Anchor Modeling
(with bitemporal data)

Lars Rönnbäck

May 2012


The only constant is change.
Heraclitus
~ 500 B.C.

Change is Accelerating

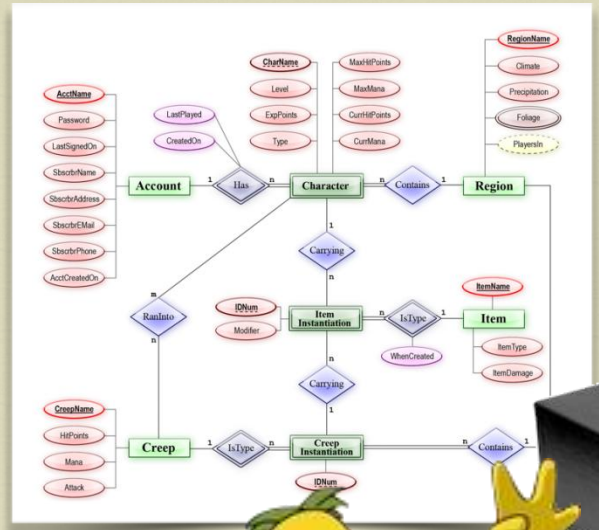
- Big Data
- Moore's Law
- Ephemeralization
- New emerging markets
- Customer "re-preferencing"
- Volatile economic environment
- Fast changing and far reaching regulations

5,000,000,000	The earth forms
500,000,000	Vertebrates
50,000,000	Mammals
5,000,000	Primates
500,000	Humans
50,000	Great migrations
5,000	Civilizations
500	Printing press
50	Television
5	Mobile Internet

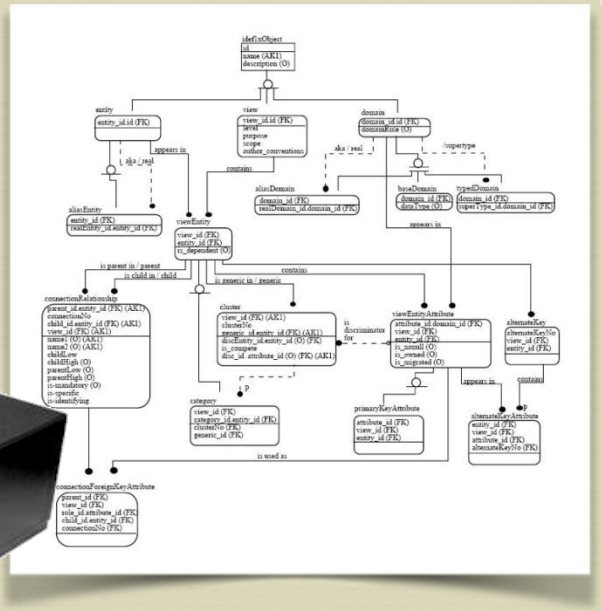
Traditional Setup

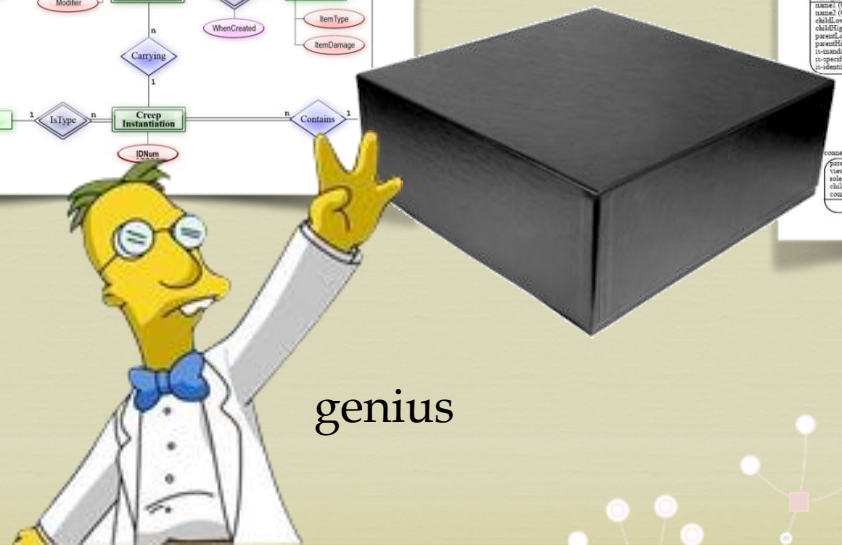
”future proof” information model

resulting static data model



opaque transformation logic



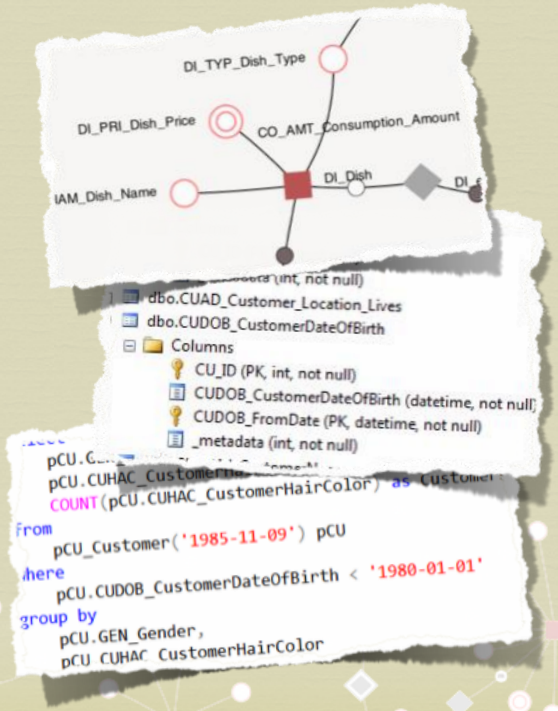


genius


With
Quicker-and-Dirtier
solutions…
redoing from scratch becomes unavoidable!

Why Anchor Modeling?

- **Anchor Modeling** is an agile technique for modeling information under evolution, and the automatic generation of corresponding database implementations, based on:
- Entity Relationship Modeling (1976 – *Chen*)
- Bitemporal Databases (1992 – *Snodgrass*)
- The Sixth Normal Form (2002 – *Date, Darwen, Lorentzos*)
- Immutability, Temporal Independency, and Timeline Annexing (2009 – *Rönnbäck, Regardt, et al.*)



Background

- Anchor Modeling is developed since 2003 in a collaboration between the Swedish industry and academia, the Department of Computer Science at Stockholm University (DSV).

