Introduction aux bases de données avancées

Master GL/RT - 2019- 2020

Abdelkader OUARED

ouared.aek@gmail.com

Pourquoi devriez-vous suivre ce cours

- Les bases de données sont encore domaine de recherche actif.
- Les développeurs de SGBD sont en demande et il existe de nombreux problèmes complexes non résolus dans la gestion et le traitement des données.
- Si vous êtes assez bon pour écrire du code pour un SGBD, vous pouvez écrire du code sur presque tout le reste.

Objectifs du cours

En savoir plus sur les pratiques modernes en matière d'interne de base de données, de programmation système et d'applications de datamining

Les étudiants deviennent compétents dans:

→ Programmation dans le SGBD

→ Mise en œuvre du modèle physique correct

→ Test des performances des requêtes + Optimisation SQL

→ Modélisation et implémentation de datawarehouse

→ Applications du Machine Learning

À propos du module

- Contenu de la matière
  - Concurrence, transactions, optimisation de requêtes
  - Programmation des bases de données
  - Modèles avancés de données
  - Bases de données XML
  - Entrepôt de données
  - Fouille de données
  - Intégration/Interopérabilité

- Connaissances préalables recommandées

  Concepts fondamentaux des bases de données

Déroulement du module

- 1 cours hebdomadaire (1h 30) sur les principaux concepts.
- 1 séance de TP hebdomadaire (1h 30) de prise en main

Évaluation

- Travaux individuels (TP).
- Mini-Project
- Soumission des projets: https://github.com/OUARED-A/Course-Masters-GL-RT-DBA--2019-2020
- Les projets seront implémentés dans les bases de données MySQL, Oracle, PostgreSQL et nous utilisons Python pour les applications de machine learning.
- Examen Final

Cours 1
Un tour d'horizon de l’évolution des systèmes de base de données

Abdelkader OUARED

a_ouared@esi.dz

Agenda d'Aujourd 'hui

- La notion de cycle de vie de BD
- Les principaux modèles de données
- L’évolution des systèmes de bases de données

