
1
DATA MODELS

# Data Models

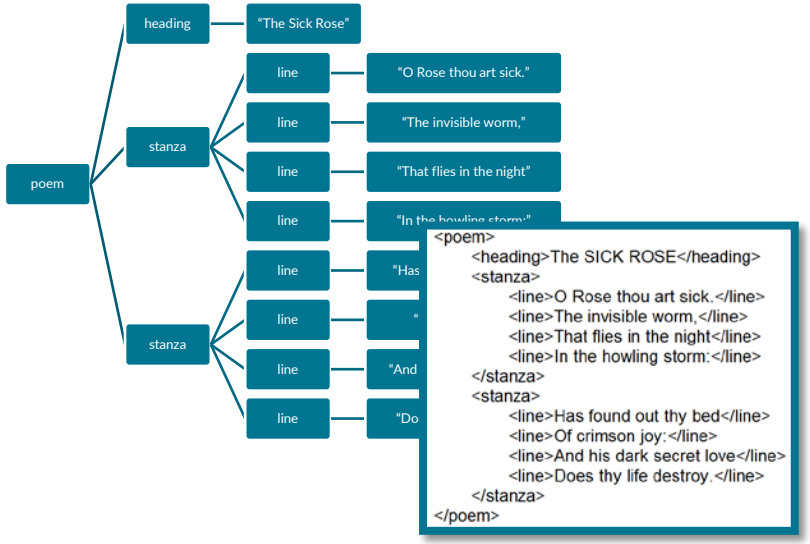
- What is a data model?
- Some examples of data models
- Why data models are important to data curation
- Towards an integrated picture of data model relationships

# Some data models you know and love

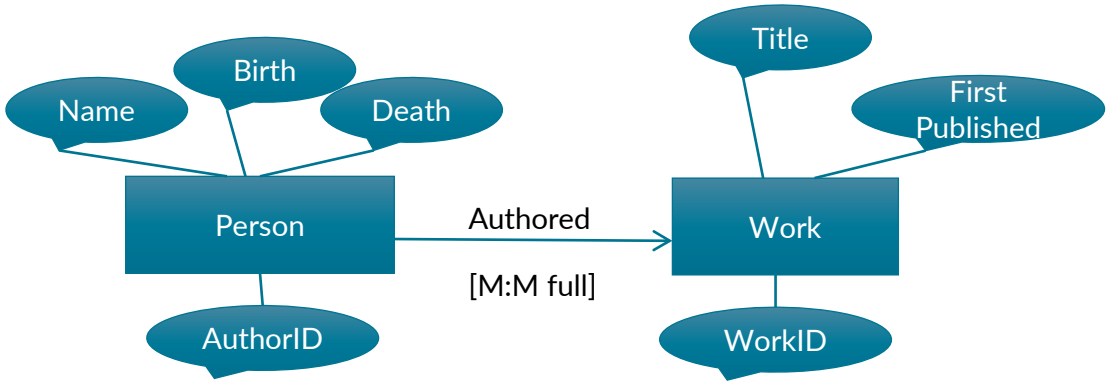
## Relations

| Work | Author | Title | Date |
|---|---|---|---|
| W58425 | P42425 | Moby Dick | 1851 |
| W85246 | P24246 | The Scarlett Letter | 1860 |
| W55427 | P24246 | Fanshawe | 1828 |

