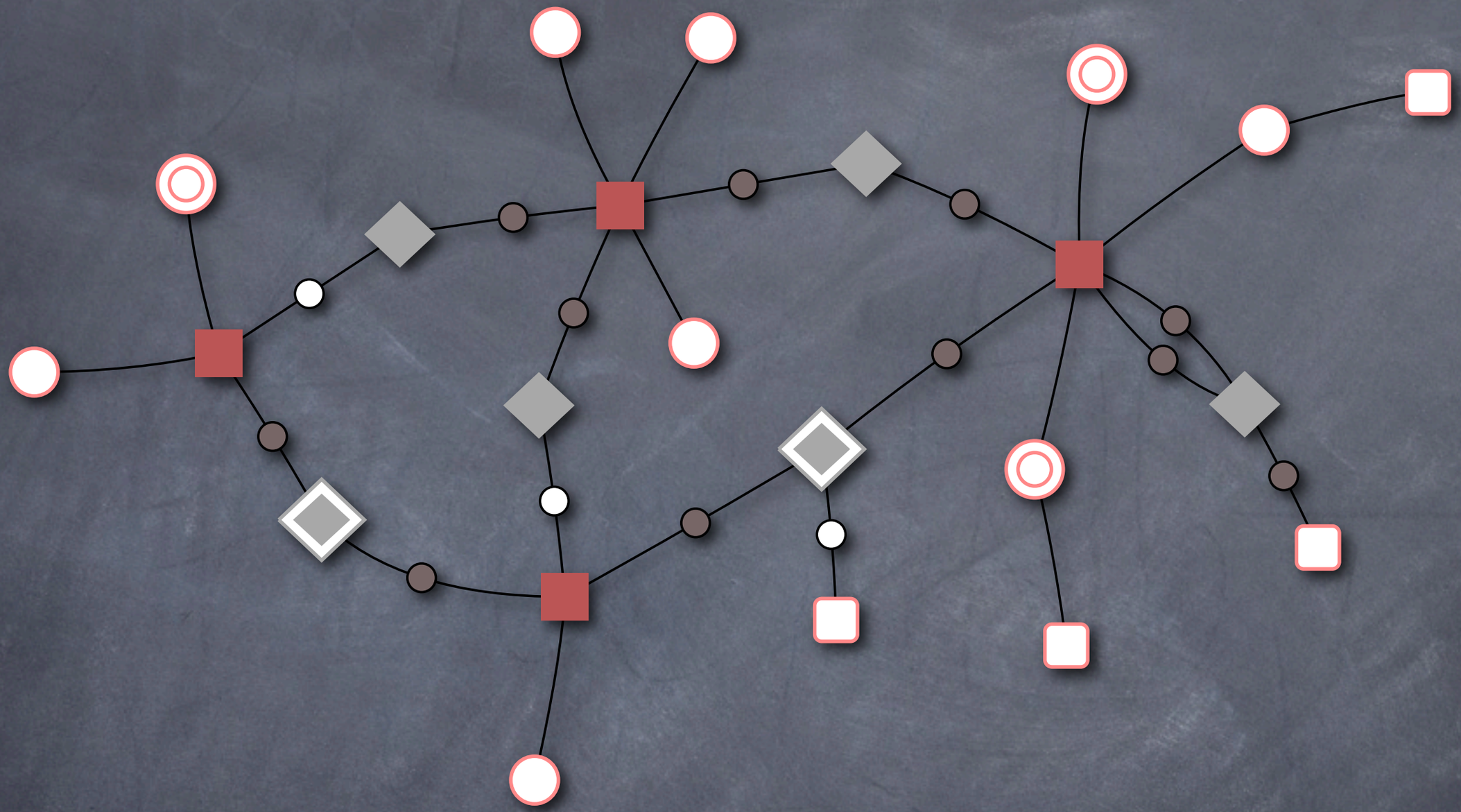
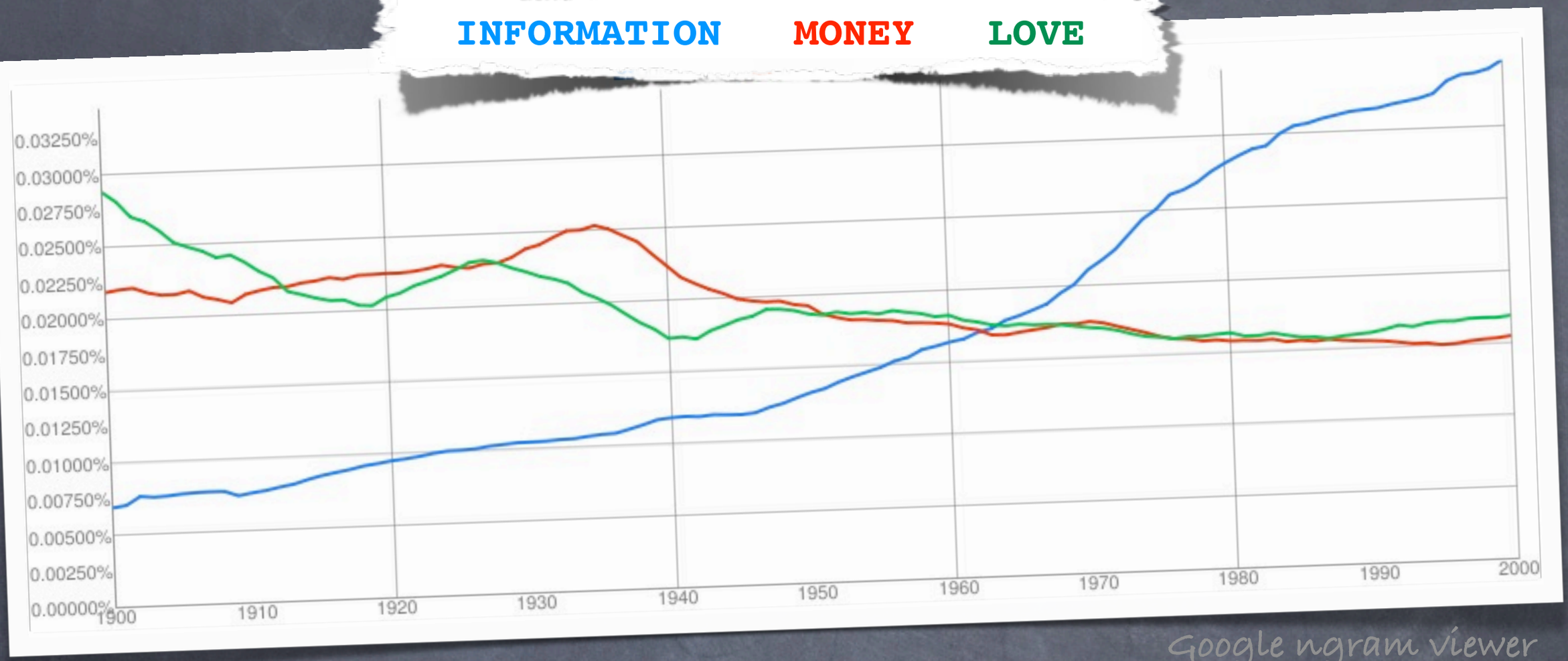


