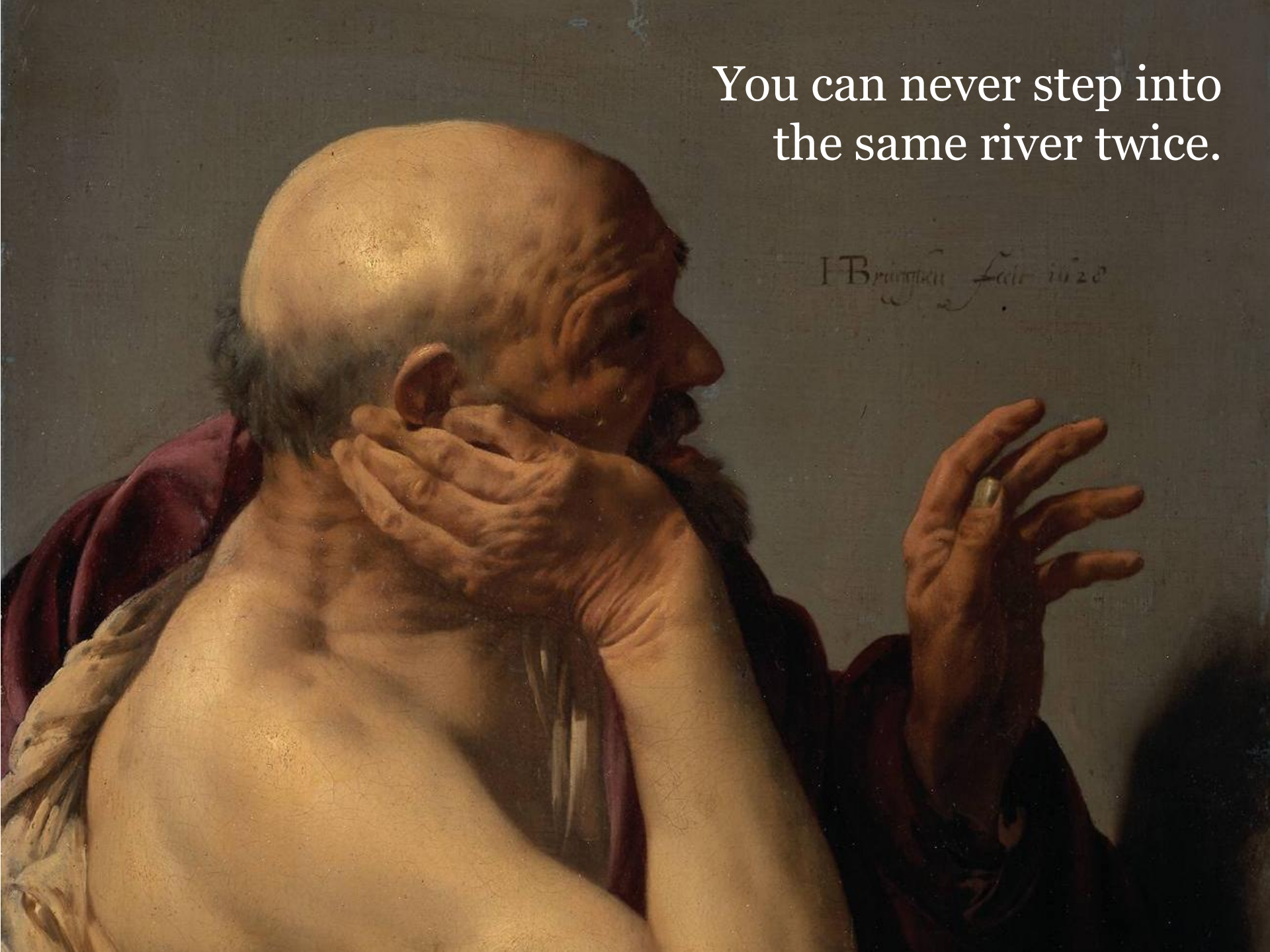


Lars Rönnbäck - Olle Regardt
Anchor Modeling
 IN THE DATA WAREHOUSE

You can never step into
the same river twice.