Turing Awards in Data Management

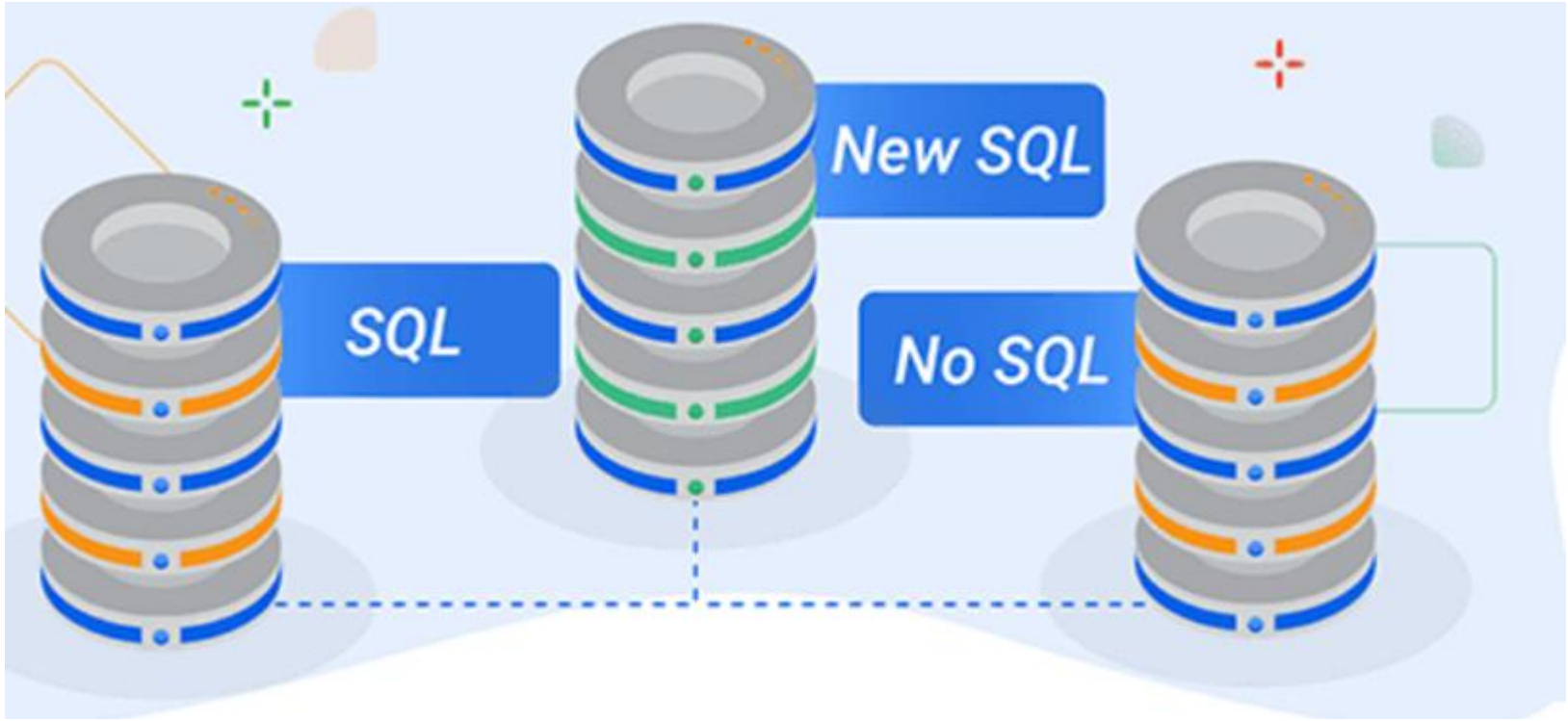
Charles Bachman, 1973
IDS and *CODASYL*

Ted Codd, 1981
Relational model

Jim Gray, 1998
Transaction processing

Michael Stonebraker, 2014
INGRES and Postgres

Evolution des systèmes de bases de données (SQL, NoSQL, NewSQL)



Évolution du système de base de données:
~ Une longue histoire

The "SQL vs. NoSQL"

History Repeats Itself

"Relational vs. CODASYL".

One Size Fits All
An Idea Whose Time Has Come and Gone
by
Michael Stonebraker

MICHAEL STONEBRAKER, 2014
-Research and products are central to many relational database systems

In Science
- Turing Awardees

1960S – IBM IMS → Hierarchical data model.

1970s – RELATIONAL MODEL

1980s – The relational model wins

Transaction processing **1998**

2000s – INTERNET BOOM

2000s – DATA WAREHOUSES

2009– NoSQL

2010s – NewSQL

2010s – HYBRID SYSTEMS

- **NoSQL projects**
- Graph DB Neo4j (2000)
- CouchDB (2005)
- MongoDB (2007)

CHARLES BACHMAN, 1973
-Known for his work in the early development of database management systems.

EDGAR CODD, 1981
-Invented Relational model (RM), the theoretical basis for relational databases and relational database management systems.

Gray

Stonebraker

Ellison

JAMES GRAY, 1998
-For seminal contributions to database and transaction processing research

DB-Engine*: **316** DBMS en 2016
*http://db-engines.com/en/ranking

Cycle de vie de base de données

Aujourd'hui

➔ La gestion de données est devenue complexe ☹

real world systems and processes

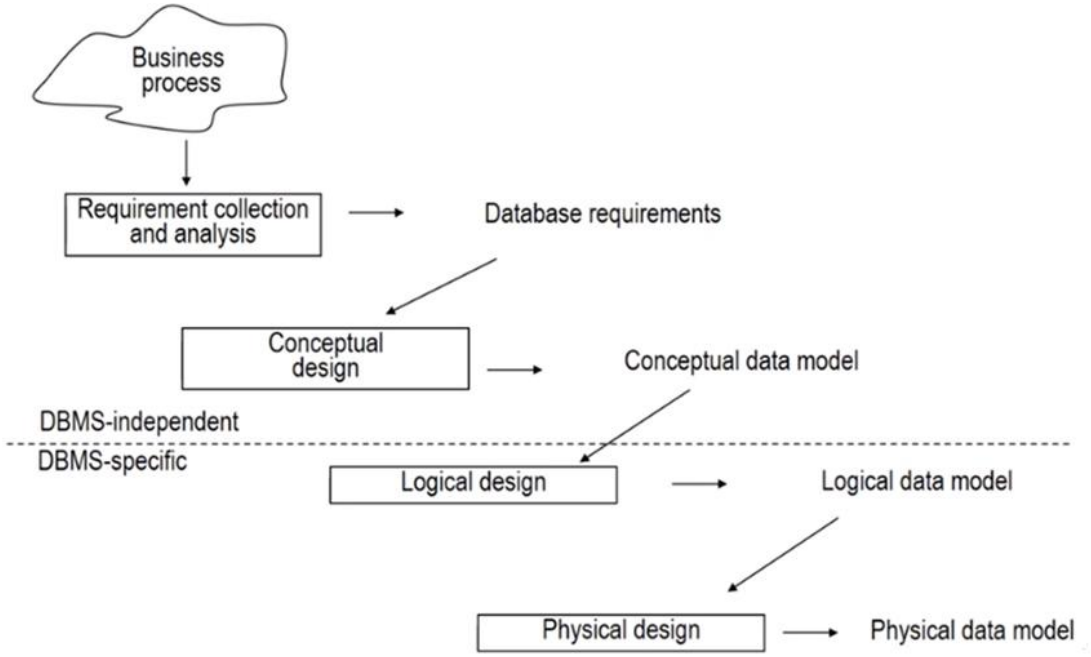
Transformation

a framework was need

Structures: be implemented in a computer ☺

ANSI/X3/SPARC, 1978:

- External model
- Conceptual data model
- Logical data model
- Internal data model



⇨Les **modèles** sont au **centre** du processus de conception de BD

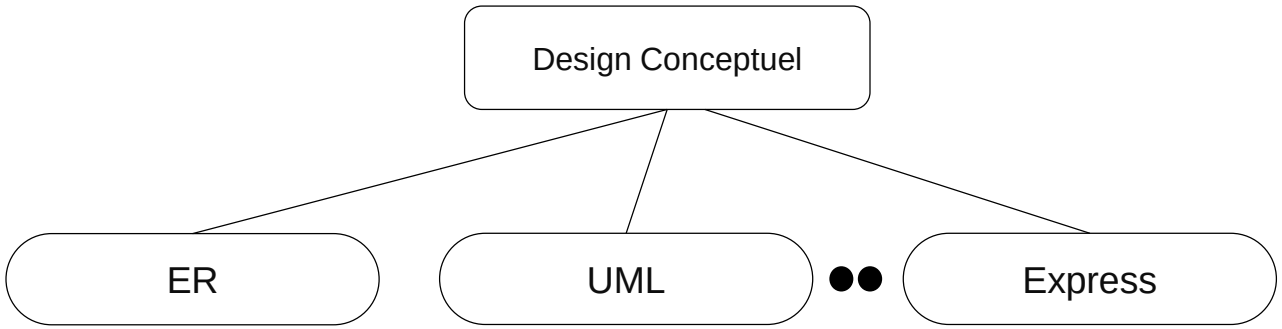
Modèle de données (Data Model)

- It defines the structure of database:
- A database model shows the **conceptual/logical and physical** structure of database
  - How databse is modeled ?
  - How data is connected to each other?
  - Describes the **data relationships**, the **semantics** and the **data integrity constraints**
  - Show how data can be organized, stored and processed in system?
- **Categories of Data Models**
  - **Object based logical Model** (e.g. ER Model, Object Oriented Model)
  - **Record Based logical Model** (e.g. Relational Model)

Modèles Conceptuel de Données (MCD)

❑ Phase conceptuelle:

- Représentation structurée et abstraite de données
- N'inclut pas de détails sur la manière dont les données sont stockées ou les opérations sont implémentées.
- **Langages de modélisation** : *Entité/Association (EA), UML, Express, etc.*
- L'expressivité et la qualité du modèle conceptuel dépend fortement des compétences du concepteur, plutôt que du formalisme utilisé
- Cette phase ne peut s'automatiser totalement



Le Modèle Entité – Association « Entity Relationship (ER) Model »

Entity Relationship (ER) Model by Peter Chen

- Born in Taiwan
- Ph.D from Harvard University in 1973
- Professor at Louisiana State University

A Relational Model of Data for Large Shared Data Banks — 1970

The Entity-Relationship Model - - Toward a Unified View of Data ** — **1976**

- Formalisé en **1976** par **P. Chen**
- Ensemble de concepts pour modéliser les données d'une application d'une entreprise (les liens, la sémantique et les contraintes.)
- Etendu vers E/R généralisé puis vers l'objet
  - L'approche parait plus naturelle ➔ obtention direct de résultat
  - Outil de communication efficace
  - Représentation graphique accessible pour les utilisateur finaux

Peter P. Chen: **The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Syst. 1(1): 9-36 (1976)

Types des Modèles de données: **Modèle Logique**

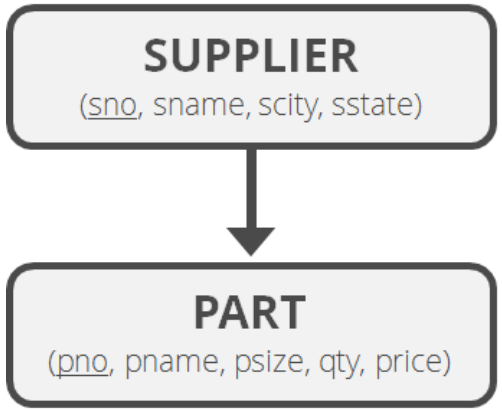
❑ Phase logique

- Transformation (**mapping**) du modèle conceptuel en un modèle logique lié à l'implémentation de BD dans un SGBD,
- **Modèle Logique de Données (MLD)**: fournir les formes explicites (structure de données) que les modèles conceptuels peuvent prendre et constitue la première étape pour avoir un modèle machinal (par ex. hierarchical, network, relational; etc.).
- Cette phase peut ainsi être automatisée par les outils de conception de BD (outils **CASE**)
- formalismes existent pour représenter logiquement les données:
  i. **modèles orientés enregistrements** : le relationnel, multidimensionnel et réseau,
  ii. **orientés-objet :** l'objet et le relationnel-objet, et
  iii. **orientés NoSQL** : les modèles clé-valeur, graphes, documents ou colonnes .

