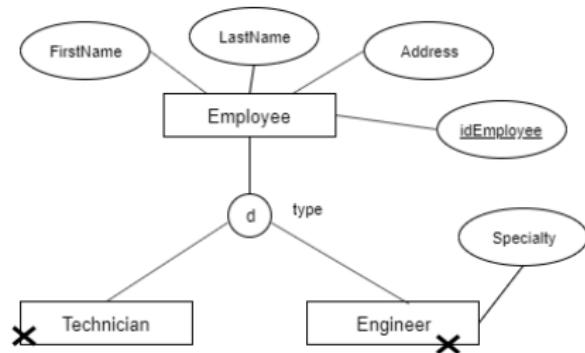
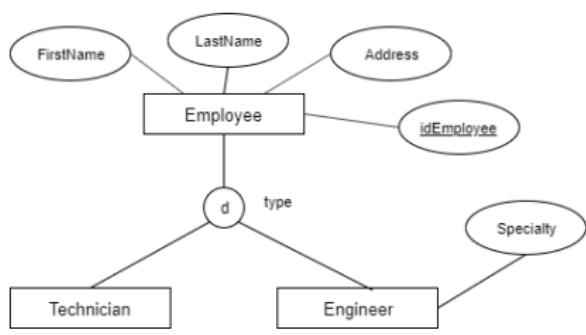


La propia semántica de la cardinalidad de la ternaria nos facilita este modelo

Jerarquías



Jerarquías



La facilidad de tener esquema flexible nos facilita el diseño. Podemos usar sólo un tipo de documentos para toda la jerarquía.

Jerarquías

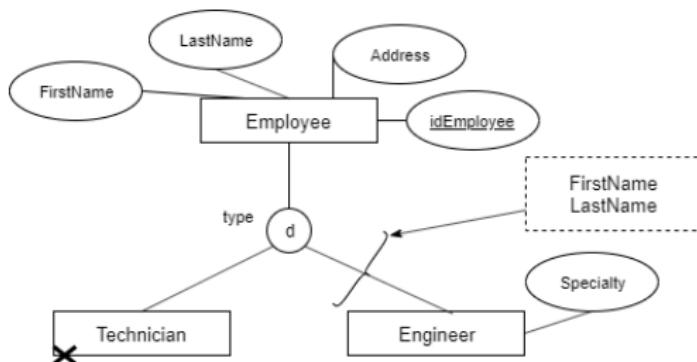
Supongamos que:

Queremos conservar la entidad *Engineer* como tipo de documento independiente porque una consulta importante es listar todos los ingenieros con sus datos.

Jerarquías

Supongamos que:

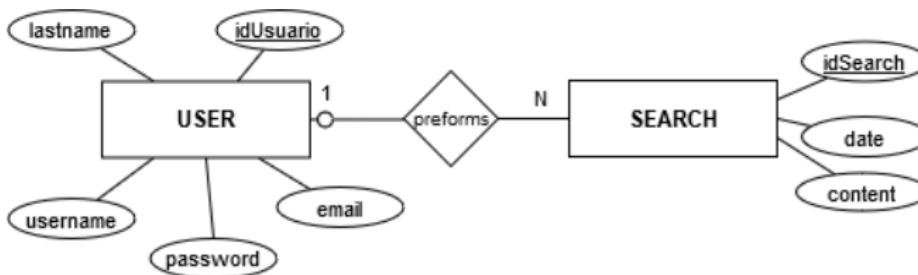
Queremos conservar la entidad *Engineer* como tipo de documento independiente porque una consulta importante es listar todos los ingenieros con sus datos.



Caso especial

Supongamos que:

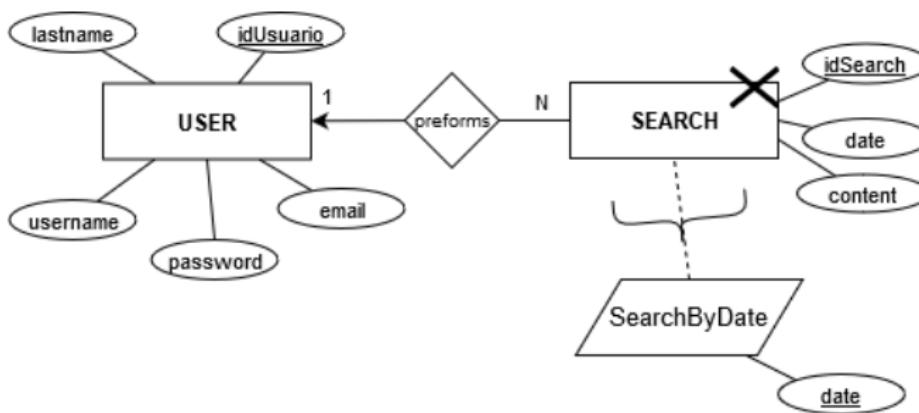
Un caso a considerar es cuando se hace necesario agrupar múltiples instancias de una entidad, por uno o más atributos, en un solo documento.



Caso especial

Supongamos que:

Un caso a considerar es cuando se hace necesario agrupar múltiples instancias de una entidad, por uno o más atributos, en un solo documento.



Bibliografía

- *NoSQL for Mere Mortals* - Dan Sullivan
- *NoSQL Distilled. A Brief Guide to the Emerging World of Polyglot Persistence* - Pramod J. Sadalage y Martin Fowler
- *A Big Data Modeling Methodology for NoSQL Document Databases* . Database Systems Journal, Vol XI, 2020. ISSN 2069 - 3230. Gerardo Rossel, Andrea Manna
- *Diseño de Bases de Datos Basadas en Documento: Modelo de Interrelación de Documentos* - Gerardo Rossel y Andrea Manna
- *MongoDB Applied Design Patterns* - Rick Copeland
- *CouchDB- The Definitive Guide* - J. Chris Anderson, Jan Lehnardt, Noah Slater
- *RavenDB in Action* - Itamar Syn-Hershko



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Apunte de Modelización

Autores

Sergio D'Arrigo - Leticia Seijas - Cecilia Ruz

Colaborador

Diego A. Castro

Revisiones y Actualizaciones

Alejandro Eidelsztein

| | | |
|----------|--|-----------|
| 1 | INTRODUCCIÓN | 2 |
| 2 | DE LOS REQUERIMIENTOS AL MODELO ENTIDAD-RELACIÓN..... | 2 |
| 2.1 | DESCRIPCIÓN GENERAL | 2 |
| 2.2 | MODELO CONCEPTUAL..... | 2 |
| 2.3 | MODELO ENTIDAD-RELACIÓN | 3 |
| 2.4 | ELEMENTOS DE UN DER..... | 4 |
| 2.4.1 | <i>Entidades.....</i> | 4 |
| 2.4.2 | <i>Atributos</i> | 4 |
| 2.4.3 | <i>Interrelaciones.....</i> | 6 |
| 2.4.4 | <i>Jerarquías.....</i> | 13 |
| 2.4.5 | <i>Agregación</i> | 15 |
| 2.4.6 | <i>Restricciones adicionales al modelo.....</i> | 16 |
| 3 | DEL MODELO ENTIDAD-RELACIÓN AL MODELO RELACIONAL..... | 17 |
| 3.1 | MODELO LÓGICO RELACIONAL..... | 17 |
| 3.2 | TRANSFORMACIÓN DEL MER AL MR..... | 18 |
| 3.2.1 | <i>Entidades Fuertes</i> | 18 |
| 3.2.2 | <i>Entidades Débiles.....</i> | 19 |
| 3.2.3 | <i>Interrelaciones uno a uno (1:1).....</i> | 20 |
| 3.2.4 | <i>Interrelaciones uno a muchos (1:N).....</i> | 20 |
| 3.2.5 | <i>Interrelaciones muchos a muchos (N:M)</i> | 21 |
| 3.2.6 | <i>Interrelaciones muchos a muchos (N:M) con atributos</i> | 21 |
| 3.2.7 | <i>Interrelaciones unarias</i> | 22 |
| 3.2.8 | <i>Interrelaciones ternarias.....</i> | 22 |
| 3.2.9 | <i>Jerarquías.....</i> | 25 |
| 3.2.10 | <i>Agregación</i> | 27 |
| 4 | BIBLIOGRAFÍA | 28 |

IMPORTANTE:

Este apunte es un complemento de la bibliografía que la materia utiliza como base para el tema. No contiene una visión exhaustiva ni completa de los contenidos, sólo pretende ser un elemento más de ayuda para la comprensión de los mismos.

1 Introducción

El presente apunte está orientado al diseño de Base de Datos Relacionales. Para un sistema cuya persistencia de datos se plantee sobre una base de datos relacional, deberemos definir en tiempo de análisis/diseño las características de esa base de datos. La notación más común para esto es la de Diagramas de Entidad Relación (DER), que es la que utilizaremos en la materia.

El diseño de las bases de datos relacionales puede darse en el contexto de proyectos encarados con diversas metodologías. Si con la metodología utilizada se construye un Modelo Conceptual (MC) con alguna técnica distinta de DER (por ejemplo, los diagramas de clases de UML), éste será uno de los inputs en la construcción del Modelo de Entidad Relación (MER). De no contarse con esto, partiremos de realizar un MC y luego refinarlo de ser necesario para llegar al MER, todo utilizando la notación de DER.

En la materia nos basaremos fundamentalmente en la Metodología de Diseño Lógico para Bases de Datos Relacionales (*LRDM* – Logical Relational Design Methodology), que utiliza la técnica de DER extendido.

En líneas generales, los grandes pasos a realizar para lograr un diseño de la Base de Datos Relacional con esta metodología son:

1. Construcción del Modelo Entidad Relación de los Requerimientos
2. Transformación del Modelo Entidad Relación a Relaciones
3. Normalización de las Relaciones

Con posterioridad al paso 3, se pasará a la construcción física.

En este apunte cubriremos los pasos 1 y 2, que son los que tienen que ver con tareas de modelización.

2 De los Requerimientos al Modelo Entidad-Relación

2.1 Descripción general

El objetivo de este paso es construir un MER que represente los datos a los cuales dar persistencia en la base de datos.

A partir del Modelo de Entidad-Relación, generaremos el Modelo Relacional que servirá para crear la Base de Datos (BD) concreta.

Requerimientos ↔ MER ↔ MR ↔ Normalización ↔ Diseño Físico ↔ BD

2.2 Modelo Conceptual

Un modelo conceptual es una conceptualización formal del mundo real (más precisamente de un dominio específico del mundo real). Para construirlo necesitamos modelar las cosas u objetos existentes, sus características y sus relaciones.