I. Brington Fair 1620



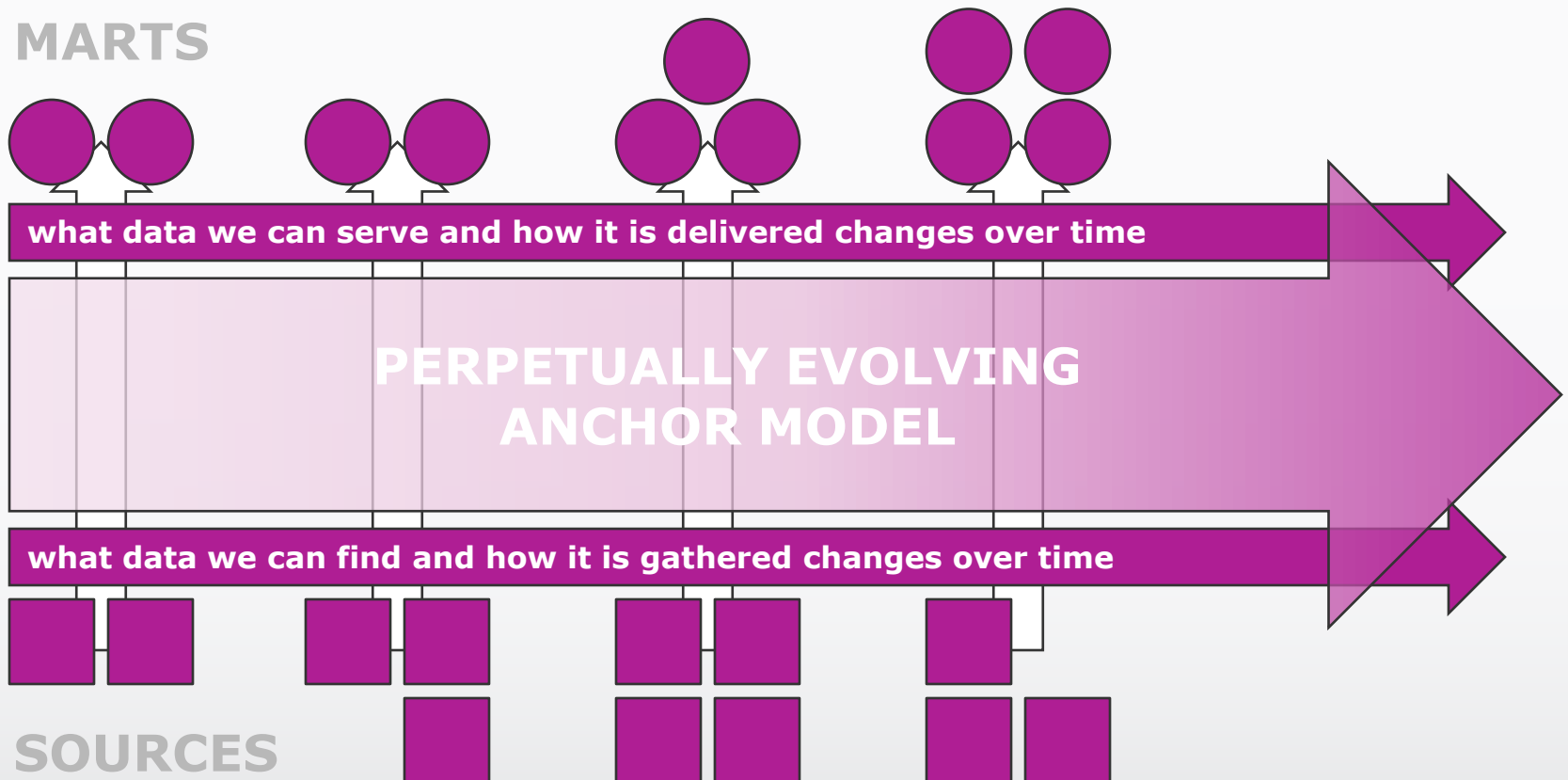
Five Essential Criteria

A future-proof data warehouse must at least fulfill:

- Value
- Maintainability
- Usability
- Performance
- Flexibility

Fail in one and there will be consequences

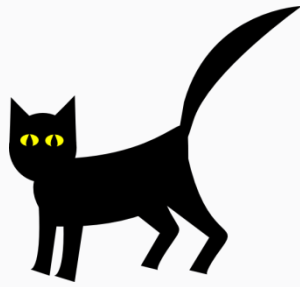
The Hybrid Architecture



Best practices for Anchor Models

- modeling constructs
- design patterns
- physical implementation
- naming convention
- collapsing views
- table elimination
- loading practice
- identity management
- flexibility
- historization
- null-handling
- orphaned relations
- project scoping
- storage
- temporal querying
- performance

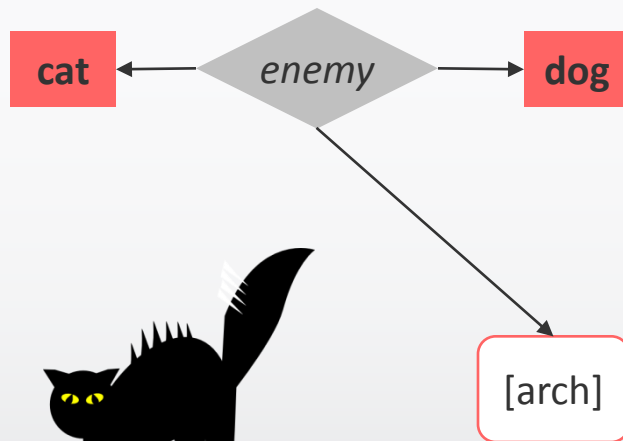
A large change outside the data warehouse should result in a small change within.