## Trees




## Entity/Relationship (ontologies)



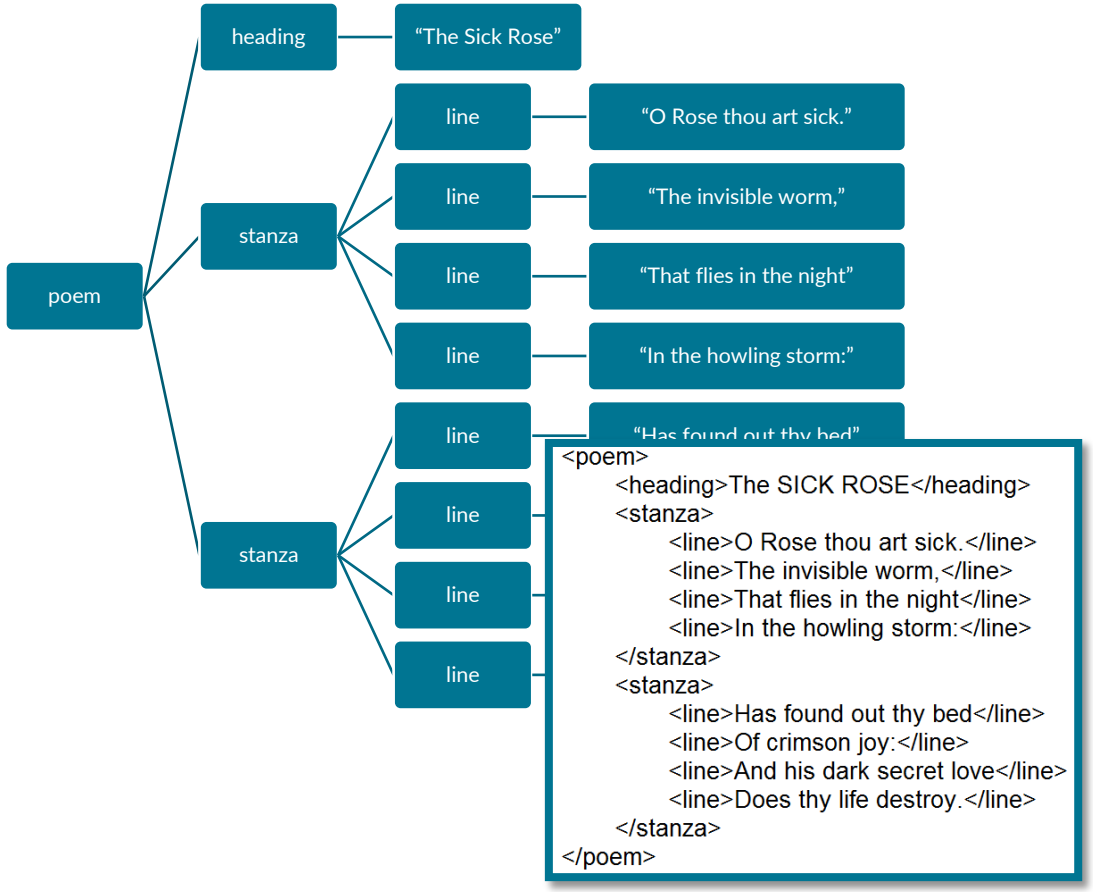
# The relational model

| Work | Author | Title | Date |
|---|---|---|---|
| W58425 | P42425 | Moby Dick | 1851 |
| W85246 | P24246 | The Scarlett Letter | 1860 |
| W55427 | P24246 | Fanshawe | 1828 |

Here a **relational** model is being used:

- Relations (tables) are well-suited for data that conceptualized as attribute/value pairs.
- This particular relational model includes the *attributes* **Title** and **Date**.
- It is modeling a state of affairs where a novel, *Moby Dick*, was published in 1851.

# The tree model

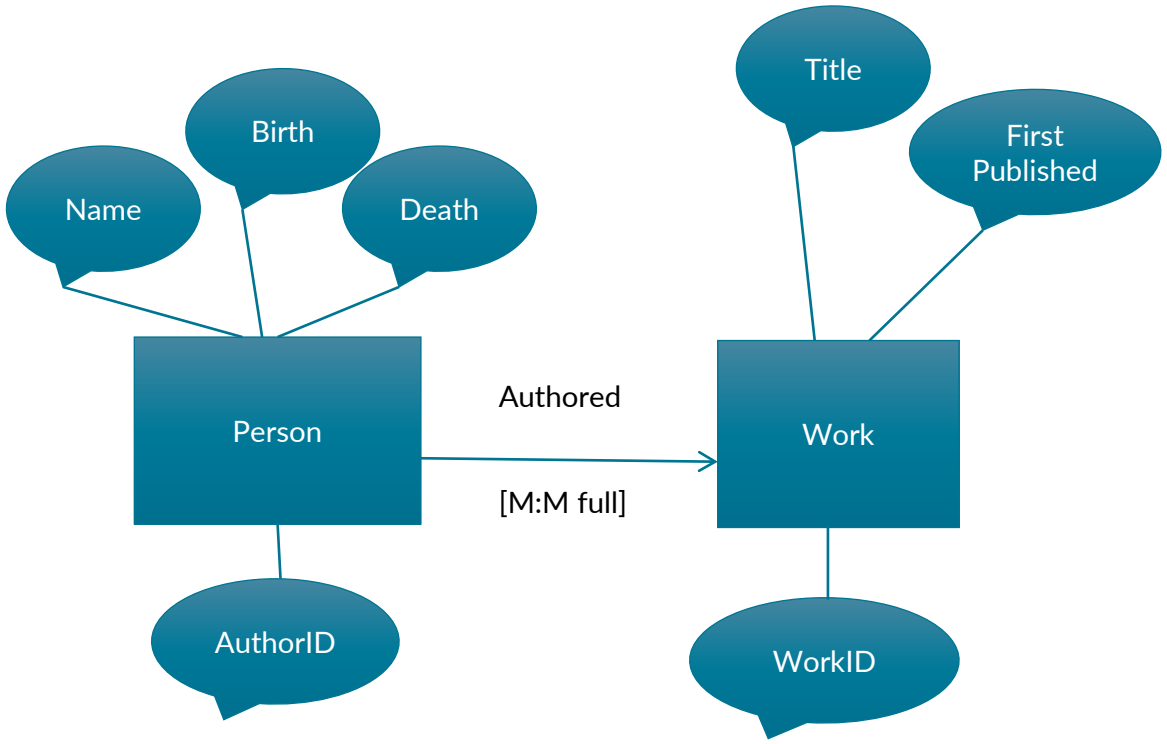



Here a **tree** model is being used:

This particular tree is serialized in XML.

- tree models are well-suited for data that has a tree-like or hierarchical structure, such as documents. But they can also be used to serialize relations and other model instances.
- In this tree the *nodes* have labels such as **stanza** and **line**.
- It is modeling a poem that has two stanzas each with four metrical lines.

# The entity/relationship model




Here is an ER schema,

- ER models operate at a high of abstraction, representing the things and relationships of a domain.
- In this ER schema there are two *entity* classes: **Person** and **Work; e**ach entity class has several *attributes*, and there is a *relationship* (**Authored**) that obtains between entities in those classes.
- Both relation and tree data models and be used to implement ER models.
  - [For our purposes ER diagrams, UML class diagrams, other conceptual modeling approaches, RDFS and OWL, and other ontology languages are all fundamentally similar and may be all be considered ways of specifying an ontology]

# What, exactly, is a data model?

The phrase “data model” has three common senses:

1. A *type* of framework for representing information

2. A *particular* framework for representing information (typically specified by a *schema*)

3. The *application* of a particular framework to represent information

# Sense 1: A *type* of framework

1) "The relational model, with attributes, tuples, and values, is a good one for organizing course registration information."

2) "The tree model, with nodes, labels, and edges, excels at organizing natural language text."

3) "The entity relationship model, with entities and relationships, identifies the things and relationships in a domain of interest."

# Sense 2: A particular framework (schema)

1) "The registrar's **relational model** includes these attributes: *course*, *prerequisites*, *credits*, *department* . . . and assigns *credits* the datatype *integer* . . ."

2) "The journal uses an **XML tree model**. It includes the nodes *article*, *title*, *author*, *affiliation* … It requires that *title* node must (and may only) appear as the first child of an *article* node . . ."