Anchor Modeling

A Technique for Information under Evolution

Lars Rönnbäck #AMweek 28/6 to 1/7, 2011



Google ngram viewer

- Graph showing relative occurrence of words in literature over the last century
- Information is rapidly becoming the most important asset


Heraclitus
500.BC
“Panta rhei”
Everything flows

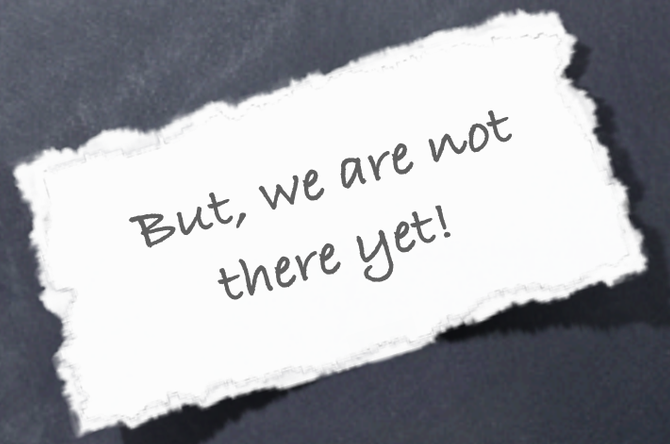
Evolving Information

- Changing content
- Changing structure
- Changing constraints
- Changing interpretation
- Changing origins
- Changing reliability

There's a big difference between saying: "This information has a 95% reliability" and "This information is 100% reliable".

What is a 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.

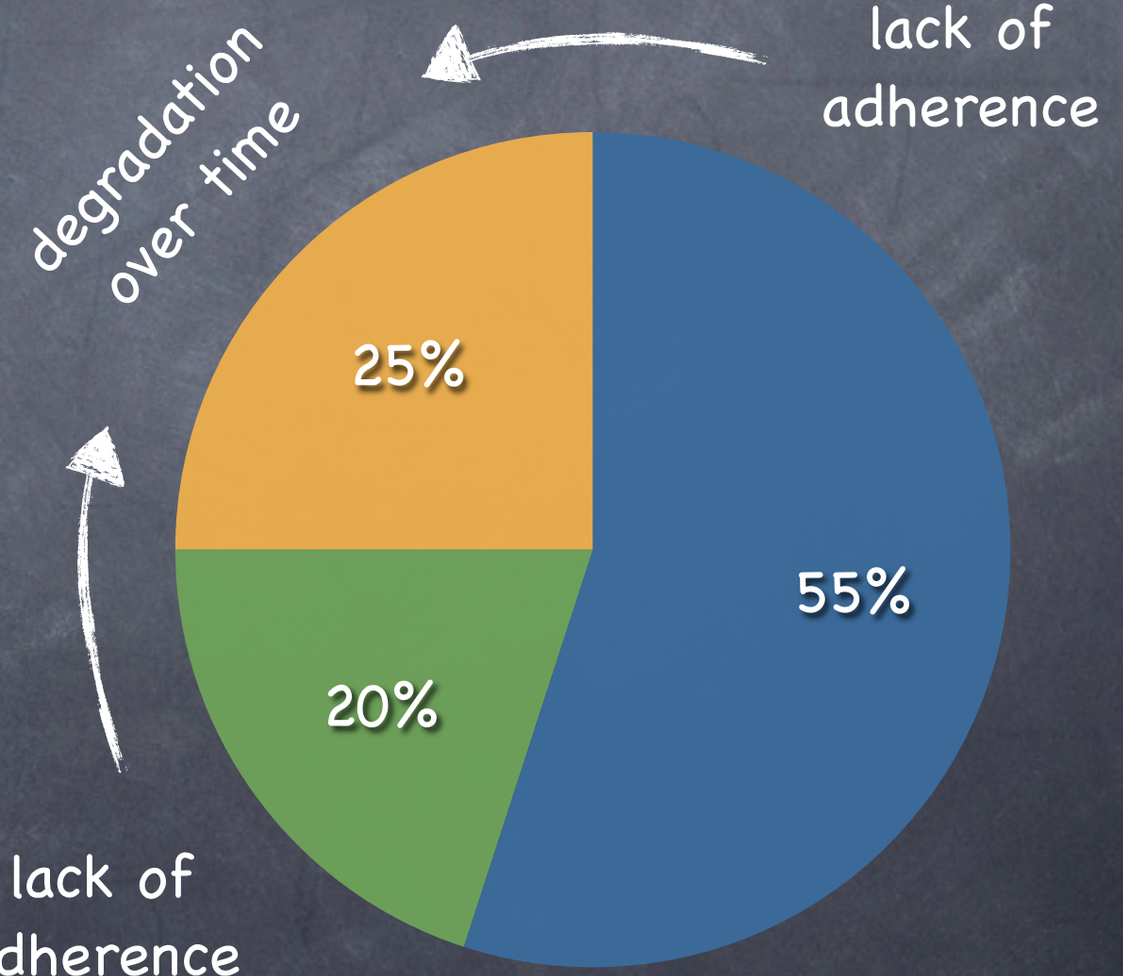


What is a Data Warehouse?

- Integrates information from many sources
- Keeps a history of changes
- Provides "one version of the truth"
- Enables reporting, ad-hoc analysis, mining
- Calculates and stores new information

The dilemma

- Many sources and many users naturally result in many changes



- Dimensional Modeling
- Normalized
- Haphazard

Patch or Redo?

