Variability in Big Data Design

Big Data Management

Knowledge objectives

1. Explain the variability dimension in Big Data
2. Name four different kinds of NOSQL systems
3. Explain the two dimensions to classify NOSQL systems according to how they manage schema
4. Explain the consequences of schema variability
5. Explain the consequences of physical (in)dependence
6. Explain the concept of impedance mismatch

Understanding objectives

1. Decide whether two NOSQL designs have a more or less explicit/fix schema
2. Understand the factors that affect the design of a NOSQL database

Application objectives

1. Given a relatively small UML conceptual schema, translate it into a logical representation of data considering flexible schema representation

Motivation

From SQL to NOSQL

What is Big Data?

VOLUME

Veracity

Velocity

Value

vArIaBiLiTy

Variety

What is Variability?

- Different definitions:
  - Inconsistent meaning depending on context
    - Applies to NLP
  - Inconsistent speed in the data flow
    - Applies to stream processing
  - Inconsistent data
    - Requires quality processes to clean, validate…

RDBMSs help to minimize it

RDBMS: One Size Fits All

- Mainly-write-only Systems (e.g., OLTP)
  - Data storage
    - Normalization
  - Queries
    - Indexes: B+, Hash
    - Joins: BNL, INL, Hash-Join, Merge Join
- Read-only Systems (e.g., DW)
  - Data Storage
    - Denormalized data
  - Queries
    - Indexes: Bitmaps
    - Joins: Star-join
    - Materialized Views

BUT, WHAT IF I NEED TO DEAL WITH MASSIVE READS AND WRITES AT THE SAME TIME?

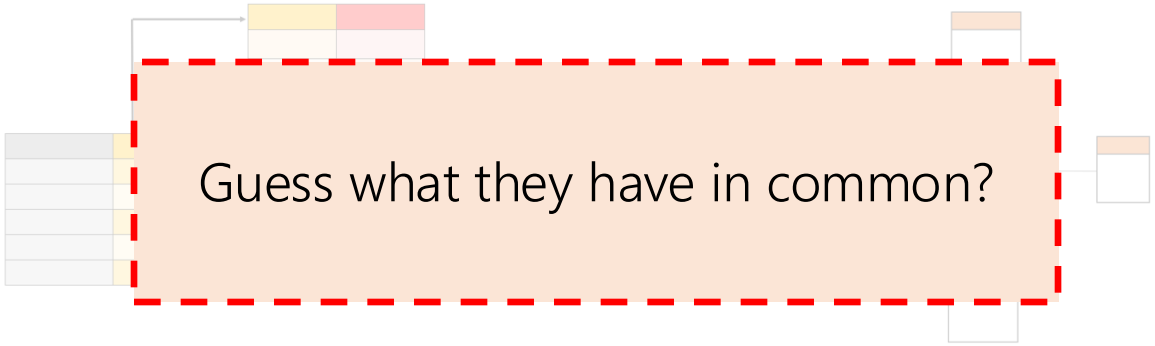
AND DYNAMICALLY INGEST NEW DATA FROM DIFFERENT SOURCES?

FLEXIBILITY

Different data models

Relational (OLTP)

Multidimensional (OLAP)



Guess what they have in common?