Los modelos conceptuales, como todo modelo, son prácticos para comunicar ideas y buscar consensos. La elaboración de un MC es de gran importancia al desarrollar el modelo de análisis de una aplicación y de gran utilidad en la validación con los usuarios.

Para representar un modelo conceptual utilizamos lenguajes de ontologías (lenguajes que nos permiten representar el conocimiento del mundo o parte de él). Estos lenguajes por lo general permiten introducir conceptos, propiedades de los conceptos, relaciones entre los conceptos y algunas restricciones adicionales. Típicamente son expresados por medio de diagramas.

Los Diagramas de Entidad Relación (orientados a Base de Datos Relacionales) y los Diagramas de Clases de UML (orientados al paradigma de objetos), pueden ser considerados lenguajes de ontología.

En Ingeniería de Software I ya se ha trabajado con modelos conceptuales utilizando UML. En Bases de Datos trabajaremos con modelos conceptuales utilizando los Diagramas de Entidad Relación, la técnica más difundida orientada a Bases de Datos Relacionales.

2.3 Modelo Entidad-Relación

El Modelo Entidad-Relación (MER) es una herramienta que permite realizar una abstracción o modelo de alguna situación de interés presente en el mundo real. El MER se realizará utilizando la técnica Diagramas de Entidad Relación (DER). Para construirlo necesitamos modelar las cosas u objetos existentes, sus características y sus relaciones.

El MER se basa en los siguientes elementos:

Entidades

Todo objeto o concepto del cual queremos registrar información constituye una entidad. Las entidades representan conjuntos de elementos¹.

Atributos

Los atributos son propiedades descriptivas de las entidades. Constituyen la información concreta que queremos mantener para cada elemento de una entidad.

Interrelaciones

Las diferentes entidades no están aisladas en el dominio del problema, muchas de ellas van a estar vinculadas entre sí. A esta vinculación la llamamos interrelación. En una interrelación pueden participar una entidad (interrelaciones unarias) o muchas entidades (interrelaciones binarias, ternarias, n-arias en general).

Las interrelaciones unarias son **conjuntos** de pares ordenados de elementos de la misma entidad. Las interrelaciones n-arias ($n > 1$) son conjuntos de n-uplas de elementos de entidades. Notemos que, como son conjuntos, cada tupla de elementos no puede repetirse dentro de una interrelación (lo mismo pasa con los elementos de una entidad).

Reglas de dominio adicionales

Al realizar el MER tenemos la necesidad de modelar ciertas reglas de dominio. En muchos casos el lenguaje que utilizamos no permite expresar a todas ellas. En ese caso, podremos recurrir a ciertas herramientas complementarias que nos permitan expresar ese conocimiento adicional del dominio del problema que no queda expresado en el diagrama.

Se podrá trabajar con lenguaje formal (por ejemplo, OCL) o con lenguaje informal (por ejemplo, lenguaje natural). Lo importante es que haya la menor pérdida de información entre lo conocido relevante del dominio del problema y lo que se exprese con el modelo.

¹ En realidad, siguiendo a alguna bibliografía, estamos llamando entidades a los conjuntos de entidades. Cada entidad es un elemento puntual, pero aquí usaremos indistintamente ambos nombres para representar al conjunto.

En la materia utilizaremos **lenguaje natural** para expresar toda restricción no modelada con DER.

2.4 Elementos de un DER

Como dijimos anteriormente, en la materia nos basaremos en la técnica de DER para elaborar modelos. A continuación describiremos la notación de esta técnica y algunas consideraciones del modelado a partir de ella.

2.4.1 Entidades

Todos los conceptos, cosas u objetos del mundo real que queremos modelar constituyen las **entidades** de nuestro modelo.

Las entidades se denotan con un rectángulo, con el nombre dentro de él.

Por ejemplo, la entidad *Estudiante* se representa así:



El nombre de la entidad lo escribimos en singular. Debe ser claro y explícito de la información contenida en la entidad, ya que nos da la semántica de la entidad.

Las entidades pueden ser **fuertes** o **débiles**.

Las entidades fuertes son aquellas que tienen una existencia independiente de cualquier otra entidad, se identifican sólo por atributos propios. Las entidades débiles son aquellas que derivan su existencia de otra entidad y necesitan la identificación de dicha entidad para distinguirse de otras (más adelante volveremos sobre las débiles).

Las entidades fuertes se denotan con un rectángulo con línea simple, como el que vimos en el ejemplo anterior. Las entidades débiles se denotan con un rectángulo con línea doble.

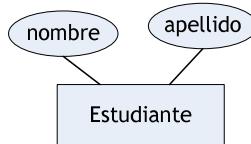


2.4.2 Atributos

2.4.2.1 Descripción General

Los atributos describen a las entidades. Son la información concreta que queremos mantener para una entidad. Se denotan con una elipse vinculada a una entidad, con el nombre dentro de la elipse.

Por ejemplo, si la entidad *Estudiante* tuviera un atributo *apellido* y otro atributo *nombre*, se representaría así:



Este tipo de atributos se denominan *atributos simples*. Son simples o atómicos en su estructura (no pueden ser subdivididos en subpartes) y también en sus valores (tienen un solo valor para cada entidad).

2.4.2.2 Atributos identificatorios

Los elementos de cualquier entidad se distinguen por el valor de algún atributo identificatorio (o de varios atributos en conjunto). A este atributo (o conjunto de atributos) se lo denomina **clave primaria** (PK – Primary Key).

La clave primaria es un conjunto minimal de atributos identificatorios y se denota subrayando a los mismos con una línea continua.

Por ejemplo, si el estudiante se identificara por su número de libreta universitaria, se representaría así:



En lo posible hay que tratar de que la clave primaria esté conformada por la menor cantidad de atributos, lo ideal es que sea un único atributo.

IMPORTANTÉ Una característica del MER es que **toda** entidad debe tener indefectiblemente una clave primaria.

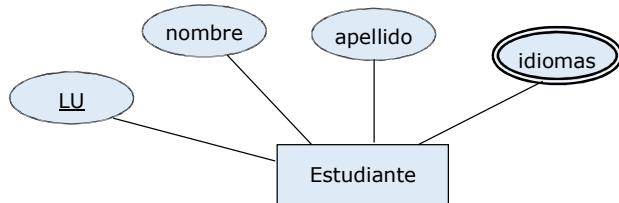
Hay ocasiones donde hay más de un atributo que distingue únicamente a los elementos de una entidad, en forma independiente uno de otro (por ejemplo, un empleado que se distinga por su número de legajo o por su CUIL). Todos ellos son **claves candidatas**, y se deberá elegir uno de ellos como clave primaria. Las claves candidatas no se modelan en el MER, recién lo veremos al trabajar con el Modelo Relacional.

2.4.2.3 Atributos Multivaluados

La mayoría de los atributos son simples, o sea que tienen un solo valor para cada entidad, como por ejemplo la fecha de nacimiento de un estudiante. Sin embargo, en algunos casos un atributo puede tener un conjunto de valores para una misma entidad, como por ejemplo los idiomas que sabe una persona o los títulos que posee un profesional.

Se los denota con una elipse con línea doble.

Por ejemplo, si la entidad *Estudiante* tuviera un atributo *idiomas*, se representaría así:



2.4.2.4 Atributos Compuestos

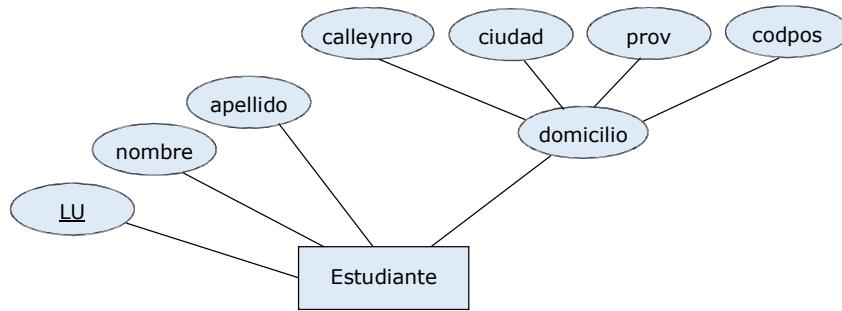
Los atributos compuestos pueden subdividirse en subpartes más pequeñas, las cuales a su vez son atributos con significado independiente. Son útiles para modelar situaciones en las que algunas veces preferimos referirnos al atributo compuesto como una unidad y en otras a sus componentes individualmente.

Los atributos compuestos pueden formar una jerarquía.

El valor de un atributo compuesto es la concatenación de los valores de los atributos simples que lo componen.

Se denotan con una jerarquía de elipses.

Por ejemplo, si la entidad *Estudiante* tuviera un atributo *domicilio* compuesto por *calleynro*, *ciudad*, *prov*, *codpos* se representaría así:



2.4.3 Interrelaciones

2.4.3.1 Descripción general

Una interrelación es una asociación entre entidades. Formalmente una interrelación es una relación matemática sobre $n \geq 2$ entidades (*no necesariamente distintas*).

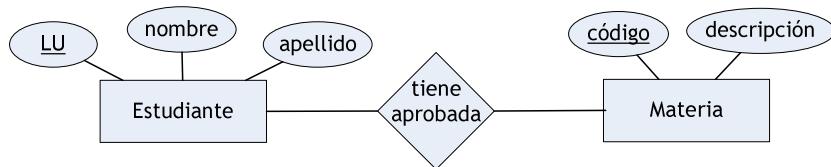
Las interrelaciones se representan con un rombo que vincula las distintas entidades participantes de la misma.

Cada interrelación tiene un nombre, que debe ser lo más representativo posible, ya que ese nombre nos va a indicar la semántica de la interrelación.

El nombre de la interrelación en lo posible debe estar expresado como un predicado (de hecho, una interrelación se puede mapear a un predicado sobre los elementos de las dos entidades).

Por ejemplo, si en nuestro ejemplo quisieramos representar las materias aprobadas que tiene un estudiante, podríamos agregar una entidad *Materia*, con atributos *código* (PK) y *descripción*, y una interrelación *tiene aprobada* que vincula al estudiante y a las materias aprobadas.

Esta situación se representaría así:



En este caso está claro que la interrelación *tiene aprobada* tiene origen en *Estudiante* y destino en *Materia* (el estudiante x tiene aprobada la materia y). Por convención, se tratará que las interrelaciones se lean de *izquierda a derecha* o de *arriba hacia abajo*.