A table is in **sixth normal form** if and only if it satisfies no non-trivial join dependencies at all.



An **entity-relationship model** adopts the natural view that the real world consists of entities and relationships.



Anchor modeling

We use EER to draw our modeling diagrams (logical model) and we implement it in 6NF.

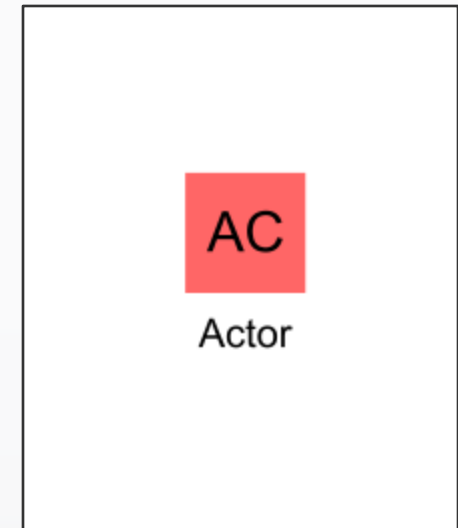
Modeling constructs

- Three table types (ER) are sufficient
 - Anchors
 - Attributes
 - Ties
- Using a fourth table type (EER) we can simplify models
 - Knots

Anchors

An anchor holds the identity of an entity in the data warehouse.

The identity is a technically generated surrogate key, rather than the natural key.



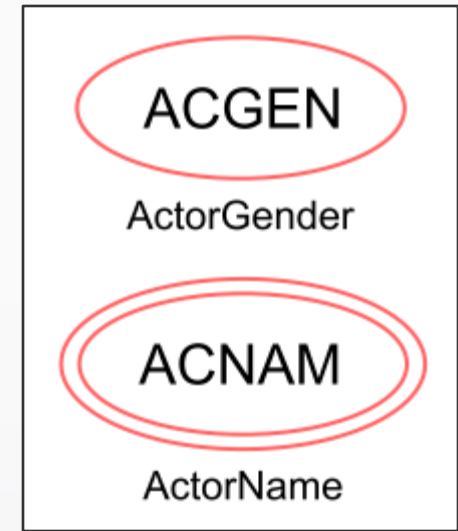
AC = <#789>

Attributes

Attributes always belong to an anchor. They hold actual attribute values that can be used to describe the entity whose identity is stored in the anchor.

The value stored in an attribute is strongly typed and can be of any data type.

A historized attribute has a double outline.



AC	=	<#789>
ACGEN	=	<#789, Female>
ACNAM	=	<#789, Marilyn, 19580101>

Ties

Relationships between entities are modeled as ties between anchors.

A tie contains the identities from the adjoining anchors. Usually two, but can be an n -tuple.

A historical tie has a double outline.

AC	=	<#789>
ACGEN	=	<#789, Female>
ACNAM	=	<#789, Marilyn, 19580101>
PEAC	=	<#256, #789>
ACPR	=	<#789, #345, 19710819>



Performance_Actor
(Cast)



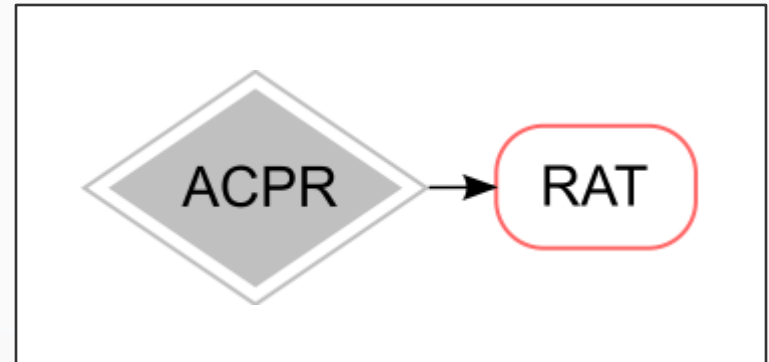
Actor_Program
(GotRating)

Knots

A combination of an anchor and an attribute may be assembled into a knot.