3) "The **ER model** for registration includes the entities *person*, *course*, and *department*, and the relationships *enrolled in*, *sponsored by*, and *teaches*. It allows *persons* to teach multiple *courses* but requires that a *course* be sponsored by just one *department* . . ."

# Sense 3: The *application* of a particular framework

1) "The registrar's [relational] **model** has the value "IS501" for *course* in the only tuple that has "Smith" for *instructor*.

2) "In the [XML tree] **model** for this article the node labelled *author* has the content "Alonzo Church", and the following sibling node, *affiliation*, has the content "Princeton University".

3) "The RDF instance of that [ontology] **model** shows that Anton Marty is enrolled in Dr. Brentano's course"

# What is a Data Model? (Elaborated)

Data models typically have three sorts of components:

1. Structure: sets and tuples, nodes and arcs, …
2. Things: values, labels, entities, relationships …
3. Constraints: datatypes, grammars, cardinality …

Often the specification of *operations* is considered essential:

> "A data model is a mathematical formalism with two parts:
>
> 1. A notation for describing data
> 2. A set of operations used to manipulate that data" – Ullman, 1988

# Why are we talking about data models?

Because critical activities in data curation include

**Select** data model types

**Select** data model schemas

**Develop** data model schemas

**Revise** data model schemas

**Document** data model schemas

**Validate** dataset instances with schemas

**Transform** data in one model (type) to another data model (type)

**Transform** data in one model (schema) to another data model (schema)

**Transform** data from one representation (e.g. serialization) to another (with same schema)

**Integrating** data from two different data models (schema or type)

*and more*



# Looking ahead: data model relationships

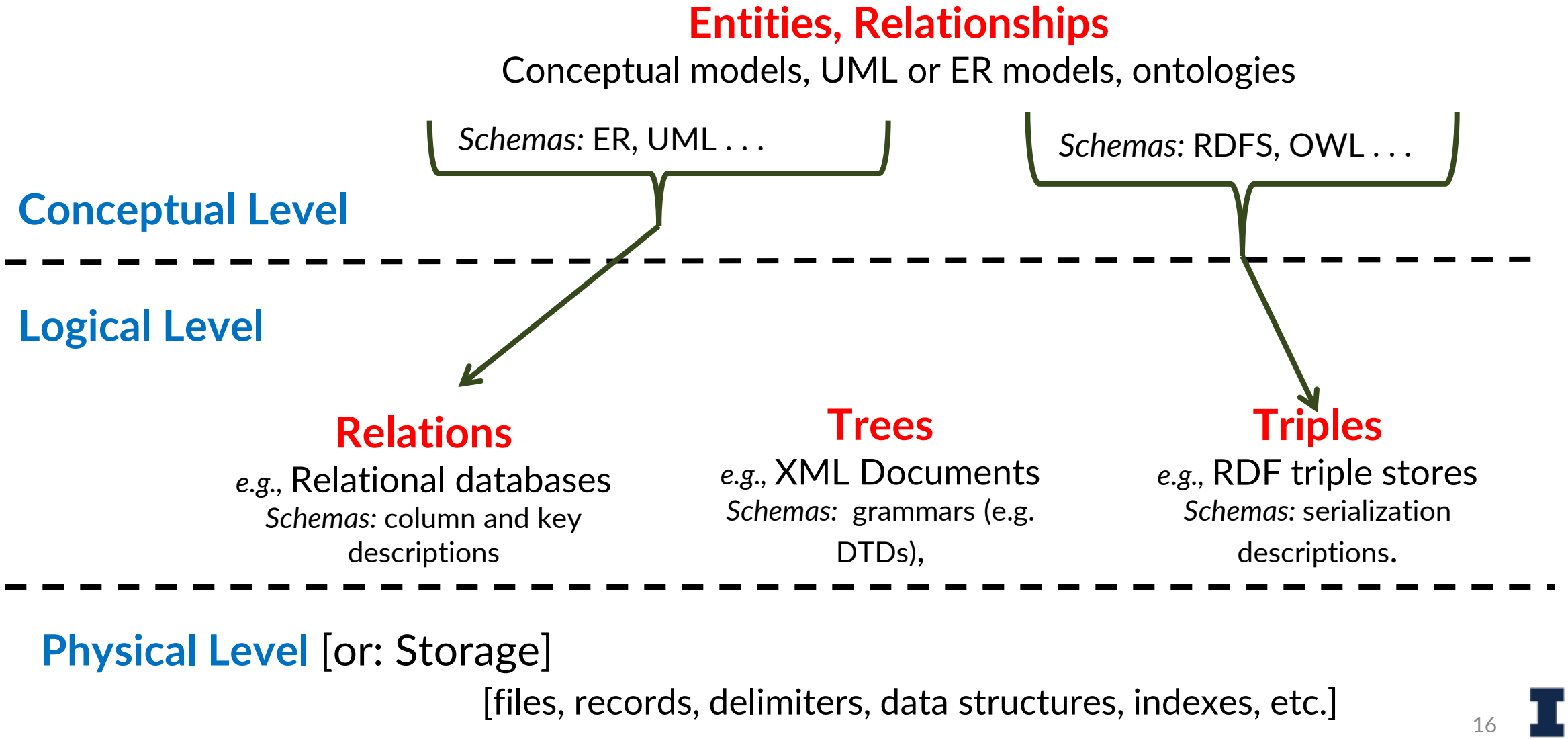
A critical issue in data curation has to do with:

How different types of data models, and different data models of the same type, are related to one another

This is a question we will take up in detail later in the course

*But here's a partial diagram of the territory ahead . . .*

# Data model relationships




2
THE PROBLEM

# What's The Problem?

- The situation (circa 1960)
  - Data is stored in radically different ways
  - Interaction with data is immediately and directly via storage methods
  - Explicit and formal conceptualization of data *as data* is rare
    (and typically only in human memory)
- Why is this a problem?
  - Huge operational inefficiencies
  - Lack of functionality
  - Lack of data independence