En caso de que esto no sea posible o haya ambigüedad en la interpretación de los roles de cada entidad, se escribirá el rol de cada entidad interviniente en la relación junto a la entidad.

Cuando se trata de interrelaciones, hay 3 características a considerar: el **grado**, la **cardinalidad** y la **participación** de las entidades.

En lo que sigue explicaremos la notación de las tres por separado, pero en un modelo deben incluirse todas simultáneamente.

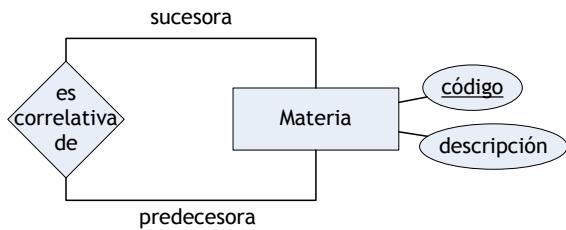
2.4.3.2 Grado

El grado de una interrelación se refiere a la cantidad de entidades que intervienen en ella. En la materia veremos interrelaciones unarias, binarias y ternarias (aunque puede haber de mayor grado, no son frecuentes).

2.4.3.2.1 Interrelación Unaria

En estas interrelaciones participa una única entidad, desempeñando los dos roles de la interrelación. Se la llama también interrelación *reflexiva*. Los elementos de esta entidad se relacionan con elementos de la misma entidad. Notar que a pesar de haber una entidad, constituyen un par ordenado donde cada componente del par desempeña un rol particular en la interrelación.

Un ejemplo de interrelación unaria sería la interrelación *es correlativa de*, que indica las correlatividades entre materias.



IMPORTANTES En relaciones unarias siempre se debe aclarar los **roles** de los vínculos para evitar ambigüedad.

2.4.3.2.2 Interrelación Binaria

En esta interrelación participan dos entidades. Los elementos de una entidad se relacionan con los elementos de otra entidad. Las interrelaciones con este grado son las más frecuentes.

Un ejemplo de esta interrelación es la interrelación *tiene aprobada* que vimos anteriormente.

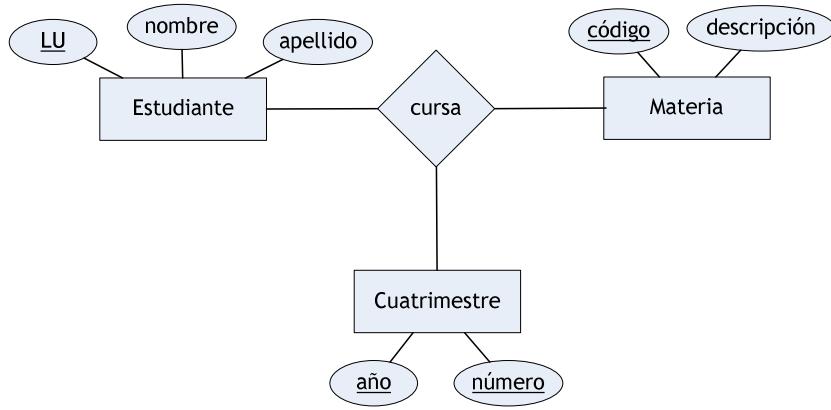
2.4.3.2.3 Interrelación Ternaria

Esta interrelación es más compleja que las anteriores y se da con la participación de tres entidades **simultáneamente**. La interrelación está representada por una 3-upla, donde cada elemento corresponde a un elemento de una de las 3 entidades vinculadas.

Se denota con el mismo rombo, con vínculos hacia las 3 entidades participantes.

IMPORTANTES En un elemento de una interrelación ternaria, siempre se requiere la participación de las 3 entidades. Dicho de otra manera, en el marco de estas interrelaciones, no existe la asociación de elementos de dos entidades sin la participación de un elemento de la tercera.

En nuestro ejemplo, si además de tener las entidades *Estudiante*, y *Materia*, tuviésemos también la entidad *Cuatrimestre* (con atributos *año* y *número*), y quisieramos modelar las materias cursadas por los estudiantes en los diferentes cuatrimestres, podríamos hacerlo con la siguiente relación ternaria:



2.4.3.3 Cardinalidad

La cardinalidad de una interrelación se refiere a la cantidad de elementos de un rol de la interrelación que puede estar vinculado a un elemento de otro rol de la interrelación.

2.4.3.3.1 Interrelaciones UNO A UNO (1:1)

En las interrelaciones uno a uno (unarias o binarias) los elementos de una entidad se vinculan a lo sumo a un elemento de la otra entidad relacionada.

Se denota escribiendo un 1 junto al vínculo cerca de cada entidad interviniente.

Por ejemplo, si tuviésemos que modelar que cada carrera de la facultad tiene un director que debe ser un profesor, y que cada profesor puede ser director de a lo sumo una carrera, lo podríamos modelar como sigue:



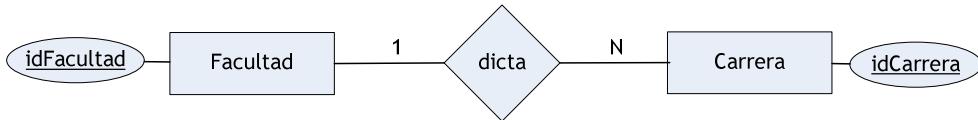
Notar que el diagrama anterior está incompleto, ya que para ajustarse a la realidad faltaría modelar la noción de participación (que veremos más adelante).

2.4.3.3.2 Interrelaciones UNO A MUCHOS (1:N)

En las interrelaciones uno a muchos (unarias o binarias), cada elemento de una entidad se puede vincular a muchos elementos de la entidad relacionada, pero los elementos de esta última se vinculan a sólo una instancia de la primera.

Se denota con una N junto al vínculo cerca de la entidad “muchos” y un 1 junto al vínculo cerca de la entidad “uno”.

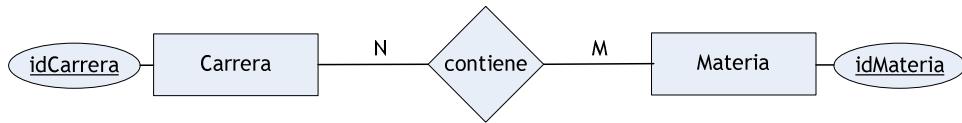
Por ejemplo, si tuviésemos que modelar que cada facultad puede dictar muchas carreras y que cada carrera puede ser dictada por una facultad, lo podríamos modelar como sigue:



2.4.3.3.3 Interrelaciones MUCHOS A MUCHOS (N:M)

En las interrelaciones muchos a muchos (unarias o binarias), cada elemento de cualquiera de las entidades se puede vincular a muchos elementos de la otra entidad relacionada.

Por ejemplo, si tuviésemos que modelar que las carreras pueden contener muchas materias, y que cada materia puede estar contenida en muchas carreras, lo podríamos modelar como sigue:



2.4.3.4 Participación de las entidades

Decimos que los elementos de una entidad *participan* de una interrelación. Si todo elemento de la entidad tiene que estar necesariamente en la interrelación (al menos una vez), decimos que la participación es total. Caso contrario (puede haber elementos que no participen) decimos que es parcial u opcional.

2.4.3.4.1 Participación Parcial u Opcional

La participación parcial de una entidad en una interrelación se denota poniendo una "0" sobre el vínculo que une el rombo de la relación con la entidad.

En nuestro ejemplo de los directores de carreras, tenemos que cada profesor puede ser director de a lo sumo una carrera, pero puede no serlo de ninguna. Por lo tanto, la participación de *Profesor* en la interrelación es **parcial**, ya que vamos a poder tener profesores que no estén asociados a ninguna carrera por esta interrelación. Este caso se denotaría así:



Esto anterior podría leerse también como que todo profesor puede dirigir a 0 o 1 carreras.

2.4.3.4.2 Participación Total

En el mismo ejemplo anterior, tenemos que cada carrera tiene que tener al menos un director. Por lo tanto, la participación de *Carrera* en la interrelación es **total**, ya que toda carrera debe estar asociada a algún profesor por esta interrelación.

Este caso se denotaría como está en el diagrama anterior.

2.4.3.5 Atributos de Interrelaciones

Las interrelaciones también pueden tener sus atributos en determinadas circunstancias y estos pueden ser descriptivos o identificatorios.

Los atributos de una interrelación se representan igual que los de las entidades, con la diferencia que están vinculados al rombo de la interrelación.

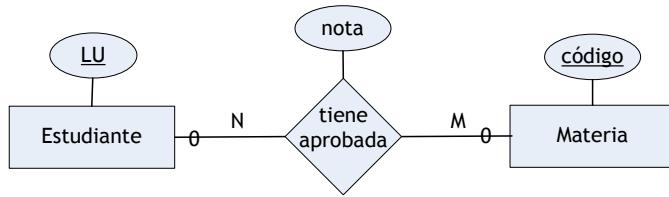
IMPORTANT En nuestra notación, los atributos de interrelaciones (tanto descriptivos como identificatorios) son permitidos **únicamente** en interrelaciones N:M.

2.4.3.5.1 Atributos descriptivos de interrelaciones

Un atributo descriptivo de una interrelación permite registrar información adicional a la que aportan de las entidades intervenientes en ella.

Un ejemplo de esto sería si tenemos una interrelación binaria *tiene aprobada* entre *Estudiante* y *Materia* y queremos registrar la nota de aprobación. Este atributo no puede ser de la materia ni del estudiante, ya que depende a la vez de ambas entidades en el contexto de la aprobación. Lo modelaremos como un atributo descriptivo de la interrelación.

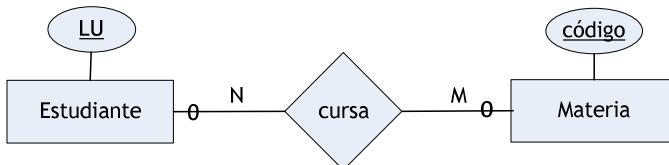
Se denotaría como sigue:



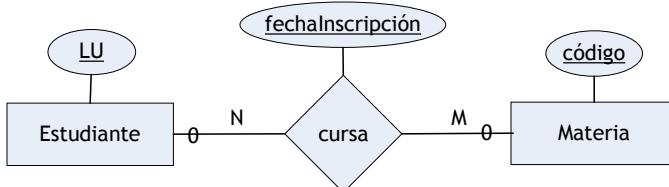
2.4.3.5.2 Atributos identificatorios de interrelaciones

El caso de los atributos identificatorios es un poco más complejo, lo veremos con un ejemplo.