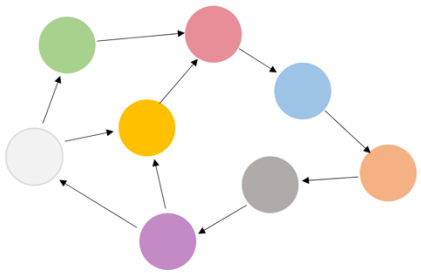
Key-Value

KEY	VALUE
KEY	VALUE
KEY	VALUE
KEY	VALUE
KEY	VALUE

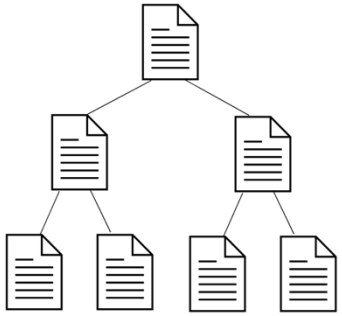
Wide-Column

	Family1	Family2	Family3	Family4
Key				
Key				
Key				
Key				

Graph



Document



By Aina Montalban, inspired by Daniel G. McCreary and Ann M. Kelly

Different kinds of schemas

- In a Relational Database:

```
CREATE TABLE Students(id int,name varchar(50),surname varchar(50),enrolment date);
INSERT INTO Students (1,'Sergi','Nadal','01/01/2012',true,'Igualada'); WRONG
INSERT INTO Students (1,'Sergi','Nadal',NULL); OK
INSERT INTO Students (1,'Sergi','Nadal','01/01/2012'); OK
```

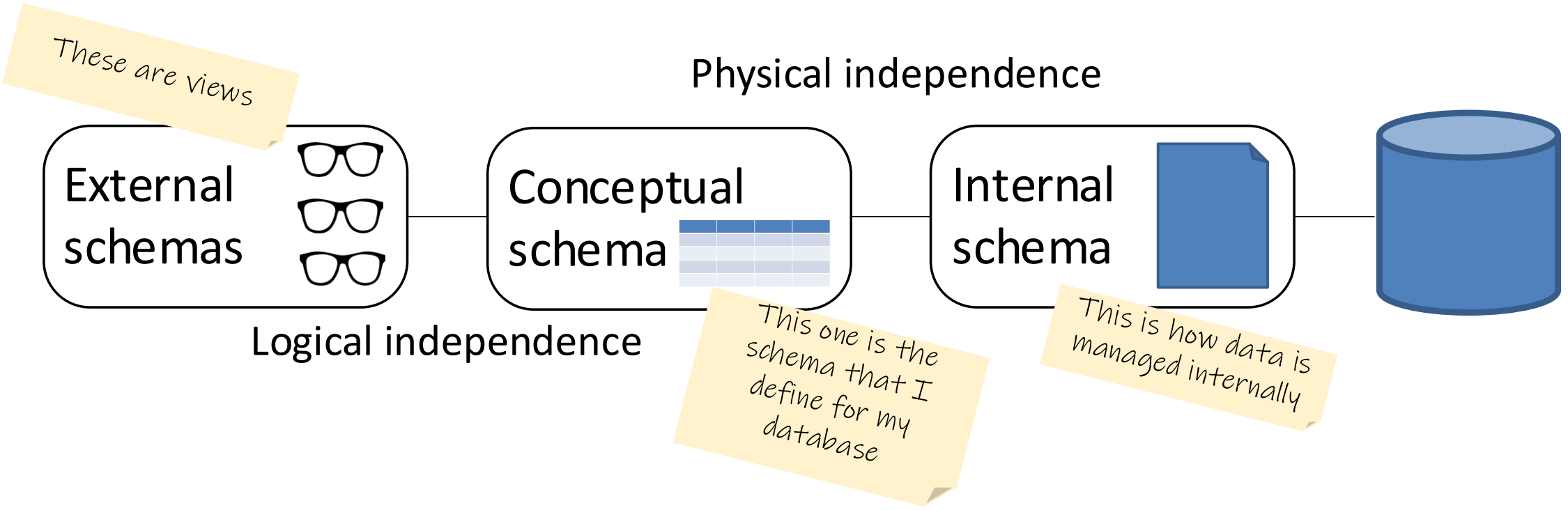
- In a NOSQL database:

```
CREATE "TABLE" Students
INSERT INTO Students (1, {name:'Sergi', surname:'Nadal', enrolment:'01/01/2012'});
INSERT INTO Students (1, {'Sergi', 'Nadal', '01/01/2012'});
```