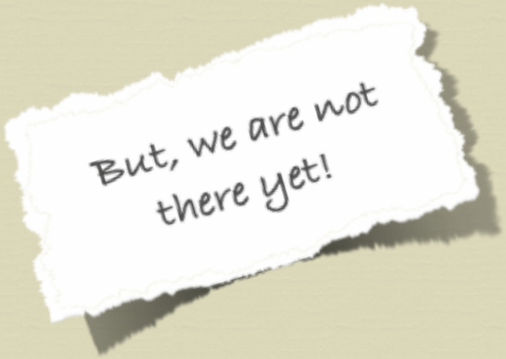
Research Group
Professor Paul Johannesson, *DSV*
M.Sc. Lars Rönnbäck, *Resight*
M.Sc. Olle Regardt, *Teracom*
Ph.D. Maria Bergholtz, *DSV*
Ph.D. Petia Wohed, *DSV*

Adapting to Change

- Anchor Modeling is particularly suited for modeling information that evolve over time, *both* in content and structure.
- Almost all information evolve in this way, with new versions and corrections of the content and new entities, properties and relationships emerging within the structure.

Temporal Database

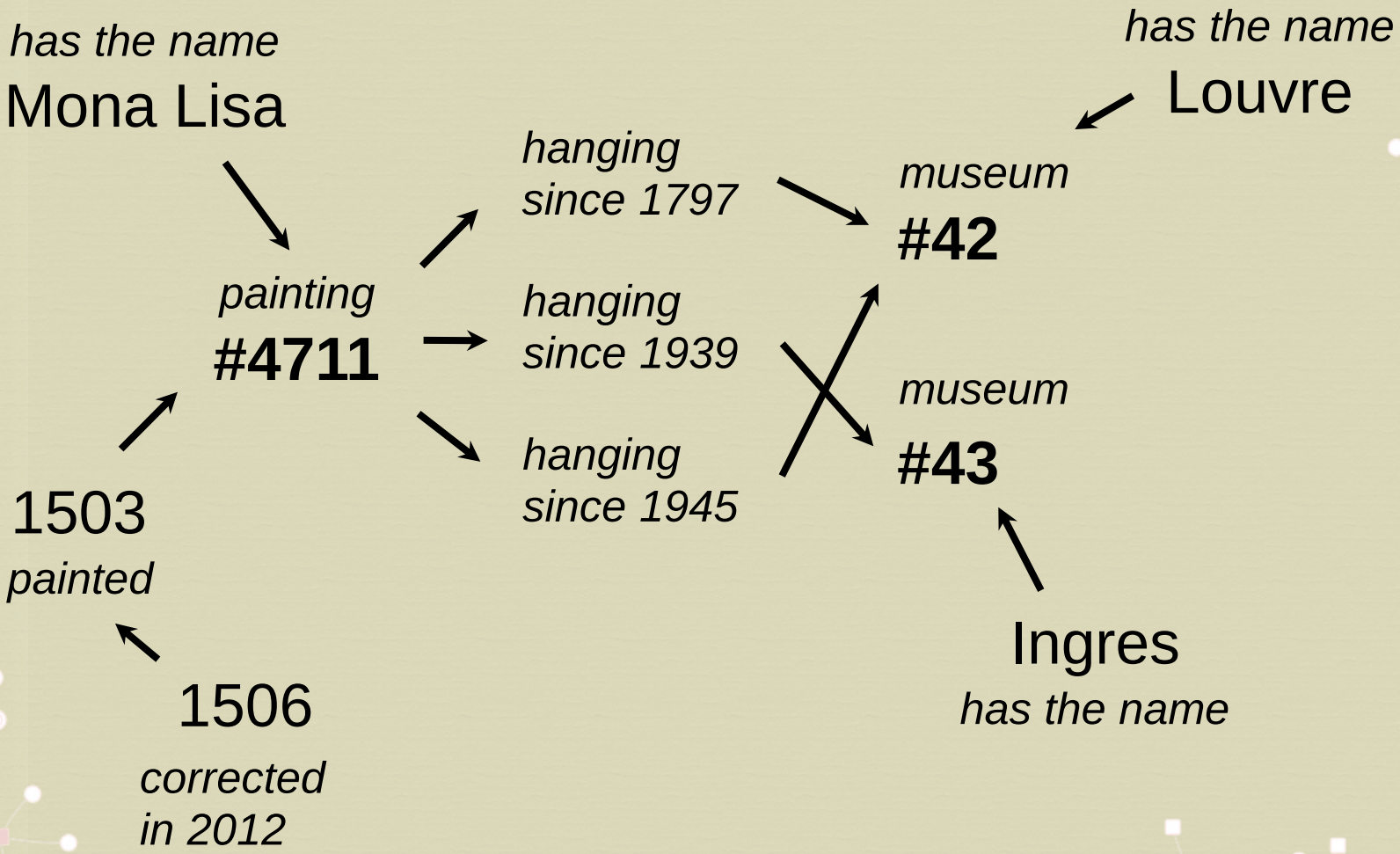
- The purpose of a **database** is to store a body of information and allow searches over it.
- The purpose of a **temporal database** is to store a body of information *under evolution* and allow *historical* searches over it.