Modèle logique (Hierarchical data model)

Hierarchical Data Model:

- In hierarchical model, records are organized as trees (describing similar entities)
- Each entity has only one parent but can have several children
- Only 1:N relationship types (can be nested)



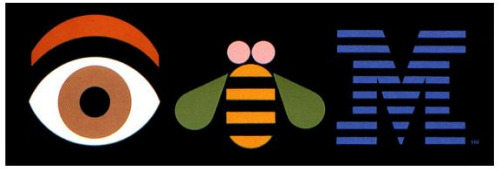
+ Very Structured Organization

⚠ Duplicate Data

⚠ No Independence

1960s – IBM IMS (Message Based Transaction Processor)

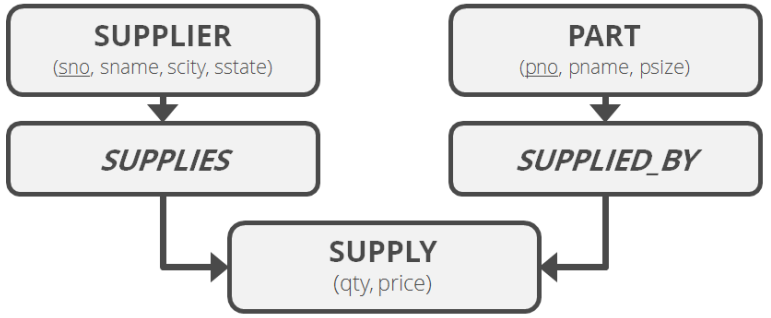
- First database system developed to keep track of purchase orders for Apollo moon mission.
  → Hierarchical data model.
  → Programmer-defined physical storage format.
  → Tuple-at-a-time queries



Modèle logique (Network data model)

CHARLES BACHMAN , 1973

1970s – CODASYL (Conference on Data Systems Languages**)**
COBOL people got together and proposed a standard for how programs will access a database**. Lead by** Charles Bachman**.**
→ Network data model.
→ Tuple-at-a-time queries



- **Network Data model** - many to many relationships.
  - In this data is represented by collection of records and relationships among data are represented by links
  - Record linked together like a family tree
    - Each record type can have **more that one** owner

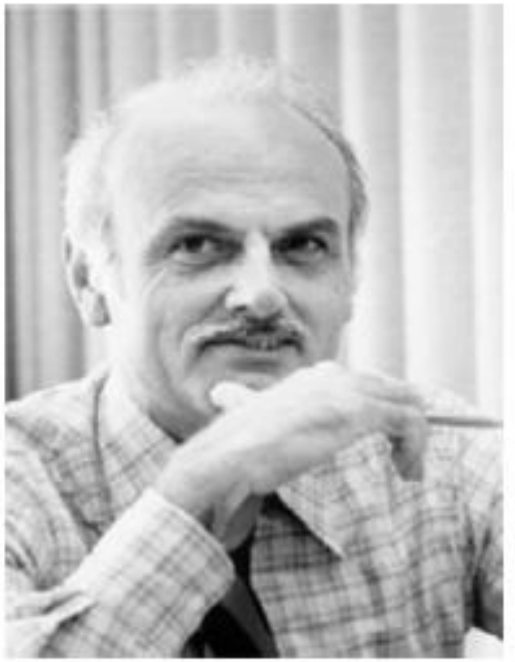
+ Reduces Redundancy

Complex Queries: pointers expensive and difficult to update when inserting and deleting

Modèle logique (Relational Model)

- Les recherches de Codd 1970 : (IBM Research).

- Structure d'une BD relationnelle (ensembles de n-uplets/tables)
- Simple et bien formalisé
- Basé sur l'algèbre relationnelle et le et langage SQL,
- Utilisé par la majorité des systèmes actuels (70% de SI)
- Théorie de la normalisation: 1NF, 2NF, 3NF, BCNF, 4NF, et 5NF



Codd Edgar, 1970

HISTORY REPEATS ITSELF

Old database issues are still relevant today.
The "**SQL vs. NoSQL**" debate is reminiscent of
"**Relational vs. CODASYL**" debate.
Many of the ideas in today's database systems are not new

1970s – RELATIONAL MODEL

- Early implementations of relational DBMS:
  → **System R** – IBM Research
  → **INGRES** – U.C. Berkeley
  → **Oracle** – Larry Ellison


Gray


Stonebraker


Ellison

1980s – RELATIONAL MODEL

- The relational model wins.
  → IBM comes out with DB2 in 1983.
  → SQL becomes the standard.

→ Many new "enterprise" DBMSs
but Oracle wins marketplace.

→ Stonebraker creates Postgres.