# What's the Problem?

Imagine a large company with many divisions

Left to itself each division will probably:

(i) conceptualize their domain in different ways

(ii) develop different methods of representation and storage

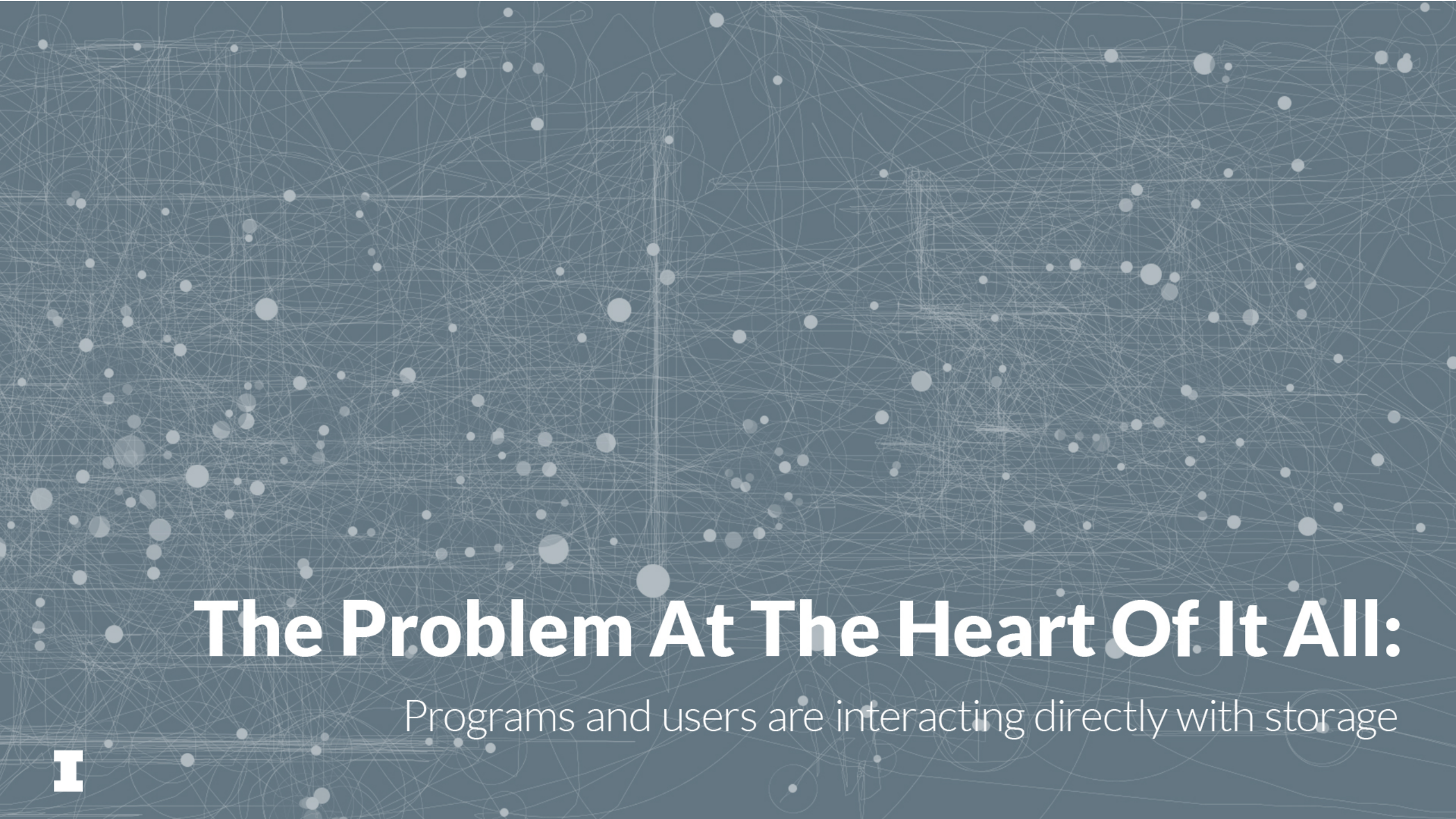
(iii) mix processing instructions with data when convenient

There will typically be

- no clear separation between storage methods and the intrinsic structure of the information being represented
- no general abstract understanding of the nature of the information being represented [other than what is in the memories of staff and programmers]



Flickr | ArnoldReinhold



# The Problem At The Heart Of It All:

Programs and users are interacting directly with storage

# Storage representations

Variable-length fields, delimited

```
W54825,Moby Dick,1851
W85246,The Scarlett Letter,1860
W55427,Fanshawe,1828
```

```
W54825    Moby Dick    1851
W85246    The Scarlett Letter        1860
W55427    Fanshawe     1828
```

Indexed byte offsets

| | |
|---|---|
| 00000000 | (WorkID) |
| 00000010 | (Title) |
| 00000020 | (Year) |

Fixed-length fields

…counting from the left: 0, 6, 25

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | 5 | 8 | 4 | 2 | 5 | M | o | b | y | | D | i | c | k | | | | | | | | | | | 1 | 8 | 5 | 1 |
| W | 8 | 5 | 2 | 4 | 6 | T | h | e | | S | c | a | r | l | e | t | t | | L | e | t | t | e | r | 1 | 8 | 6 | 0 |
| W | 5 | 5 | 4 | 2 | 7 | F | a | n | s | h | a | w | e | | | | | | | | | | | | 1 | 8 | 2 | 8 |

…or from the right: 28, 22, 3

# The problems this causes (1)

With no formally defined general approach application development is arduous.

- Many different unique access subroutines must be developed, tested, and maintained.
- Tools developed for different divisions are not interoperable.
- There cannot be a sustainable 3rd party industry of common applications and tools.
- Specialized applications (searching, analysis, etc.) must be custom developed, cannot reference high level constructs
- All tools must be be modified often as storage formats change.