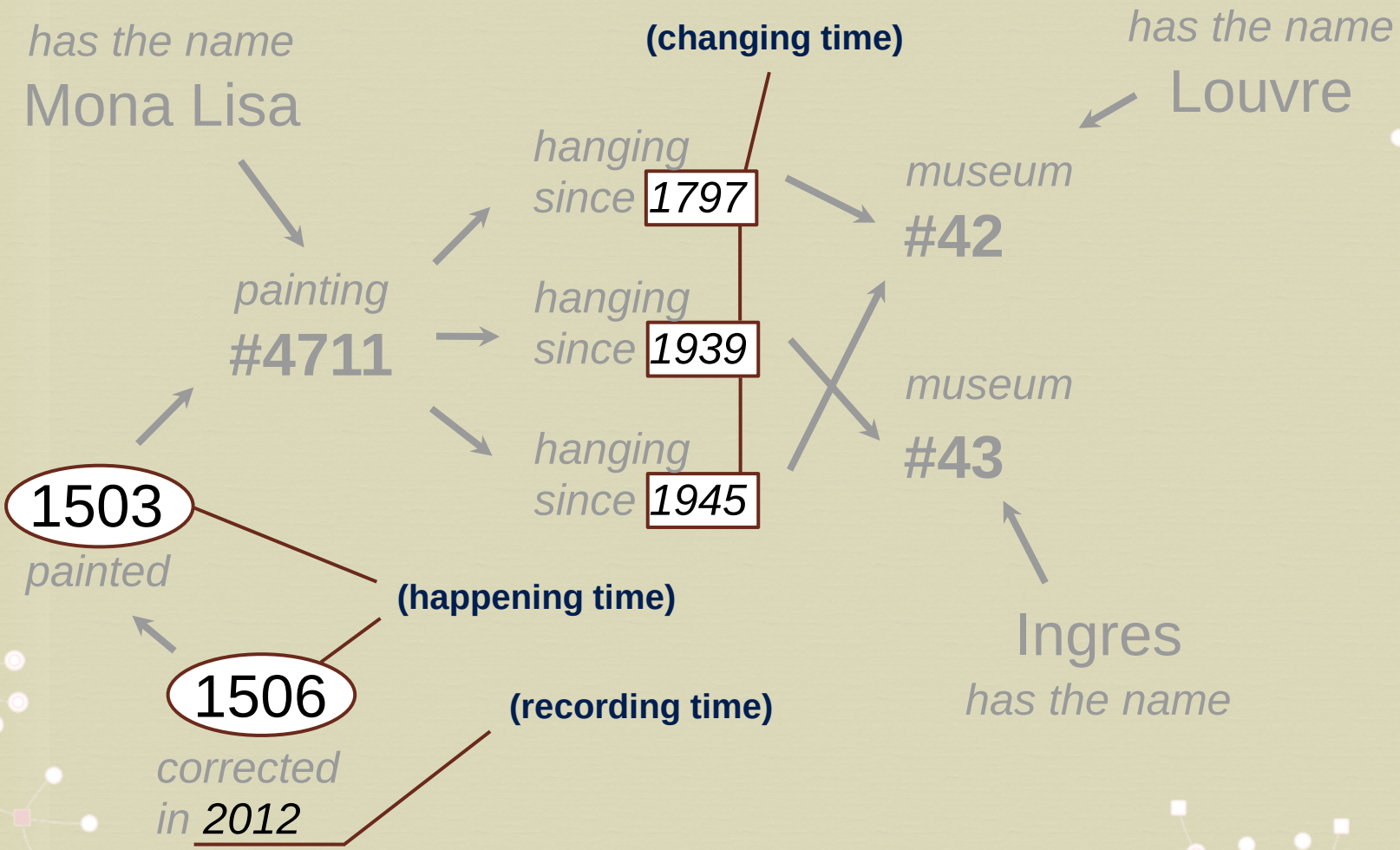
Bitemporal Modeling

- Information is not bitemporal in itself. However, most information can be bitemporally modeled:
  - Mona Lisa was painted in 1503 and has been hanging in the Louvre since 1797.
  - Research done in 2012 has shown that Mona Lisa was in fact painted somewhat later, in 1506.
  - During the Second World War, 1939–1945, the painting was moved to safety in the Ingres Museum.