Supongamos que queremos modelar las cursadas como en nuestro ejemplo de ternarias, pero en el dominio de nuestro problema no fuese relevante la entidad *Cuatrimestre* y no se justifique modelarla como entidad. En ese caso la interrelación se daría sólo entre *Estudiante* y *Materia*, constituyendo una interrelación binaria. Esta situación se denotaría como sigue:



Notar que esta interrelación sólo nos permite vincular una vez a cada estudiante con cada materia, ya que los pares de elementos (*Estudiante, Materia*) no pueden repetirse (las interrelaciones son un conjunto de pares ordenados). ¿Qué pasaría si quisieramos reflejar la historia de las cursadas? En este caso necesitamos que esos pares se puedan repetir. La alternativa de notación que podríamos usar en este caso es agregar un atributo identificatorio que nos permita distinguir las elementos de la interrelación con repeticiones, por ejemplo, la fecha de inscripción en la cursada. Se denotaría así:



En este caso, cada elemento de la interrelación estaría identificado por el estudiante (con su *LU*), la materia (con su *código*) y la *fechalinscripción*. Aquí podría repetirse un elemento de estudiante y materia, siempre y cuando tenga diferente fecha de inscripción.

IMPORTANTES Notar que los atributos identificatorios sí permiten que se repitan pares ordenados pero éstos deben tener diferentes valores para el atributo identificatorio.

2.4.3.6 Interrelaciones con “muchos” dentro de un rango

Hay casos en que se sabe que la cardinalidad “muchos” está dentro de un rango de elementos (por ejemplo, si dijéramos que cada materia tiene que tener entre 5 y 40 estudiantes inscriptos).

Con la notación que utilizaremos en la materia esta situación **no** se puede modelar.

La forma de expresar este tipo de situaciones es como restricción adicional, en lenguaje natural.

2.4.3.7 Cardinalidad y participación en interrelaciones n-arias

Analicemos la cardinalidad y participación para el caso de interrelaciones n-arias ($n > 2$).

El caso de la participación de entidades en las interrelaciones ternarias es similar al de las binarias. Si todo elemento de una entidad interviniente tiene que estar necesariamente en la interrelación (al menos una vez), decimos que la participación es total. Caso contrario (puede haber elementos que no

participen) decimos que es parcial u opcional. Esto último se denota de la misma manera, con una "0" sobre el vínculo que une al rombo de la interrelación con la entidad en cuestión.

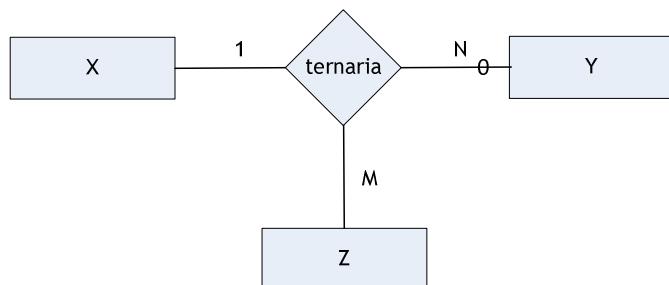
Una interrelación con cardinalidad 1 en una entidad X significa que para cada par de elementos de las entidades restantes que participa de la interrelación, existe un elemento de la entidad X a la que el par está vinculado.

Análogamente, una interrelación con cardinalidad muchos en la entidad X significa que para cada par de elementos de las entidades restantes que participa de la interrelación, pueden existir muchos elementos de la entidad X a las que el par está vinculado.

Notar que el concepto de participación en las interrelaciones está siempre referido a una única entidad. Si en el dominio del problema a modelar existiese alguna restricción de participación que afectase a pares de entidades, esta restricción debería explicitarse como restricción adicional.

2.4.3.8 Interpretación de un caso de interrelaciones ternarias con participación y cardinalidad

Tomemos el siguiente caso de interrelación ternaria.



Se interpretan las siguientes reglas

- Todo elemento de la entidad X debe participar al menos una vez en la interrelación.
- Los elementos de la entidad Y pueden no participar de la interrelación.
- Todo elemento de la entidad Z debe participar al menos una vez en la interrelación.
- Cada par de elementos de las entidades X e Y que participa de la interrelación puede estar vinculado a muchos elementos de la entidad Z.
- Cada par de elementos de las entidades X y Z que participa de la interrelación puede estar vinculado a muchos elementos de la entidad Y.
- Cada par de elementos de las entidades Y y Z que participa de la interrelación está vinculado a un elemento de la entidad X.

Si por ejemplo tuviésemos los siguientes elementos de entidades:

Entidad X: {X1, X2}

Entidad Y: {Y1, Y2, Y3}

Entidad Z: {Z1, Z2}

- Si la interrelación fuese $\{<X1, Y1, Z1>\}$ sería *inconsistente* con el modelo, ya que se violan las reglas:
 - a: X2 no participa de la relación
 - c: Z2 no participa de la relación
- Si la interrelación fuese $\{<X1, Y1, Z1>, <X2, Y2, Z2>\}$ sería *consistente* con el modelo.
- Si la interrelación fuese $\{<X1, Y1, Z1>, <X2, Y2, Z2>, <X1, Y3, Z2>, <X2, Y1, Z1>\}$ sería *inconsistente* con el modelo, ya que se viola la regla:
 - f: El par $<Y1, Z1>$ está vinculado a dos elementos de la entidad X: X1 y X2
- Si la interrelación fuese $\{<X1, Y1, Z1>, <X2, Y2, Z2>, <X1, Y3, Z2>, <X2, Y3, Z1>\}$ sería *consistente* con el modelo.

La generalización de los conceptos de cardinalidad y participación para interrelaciones n-arias con $n \geq 4$ es más compleja aún y está fuera del alcance de la materia.

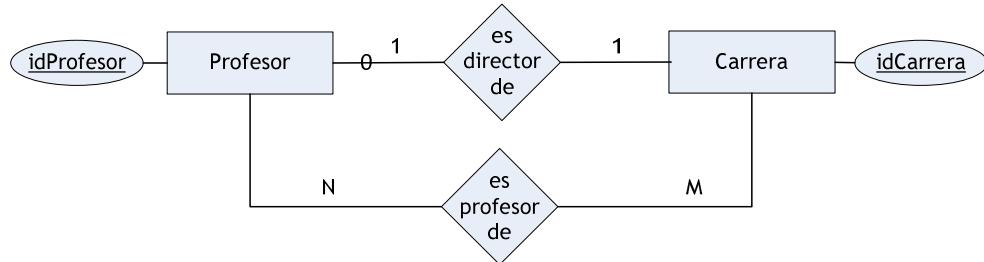
2.4.3.9 Múltiples interrelaciones sobre similares entidades

También puede darse que tengamos más de una interrelación sobre un mismo conjunto de entidades, en este caso cada una de esas interrelaciones tendrá diferente semántica.

En este caso, debemos representar en el modelo las diferentes interrelaciones con rombos independientes, poniendo especial énfasis en los nombres de las mismas, ya que es necesario que su semántica esté bien clara.

Por ejemplo, si tenemos las entidades *Carrera* y *Profesor*, podríamos tener las relaciones *es profesor de* y *es director de* que vinculen a ambas entidades.

Se denotaría así:

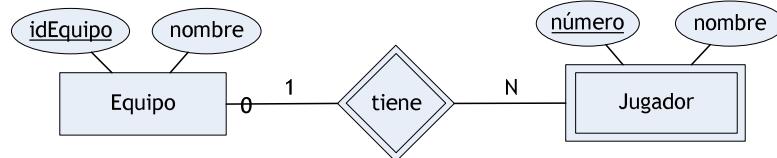


2.4.3.10 Interrelaciones identificadorias de entidades débiles

Como dijimos, las entidades débiles se denotaban con un rectángulo con línea doble. De todos modos, con esto no es suficiente.

Una entidad débil es una entidad que necesita a otra para identificarse. Por lo tanto, tenemos que tener una manera de indicar cuál es la entidad fuerte correspondiente a una entidad débil. Si bien la entidad débil va a estar interrelacionada con su entidad fuerte, también podría estarlo con otras de las cuales no depende su identificación. Para no tener ambigüedades, a la interrelación que vincula una entidad débil con su entidad fuerte correspondiente (llamada *interrelación identificatoria*) la denotaremos también con un rombo con línea doble.

Por ejemplo, si queremos modelar que un equipo puede tener varios jugadores, podríamos modelar las entidades *Equipo* (fuerte) y *Jugador* (débil), con la interrelación *tiene* que las vincula. En este caso, esta interrelación es la interrelación identificatoria.



IMPORTANTÉ Notar que la entidad débil no se identifica por sí sola. El atributo identificatorio de la entidad débil es parte de la clave y no la clave en su totalidad, y por lo tanto no garantiza unicidad de un elemento. La unicidad va a estar dada por la clave completa de la entidad débil, que está compuesta por la clave de la entidad fuerte (heredada, no se debe explicitar en la entidad débil) más el atributo identificatorio propio.

IMPORTANTÉ Notar también que, como consecuencia de lo anterior, la entidad débil siempre debe tener participación total en la interrelación identificatoria.

2.4.4 Jerarquías

2.4.4.1 Descripción general

Puede darse que haya entidades que constituyan casos particulares de otras entidades, por ejemplo *Materia con Laboratorio* podría ser un caso especial de *Materia*.

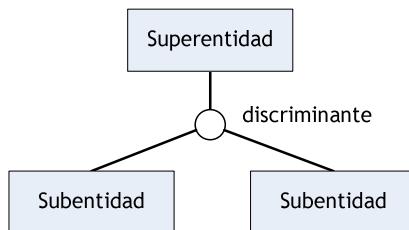
Esta situación la modelamos de una forma especial, con una jerarquía o especialización. Esta relación nos da una jerarquía, una especie de herencia (no es herencia en el sentido completo de objetos).

Con estas relaciones, las propiedades de la entidad padre (la superentidad) son *heredadas* por las entidades hijas (las subentidades).

La identificación de las entidades hijas también es la misma que la de la entidad padre. No puede existir un elemento en una subentidad que no figure en la superentidad.

Cada especialización tiene una semántica y decimos que hay un discriminante que determina la especialización.