- Patching works initially to cope with new requirements
- Maintenance costs usually rise proportionally to the lifetime of the data warehouse
- Redoing is unavoidable at some point
  (and for dimensional modeling sometimes accounted for)
- The average lifetime is five years
- The return of investment should and could be much better with a longer lifetime!

What is Anchor Modeling?

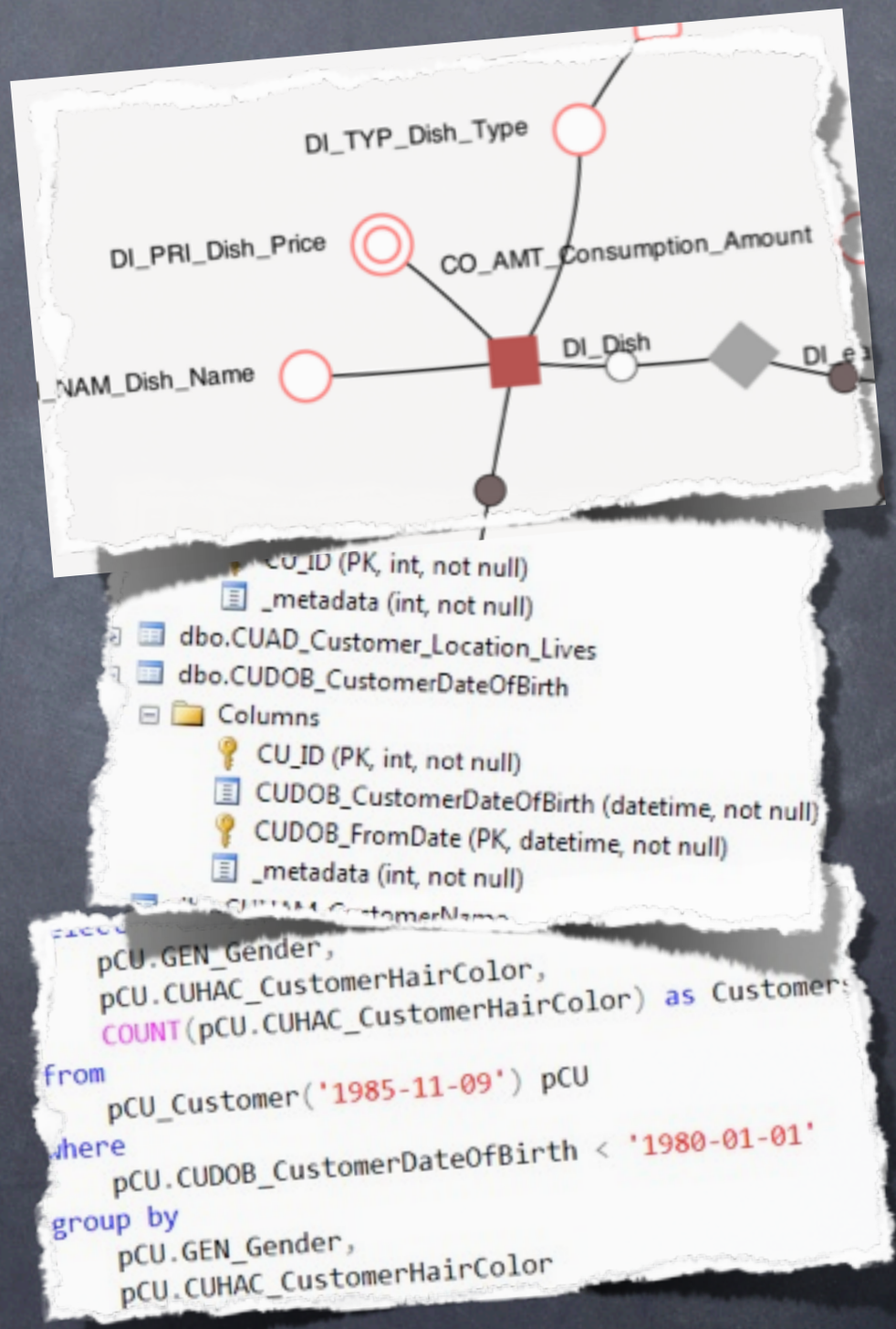
- Anchor Modeling combines normalization and emulation to provide an agile database modeling technique for evolving information that is implementable in current relational databases.
- Most, if not all, of what Anchor Modeling is doing in its physical (relational) representation could be "hidden" from the end-user in a true temporal database.