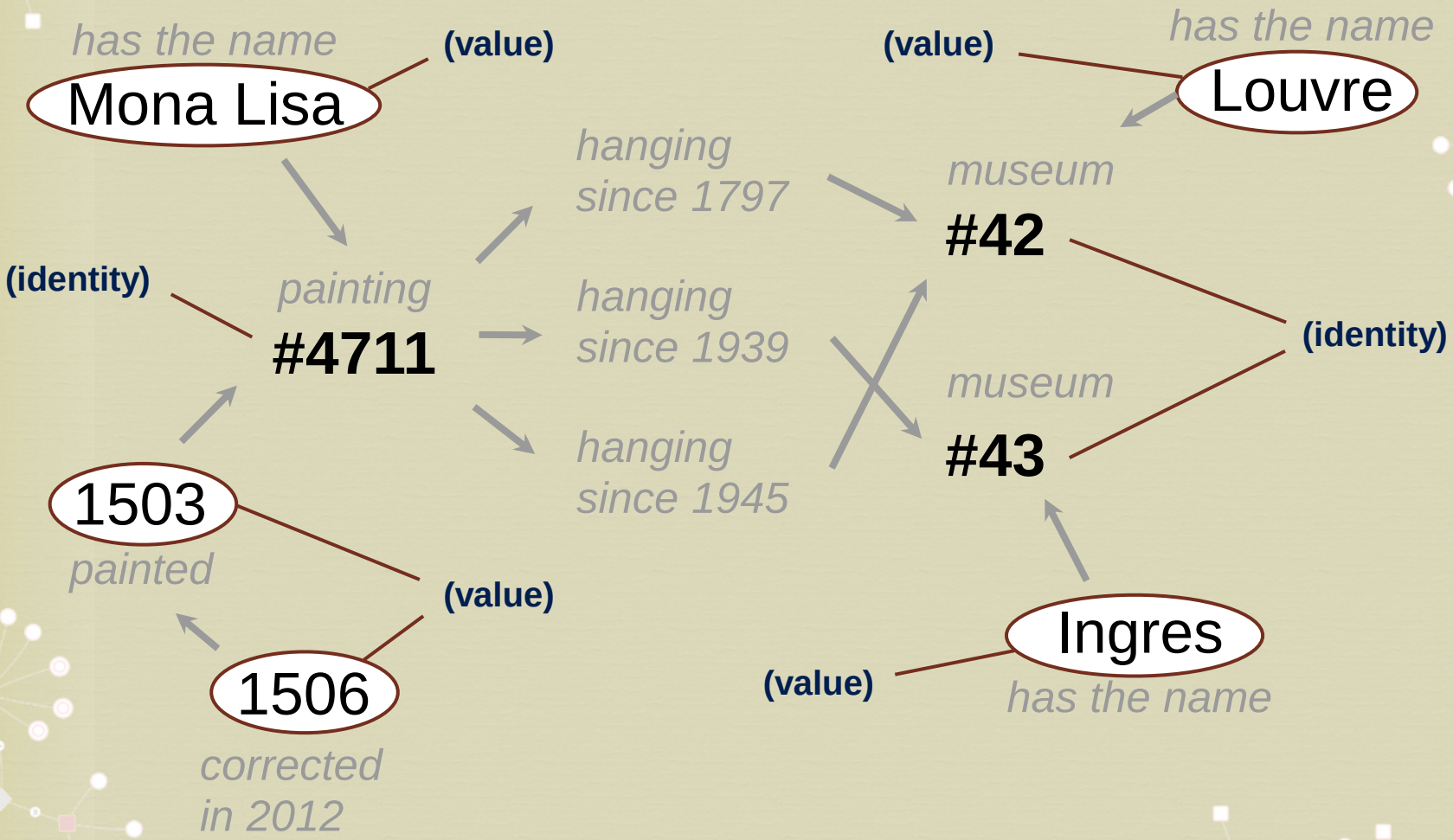
A Bitemporal Model



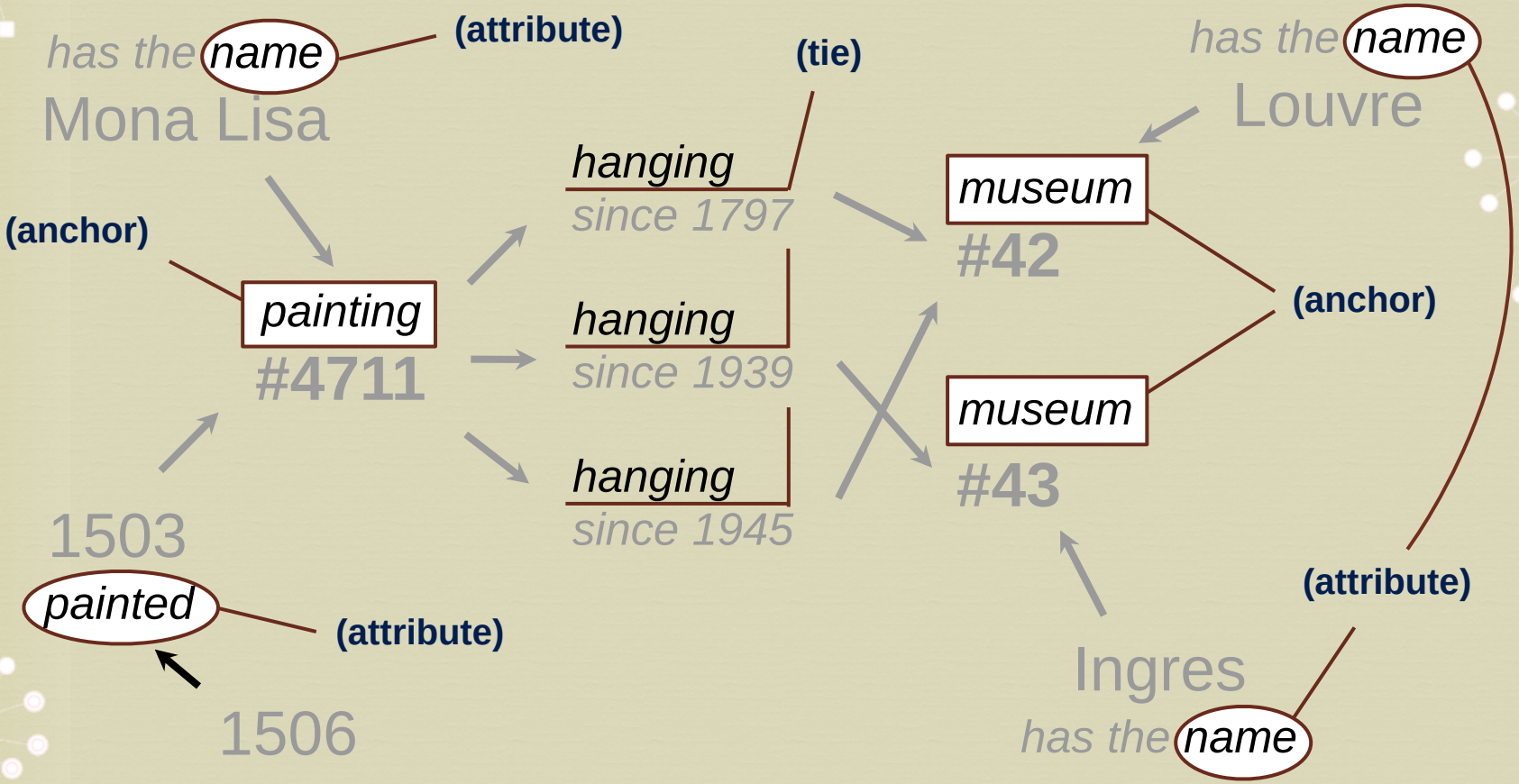
Temporal Terminology



Content Terminology



Structure Terminology



Graphical Notation

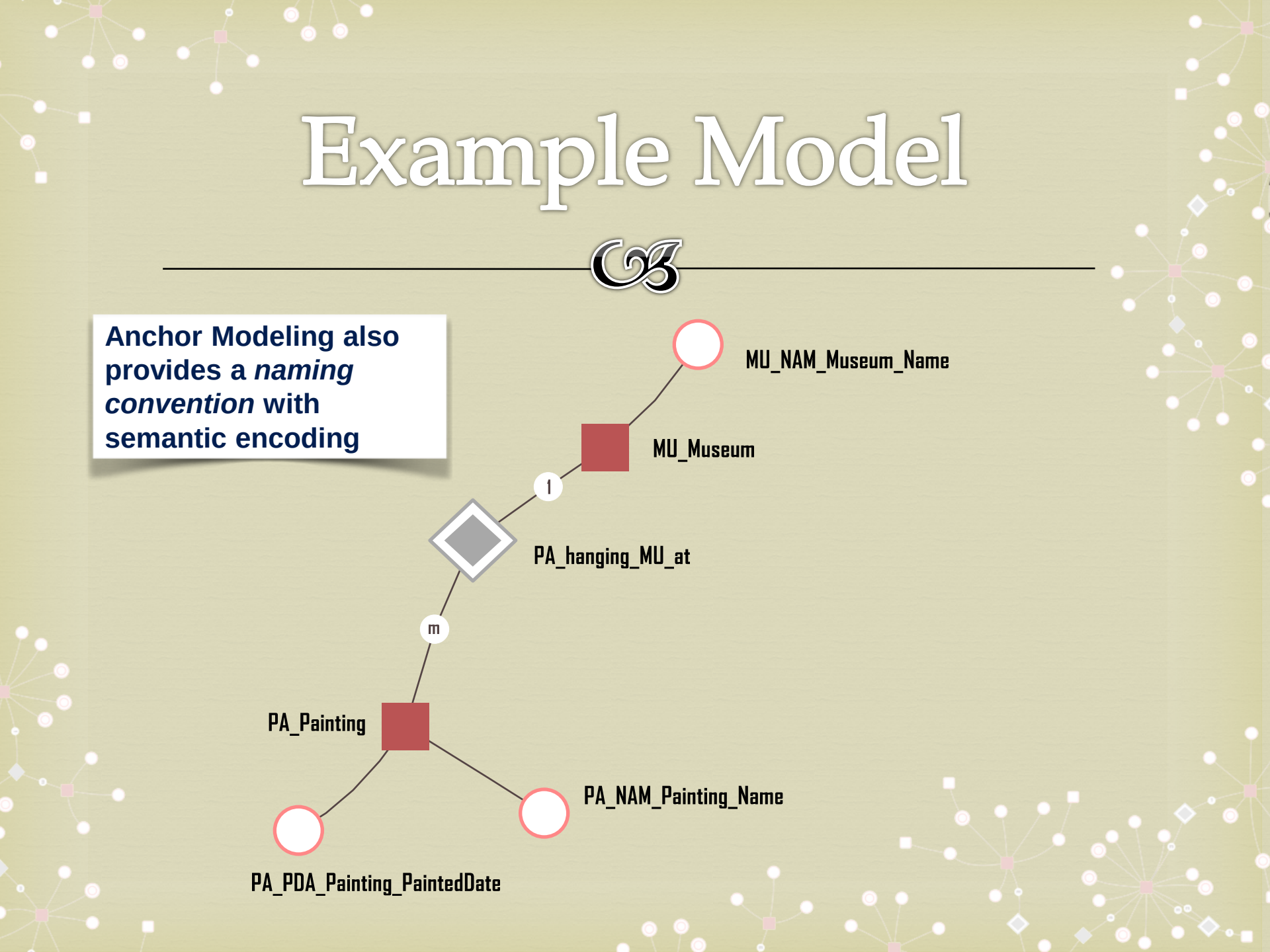
Anchors
store identities of entities

Knots
store value domains

Attributes
store values of properties
(with optional history over changing time)