# The problems this causes (2)

There are more problems caused by a lack of a general conception of data:

- Workflow and transformation cannot be usefully tracked, audited, or logged
- Documentation cannot exploit a conceptual understanding of the data
- Documentation must be updated frequently for changes in storage format or processing changes.
- Data validation and quality assurance is difficult as standard tools for syntax checking, typing, constraint management, etc., cannot be used.
- Schemas, if they exist, focus only on storage and do not help us with general data management.

# As a result. . .

This result is that systems and practices are:

- Inefficient
- Error-prone
- Untrustworthy
- Difficult to document
- Difficult to repurpose and reuse
- Difficult to preserve for future use
- Dependent on memory and workplace practices
- Dependent on custom tools and applications

# Data Independence

One significant consequence of this chaos is a failure of data independence.

This failure comes in two varieties:

**Type 1: If the storage method changes**, then the end user programs accessing the data will fail to perform as expected.

**Type 2: If new kinds of data need to be represented,** then again end user programs may fail or give the wrong result.



# First, keep these variations in mind

Variable-length fields, delimited

```
W54825,Moby Dick,1851
W85246,The Scarlett Letter,1860
W55427,Fanshawe,1828
```

```
W54825      Moby Dick     1851
W85246      The Scarlett Letter          1860
W55427      Fanshawe      1828
```

Indexed byte offsets

| | |
|---|---|
| 00000000 | (WorkID) |
| 00000010 | (Title) |
| 00000020 | (Year) |

Fixed-length fields

…counting from the left: 0, 6, 25

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | 5 | 8 | 4 | 2 | 5 | M | o | b | y | | D | i | c | k | | | | | | | | | | 1 | 8 | 5 | 1 |
| W | 8 | 5 | 2 | 4 | 6 | T | h | e | | S | c | a | r | l | e | t | t | | L | e | t | t | e | r | 1 | 8 | 6 | 0 |
| W | 5 | 5 | 4 | 2 | 7 | F | a | n | s | h | a | w | e | | | | | | | | | | | | 1 | 8 | 2 | 8 |

…or from the right: 28, 22, 3

# Lack of data independence (type 1)

If the physical storage method changes, then the end user programs accessing the data will fail to perform as expected.

For instance, if the storage method switches from a fixed field approach to a delimited field approach then the access programs (and other tools) will return the wrong results.

# Lack of data independence (type 2)

If new kinds of data need to be represented then, again, end user programs may fail or give the wrong result.

For instance, if a new attribute is accommodated by adding a delimited field to the right side of a record, then any program or other tool that has been identifying fields by counting delimiters right to left will probably return the wrong result.



# 3

# THE RELATIONAL MODEL

# The Relational Model

- The problem, the solution
- Relations
    - The relational model
    - How the relational model addresses the problem
    - Relations from a mathematical POV
- The two fundamental principles of data organization
    - Abstraction
    - Indirection

# The problem, again

We have just described two fundamental problems facing data management:

**Programs and users interact with data directly via its storage structure**
(And those storage structures vary wildly).

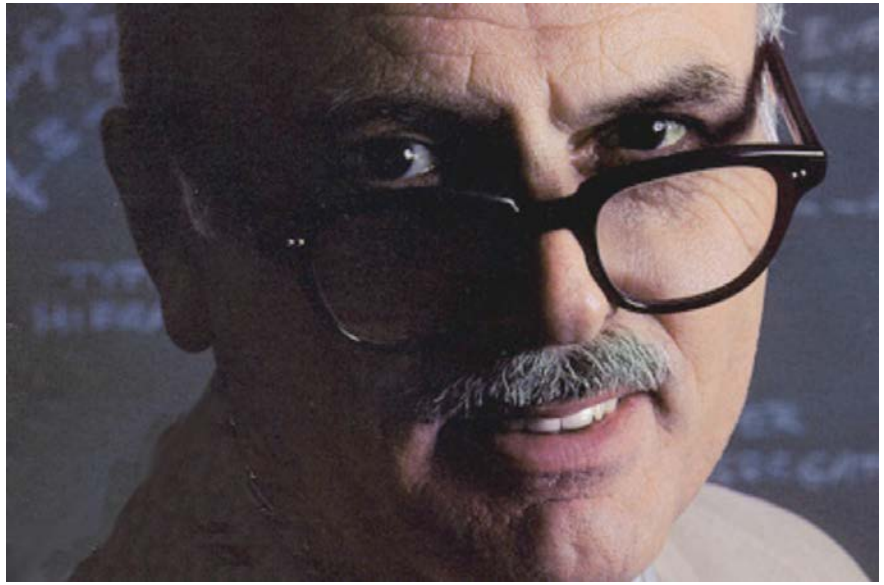
**The intrinsic nature of the information is not reflected in the management systems.**
These systems to not explicitly reflect the attributes, relationships, etc. that are the genuine components of the information being stored and managed.

# The Solution

In 1970 E. F. Codd proposed a simple solution.

*Conceptualize data as relations (tables)*
*and then map those relations*
*to whatever storage methods are being used*

It changed the world.



*E. F Codd*

*EF Codd in* "A Relational Model of Data for Large Shared Data Banks" (1970).
*Perhaps the most cited paper in computer science.*

# Relations (aka Tables)

| Works | Work | Author | Title | Date | Last_Name | First_Name |
|---|---|---|---|---|---|---|
| | W58425 | P42425 | Moby Dick | 1851 | Melville | Herman |
| | W85246 | P24246 | The Scarlett Letter | 1860 | Hawthorne | Nathaniel |
| | W55427 | P24246 | Fanshawe | 1828 | Hawthorne | Nathaniel |



# Why it works

The relational model is a simple single high-level abstraction for conceptualizing information;

It is indifferent to the details of physical storage and processing

All interactions with the data are in relational (tabular) terms, such as attributes, values, tuples. Those interactions are translated into instructions expressed in terms of storage data instructions.