Technologies

- Entity-Relationship Modeling

one-to-one

- Sixth Normal Form Tables
- Temporal Database Emulation



History

Paul Johannesson

Lars Rönnbäck

Olle Regardt

Maria Bergholtz

Petia Wohed

Best Paper Award @ ER'09

research

consulting

03	04	05	06	07	08	09	10	11
DW	MDM	EDW	DW	TDWI	SU WWW	DW ER09	DKE TOOL	DW GSE AMW

Philosophy

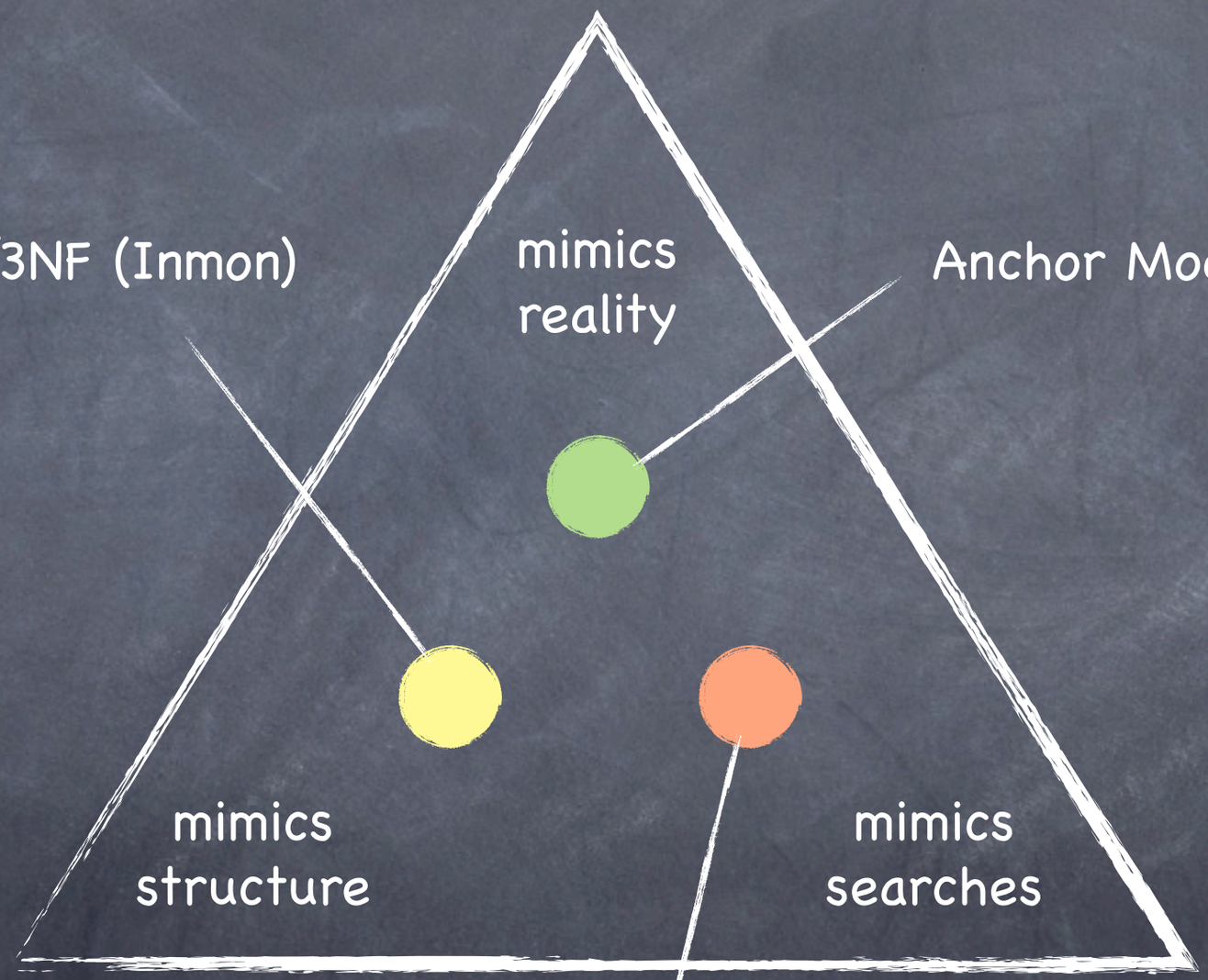
- Make modeling free from assumptions
- Make modeling agile and iterative
- Make evolution non-destructive
- Do not duplicate information
- Do not alter existing information
- Decouple metadata from the model
- Provide a simple interface for queries

Evolution in Anchor Modeling

- **Changing content**
  full support [6NF + time of change]
- **Changing structure**
  full support (through extensions) [non-destructive schema evolution]
- **Changing constraints**
  minimal support [only primary and foreign keys]
- **Changing interpretation**
  achievable [explicitly modeled]
- **Changing origins**
  restricted support [using metadata]
- **Changing reliability**
  restricted support [using metadata]

Positioning
Anchor Modeling

Domain driven modeling



Data driven modeling

Use-case driven modeling

Basic Notions

Attributes - properties

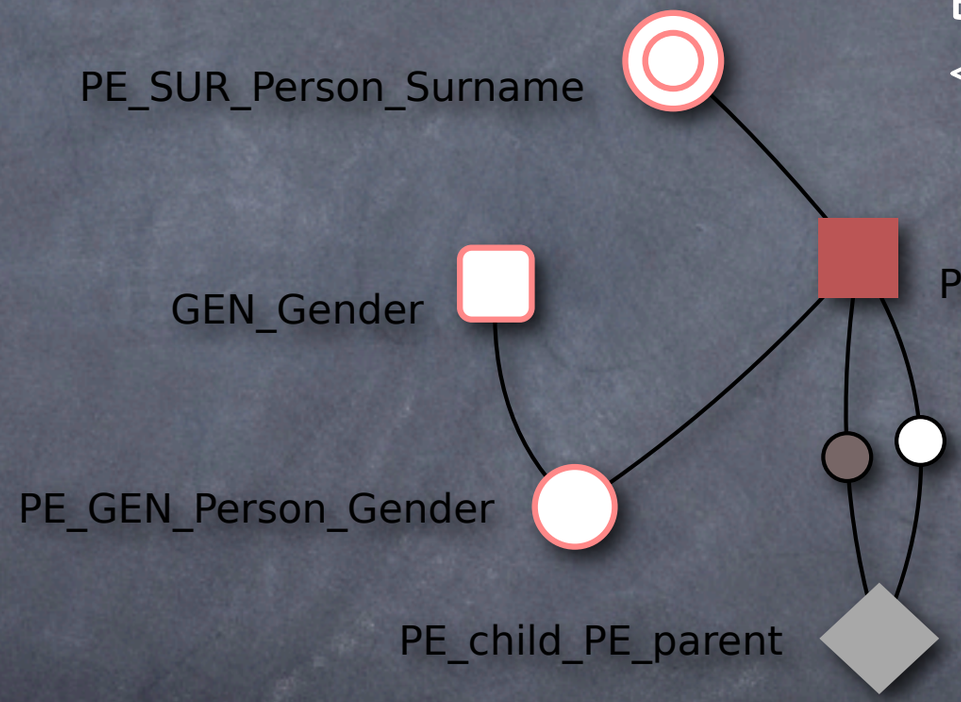
Example: The surname of a Person

<#42, ‘Rönnbäck’, 2004-06-19>

Anchors - entities

Example: A Person

<#42> (holds only identities of entities)



filled = primary

Knots - shared properties

Example: The gender of a Person

<#1, ‘Male’> + <#42, #1>

Ties - relationships

Example: The children of a Person

<#42, #4711>

Naming convention

- **Anchors**
  Two letter mnemonic + descriptor
  `PE_Person`

  unique in the model

- **Knots**
  Three letter mnemonic + descriptor
  `GEN_Gender`

  unique on the anchor

- **Attributes**
  Inherited anchor mnemonic + three letter mnemonic + inherited anchor descriptor + descriptor
  `PE_SUR_Person_Surname`

- **Ties**
  Inherited anchor and knot mnemonics separated by the roles they play in the relationship
  `PE_child_PE_parent`

Historization

```
<#42, 'Samuelsson', 1972-08-20>
<#42, 'Rönnbäck',   2004-06-19>
```

closed interval
historical information

open interval
current information

asynchronous insert

```
<#42, 'Sommare',    1982-03-01>
```

Note that **UPDATE** is never allowed or needed in an anchor database

Historization is done using the time of change as the start of an interval implicitly closed by another instance of the same identity with a later time of change.