La notación está dada por un pequeño círculo vinculado a la superentidad, al cual se vinculan las subentidades, colocando el nombre del discriminante junto al círculo (el discriminante nos indica la semántica de la jerarquía).



En las jerarquías existen 2 características a considerar: la cobertura y el solapamiento.

En lo que sigue explicaremos la notación de las dos por separado, pero en un modelo deben incluirse todas simultáneamente.

2.4.4.2 Jerarquías según cobertura

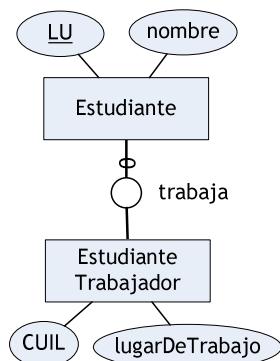
La cobertura nos indica si todos los elementos de la superentidad van a tener algún elemento correspondiente en alguna subentidad.

2.4.4.2.1 Cobertura Parcial u Opcional

En este caso, un elemento de la superentidad no necesariamente está en una subentidad.

Se lo denota colocando una "O" en el vínculo entre el círculo y la superentidad.

Por ejemplo, si quisieramos registrar alguna información adicional para los estudiantes que trabajan, por ejemplo su CUIL y el lugar de trabajo, lo podríamos representar así:



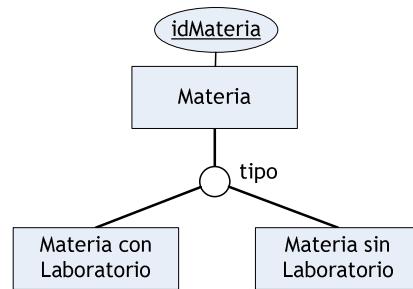
☞ Notar que el diagrama anterior está incompleto pues falta decir qué tipo de jerarquía es (esto se presentará en breve).

2.4.4.2.2 Cobertura Total

En este caso, todo elemento de la superentidad debe estar en al menos una subentidad.

Se lo denota con una línea simple uniendo el círculo con la superentidad.

Por ejemplo, si tenemos que las materias pueden ser con o sin laboratorio (por simplicidad obviamos en este ejemplo los atributos particulares de cada uno), podríamos denotarlo como sigue:



2.4.4.3 Jerarquías según solapamiento

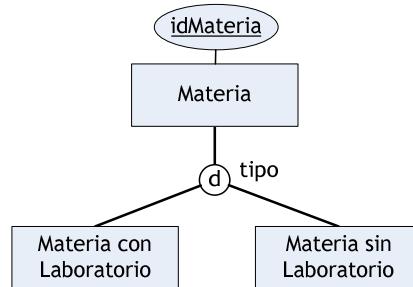
El solapamiento nos indica si los elementos de la superentidad pueden estar en más de una subentidad simultáneamente.

La especialización puede ser disjunta o con solapamiento y se diferencia registrándolo en el círculo de la especialización.

2.4.4.3.1 Jerarquía Disjunta

Los elementos de la superentidad pueden estar a lo sumo en una subentidad. Se lo denota con una **d** en el círculo.

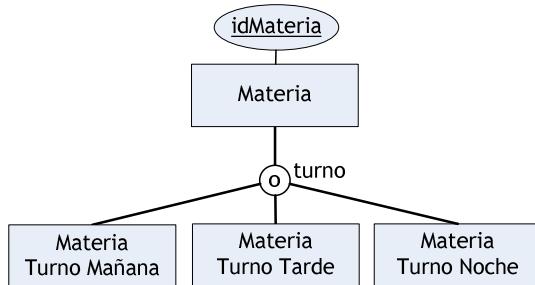
Por ejemplo, con el caso de las materias con y sin laboratorio, quedaría así:



2.4.4.3.2 Jerarquía con Solapamiento

Los elementos de la superentidad pueden estar en más de una subentidad. Se lo denota con una **o** en el círculo (del inglés overlapping).

Por ejemplo, si quisieramos modelar que una materia puede ser de turno mañana, tarde o noche, pudiendo ser de varios turnos a la vez, lo podríamos modelar así:

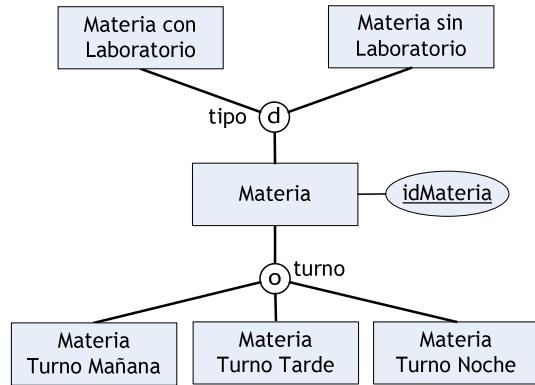


2.4.4.4 Varias jerarquías de una entidad con diferente semántica

Cada jerarquía tiene una semántica. Es posible que una misma entidad tenga diferentes especializaciones con distinta semántica. Se representa vinculando todas las especializaciones a la entidad, escribiendo al lado del círculo el “discriminante” que da la semántica de la entidad.

Por ejemplo, podríamos tener clasificadas nuestras materias según *tipo* (con o sin laboratorio) y también según *turno* que se dictan (mañana, tarde o noche).

Se podría representar así:



2.4.5 Agregación

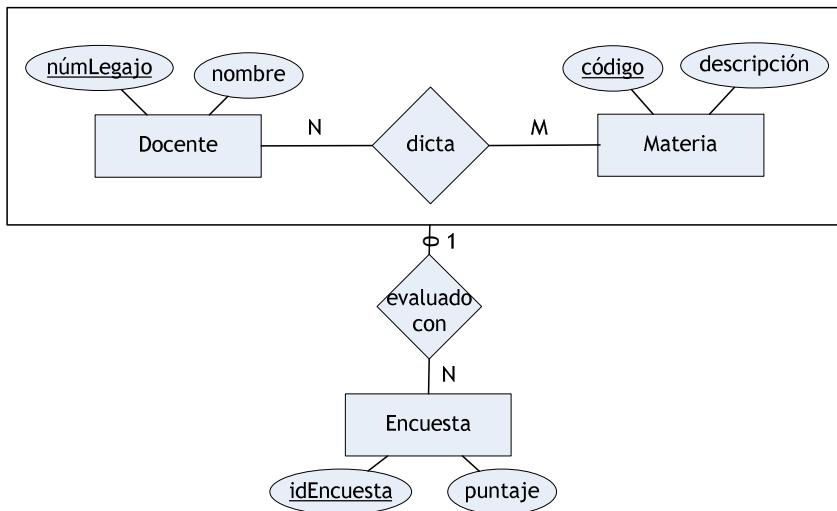
En algunas ocasiones, durante el proceso de modelado surge la necesidad de representar interrelaciones donde participan otras interrelaciones y, por lo que vimos hasta ahora, en nuestro MER esto no está permitido.

Consideremos la siguiente situación: se registran datos de los docentes y de las materias dictadas. Una materia puede ser dictada por muchos docentes y un docente puede dictar muchas materias. Además, en algunos cursos (*no en todos*) se realizan encuestas de evaluación docente. Cada encuesta corresponde a un docente que dicta una materia, y cada docente en una materia puede tener más de una encuesta (o ninguna).

Para modelar este enunciado utilizamos una entidad *Docente*, otra *Materia* y otra *Encuesta* (con las correspondientes claves y atributos descriptivos). Ahora, ¿cómo las relacionamos?

Si pensáramos en una ternaria entre *Docente*, *Materia* y *Encuesta*, estaríamos ante un inconveniente. Como vimos anteriormente, en una tupla de la ternaria **siempre** están los 3 elementos presentes. Por ejemplo, no podemos tener una tupla con 2 elementos y el tercero nulo. Es decir, no podemos tener un docente con una materia sin encuesta (y es justamente lo que queremos modelar).

Aquí es donde aparece en escena la agregación. Entre *Docente* y *Materia* podemos usar una interrelación binaria convencional N:M (*dicta*). Esta interrelación podemos verla como una especie de entidad y relacionarla con *Encuesta* utilizando una binaria (*evaluado con*). Así quedaría representado:



Notar que la binaria *evaluado con* tiene participación parcial del lado de la agregación (esto indica que algunos pares *docente-materia* pueden no tener *encuesta* asociada).

La agregación es una abstracción en la cual una interrelación (junto con sus entidades vinculadas) es tratada como una entidad de alto nivel y puede participar de interrelaciones. Por supuesto, las entidades vinculadas a una agregación también pueden tener sus propias interrelaciones individualmente.

El concepto de agregación se denota con un rectángulo conteniendo a la interrelación y sus entidades vinculadas.

2.4.6 Restricciones adicionales al modelo

Muchas veces, tenemos conocimiento de restricciones que no pueden representarse en nuestro diagrama pero sabemos que constituyen un invariante de nuestro modelo. En este caso, requeriremos que dichas restricciones sean explicitadas en lenguaje natural complementando al modelo.

En nuestro ejemplo de los profesores y directores de los departamentos (sección 2.3.3.9), existiría la siguiente restricción adicional que podríamos enunciar como sigue:

“Todo director de un departamento debe ser profesor de ese departamento”

Es muy común que se requieran restricciones adicionales para interrelaciones “circulares” entre entidades.

3 Del Modelo Entidad-Relación al Modelo Relacional

Una vez que ha sido realizado el *diseño conceptual* de la base de datos, es necesario transformarlo en el *diseño lógico*, dando un paso más hacia la implementación física.

Para expresar el esquema lógico utilizamos un modelo de datos lógico, es decir, un conjunto de herramientas formales que nos permitirán especificar la estructura lógica de la base de datos junto con una descripción de alto nivel de la implementación.

3.1 Modelo Lógico Relacional

El Modelo Lógico Relacional o Modelo Relacional (MR), presentado por Codd en 1970, está ampliamente difundido entre las implementaciones de sistemas de gestión de base de datos. Se basa en una estructura de datos sencilla denominada *relación* y lo sustenta una sólida base teórica.

En este modelo, una base de datos es un conjunto de *relaciones*.

Intuitivamente, una *relación* puede pensarse como una tabla, con filas y columnas. Cada fila, a la que denominaremos *tupla* está formada por un conjunto de valores de datos relacionados que representan un hecho de la realidad. Cada columna de la tabla representa un *atributo*, asociado a un conjunto de valores posibles que puede tomar. A este conjunto lo llamamos *dominio* del atributo.