Operations on data are based on formally defined, well understood operations from logic and set theory.



# How it works

Many of the problems described earlier are solved or mitigated by this approach

- Programmers and other users need know nothing about storage methods
- Programmers can learn a common language and approach to data management
- Documentation will have a common structure and organization
- Established mathematical methods may be used (such as set theory and logic)
- Data from different sources can be integrated more easily
- Data will be easier to check for validity and quality
- Data independence is supported:
  - storage methods can be changed without impacting programs
  - new data constructs can be added without impacting programs

# Simple and relentless

Conceptualize all information in terms of relations (rows, columns, values)

Whenever you say anything , *say it in relations*
Whenever you do anything, *do it with relations*
Whenever you talk, *talk in relations*
Whenever you think, *think in relations*

Rows, columns, and values. Nothing else. ***Ever!***

*Upload your data in rows and columns, query your data in rows and columns, receive your query results in rows and columns, only buy software that works on rows and columns, eat, drink and sleep rows and columns*

**Pay no attention to how the information is stored— that's not your problem!**

relations, relations, relations, relations

*The end*

# Internal/External

“Whatever you do in the privacy of your own CPU is your business,
but the interface you present in public must be: *relations*.”
[adapted from Michael Sperberg-McQueen]

# Relations, from a mathematical POV

- A relation is a set of n-tuples:

{

| | | | | | | |
|---|---|---|---|---|---|---|
| < | W58425 | P42425 | Moby Dick | 1851 | Melville | Herman > |
| < | W85246 | P24246 | The Scarlet Letter | 1860 | Hawthorne | Nathaniel > |
| < | W55427 | P24246 | Fanshawe | 1828 | Hawthorne | Nathaniel > |

}

# Relations, from a mathematical POV

More formally:

An ***n-ary relation*** on sets $A_1$, $A_2$, …,$A_n$ is any subset ***R*** of $A_1$ x $A_2$ x … x $A_n$.

Where each set $A_1$, $A_2$, …,$A_n$ is the set of possible values for an attribute

And $A_1$ x $A_2$ x … x $A_n$. is the cartesian product of those sets and so is the set of all sets of n-tuples (relations) for those attribute values. And **R** will thus be one of those sets of n-tuples.

The sets $A_i$ are the ***domains*** of the relation, and *n* is the ***degree*** of the relation.

[From Rosen. . .]

# Two key principles: abstraction and indirection

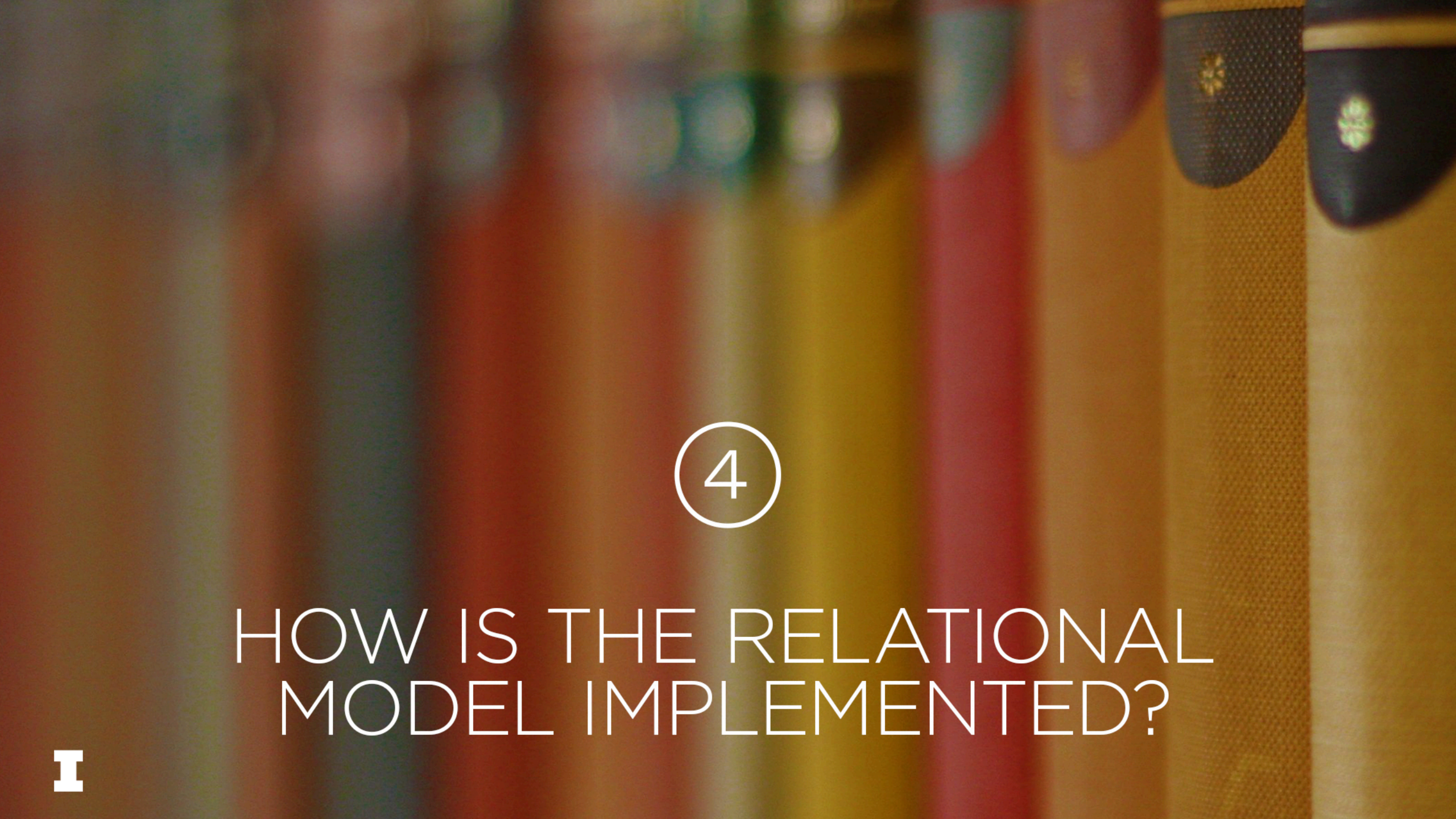
**Abstraction**

Our data model is an *abstraction*; it abstracts away from the transient and varying details of storage and processing and focuses on the essential features of the data itself.