1980s – Bases de Données Orientées Objet

Application Code

```
class Student {
    int id;
    String name;
    String email;
    String phone[];
}
```

Student

```
{
  "id": 1001,
  "name": "M.O.P.",
  "email": "ante@up.com",
  "phone": [
      "444-444-4444",
      "555-555-5555"
   ]
}
```

- Avoid "relational-object impedance mismatch" by tightly **coupling objects and database**.
- Few of these original DBMSs from the 1980s still exist today but many of the technologies exist in other forms (JSON, XML)

1990s – Pas de grande évolution

- Aucune avancée majeure dans les systèmes de base de données ou les charges de travail des applications.

→ Microsoft crée Sybase et crée SQL Server.

→ MySQL est écrit en remplacement de mSQL.

→ Postgres obtient le support SQL.

2000s – Bases de données open source

- All the **big players** were heavyweight and expensive.
- **Open-source** databases were missing important features.

- Many companies wrote their own custom middleware to scale out database across single node DBMS instances.

2000s – Data Warehouses

Rise of the special purpose OLAP DBMSs.

→ Distributed / Shared-Nothing

→ Relational / SQL

→ Usually closed-source.

Significant performance benefits from using Decomposition Storage Model (i.e., **columnar, In-Memory Columnstore**)

Star Join Query

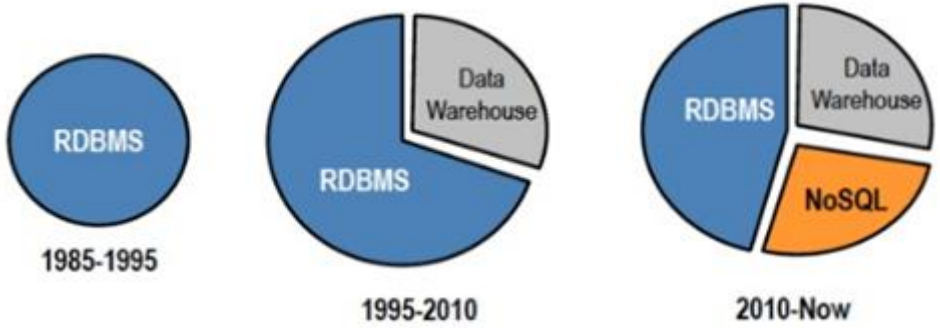
```
select d_year, c_nation,
sum(lo_revenue - lo_supplycost)
 as profit
from date, customer, supplier,
 part, lineorder
 where lo_custkey = c_custkey
 and lo_suppkey = s_suppkey
 and lo_partkey = p_partkey
 and lo_orderdate = d_datekey
 and c_region = 'AMERICA'
 and s_region = 'AMERICA'
 and (p_mfgr = 'MFGR#1'
     or p_mfgr = 'MFGR#2')
 group by d_year, c_nation
 order by d_year, c_nation ;
```

2000s – Systèmes NoSQL

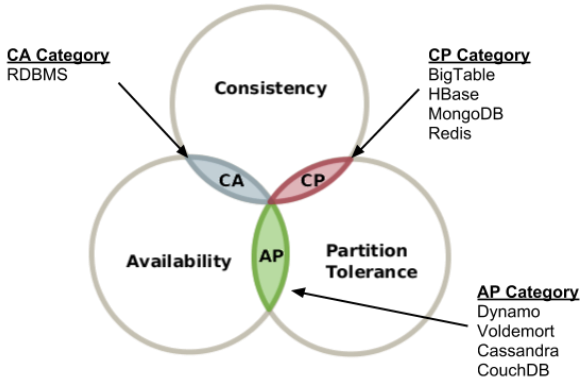
Eras of Database

Why NoSQL?

RDBMS one-size-fits-all needs



- Focus on high-availability & high-scalability (cap theory):
  → Schemaless (i.e., "Schema Last")
  → Non-relational data models (document, key/value, etc)
  → No ACID transactions
  → Custom APIs instead of SQL
  → Usually open-source



ICDE 2005 conference

"One Size Fits All": An Idea Whose Time Has Come and Gone

Michael Stonebraker
Computer Science and Artificial Intelligence Laboratory, M.I.T., and StreamBase Systems, Inc.
stonebraker@csail.mit.edu

Uğur Çetintemel
Department of Computer Science Brown University, and StreamBase Systems, Inc.
ugur@cs.brown.edu

The last 25 years of commercial DBMS development can be summed up in a single phrase: "one size fits all". This phrase refers to the fact that **the traditional DBMS architecture (originally designed and optimized for business data processing) has been used to support many data-centric applications** with widely varying characteristics and requirements. In this paper, we argue that this concept is no longer applicable to the database market, and that the commercial world will fracture into a collection of independent database engines, some of which may be unified by a common front-end parser. We use examples from the stream-processing market and the data-warehouse market to bolster our claims. We also briefly discuss other markets for which the traditional architecture is a poor fit and argue for a critical rethinking of the current factoring of systems services into products.