El nombre asociado a la tabla y cada nombre asociado a una columna o atributo, nos ayudan a interpretar el significado de los valores en cada tupla. El siguiente ejemplo presenta la relación *Empleado*, conformada por atributos *númLegajo*, *nombre*, *apellido*, *fechaNacimiento*, entre otros. La primera fila/tupla nos dice que el empleado con legajo 1102 se llama Martín Gómez y nació el 25 de enero de 1980.

Empleado

| <i>númLegajo</i> | <i>nombre</i> | <i>apellido</i> | ... | <i>fechaNacimiento</i> |
|------------------|---------------|-----------------|-----|------------------------|
| 1102 | Martín | Gómez | | 25/01/1980 |
| 1105 | Natalia | Rodriguez | | 05/04/1975 |
| 1110 | Manuel | Pineda | | 03/10/1979 |

El conjunto de datos de la relación en un momento dado se denomina *instancia de relación* y la estructura de la relación, es decir su nombre y lista de atributos, se denomina *esquema de relación*.

A continuación describimos los esquemas de relación *Empleado* y *Departamento*, los cuales permiten almacenar los datos de los empleados y departamentos de una organización, además de especificar en qué departamento trabaja cada empleado, si es que está asignado a alguno:

Esquemas de relación

Empleado(númLegajo, nombre, apellido, fechaNacimiento, domicilio, númDept)

Departamento(númDept, nombre)

Los atributos subrayados con línea continua son identificadores y constituyen la **clave primaria** de cada esquema.

Los atributos subrayados con línea punteada constituyen una **clave foránea**, es decir, hacen referencia a valores existentes en atributos *clave* de otra relación. En algunas ocasiones que detallaremos más adelante, las claves foráneas pueden tomar valor nulo.

El conjunto de esquemas de relación, junto con las restricciones adicionales asociadas, constituyen el diseño lógico de la base de datos. Es deseable y necesario que el diseño lógico cumpla con ciertas propiedades y no presente anomalías. El proceso de *normalización* es una herramienta que posibilita arribar a un buen diseño de base de datos relacional.

3.2 Transformación del MER al MR

3.2.1 Entidades Fuertes

3.2.1.1 Caso de atributos simples:

Cada conjunto de entidades fuertes de un MER se asocia con un esquema de relación en el modelo relacional, que mantiene el nombre del conjunto de entidades, sus atributos y la clave primaria. A continuación, un ejemplo:

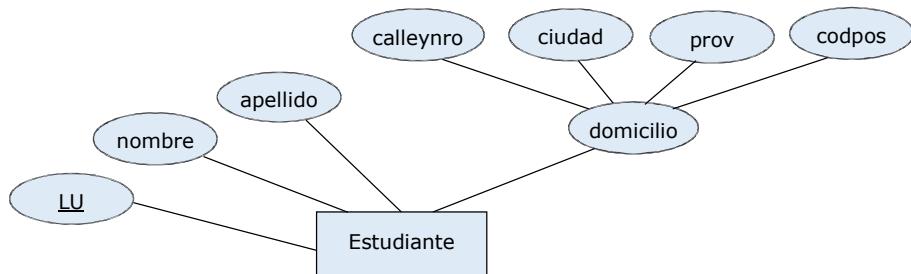


Esquemas resultantes

Estudiante(LU, nombre, apellido)

3.2.1.2 Caso de atributos compuestos:

Cada subparte del atributo compuesto se transforma en un atributo simple. La raíz del atributo compuesto se descarta. Veamos un ejemplo:

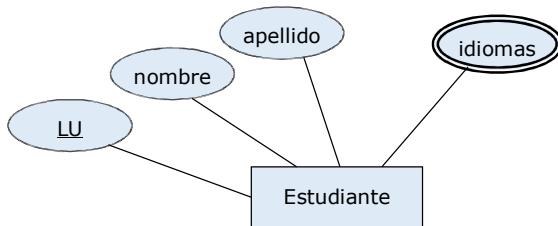


Esquemas resultantes

Estudiante(LU, nombre, apellido, calleynro, ciudad, prov, codpos)

3.2.1.3 Caso de atributos multivaluados:

La regla que se utiliza ahora crea una nueva relación que se identifica por el atributo multivaluado y la clave primaria de la entidad a la que pertenece. Veamos un ejemplo:



Esquemas resultantes

Estudiante(LU, nombre, apellido)

Idiomas (LU, idioma)

Notar que el atributo *LU* en el esquema *Idiomas* hace referencia al atributo con el mismo dominio en *Estudiante*, con lo cual los valores que tome el primero deberán existir en la relación *Estudiante* para mantener la integridad de la base de datos (esto recibe el nombre de **integridad referencial**).

Para cada uno de los esquemas, especificaremos las claves candidatas (CK), primarias (PK) y foráneas (FK) en el caso de existir.

Estudiante PK= CK={LU}

Idiomas PK= CK = {(LU,idioma)}

FK = {LU}

Además, explicitaremos las siguientes **restricciones adicionales**:

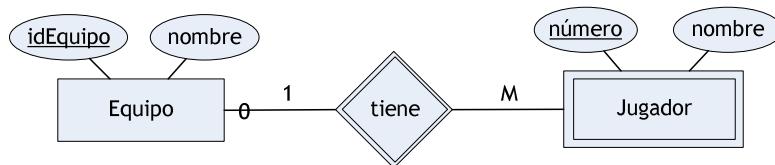
Estudiante.LU puede no estar en *Idiomas.LU*

Idiomas.LU debe estar en *Estudiante.LU*

Notar que escribimos los nombres de los atributos precedidos por el nombre del esquema de relación al que pertenecen.

3.2.2 Entidades Débiles

Una entidad débil se asocia con un esquema de relación en el MR, que mantiene el mismo nombre, sus atributos y presenta como clave primaria la clave primaria de la entidad fuerte asociada junto con la clave parcial de la débil.



Esquemas resultantes

Equipo(idEquipo, nombre)

Jugador(idEquipo, número, nombre)

Notar que el atributo *idEquipo* en el esquema *Jugador* hace referencia al atributo con el mismo dominio en *Equipo*, con lo cual los valores que tome el primero deberán existir en la relación *Equipo* para mantener la integridad de la base de datos.

Para cada uno de los esquemas, especificamos las claves candidatas (CK), primarias (PK) y foráneas (FK) en el caso de existir.

Equipo PK= CK={idEquipo}

Jugador PK= CK = {(idEquipo,número)}

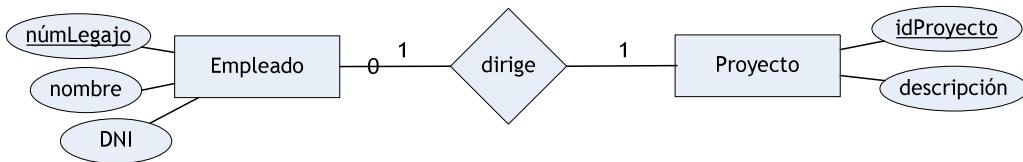
FK = {idEquipo}

Además, explicitamos las siguientes **restricciones adicionales**:

Equipo.idEquipo puede no estar en *Jugador.idEquipo*

Jugador.idEquipo debe estar en *Equipo.idEquipo*

3.2.3 Interrelaciones uno a uno (1:1)



La regla que se utiliza para la transformación en este caso plantea que se incorpore la clave primaria de una de las entidades como clave foránea de la otra entidad.

Esquemas resultantes

Empleado(númLegajo, nombre, DNI)

PK = {númLegajo}

CK = {númLegajo, DNI}

Proyecto(idProyecto, descripción, númLegajo)

PK = CK = {idProyecto}

FK = {númLegajo}

Notar que el DNI también identifica a un empleado, por lo que es una clave candidata.

Establecemos las siguientes **restricciones adicionales**:

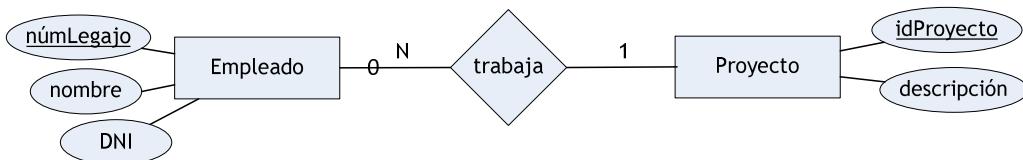
Empleado.númLegajo puede no estar en *Proyecto.númLegajo*

Proyecto.númLegajo debe estar en *Empleado.númLegajo*

IMPORTANTES ¿Por qué la FK está del lado del *Proyecto*? Si hubiéramos decidido colocarla en *Empleado*, hubiésemos conseguido una gran cantidad de valores nulos en ese atributo ya que no todos los empleados dirigen un proyecto (pero un proyecto siempre es dirigido por un empleado). En las 1:1 siempre se debe utilizar este criterio para decidir el lugar de la FK.

3.2.4 Interrelaciones uno a muchos (1:N)

Siguiendo con el ejemplo de empleados y proyectos, ahora queremos modelar el hecho de que un empleado puede estar afectado a uno o ningún proyecto, en cambio, un proyecto tiene asignados varios empleados.



La regla que se usa aquí incorpora la clave primaria de la entidad “uno” en la entidad “muchos” como clave foránea.

Esquemas resultantes

Empleado(númLegajo, nombre, DNI, idProyecto)

PK = {númLegajo}

CK = {númLegajo, DNI}

FK = {idProyecto}

Proyecto(idProyecto, descripción)

PK = CK = {idProyecto}

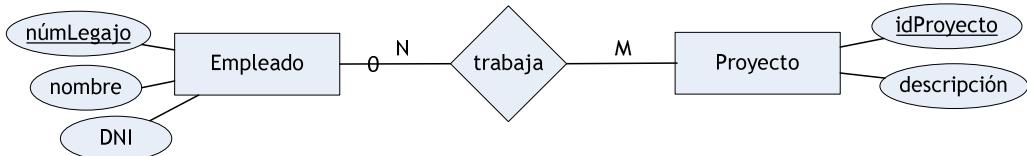
Establecemos las siguientes **restricciones adicionales**:

Empleado.idProyecto es nulo o está en *Proyecto.idProyecto*

Proyecto.idProyecto debe estar en *Empleado.idProyecto*

3.2.5 Interrelaciones muchos a muchos (N:M)

Ahora suponemos que un empleado puede trabajar en varios proyectos a la vez.