**Indirection**

Relational data management systems do not *directly* interact with the stored data representations, instead they interact *indirectly* with the stored data representations, via the relational representation that is mapped to the actual storage representation.

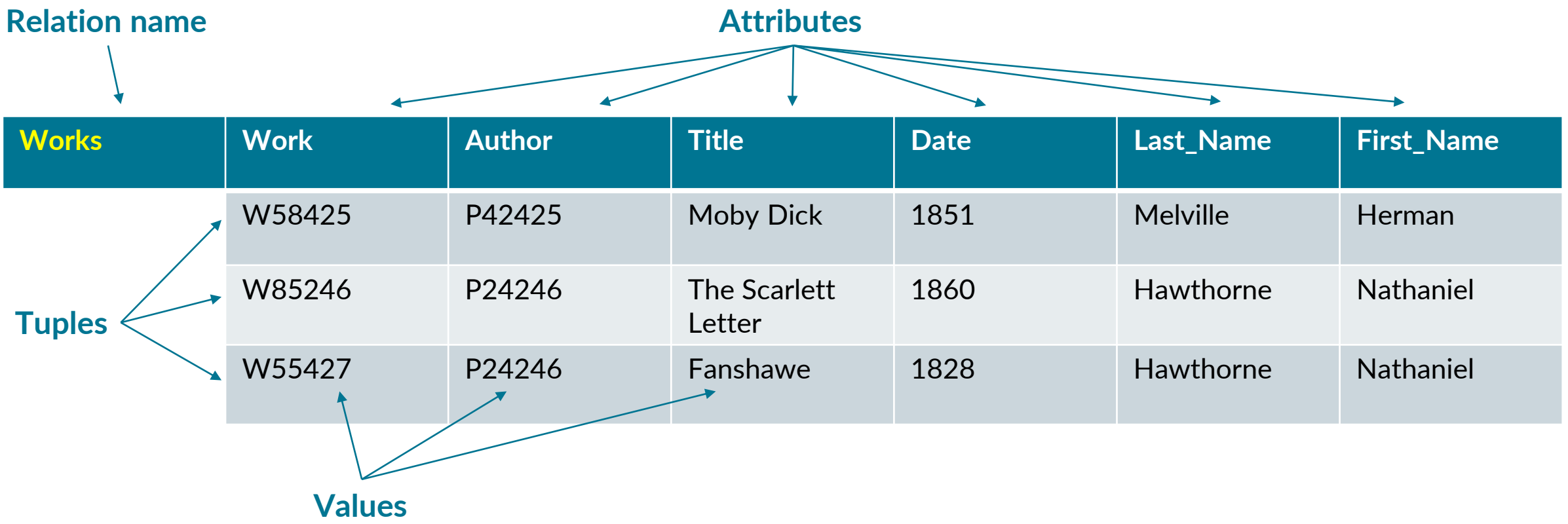



4
HOW IS THE RELATIONAL MODEL IMPLEMENTED?

# The Relational Model In More Detail

- Terminology
- Schemas
- Normalization
- Constraints
- Query languages

# Relation Terminology



# Schemas (generally)

***Schema***: a general term for a specification of how data is or will be organized
they may also specify: vocabulary, syntax, data types, attributes, value ranges, etc.

Schemas are written in a *schema language*.
Most schemas can themselves be processed by software.

Schemas can be used to:

- configure access and retrieval
- map between levels of abstraction
- support validattion
- create structured interfaces for input
- support inferencing and analysis
- support format conversions
- support documentation

# Schemas for relations

A simple table schema (or *relation schema*):

```
AuthorTable (authorID, last, first)
```

A simple relational database schema:

```
{
AuthorTable (authorID, last, first)
WorkTable   (workID, authorID, title, date)
}
```



# Normalization and Functional Dependencies

Functional dependency:

Suppose that whenever two tuples agree on *Author*, they will also agree on *Last_Name*

| Work | Author | Title | Date | Last_Name | First_Name |
|---|---|---|---|---|---|
| W58425 | P42425 | Moby Dick | 1851 | Melville | Herman |
| W85246 | P24246 | The Scarlett Letter | 1860 | Hawthorne | Nathaniel |
| W55427 | P24246 | Fanshawe | 1828 | Hawthorne | Nathaniel |

# Normalization

| Works | ***Work*** | *Author* | Title | Date |
|---|---|---|---|---|
| | W58425 | P42425 | Moby Dick | 1851 |
| | W85246 | P24246 | The Scarlett Letter | 1860 |
| | W55427 | P24246 | Fanshawe | 1828 |

| People | ***Person*** | Last_Name | First_Name |
|---|---|---|---|
| | P42425 | Melville | Herman |
| | P24246 | Hawthorne | Nathaniel |

# Keys

| Works | *Work* | *Author* | Title | Date |
|---|---|---|---|---|
| | W58425 | P42425 | Moby Dick | 1851 |
| | W85246 | P24246 | The Scarlett Letter | 1860 |
| | W55427 | P24246 | Fanshawe | 1828 |

Primary key: Each value for *Work* attribute identifies one work. Each value for *Person* attribute identifies one person.