Temporal perspectives

- **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

- **Natural perspective**
  Converts natural to surrogate identities
  (for composite identities these may span over several anchors)

They all look like **3NF**!

Table elimination

- A table *T* can be removed from the execution plan if:

  a) no column from *T* is explicitly selected

  b) the number of rows in the returned data set is not affected by the join with *T*

The temporal perspectives regain all benefits from **6NF**!

Indexing

- All primary indexes are clustered indexes (index organized tables) which use no extra space

```
1|Peter    |2004-02-13 20:08
2|Paul     |2007-01-01 13:54
2|John     |2006-08-20 15:15
2|Matthew  |2002-10-15 13:20
2|Ringo    |2001-01-02 01:18
3|George   |2007-09-19 08:00
```

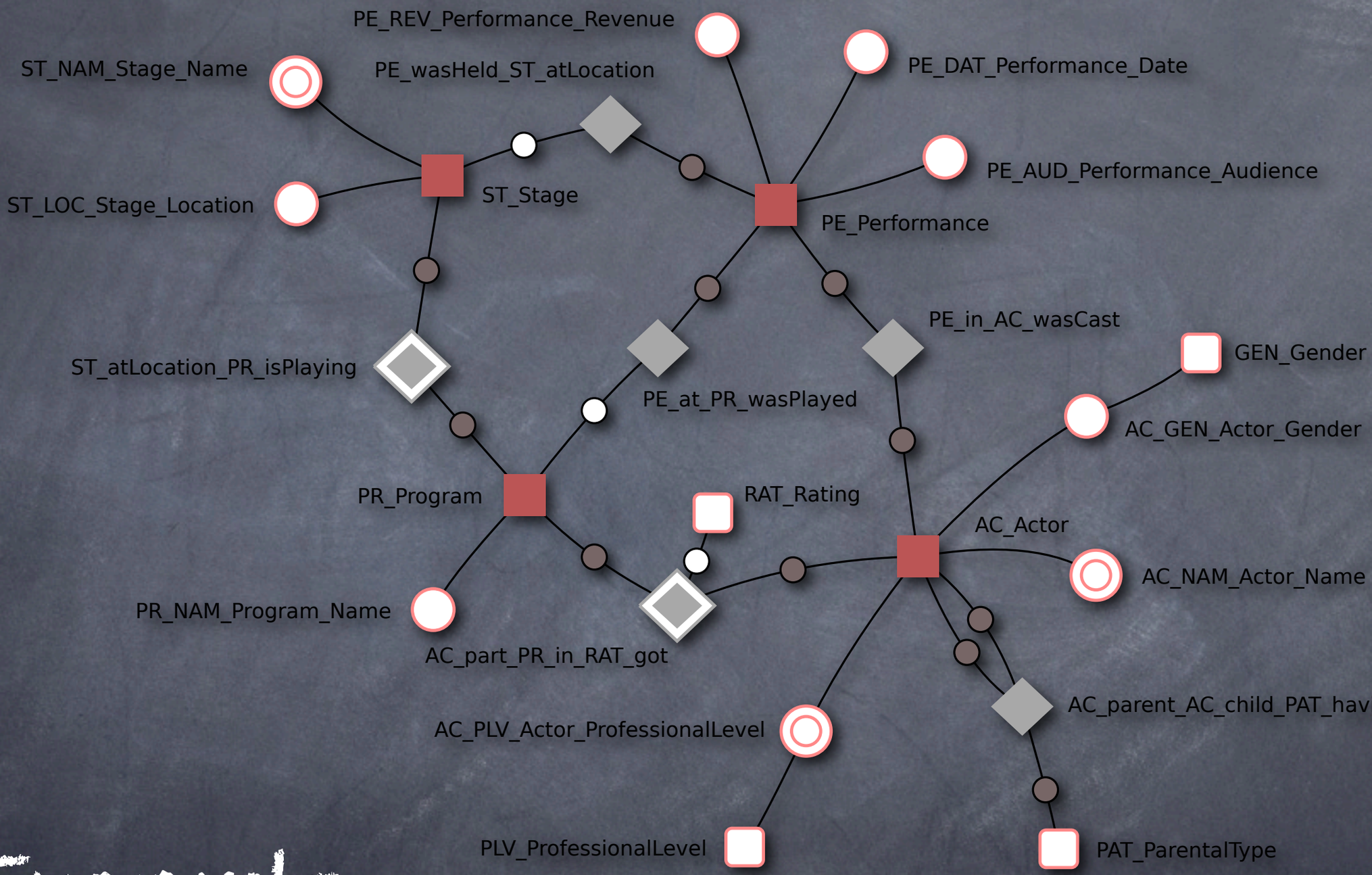
clustered index on identity + historization