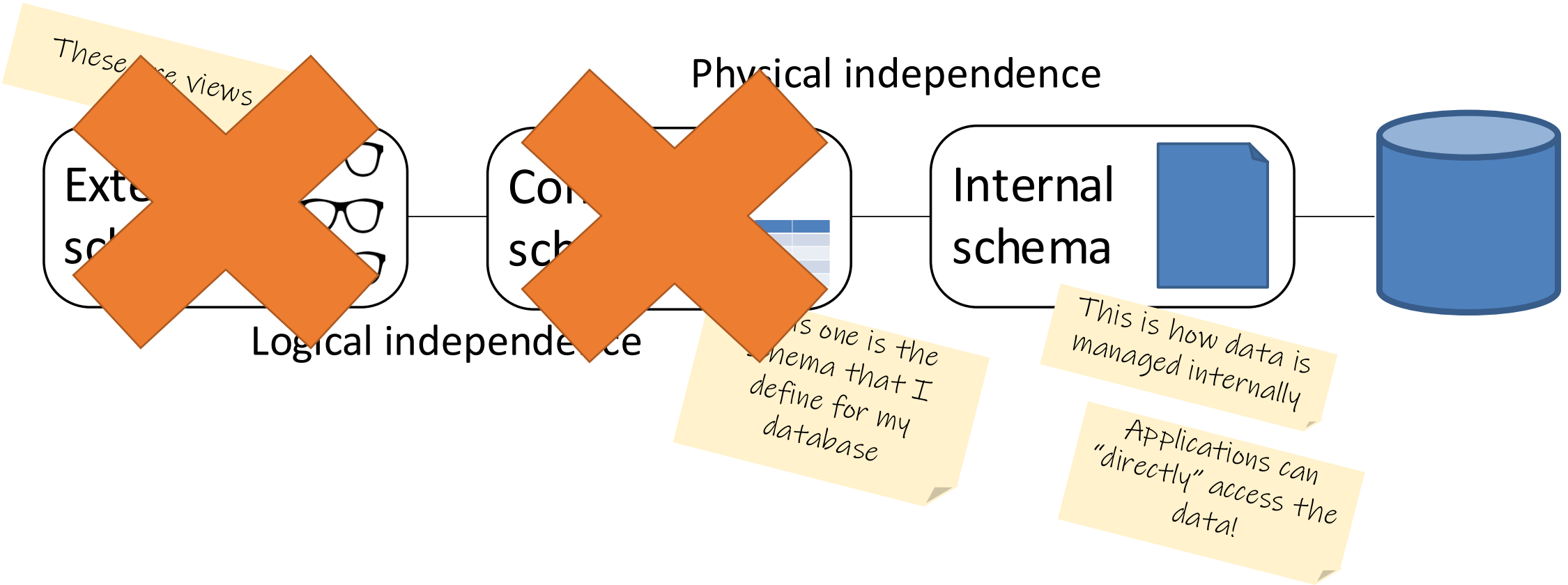
- Consequences
  - Gain flexibility
  - Gain performance
  - But…
    - Lose semantics (and also consistency!)
    - The data independence principle is lost (!)
      - The ANSI / SPARC architecture is not followed