Ties
store relationships between entities
(with optional history over changing time)

Example Model

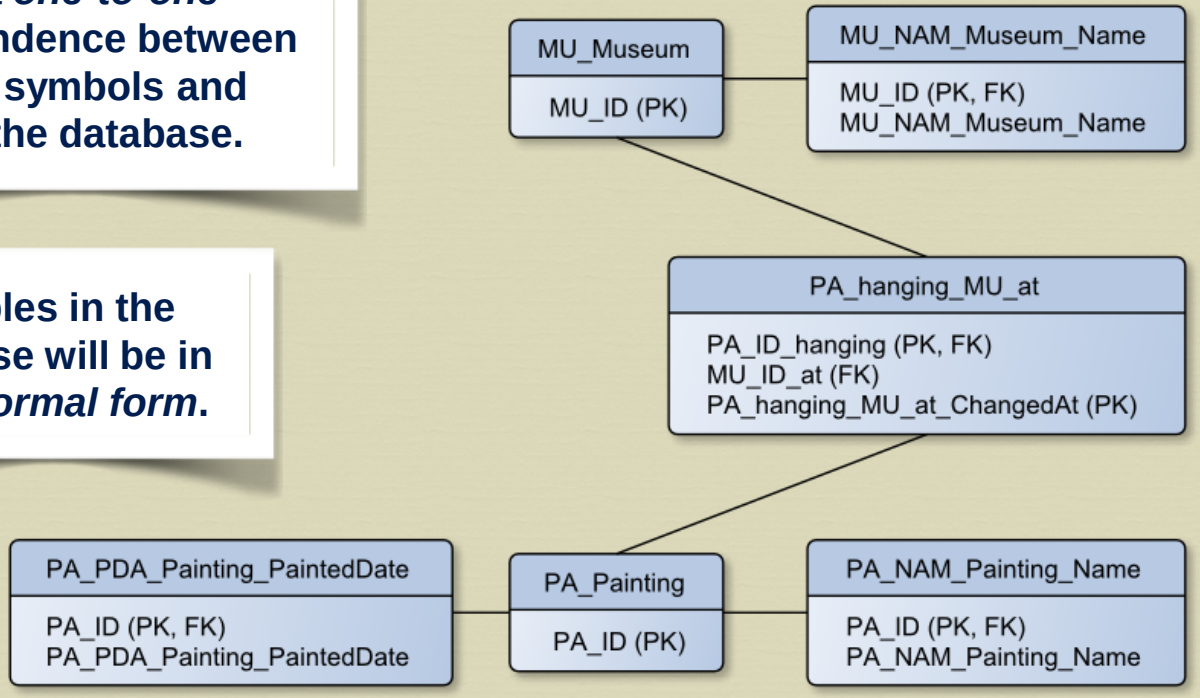


Anchor Modeling also provides a *naming convention* with semantic encoding

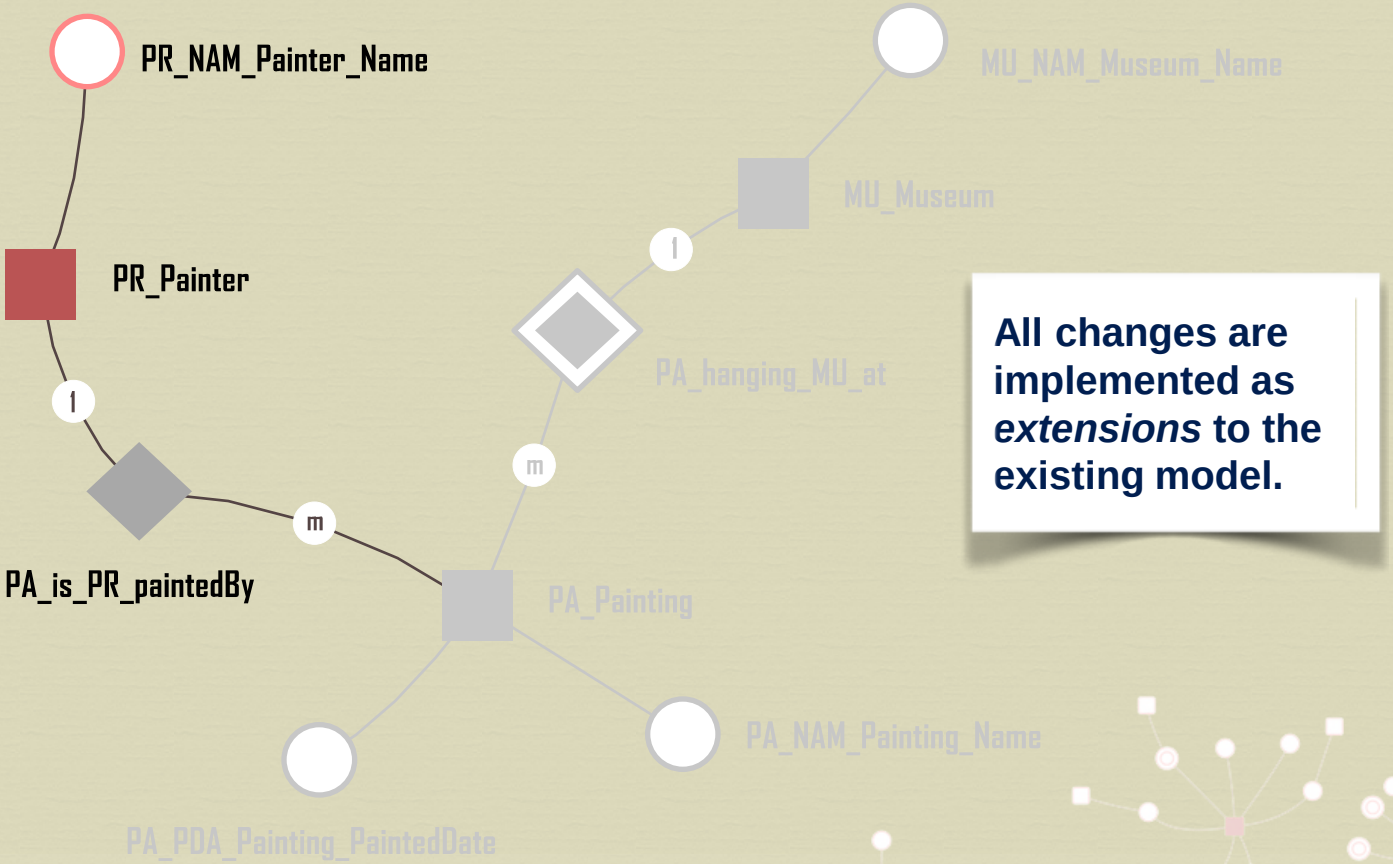
Relational Implementation

There is a *one-to-one* correspondence between graphical symbols and tables in the database.