We can not achieve all the three items
In distributed database systems (center)

2000s – Systèmes NoSQL (Types des BDDs NoSQL)

- Stocker les informations de la façon la mieux adaptée à leur représentation :
  - **Clé-valeur** : chaque objet est identifié par une clé unique constituant la seule manière de le requêter
  - **Colonne** : permet de disposer d'un très grand nb de valeurs sur une même ligne, de stocker des relations « one-to-many », d'effectuer des requêtes par clé (adaptés au stockage de listes : messages, posts, commentaires, ...)
  - **Document** : gestion de collections de documents, composés chacun de champs et de valeurs associées, valeurs pouvant être requêtées (adaptées au stockage de profils utilisateur et des fichiers JSon)
  - **Graphe** : pour gérer des relations multiples entre les objets (adaptés au données issues de réseaux sociaux,…)

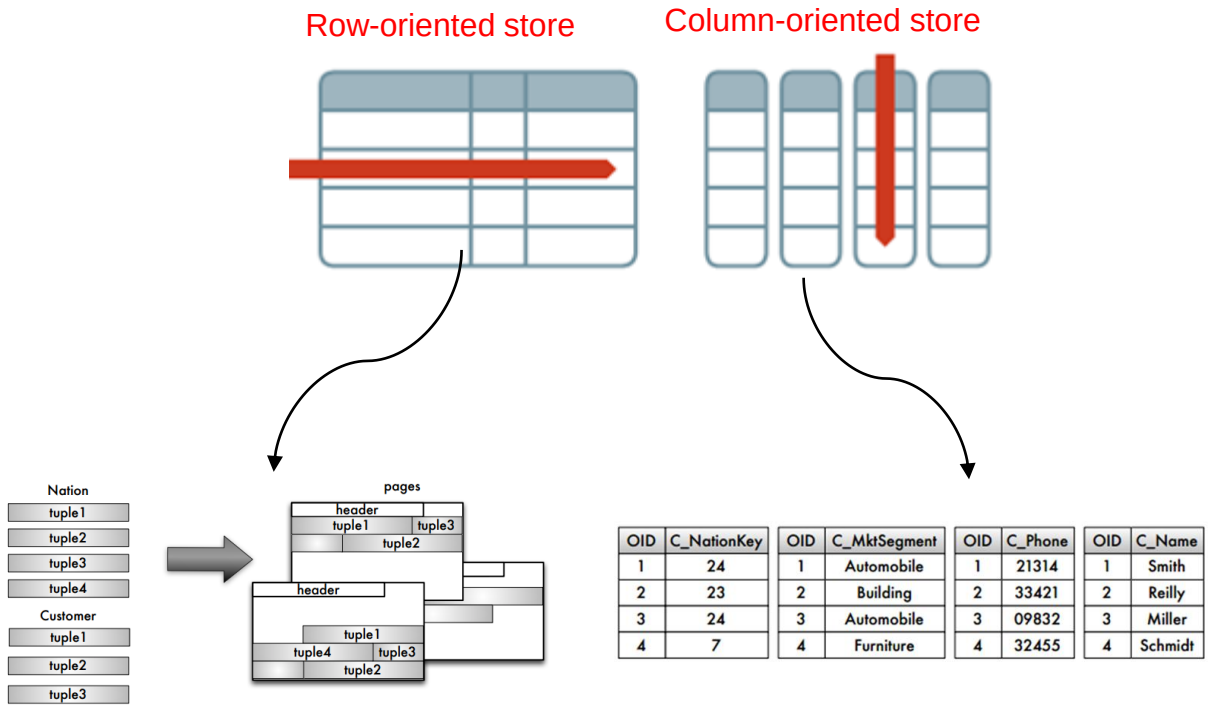
NoSQL practitioners focus on physical data model design

- Physical Data model **represents** the data at data layer or internal layer (low level data structures, records, pointers, Index etc.)
- Physical Data model **describes:**
  - How data are stored
  - How data are scattered and ordered
  - How data would be retrieved from memory
  - spécifiés des structures internes de stockage, des index, des chemins d'accès, des paramètres physiques et l'organisation des fichiers de la BD.
- **Goal of physical design** (the efficiency of data processing)