- Secondary indexes are very rarely needed

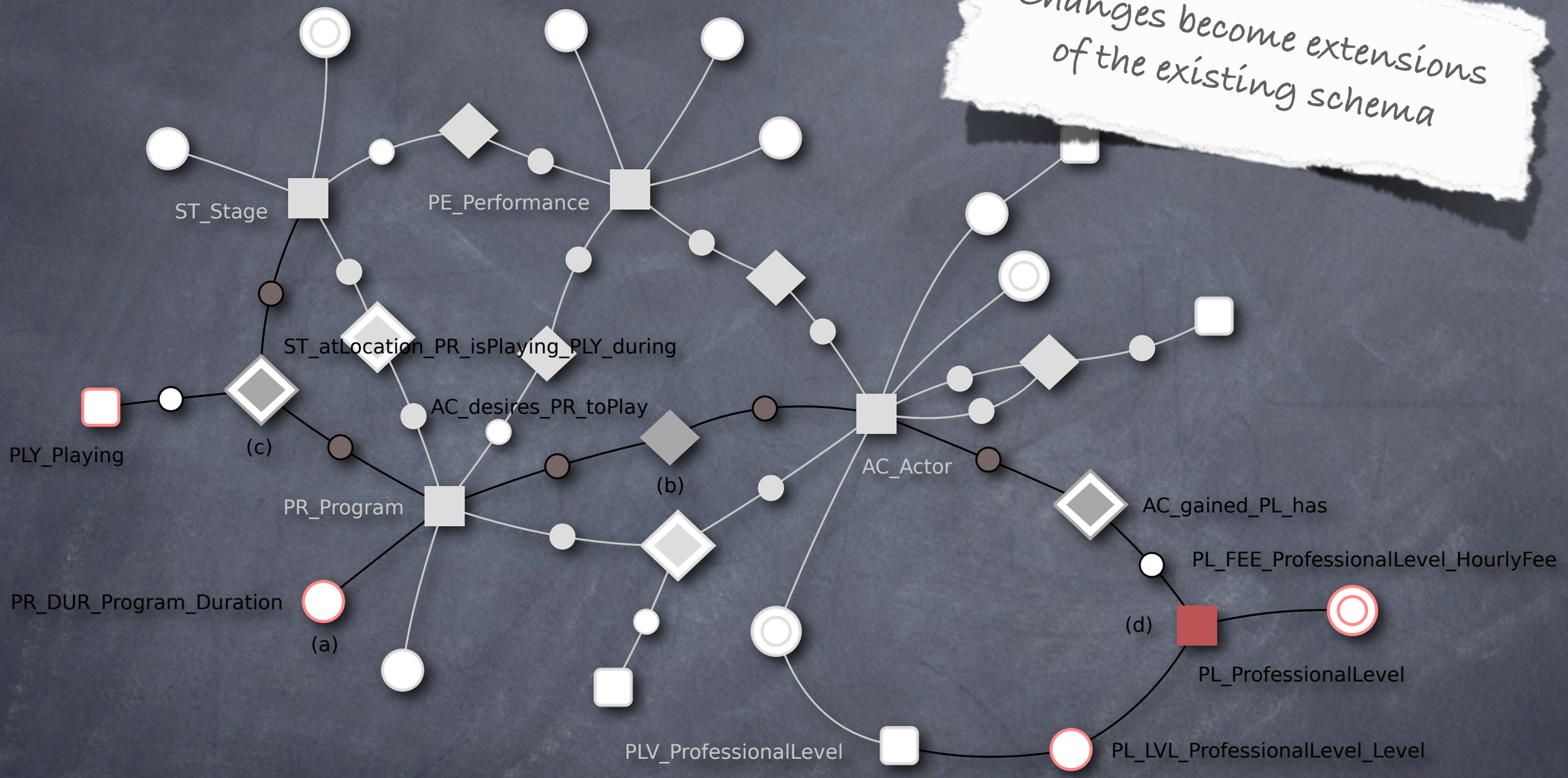
Performance boosters

1. The data is time dependent and its changes need to be kept.
2. The data is sparse, i.e. many entities lack some of their attribute values.
3. The number of distinct data values is small compared to the number of all data values.
4. Identifiers constitute a small portion of the total amount of data.
5. There are many identifiers.
6. There are many properties.
7. Searches address relatively few of the properties in the entities over which the search is done.
8. Searches include conditions that impose bounds on values.

Example model



Changes become extensions of the existing schema



Schema evolution

all previous versions of the schema are available as subsets of the current schema

Tracking changes

- **_Schema**
  Table containing the schema in XML format and when it was activated

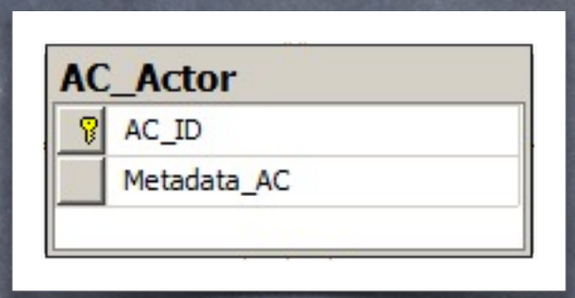
- **_Anchor**, **_Knot**, **_Attribute**, and **_Tie**
  Views shredding the XML to columns over which detailed changes can be tracked

- **_Evolution**
  Table-valued function comparing the schema valid at the given timepoint with the current database tables

Modeling anchors

- **Guideline 1:** Use anchors for modeling core entities and transactions.

relational implementation:

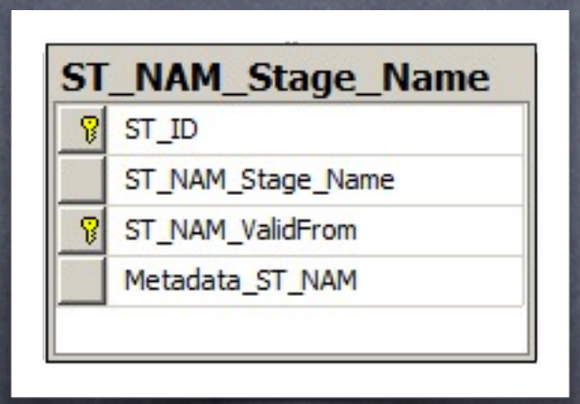


anchor

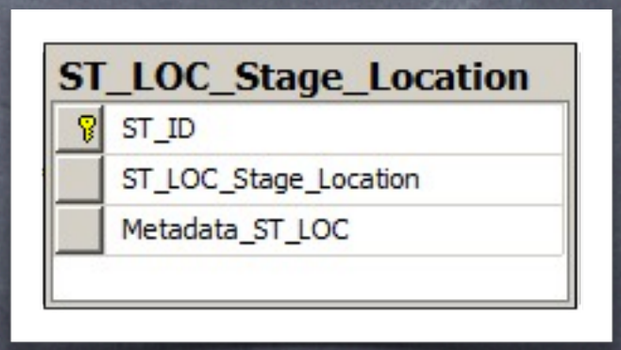
Modeling attributes

- **Guideline 2a:** Use a historized attribute if versioning of attribute values are of importance, otherwise use a static attribute.