The tables in the database will be in *sixth normal form*.

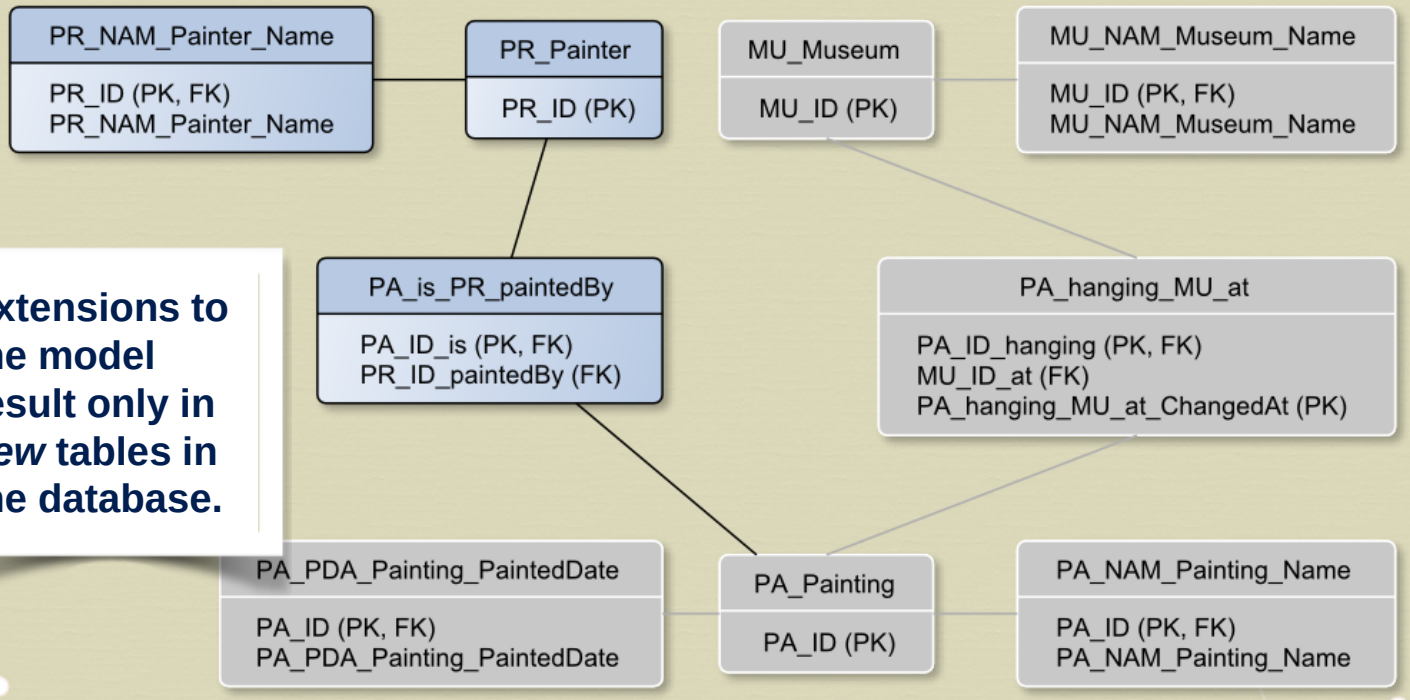


Evolved Example



All changes are implemented as *extensions* to the existing model.

Extended Implementation



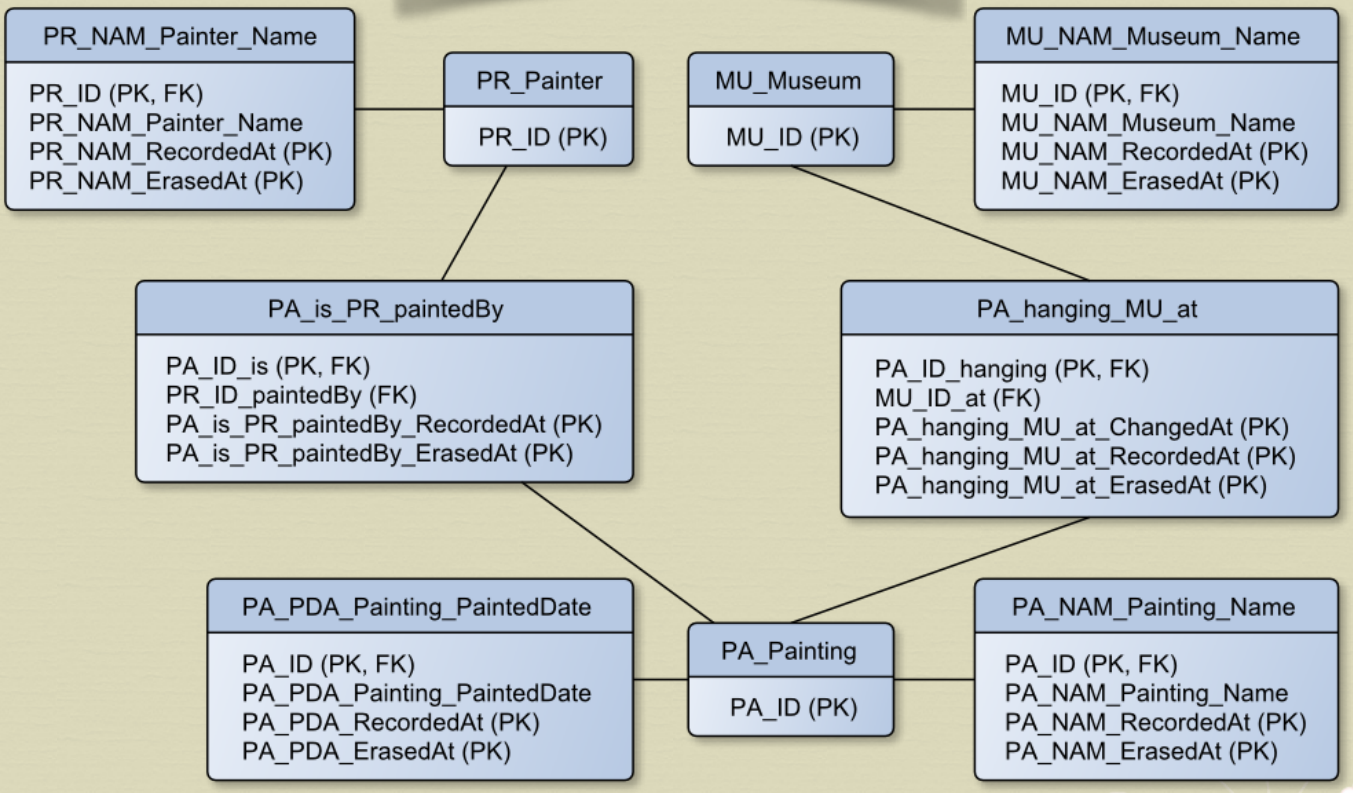
Extensions to the model result only in *new* tables in the database.