Modèle de données physique

❑ Physical data model

- Row Store
- Column Store
- Hybrid Store



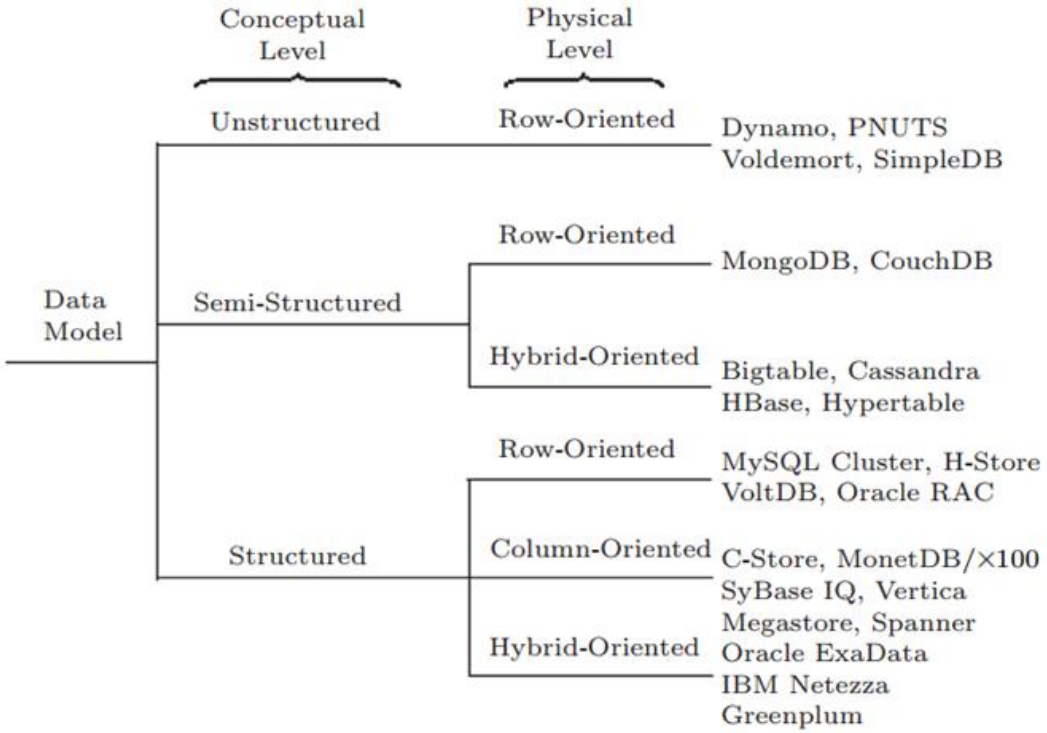
OID	C_NationKey
1	24
2	23
3	24
4	7

OID	C_MktSegment
1	Automobile
2	Building
3	Automobile
4	Furniture

OID	C_Phone
1	21314
2	33421
3	09832
4	32455

OID	C_Name
1	Smith
2	Reilly
3	Miller
4	Schmidt

❑ Taxonomy of Physical data model



Les praticiens NoSQL se concentrent sur la conception de modèles de données physiques

- **NoSQL databases** have been emerged as a revolutionary technology **for modern web-scale** and **cloud-based applications**.
- NoSQL practitioners focus on **physical data model design** rather than **the traditional conceptual / logical data model** process
  - A variety of NoSQL databases are industrialized which have different types of **physical level data models**
  - lack of common standardization among NoSQL databases
  - lack of commonly accepted conceptual model for NoSQL databases
    - ➢ Difficult to choose the right physical data model for specific application.
      - ➢ **A strong need for common conceptual model and logical level data model for those databases**

UMLtoGraphDB: Mapping Conceptual Schemas to Graph Databases

Gwendal Daniel[1], Gerson Sunyé[1], and Jordi Cabot[2]

[1] AtlanMod Team
Inria, Mines Nantes & Lina

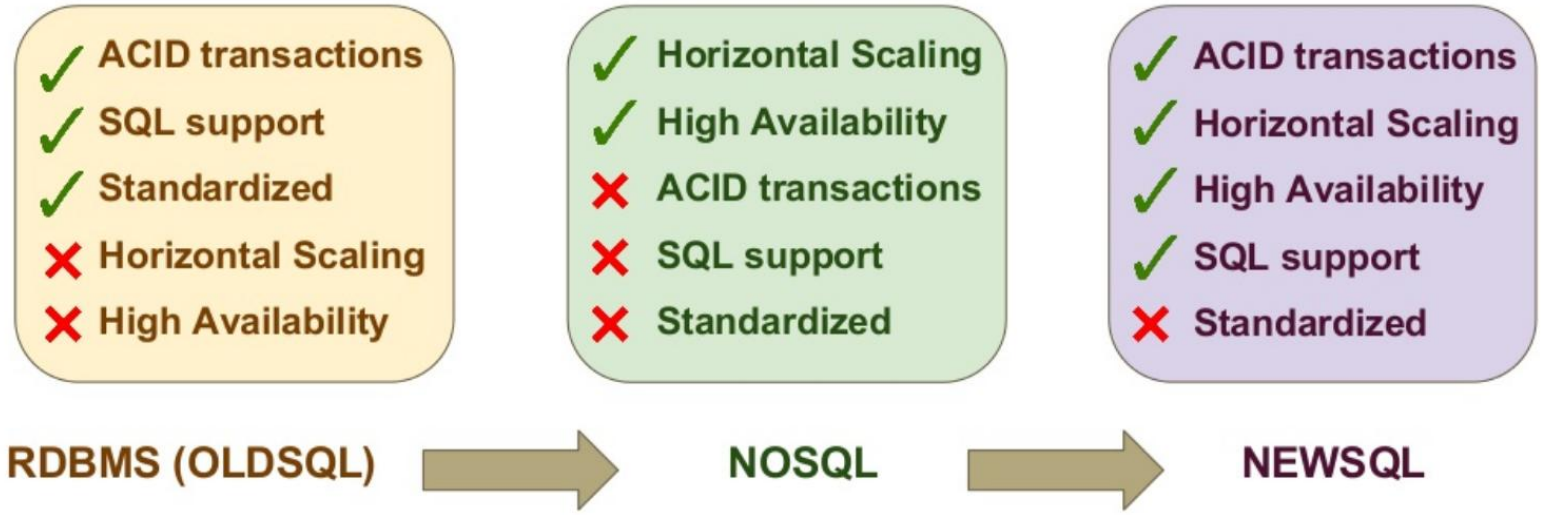
Modeling NoSQL Databases: From Conceptual to Logical Level Design