relational implementation:



historized attribute

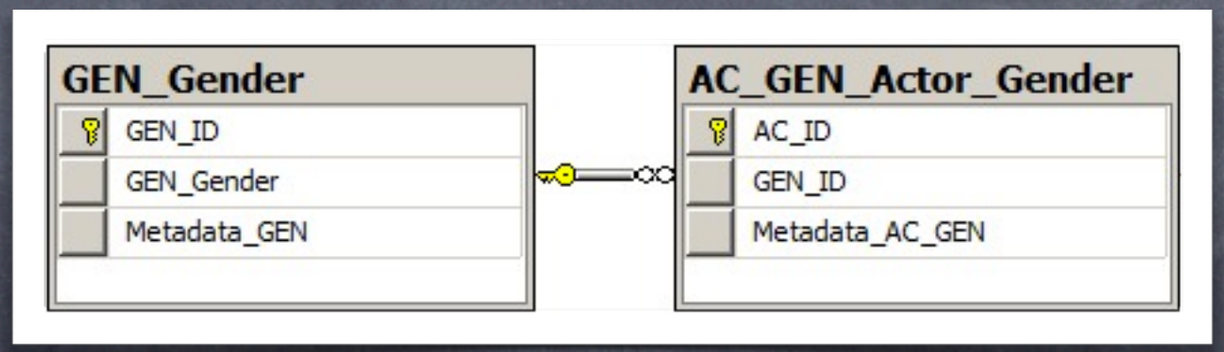


static attribute

Modeling attributes

- **Guideline 2b:** Use a knotted static attribute if attribute values represent categories or can take on only a fixed small set of values, otherwise use a static attribute.

relational implementation:

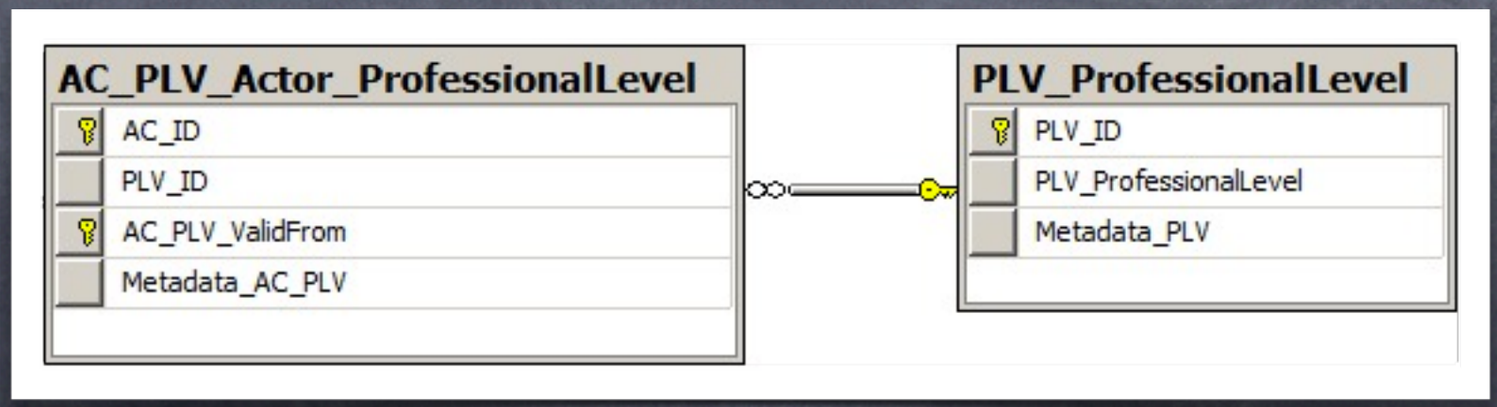


knotted static attribute

Modeling attributes

- **Guideline 2c:** Use a knotted historized attribute if attribute values represent categories or a fixed small set of values and the versioning of these are of importance.

relational implementation:



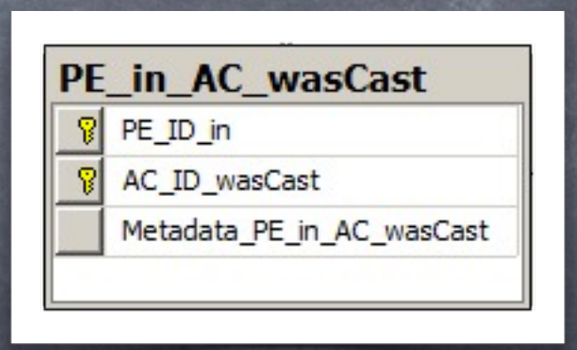
knotted historized attribute

Modeling ties

- **Guideline 3a:** Use a historized tie if a relationship may change over time, otherwise use a static tie.

relational implementation:

historized tie

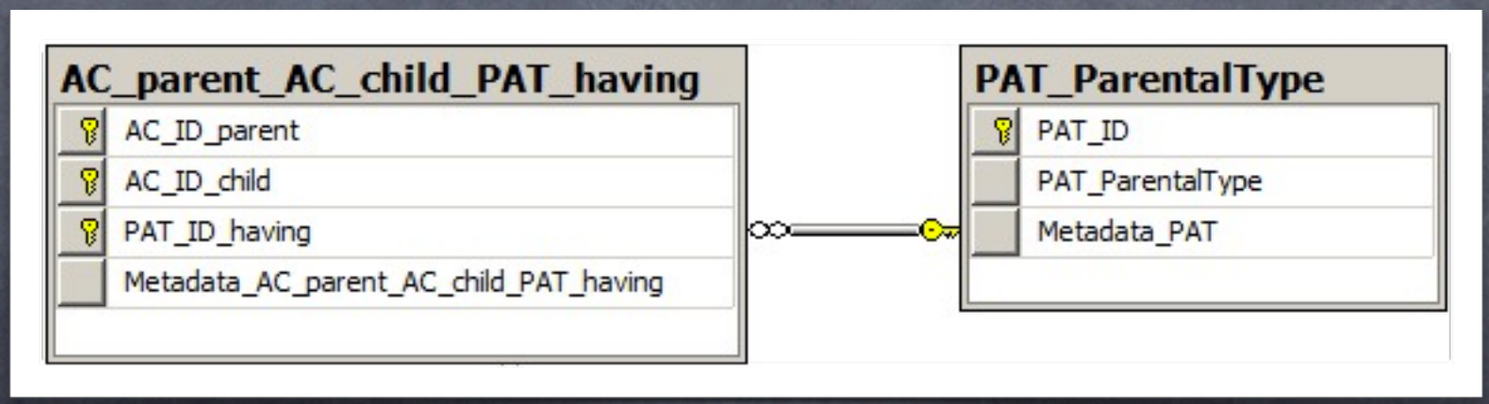


static tie

Modeling ties

- **Guideline 3b:** Use a knotted static tie if the instances of a relationship belong to certain categories, otherwise use a static tie.

relational implementation:

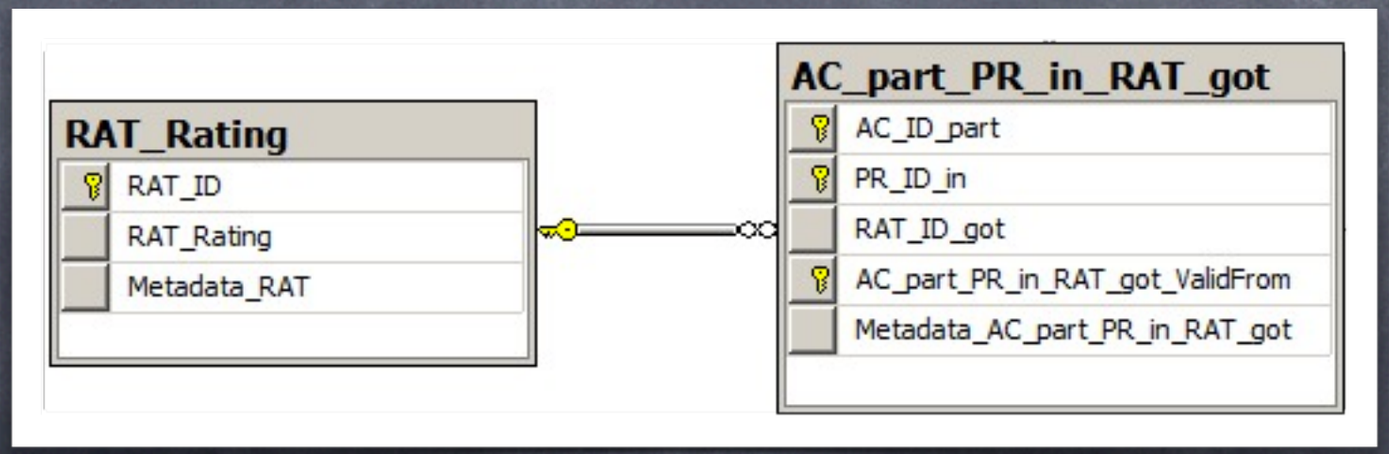


knotted static tie

Modeling ties

- **Guideline 3c:** Use a knotted historized tie if the instances of a relationship belong to certain categories and the relationship may change over time.

relational implementation:

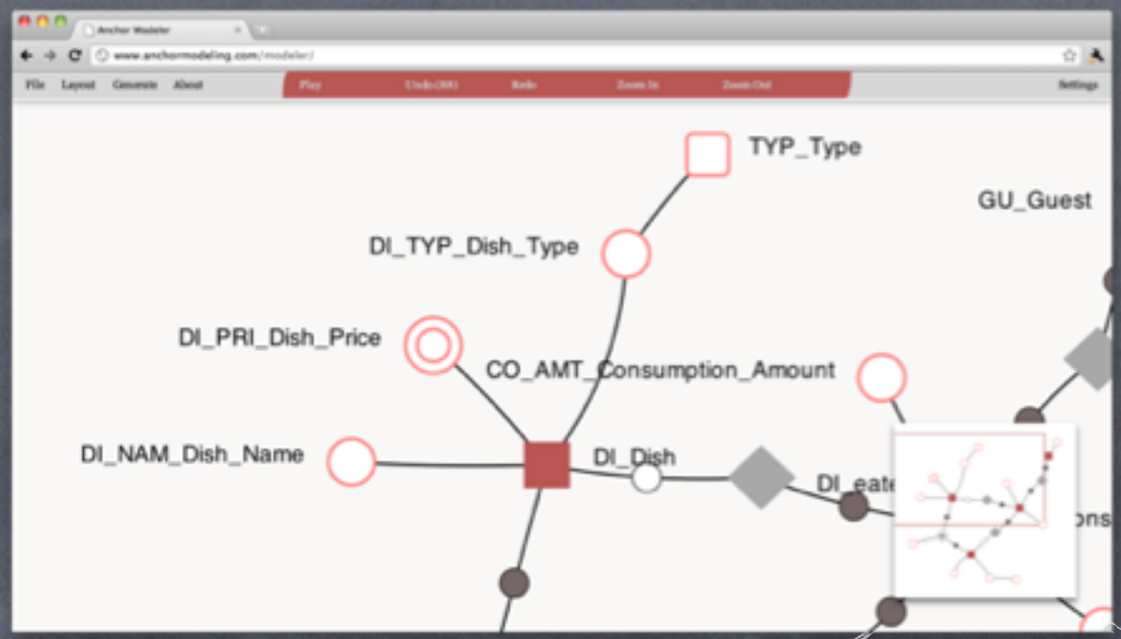


knotted historized tie

The Modeling Tool

www.anchormodeling.com/modeler

- Open Source
- Online (**HTML5**)
- Free to use
- In the Cloud
- XML Interchange Format
- Automatic generation of SQL scripts
- Interactive (force-directed) Layout Engine



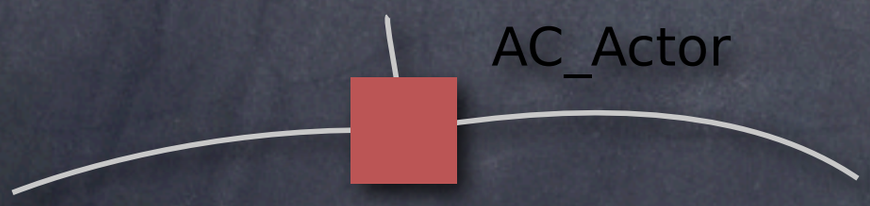
DEMO!

Inserting data

These are also automatically generated by the tool

- **OLTP-like requirements**
  Insert trigger on the latest view where an identity is created if not provided

- **DW/OLAP-like requirements**
  Stored procedure generating the given number of identities



anchors hold the identities

Important Benefits

- Handles evolving information (keeping the integrity intact)
- Increases longevity (databases with long life expectancy)
- Simplifies modeling concepts (less prone to error)
- Enables modular and iterative development
- Needs no translation logic to the physical layer
- Automates generation of scripts
- No downtime when upgrading databases
- Scans only relevant data during searches
- Sparse data cause no gaps (no null values)

More Information

Homepage:
http://www.anchormodeling.com

Blog . Video Tutorials . Modeling Tool

Twitter: **anchormodeling**

E-mail: **lars.ronnback@anchormodeling.com**

LinkedIn Groups:
Anchor Modeling
Temporal Data Modeling
Temporal Data