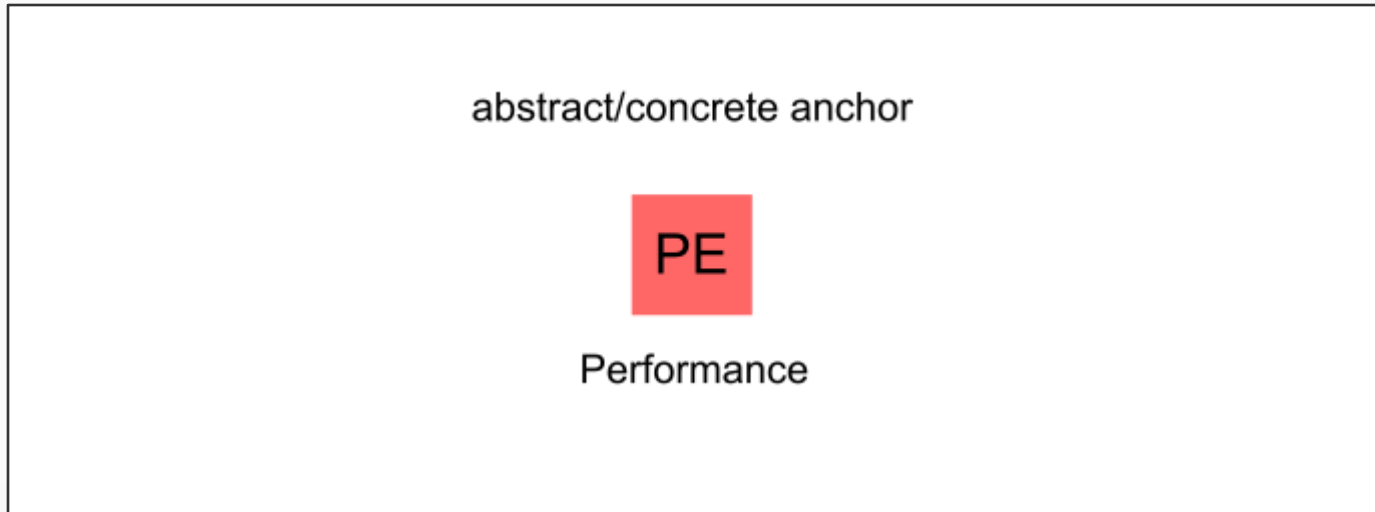
Knots are used for typing, representing states or domain values.

Since it contains both its identity and its value it may never change over time. Knots are non-historized by design.