Database schema definition

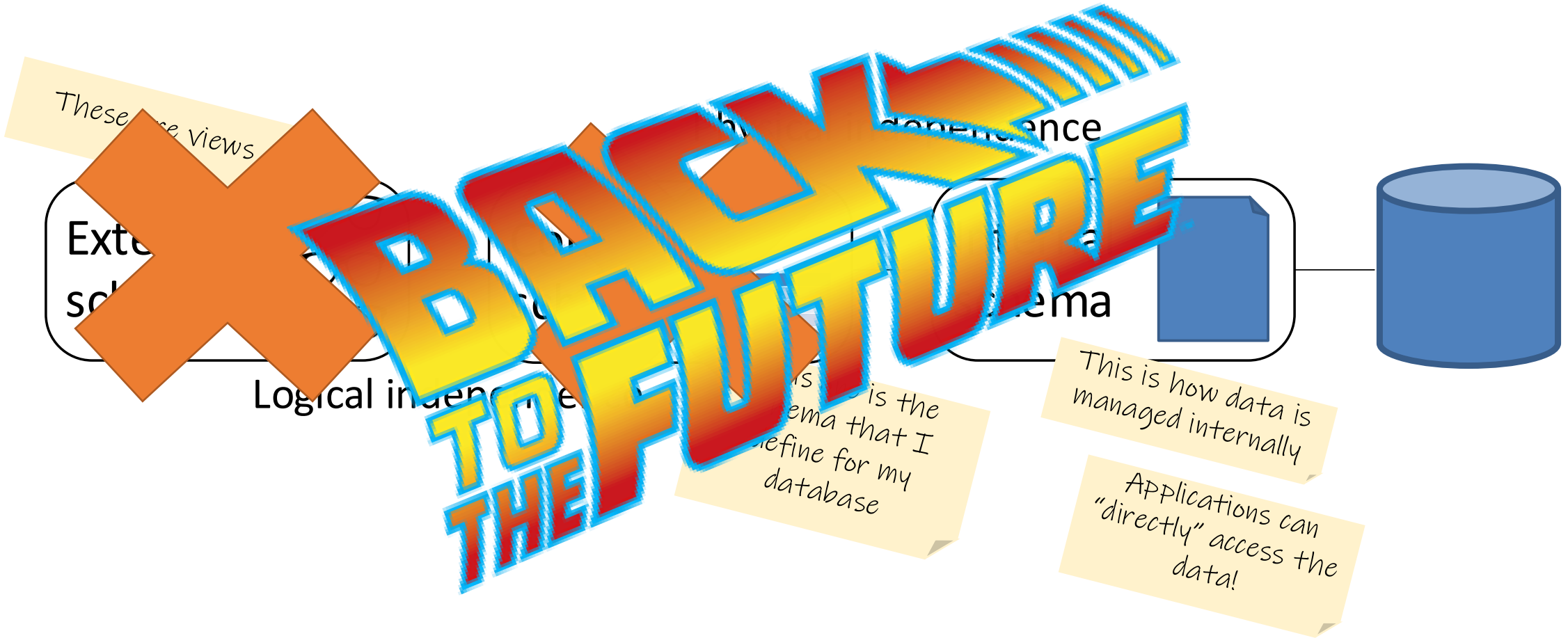
ANSI/SPARC


These are views
Physical independence
External schemas
Conceptual schema
Internal schema
This one is the schema that I define for my database
This is how data is managed internally
Logical independence

ANSI/SPARC


These are views
Physical independence
Internal schema
Logical independence
This is how data is managed internally
Applications can "directly" access the data!

ANSI/SPARC


These are views
Logical independence
This is how data is managed internally
Applications can "directly" access the data!

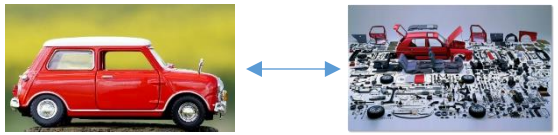
Database models

RELATIONAL

- Based on the relational model
  - Tables, rows and columns
    - Sets, instances and attributes
  - Constraints are allowed
    - PK, FK, Check, …

When creating the tables you MUST specify their schema (i.e., columns and constraints)

Data is restructured when brought into memory (impedance mismatch)



NOSQL

- No single reference model
  - Graph
  - Document-oriented
  - Key-value
  - Wide-column
  - Vector…

Schema (if any) is specified at insertion, not at definition (schemaless databases)

The closer the data model looks to the way data is stored internally, the better (read/write through)