La regla que se utiliza ahora crea una nueva relación que se identifica por las claves primarias de las dos entidades interrelacionadas. En este caso no tendrá ningún otro atributo, ya que la interrelación no lo tiene.

Esquemas resultantes

Empleado(númLegajo, nombre, DNI)

PK = {númLegajo}

CK = {númLegajo, DNI}

Proyecto(idProyecto, descripción)

PK = CK = {idProyecto}

Trabaja(númLegajo, idProyecto)

PK = CK = {(númLegajo, idProyecto)}

FK = {númLegajo, idProyecto}

IMPORTANTES Notar los paréntesis en la PK de *Trabaja*. Esto significa que la clave es compuesta y la conforman esos atributos al mismo tiempo. En cambio, las FK no cuentan con los paréntesis ya que son 2 simples.

Establecemos las siguientes **restricciones adicionales**:

Empleado.númLegajo puede no estar en *Trabaja.númLegajo*

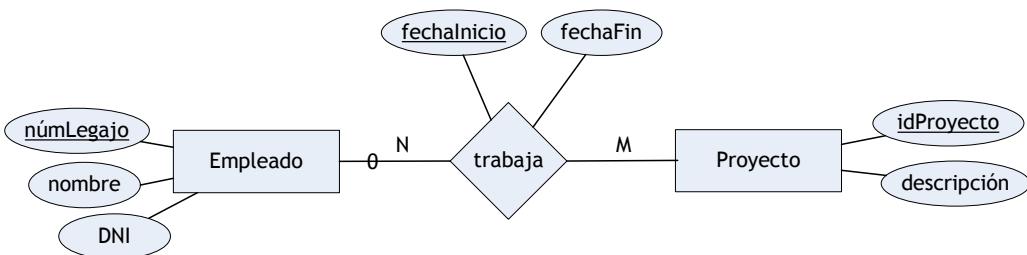
Proyecto.idProyecto debe estar en *Trabaja.idProyecto*

Trabaja.númLegajo debe estar en *Empleado.númLegajo*

Trabaja.idProyecto debe estar en *Proyecto.idProyecto*

3.2.6 Interrelaciones muchos a muchos (N:M) con atributos

Otro ejemplo de transformación de relaciones N:M, es el siguiente: supongamos que tenemos el caso anterior pero queremos modelar la historia de las asignaciones de los empleados en los proyectos.



La regla de transformación que se aplica en este caso es similar a la anterior, con la diferencia que el atributo identificatorio de la interrelación se incorpora a la nueva relación como parte de la clave primaria y el atributo descriptivo de la interrelación como un atributo descriptivo de la nueva relación.

Esquemas resultantes

Empleado(númLegajo, nombre, DNI)

PK = {númLegajo}

CK = {númLegajo, DNI}

Proyecto(idProyecto, descripción)

PK = CK = {idProyecto}

Trabaja(númLegajo, idProyecto, fechaInicio, fechaFin)

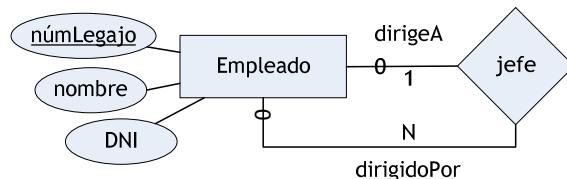
PK = CK = {(númLegajo, idProyecto, fechaInicio)}

FK = {númLegajo, idProyecto}

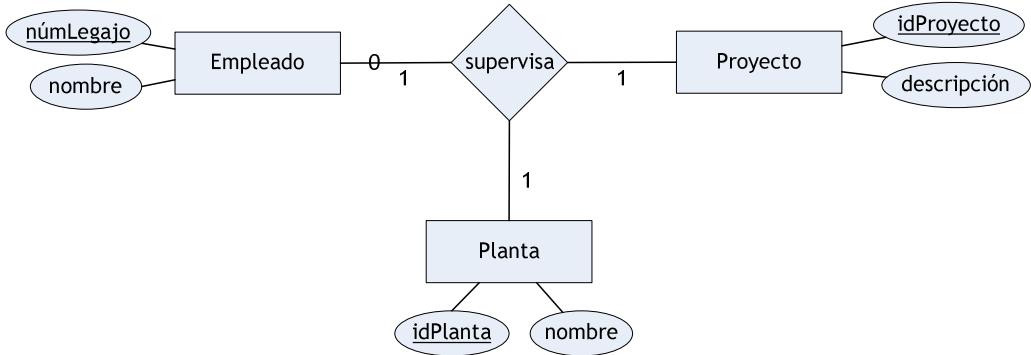
Las **restricciones adicionales** son las mismas que para el caso N:M.

3.2.7 Interrelaciones unarias

Hasta ahora hemos visto cómo transformar interrelaciones binarias al modelo relacional. Las reglas para transformar interrelaciones unarias son *análogas a las binarias*, lo único que cambia es que ahora tenemos un solo conjunto de entidades para transformar, en lugar de dos.



empleado en una planta puede supervisar un proyecto; un empleado que supervisa un proyecto lo hace sólo en una planta. Además, puede haber empleados que no supervisen nada. Modelamos la situación utilizando una interrelación ternaria con cardinalidad 1:1:1.



Esquemas resultantes

Empleado(númLegajo, nombre)

PK = CK = {númLegajo}

Proyecto(idProyecto, descripción)

PK = CK = {idProyecto}

Planta(idPlanta, nombre)

PK = CK = {idPlanta}

Supervisa(númLegajo, idProyecto, idPlanta)

PK = {(númLegajo, idProyecto)}

CK = {(númLegajo, idProyecto), (númLegajo, idPlanta), (idProyecto, idPlanta)}

FK = {númLegajo, idProyecto, idPlanta}

Si analizamos la primera clave candidata, para cada combinación de valores de número de legajo e id de proyecto, podremos determinar el id de planta asociado, que será único. El mismo razonamiento se puede aplicar para el resto de las claves candidatas.

Con algún criterio, que surge del relevamiento del problema, se ha elegido la clave candidata (**númLegajo, idProyecto**) como clave primaria del esquema *Supervisa*.

Establecemos las siguientes **restricciones adicionales**:

Supervisa.númLegajo debe estar en *Empleado.númLegajo*

Supervisa.idProyecto debe estar en *Proyecto.idProyecto*

Supervisa.idPlanta debe estar en *Planta.idPlanta*

Empleado.númLegajo puede no estar en *Supervisa.númLegajo*

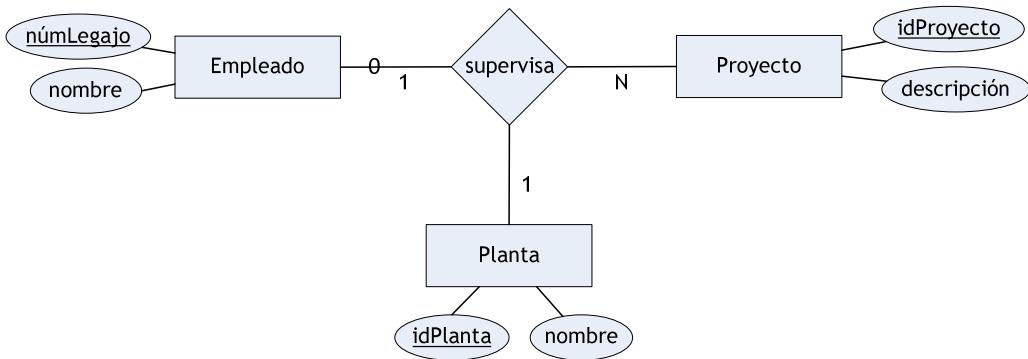
Proyecto.idProyecto debe estar en *Supervisa.idProyecto*

Planta.idPlanta debe estar en *Supervisa.idPlanta*

IMPORTANTÉ Recordar que en toda ternaria, siempre deben estar presentes los 3 integrantes de la ternaria. En este caso, en ninguna de las tuplas de *Supervisa*, el *númLegajo*, *idProyecto* o *idPlanta* podrán tomar valores nulos.

3.2.8.2 Interrelaciones ternarias 1:N

Continuando con el ejemplo anterior, ahora un empleado en una planta puede supervisar varios proyectos. Modelamos esta situación utilizando una interrelación ternaria con cardinalidad 1:1:N.



Esquemas resultantes

Empleado(númeroLegajo, nombre)
PK = CK = {númeroLegajo}

Proyecto(idProyecto, descripción)
PK = CK = {idProyecto}

Planta(idPlanta, nombre)
PK = CK = {idPlanta}

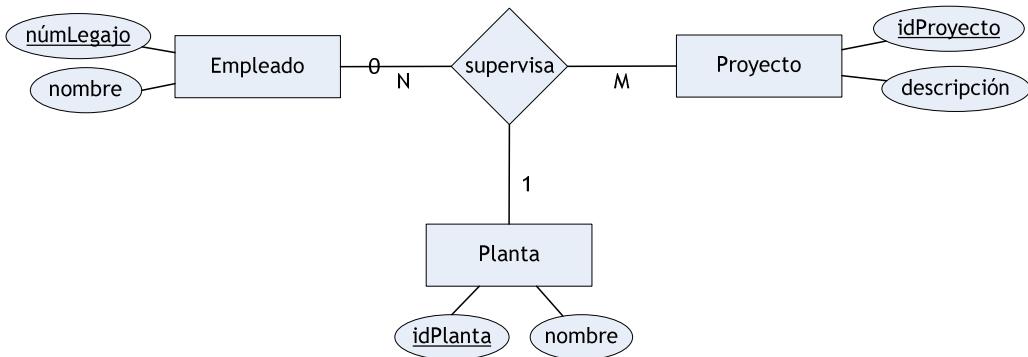
Supervisa(númeroLegajo, idProyecto, idPlanta)
PK = {(númeroLegajo, idProyecto)}
CK = {(númeroLegajo, idProyecto), (idProyecto, idPlanta)}
FK = {númeroLegajo, idProyecto, idPlanta}

Notar que, por la cardinalidad de la interrelación, un valor de número de legajo asociado con un valor de número de proyecto, determinan un único valor de número de planta. De la misma forma, un número de planta asociado a un número de proyecto determinan un único valor de número de legajo de empleado.

Establecemos las mismas **restricciones adicionales** que en el caso anterior.