RAT	= <#2, Excellent>
ACPR	= <#789, #345, #2, 19710819>

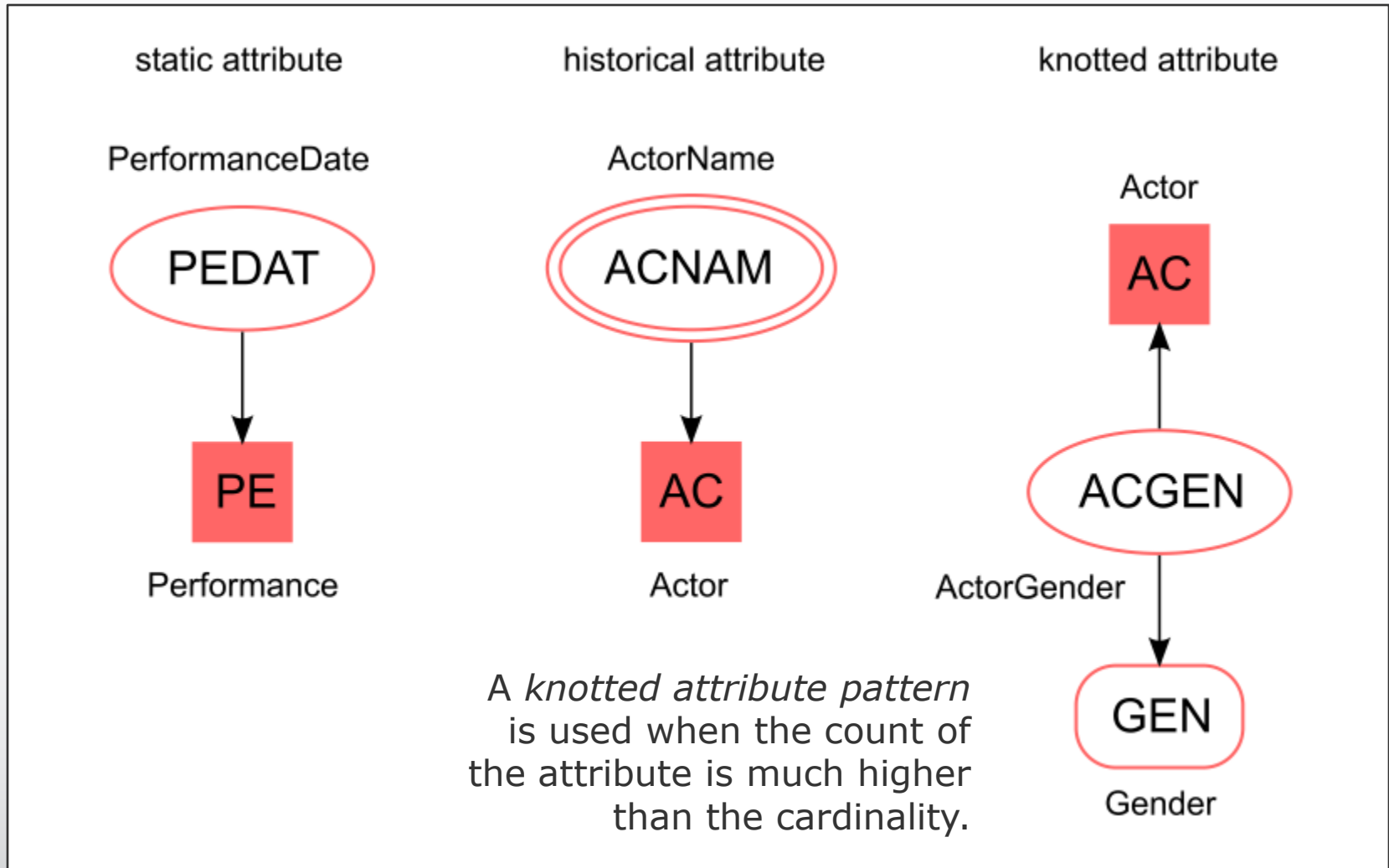
Anchor Design Patterns



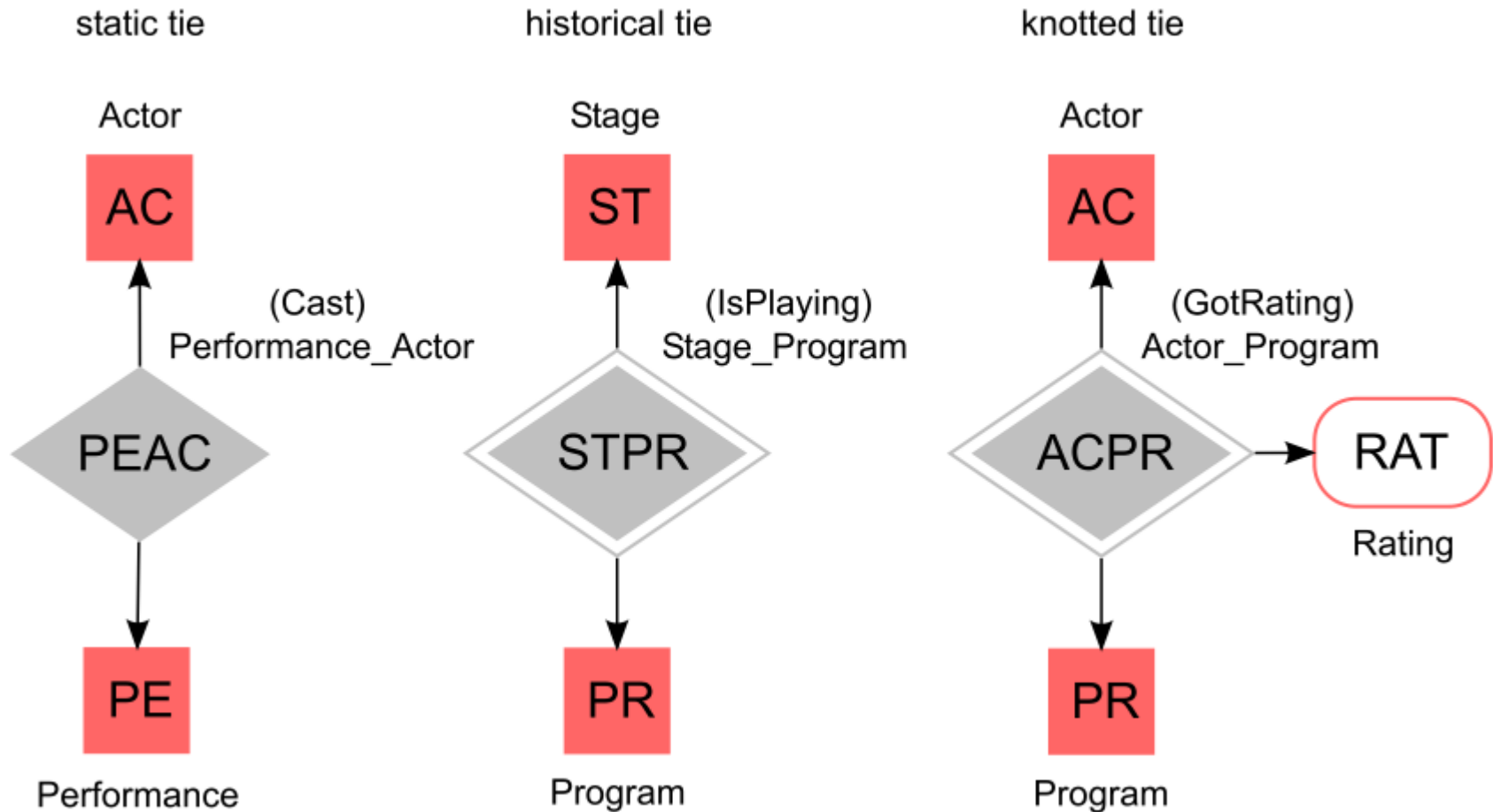
Business entities firmly anchored in the organization should become anchors. The anchors represent that with which the business works. We call these anchors *concrete*.