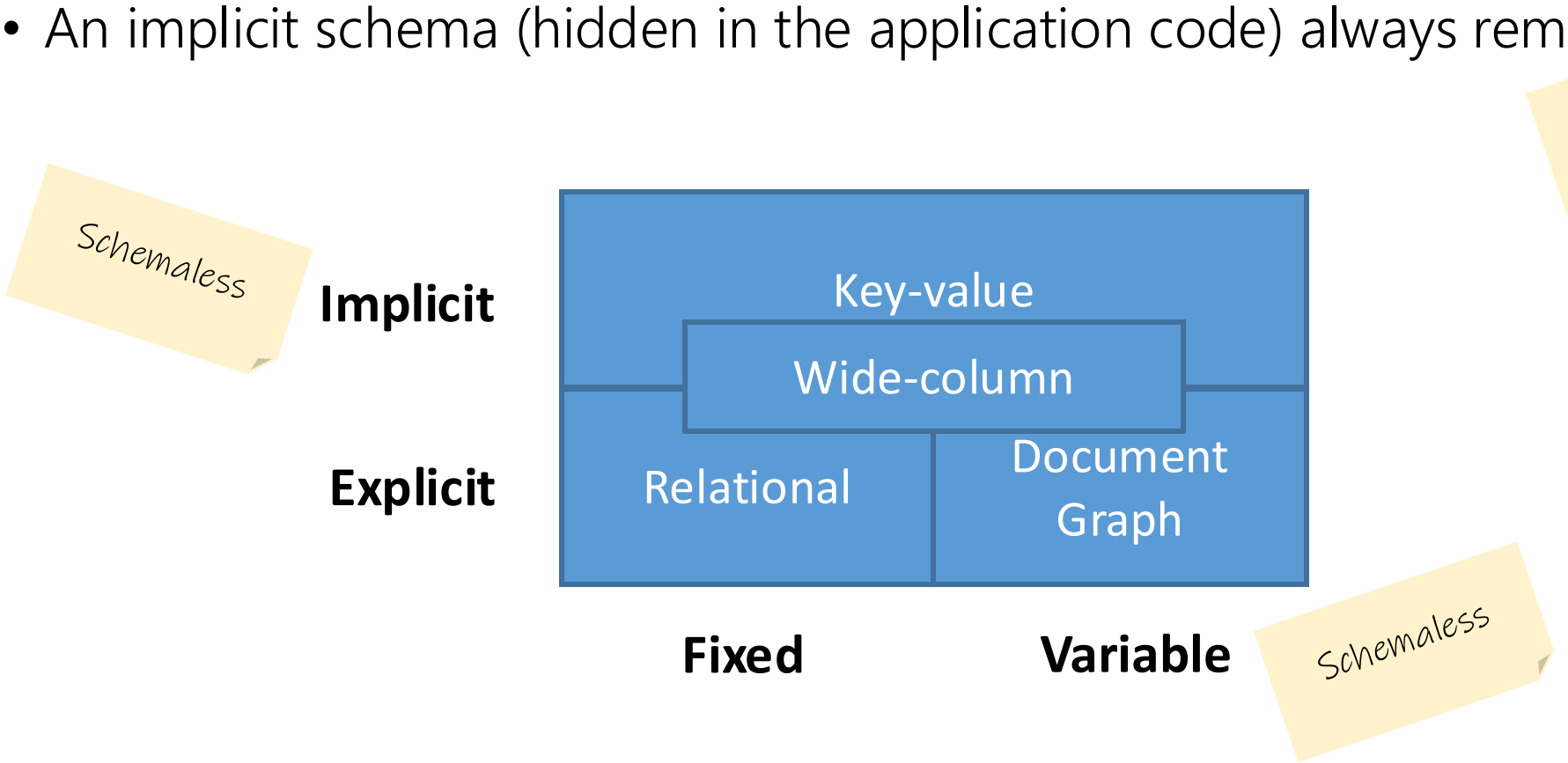
Database models

- Relational
  city(name, population, region) VALUES (*'BCN', '2,000,000', 'CAT'*)
- Key-Value
  [*'BCN', '2,000,000;CAT'*]
- Document
  {**id:***'BCN'*, **population:***'2,000,000'*, **region:***'CAT'*}
- Wide-Column
  [*'BCN'*, **population:**{*value:'2,000,000'*}, **region:**{*value:'CAT'*}]
  [*'BCN'*, **all:**{*value:'2,000,000;CAT'*}]
  [*'BCN'*, **all:**{*population:'2,000,000';region:'CAT'*}]
- Graph
  (**city:***'BCN'*, **population:***'2,000,000'*) – **'is_part_of'** -> (**region:***'CAT'*)