Shreya Banerjee, Anirban Sarkar
Department of Computer Applications, National Institute of Technology, Durgapur, India
{shreya.banerjee85@gmail.com, sarkar.anirban@gmail.com}

Implementation list is available at https://github.com/tinkerpop/blueprints

2010s – NewSQL

- Provide same performance for OLTP workloads as NoSQL DBMSs without giving up ACID:
  → Relational / SQL
  → Distributed
  → Usually closed-source (e.g. VoltDB, NuoDB)

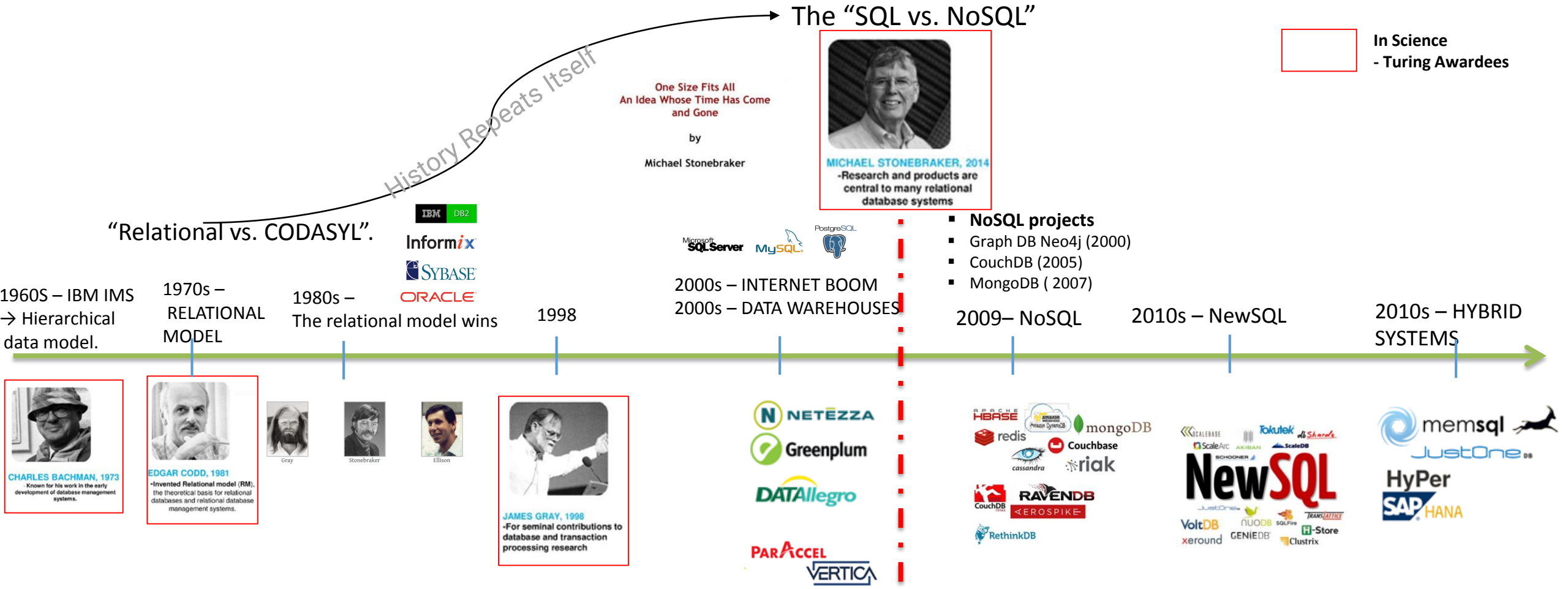


2010s – Systèmes Hybrides

- **H**ybrid **T**ransactional-**A**nalytical **P**rocessing.
  Execute fast OLTP like a NewSQL system while also executing complex OLAP queries like a data warehouse system.

  → Distributed / Shared-Nothing
  → Relational / SQL
  → All closed-source (as of 2016).

Aperçu de l'évolution de système de BD



Questions !!

I Hope I Succeeded to clarify the history of database systems