Non-destructive Evolution

- Structural changes do not affect the existing model in *any* way.
- Upgrading a database to implement such changes only result in new tables being created, therefore it can be done online and almost instantaneously.
- As a result, version *n* of an application is guaranteed to continue to run on any later version *(n+1)* of the database.

Bitemporal Implementation

Identities are *immutable.*



Temporal Independence

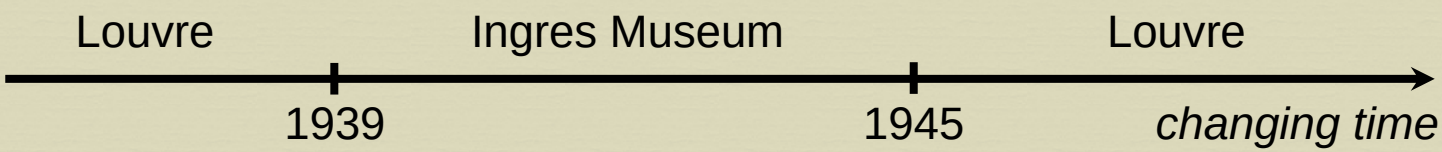
- Thanks to the structural separation of *mutable* and *immutable* content (a mutable always references an immutable) changes in any one table become *temporally independent* of those in all others.
- Anchors and knots have immutable content in Anchor Modeling. Ties and attributes are mutable both over changing and recording time.

Temporal Referential Integrity

- For every fixed point in bitemporal time: *(point-in-changing-time, point-in-recording-time)* the model behaves as if it was wholly immutable and with complete "normal" referential integrity.
- Given two points, *past* and *present*, past identities will still exist in the present due to immutability. Therefore any RI in the past will also satisfy RI in the present, yielding complete temporal RI.

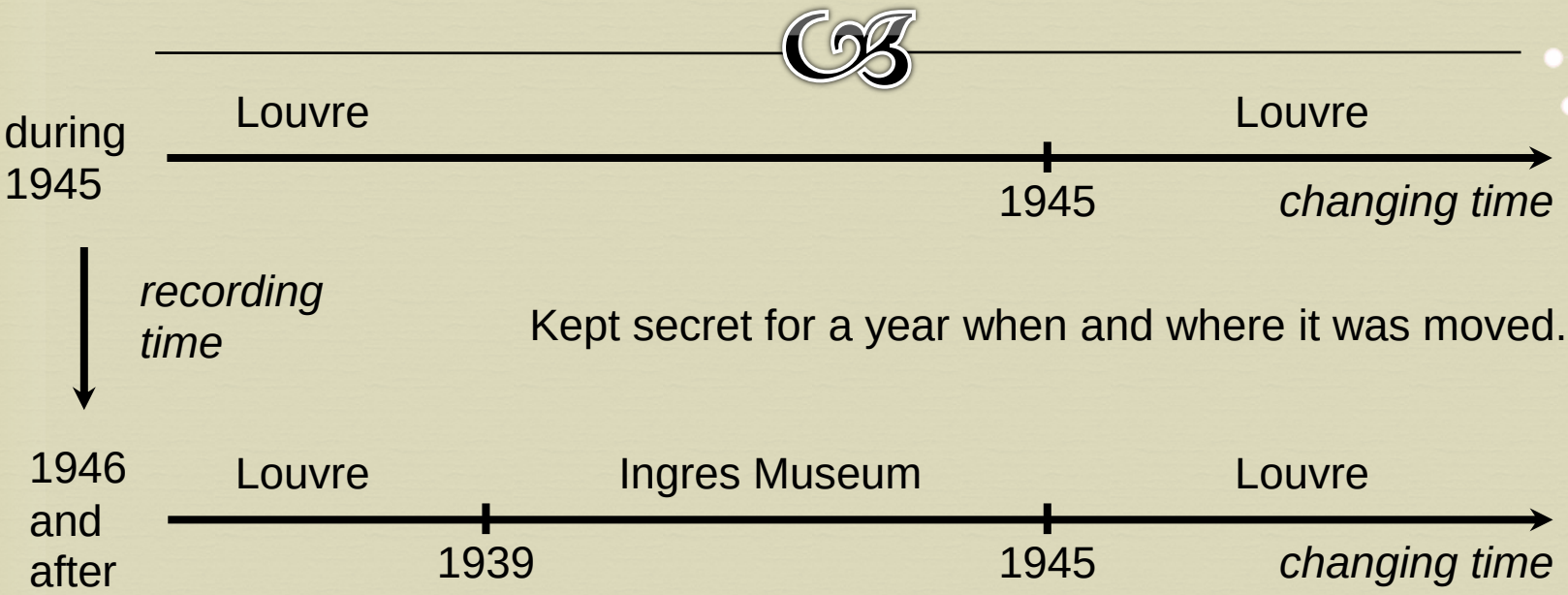
Temporal Entity Integrity

- *TEI* ensures temporal consistency for attribute and tie values, such that values undergoing evolution cannot overlap if plotted on a timeline (uniqueness over time).



- In 6NF attributes and ties reside in their own tables and TEI can be ensured for changed, recorded, and erased values using simple table constraints.

Restatement Control



- A *restatement* is a temporal no-op, desirable when information can arrive out-of-changing-time order and its source can be trusted. Restatements can be allowed or prevented individually on historized attributes and ties.

Types of Timelines

- *Happening times* are instantaneous and do not necessarily form a timeline.
  When something happens in the modeled domain (PaintDate is 1503).
- *Changing time* consists of periods forming a gapless timeline with a start, but no end.
  When a property changes its value or a relationship changes its members (hanging at Louvre changes to Ingres Museum in 1939).
- *Recording time* consists of periods forming a timeline that may contain gaps.
  The time during which values were stored in some kind of memory (in 2012 delete PaintDate 1503 and insert PaintDate 1506).