3.2.8.3 Interrelaciones ternarias 1:N:M

Retomando el ejemplo anterior, ahora un proyecto en una planta puede ser supervisado por más de un empleado. Usamos una ternaria 1:N:M.



Esquemas resultantes

Empleado(númeroLegajo, nombre)
PK = CK = {númeroLegajo}

Proyecto(idProyecto, descripción)
PK = CK = {idProyecto}

Planta(idPlanta, nombre)

PK = CK = {idPlanta}

Supervisa(númLegajo, idProyecto, idPlanta)

PK = CK = {(númLegajo, idProyecto)}

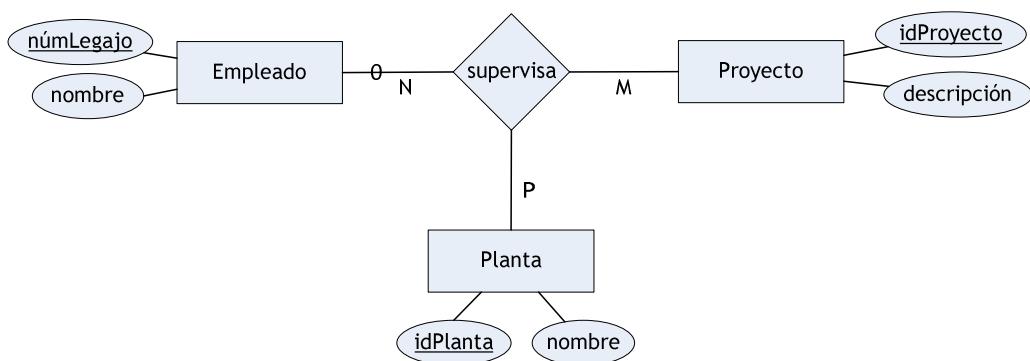
FK = {númLegajo, idProyecto, idPlanta}

A diferencia de los casos anteriores, existe una sola clave candidata. Dado un empleado y un proyecto, la planta queda determinada únicamente.

Las **restricciones adicionales** son las mismas que en el caso anterior.

3.2.8.4 Interrelaciones ternarias N:M:P

Siguiendo con el ejemplo anterior, ahora un empleado que supervisa un proyecto puede hacerlo en más de una planta. Usamos una ternaria N:M:P.



Esquemas resultantes

Empleado(númLegajo, nombre)

PK = CK = {númLegajo}

Proyecto(idProyecto, descripción)

PK = CK = {idProyecto}

Planta(idPlanta, nombre)

PK = CK = {idPlanta}

Supervisa(númLegajo, idProyecto, idPlanta)

PK = CK = {(númLegajo, idProyecto, idPlanta)}

FK = {númLegajo, idProyecto, idPlanta}

La única clave candidata de *Supervisa* ahora pasa a tener a los 3 participantes, ya que ningún par alcanza para determinar al restante.

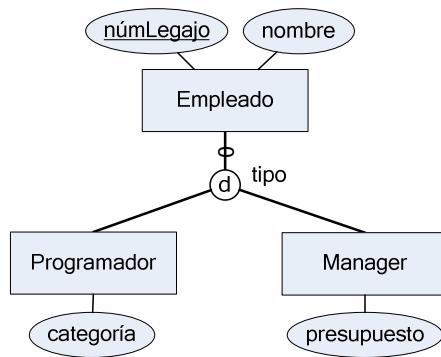
Como antes, las **restricciones adicionales** son las mismas.

3.2.9 Jerarquías

Las reglas de transformación de las jerarquías plantean crear un esquema de relación asociado a la superentidad y un esquema por cada subentidad. Los atributos clave definidos para la superentidad estarán incluidos en los esquemas de relación asociados a cada subentidad, junto con los atributos propios de éstas.

3.2.9.1 Jerarquía Disjunta

Ahora veamos cómo transformar una jerarquía disjunta.

Esquemas resultantes

Empleado(númLegajo, nombre, tipo)
 PK = CK = {númLegajo}

Programador(númLegajo, categoría)
 PK = CK = FK = {númLegajo}

Manager(númLegajo, presupuesto)
 PK = CK = FK = {númLegajo}

IMPORTANTES Notar que se agrega el atributo *tipo* en el esquema *Empleado*, que permite particionar el conjunto de empleados. Además, observar que en las subentidades *númLegajo* cumple el rol de PK y también de FK.

Establecemos las siguientes **restricciones adicionales**:

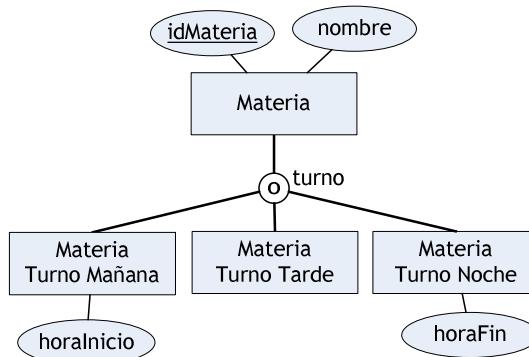
Programador.númLegajo debe estar en *Empleado.númLegajo*

Manager.númLegajo debe estar en *Empleado.númLegajo*

Empleado.númLegajo puede estar en *Programador.númLegajo* o (exclusivo) *Manager.númLegajo* o (exclusivo) no estar en ninguno de los dos anteriores.

3.2.9.2 Jerarquía con Solapamiento

Transformemos esta jerarquía sobre materias.

Esquemas resultantes

Materia(idMateria, nombre)
 PK = CK = {idMateria}

MateriaTurnoMañana(idMateria, horaInicio)
 PK = CK = FK = {idMateria}

MateriaTurnoTarde(idMateria)
 PK = CK = FK = {idMateria}

MateriaTurnoNoche(idMateria, horaFin)

PK = CK = FK = {idMateria}

IMPORTANTÉ Notar que en el caso de jerarquías con solapamiento, no se agrega un atributo en la superentidad con el nombre del discriminante ya que un elemento de la superentidad puede estar en más de una subentidad. Además, por más que no haya atributos ni interrelaciones en *MateriaTurnoTarde*, es necesario crear un esquema porque sino dejaríamos de modelar a las materias de la tarde.

Agregamos las siguientes **restricciones adicionales**:

MateriaTurnoMañana.idMateria debe estar en *Materia.idMateria*

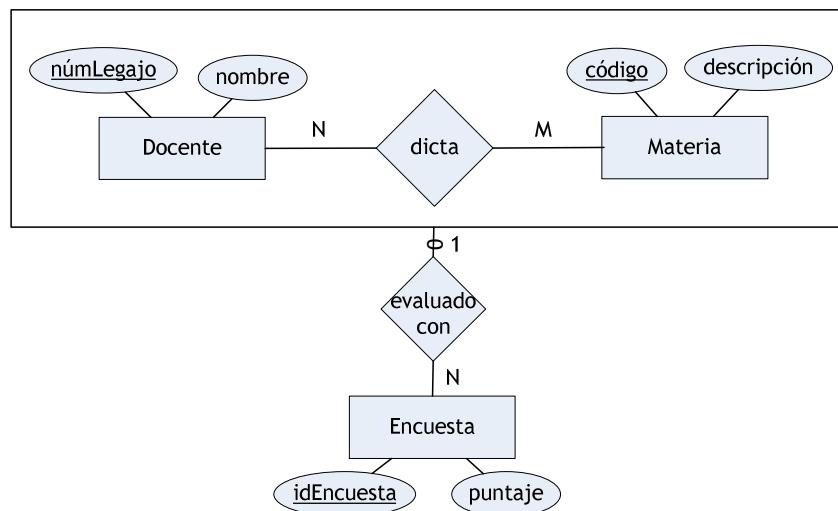
MateriaTurnoTarde.idMateria debe estar en *Materia.idMateria*

MateriaTurnoNoche.idMateria debe estar en *Materia.idMateria*

Materia.idMateria puede estar en *MateriaTurnoMañana.idMateria*, *MateriaTurnoTarde.idMateria* y *MateriaTurnoNoche.idMateria*

3.2.10 Agregación

Las reglas de transformación de la agregación al modelo relacional son parecidas a las que utilizamos para interrelaciones binarias. Se transforman tomando a la *agregación como si fuese una entidad de la binaria*.



Esquemas resultantes

Docente(númLegajo, nombre)

PK = CK = {númLegajo}

Materia(código, descripción)

PK = CK = {código}

Dicta(númLegajo, código)

PK = CK = {(númLegajo,código)}

FK = {númLegajo, código}

Encuesta(idEncuesta, puntaje, númLegajo, código)

PK = CK = {idEncuesta}

FK = {(númLegajo, código)}

IMPORTANT Notar que la binaria *evaluado con* se transformó como si participaran dos entidades: *Encuesta* y *Dicta* (claro que esta última no es una entidad sino una agregación). Por este motivo, la FK de *Encuesta* no es hacia *Docente* ni *Materia*, sino directamente hacia la agregación.

Establecemos además las siguientes **restricciones adicionales**:

Dicta.númLegajo debe estar en *Docente.númLegajo*

Dicta.código debe estar en *Materia.código*

Docente.númLegajo debe estar en *Dicta.númLegajo*

Materia.código debe estar en *Dicta.código*

(*Encuesta.númLegajo*, *Encuesta.código*) debe estar en (*Dicta.númLegajo*, *Dicta.código*)

(*Dicta.númLegajo*, *Dicta.código*) puede no estar en (*Encuesta.númLegajo*, *Encuesta.código*)

Como mencionamos previamente, los demás casos de agregación se transforman considerando a la agregación como si fuera una entidad y utilizando las reglas de las binarias.

4 Bibliografía

- Teorey T, Yang D, Fry J - *A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model* - Computing Surveys, Vol. 18, No. 2, 1986
- Elmasri, Navathe - *Fundamentals of Database Systems*, 4th Ed., Addison Wesley, 2003
- Ramakrishnan, Gherke - *Database Management Systems*, 3rd Ed. Mc Graw-Hill, 2003
- Ullman - *Principles of Database and Knowledge Base Systems*, Computer Science Press, 1988
- Chen, P.P.-S., *The entity-relationship model -- toward a unified view of data*. ACM Transactions on Database Systems, 1(1), 9-36. – 1976.
- Enrico Franconi - *Logic – The Meaning of Entity-Relationship Diagrams* – Free University of Bozen-Bolzano.
- Liwu Li, Xin Zhao - *UML Specification of Relational Database* - Journal of Object Technology, vol. 2, Nº , September-October 2003, pp. 87-100