There are however other things that is handled by or affect your business, like events or transactions. They should also be modeled as anchors. We call these anchors *abstract*.

Attribute Design Patterns

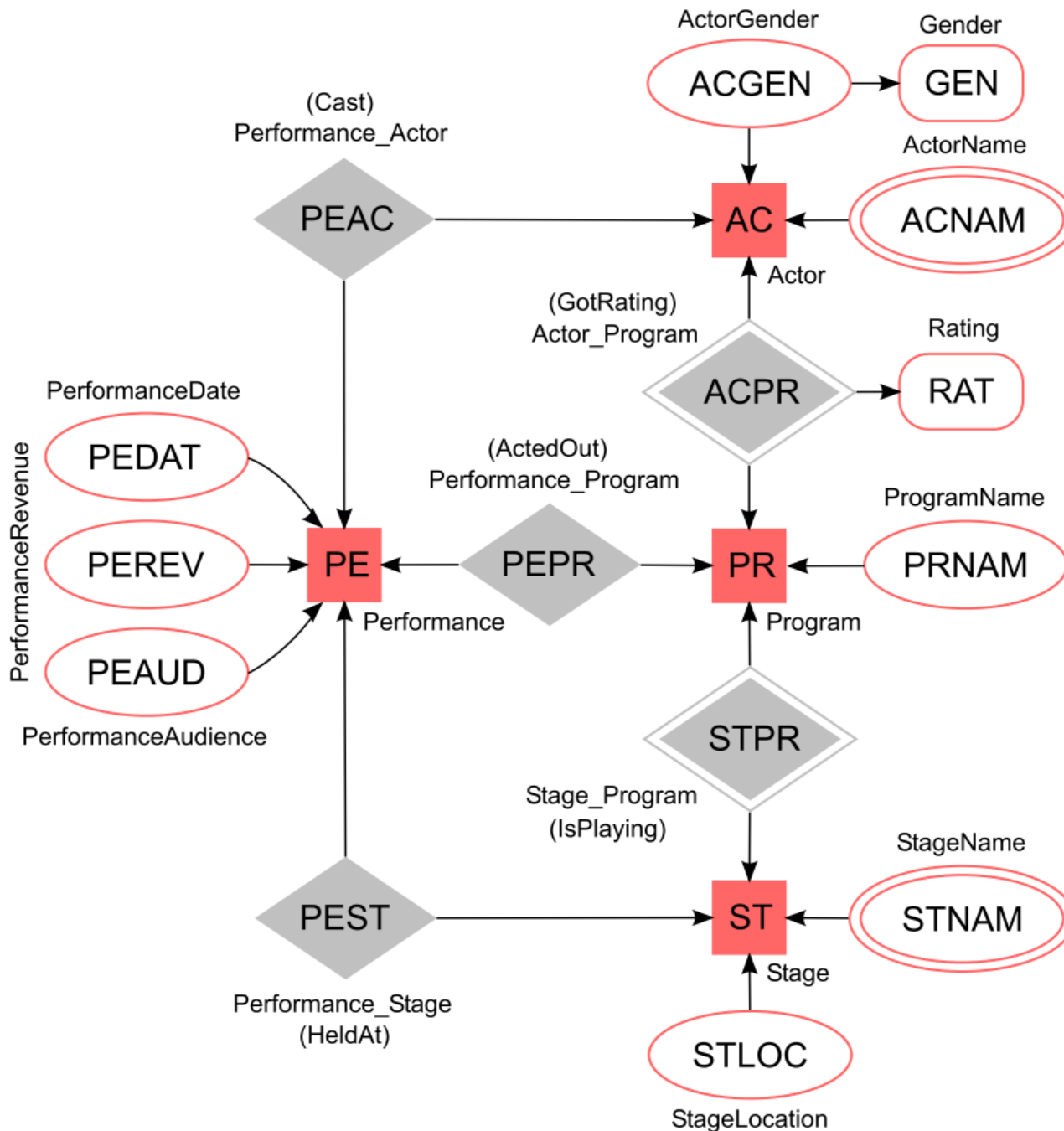


Tie Design Patterns



The *knotted tie pattern* is used for relations with a type or state. It may be used both statically or historically.

An example model



The scenario here is based on a business arranging stage performances. The business entities are the stages on which performances are held, programs defining the actual performance, and actors who carry out the performances.