Unknown Values

Unknown values either:

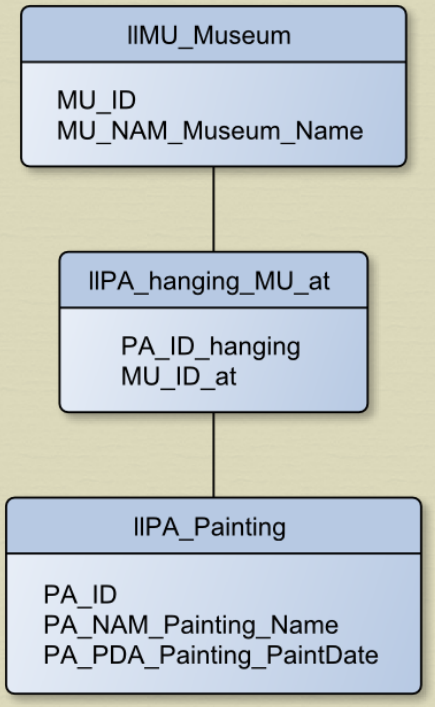
- *have business value* and should therefore be modeled explicitly, for example using the string 'Unknown' and making them visible to the end user.
- *have no business value* and should therefore be represented by gaps in the recording timeline and become invisible to the end user during the gap.

Changing time cannot be "end-dated".

Perspectives (temporality)

- Parametrized views select temporality:
  - *Latest perspective*
    Shows the latest available information.
  - *Point-in-time perspective*
    Shows information as it was on the given timepoint.
  - *Interval perspective*
    Shows information changes that happened within the given interval.

Perspectives (denormalization)



Most end users are familiar with 3NF and need not see the underlying 6NF model.

Users only need to pick the temporal perspectives suitable for their task.

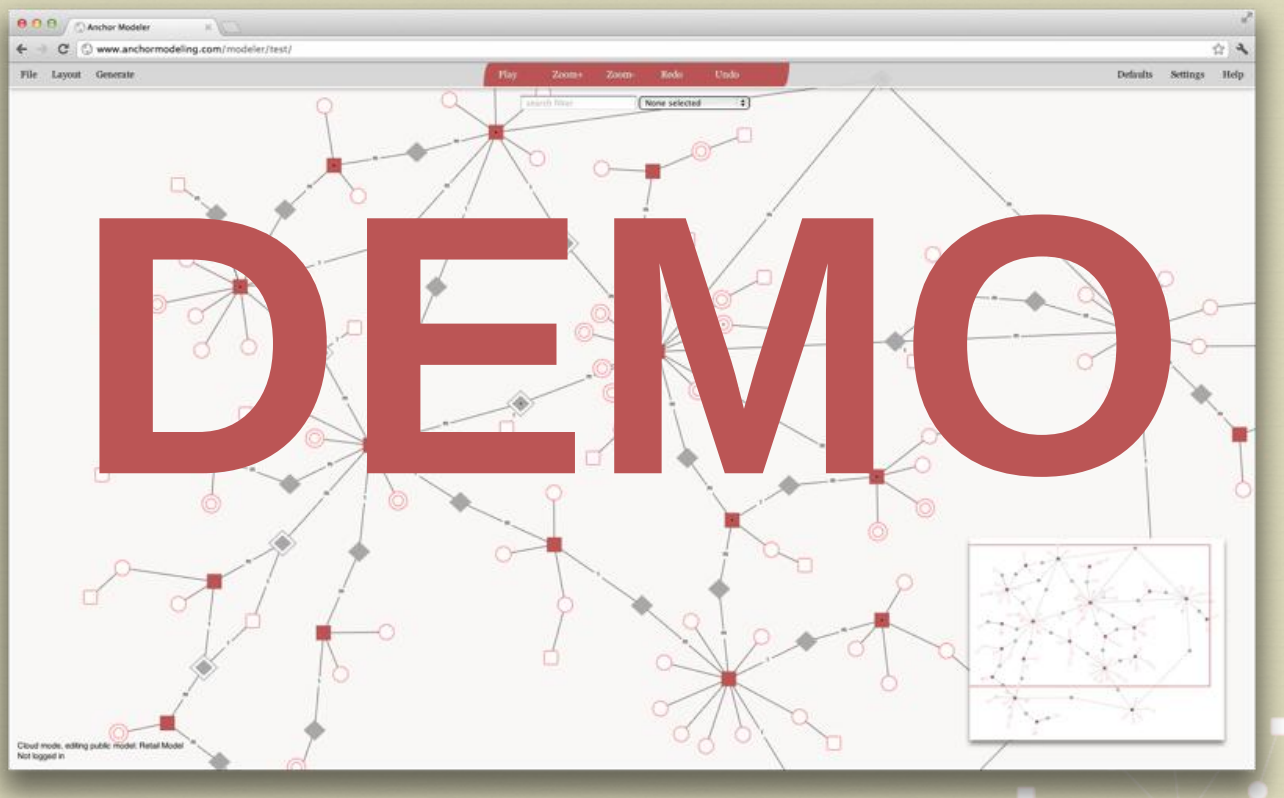
Perspectives (triggers)

```
-- happening time
insert into llPA_Painting (
  PA_NAM_Painting_Name,
  PA_PDA_Painting_PaintDate
) values ('Mona Lisa', 1503);

-- changing time
update llPA_hanging_MU_at (
set
  MU_ID = 43,
  PA_hanging_MU_at_ChangedAt = 1939
where
  PA_ID = 4711;

-- recording time
delete from llPA_Painting
where
  PA_NAM_Painting_Name = 'Mona Lisa';
```

```
insert into llPA_Painting (
  PA_NAM_Painting_Name,
  PA_PDA_Painting_PaintDate
) values ('Mona Lisa', 1506);

-- latest changing latest recording
select … from llPA_Painting;

-- point-in-changing latest recording
select … from plPA_Painting(1942);

-- point-in-changing point-in-recording
select … from ppPA_Painting(1942, 1942);
```

The *insert*, *update*, and *delete* triggers make it possible to use only the views for all data modification and querying.

The Open Source Tool



Going Beyond Bitemporal

- Anchor Modeling (but not yet the tool) supports "multi-bitemporal" modeling through the use of *timeline annexing*. This is useful for modeling concurrency and in data warehousing of multiple bitemporal information sources.
- A *timeline annex* is an extension of the primary key* to include the annex (timeline owner):

```
ID*, Value,
ChangedAt*, ChangingAnnex*,
RecordedAt*, ErasedAt*, RecordingAnnex*
```

Anchor Modeling…

- has a solid scientific formalization.
- is built on well known principles.
- is easy to learn.
- is hard to make mistakes with.
- fully supports agile development.
- shortens implementation time.
- lowers maintenance costs.
- preserves all previous versions of the database.
- increases the lifetime of the database.
- has Open Source tools.
- is free to use.

More?

- Homepage: www.anchormodeling.com
- E-mail: lars.ronnback@anchormodeling.com
- Twitter: anchormodeling
- LinkedIn: Anchor Modeling Group
- Facebook: Anchor Modeling
- Wikipedia: Anchor Modeling
- MSDN: Anchor Modeling

Do not miss our 5 minute video tutorials (total 2 hours) available on the homepage.