Relevant schema dimensions

Some *new* models lack of an explicit schema (declared by the user)

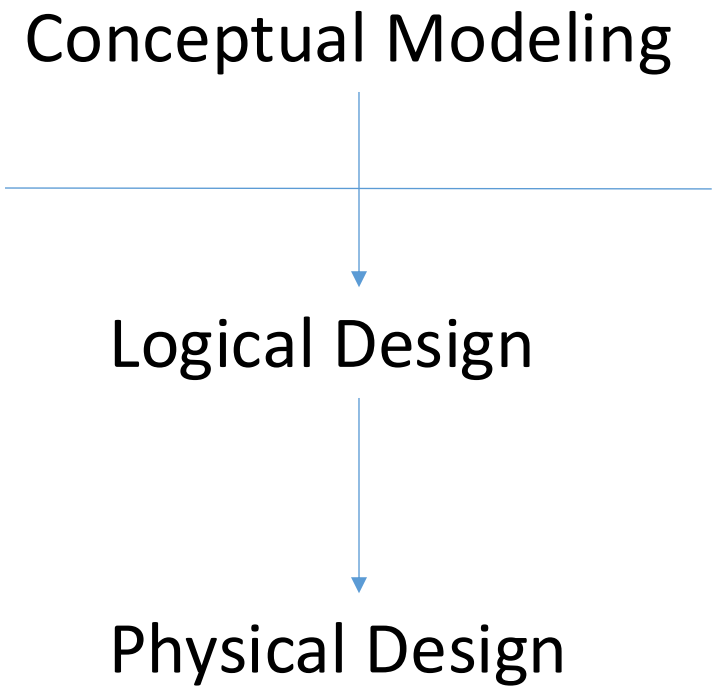
- An implicit schema (hidden in the application code) always remains