Foreign key: *Author* references primary key “Person” of “People” table

| People | *Person* | Last_Name | First_Name |
|---|---|---|---|
| | P42425 | Melville | Herman |
| | P24246 | Hawthorne | Nathaniel |

# Normalization and data curation

Understanding functional dependencies is important to data curation because they

- allow the use of normalization to reduce redundancies that cause error and inconsistency, as well as degrade efficiency of updates and validation
- allow a developer, user, or analyst to reason about how data may be manipulated or reorganized

# Constraints and data curation

Constraints such as key constraints, data types, data ranges, referential integrity, etc. are critical to data curation.

- They help to model the real world, real states of affairs, with greater complexity and expressiveness than relational model alone
- They support validation and consistency
- They reflect what may be *assumed* by users, application developers, storage structures, curators… etc.

# Query languages and data curation

The relational model supports the use of well-understood query languages, rather than idiosnycratic language based on unique structures.

This not only supports shared learning, training, documentation, and tools, but ensures that retrieval, views, and calculations have well-defined semantics and will perform as expected.

5

# ABSTRACTION, INDIRECTION AND DATA INDEPENDENCE

# Abstraction, indirection, and data independence

It is worth saying these things one more time

- Abstraction
- Indirection
- Data independence
- A fundamental principle in the data curation creed

# Abstraction and Indirection

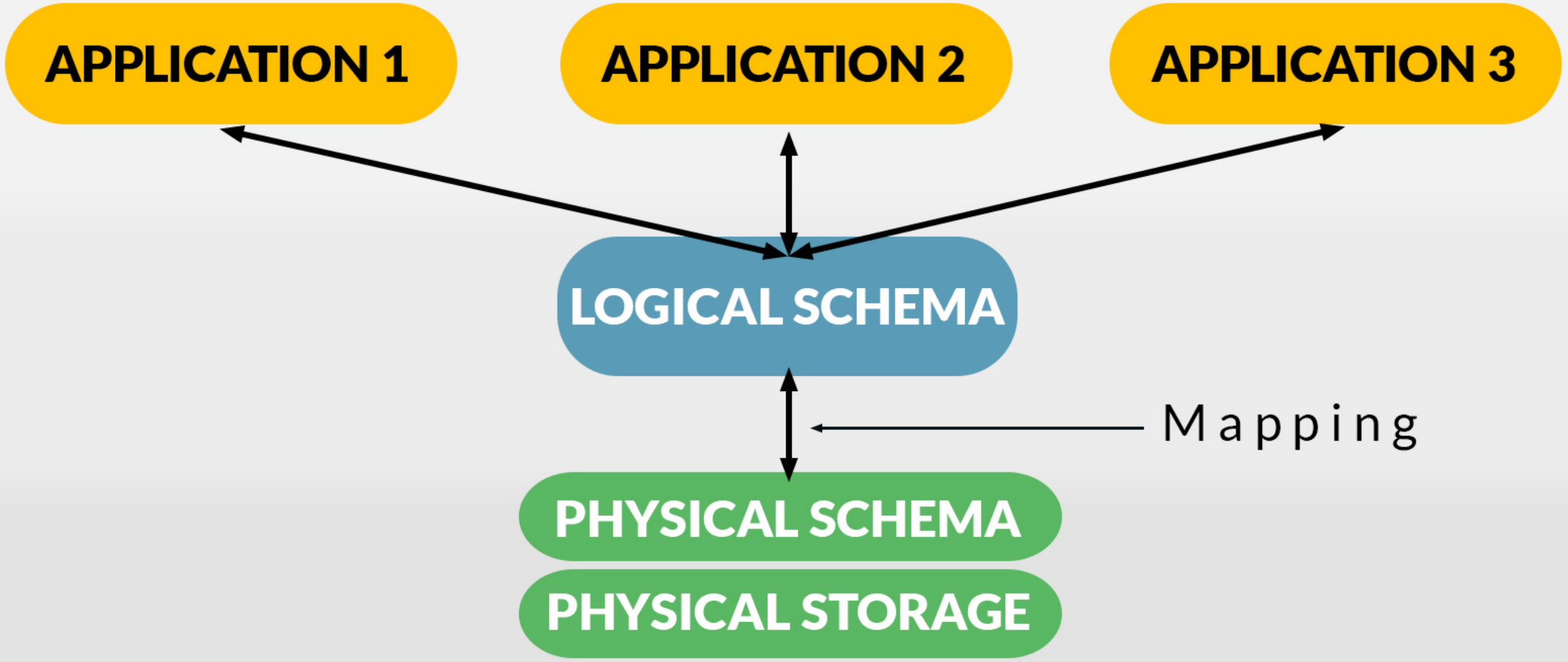
The success of the relational model is based primarily on two related principles:

## **Abstraction** and **Indirection**

The relational representation *abstracts* away from the specific and transient details of storage, presenting only the intrinsic features of the data itself.

Subsquent interaction with the stored data is then *indirect* via a mapping from the relational schemas to the stored data

# Mapping


APPLICATION 1
APPLICATION 2
APPLICATION 3
LOGICAL SCHEMA
Mapping
PHYSICAL SCHEMA
PHYSICAL STORAGE

# Adjusting the mapping manages data independence

**Physical data independence**

The **physical schema** (storage method) can change without affecting interactions with the data.

The problem with direct interactions:
if the storage method switches from a fixed field approach to a delimited field approach then the access programs (and other tools) will return the wrong results.

The solution:

*The physical schema <> logical schema mapping is adjusted to accommodate the changes*



# Adjusting the mapping manages data independence

**Logical data independence**

The **logical schema** can change without affecting interactions with the data.

For instance, if a new attribute is accommodated by adding a delimited field to the right side of a record, then any program or other tool that has been identifying fields by counting delimiters right to left will probably return the wrong result.

The solution:

*Again, the physical schema <> logical schema mapping is adjusted to accommodate the changes*

**...with feeling**

**Abstraction**

implemented with **indirection**

supports **data independence**

(among other things)

*This is a fundamental principle in the data curation creed.*

We will see it again and again in this course.