Physical Implementation

Tables in an anchor model are implemented physically in a manner that in detail reflects the logical model.

There will be **one table for each entity** in your model.

Every table will further have a relation to a metadata structure, in which metadata about specific rows are stored.

- *When it was created*
- *Who created it*
- *How it was created*
- *From where it came*

The Physical Anchor

The anchor holding the identities of the different actors will contain rows with a unique surrogate key for its identity.

AC_ID (PK)
4711
4712

The primary key in this table is the identity column, on which a unique clustered index should be created in ascending order.

The Physical Attribute

Actors have names, so there is an attribute table associated with the actor which holds the name information.

AC_ID (PK,FK)	ACNAM_Name	ACNAM_FromDate (PK)
4711	Arfle Barfle Gloop	2006-08-20
4711	Foobar Barfoo	2009-01-01

The primary key in this table is the foreign identity column together with the historization date. A composite unique clustered index should be created on the foreign identity column in ascending order together with the historization date in descending order.

The Physical Tie

The current rating that an actor has got for performing a certain program is modeled as a tie.

AC_ID (PK,FK)	PR_ID (PK,FK)	RAT_ID (FK)	ACPR_FromDate (PK)
4711	555	4	2008-02-13
4711	555	5	2009-01-01

The primary key and unique clustered index should in this case be the composite of the two foreign keys in ascending order and the historization date in descending order.

The Physical Knot

The actual rating, in cleartext is modeled as a knot.

RAT_ID (PK)	RAT_Description
4	Good
5	Outstanding

The primary key and unique clustered index should in this case be the column containing the identity of the knot.

Naming convention for tables

Anchors have a **two** character mnemonic:
AC_Actor, ST_Stage

Attributes have a **five** character mnemonic:
ACNAM_ActorName, STLOC_StageLocation

Ties have a **four** (or *2n*) character mnemonic:
PEAC_Performance_Actor, STPR_Stage_Program

Knots have a **three** character mnemonic:
RAT_Rating, GEN_Gender

Naming convention for columns

Anchors have a **two** character mnemonic:

AC_ID

Attributes have a **five** character mnemonic:

ACNAM_Name, STLOC_Location

Ties have a **four** (or $2n$) character mnemonic:

STPR_FromDate

Knots have a **three** character mnemonic:

RAT_Description

Collapsing views

Collapses the anchor model to third normal form.

We need to pick a temporal view of the data:

Latest view

Point-in-time view

Difference view

Latest view

This view shows the latest version of everything that has been historized:

```
select ST_ID, STNAM_Name, STLOC_Location
from   1ST_Stage
-----
1      Shakespeare's Globe      Maiden Lane
2      Cockpit                  Drury Lane
```


Point-in-time view

This table valued function shows the latest version of everything prior to the given date:

```
select ST_ID, STNAM_Name, STLOC_Location
from pST_Stage('1608-01-01')
-----
1      The Globe Theatre      Maiden Lane
```

Difference view

This table valued function shows the all versions of everything falling in between the two given dates:

```
select ST_ID, STNAM_Name, STLOC_Location, STNAM_FromDate
from dST_Stage('1598-01-01', '1998-01-01')
-----
1      The Globe Theatre      Maiden Lane      1599-01-01
1      Shakespeare's Globe    Maiden Lane      1997-01-01
2      Cockpit                Drury Lane       1609-01-01
```

Table elimination

Even though all the attributes surrounding an anchor is joined into the view, those tables can many times be eliminated from the query execution.

Typical data warehouse queries do not retrieve all attribute values in a single query.

Most of the time few selected attributes are put against each other in a query.

Table elimination

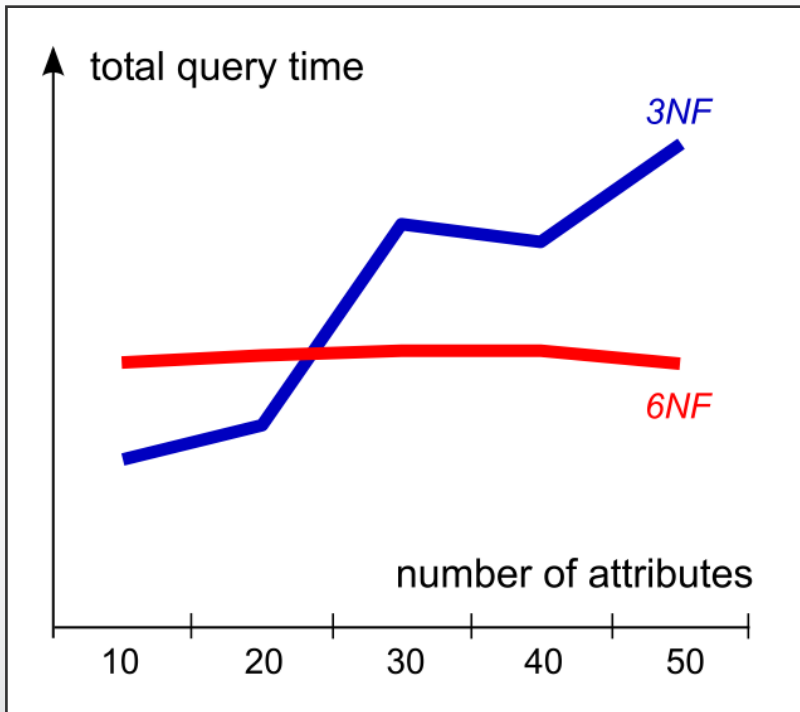
Even though all the attributes surrounding an anchor is joined into the view, those tables can many times be eliminated from the query execution.