Database design in NOSQL

From Conceptual Schema to Logical Design

Phases of database design



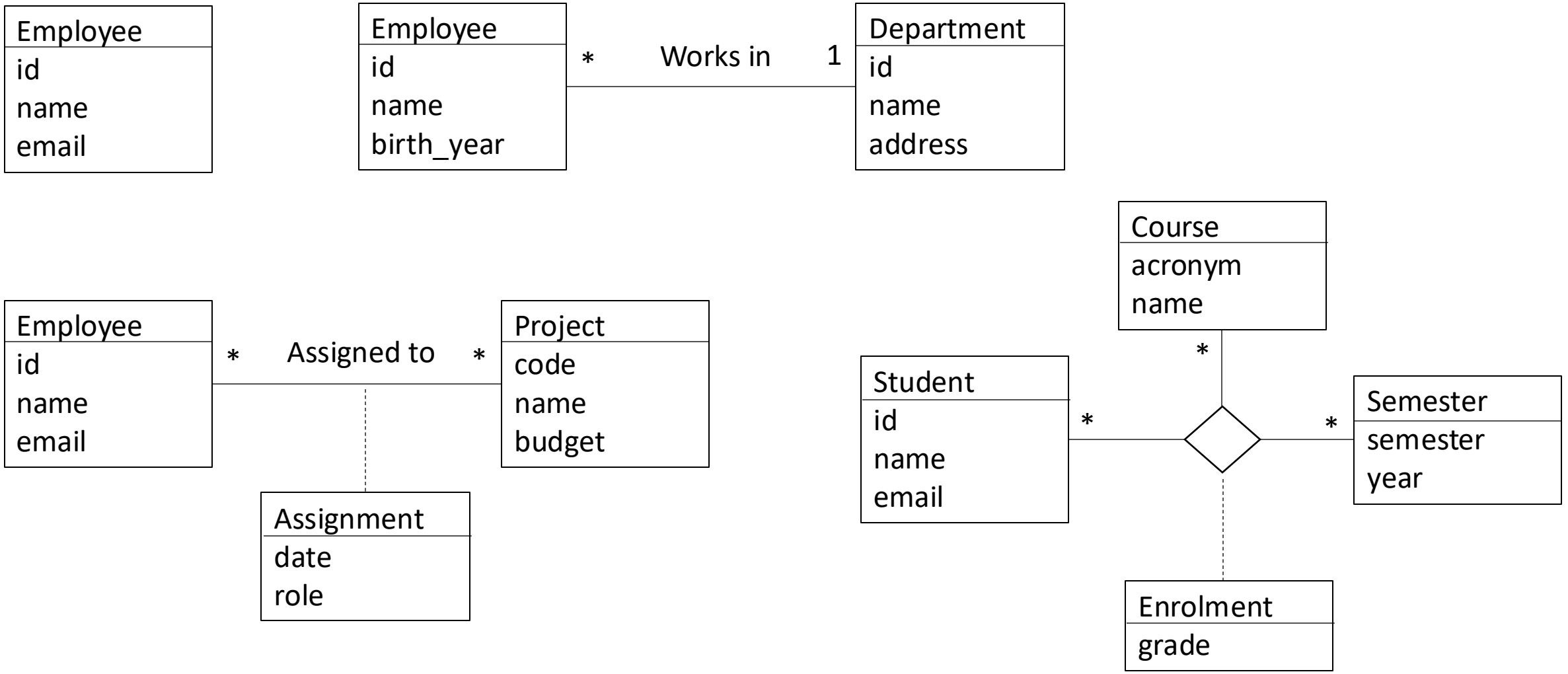
Technology independent

WHAT?

Depend on technology

HOW?

Conceptual Schema (UML)



From Conceptual Schema to Logical Design (RDBMS)

- The **Conceptual Schema** is expressed in a technology-independent language, such as UML.
- The **Logical Design** consists in defining tables, their attributes, PKs and FKs

- Class → **Table**
  - *Ex: Employee*
- 1:N Association → **Foreign Key**
  - *Ex: Employee – Department*
- M:N Association → **Table**
  - *Ex: Employee – Project*
- N-ary association → **Table**
  - *Ex: Student – Course – Semester*
- Association class → **Table**
  - *Ex: LineItem (Order – Item)*

Possible thanks to Normalization!
- Reduces redundancy
- Improves consistency → reduces variability

Activity: From Conceptual Schema to Logical Design (NOSQL)

- *Objective: Lean how to design a schemaless database*
- *Tasks:*
  1. *(15') By pairs, given the following conceptual schema, how many alternative designs for a (generic) schemaless database can you think of?*
     - *Evaluate the pros and cons from the point of view of reads, writes/updates, space, consistency, …*
     - *Which is the best design?*
  2. *(15') Discussion*

Data Storage

RELATIONAL

Generic architecture that can fairly solve many problems by means of:

- Schema
- Normalization
- Indexes
- Joins
- …

The database can be designed following a set of **rules**.

NOSQL

Specific architectures/techniques to optimize particular needs by means of:

- Indexes
- Sequential reads
- Fragmentation
- Replication
- In-memory processing
- …

Workload + architecture must be taken into account when designing the database.

Closing

Summary

- Variability
- NOSQL systems
- Schemaless databases
- Physical (in)dependence
- Database modeling in NOSQL

References

- W. J. Brown et al. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. Wiley, 1998
- M. Stonebraker et al. *The End of an Architectural Era (It's Time for a Complete Rewrite)*. VLDB, 2007
- L. Liu, M.T. Özsu (Eds.). *Encyclopedia of Database Systems*. Springer, 2009
- R. Cattell. *Scalable SQL and NoSQL Data Stores*. SIGMOD Record 39(4), 2010
- M. Stonebraker. *SQL Databases vs. NoSQL Databases*. Communications of the ACM, 53(4), 2010
- E. Meijer and G. Bierman. *A Co-Relational model of data for large shared data banks*. Communications of the ACM 54(4), 2011
- P. Sadagale and M. Fowler. *NoSQL distilled*. Addison-Wesley, 2013