Typical data warehouse queries do not retrieve all attribute values in a single query.

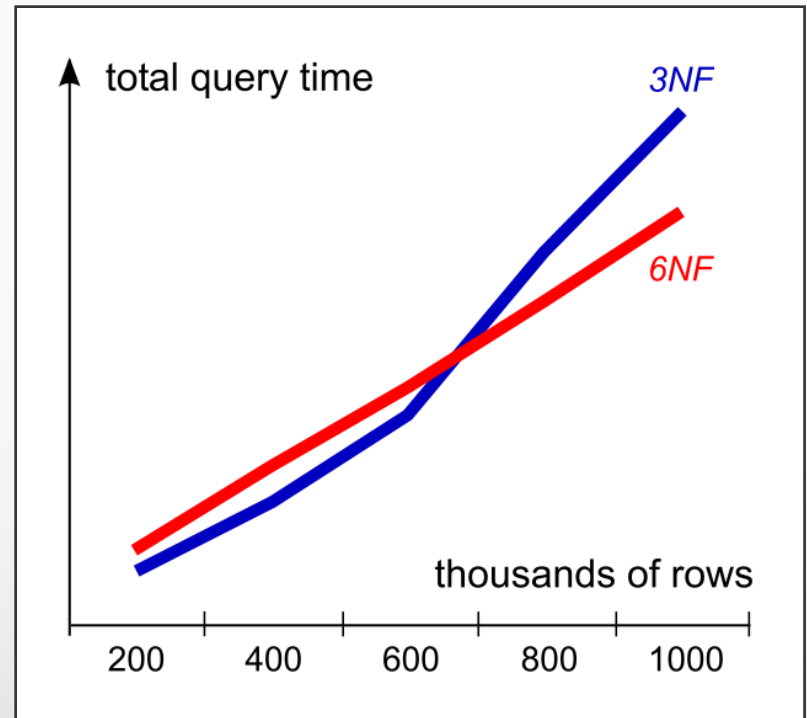
Most of the time few selected attributes are put against each other in a query.

Performance comparison

Increasing number of attributes



Increasing number of rows



Loading without updates

Only insert statements are allowed, which means that data is always added, never updated. Delete statements are allowed, but then only when applied to remove erroneous data.

Will affect your model:

Use state knots to model non-persistancy

Loading without updates

Only insert statements are allowed, which means that data is always added, never updated. Delete statements are allowed, but then only when applied to remove erroneous data.

Will affect your model:

Use state knots to model non-persistancy

Benefits:

No need for transactions – backing through scripting

Identity Management

To make the management of identities easier we create a helper view called the natural key view for every anchor.

For example `iPE_Performance`, in which you can find the natural key for a performance together with the technical surrogate key.

Acts as a translation table between natural keys and their corresponding surrogate identities.

Can be materialized if needed to increase performance.

Benefits: Flexibility

Since none of the existing model is affected by changes, we can guarantee that applications using it will remain unaffected.

Of course, once you incorporate your changes into the collapsing views, anything using the views will have to be looked over.

Rather than trying to encompass the entire business with all its entities when modeling, we can start with a first tightly scoped model and grow from there.

An anchor model is an evolving model by design.

Benefits: Historization

Anchor models are bi-temporal by design. Only using a FromDate, never a ToDate.

Every attribute you have in a business can be historized and different versions can easily be found using the collapsing views.

We also keep track of when, who, how and from where data was entered using metadata.

Compared with the third normal form, an anchor model will not duplicate data when historization is done.

Benefits: Performance

The immediate fear of breaking up data into many tables is that performance will suffer due to the extra joins that will have to be performed.

All in all, an anchor model is bound to be much less I/O intense than a corresponding third normal form model. This makes anchor models even more suitable for data warehouses, where I/O often is one of the first obstacles you will hit with respect to performance.

We can also “hot partition” individual columns, or compress them.

Benefits: Storage

Thanks to the high degree of normalization in an anchor model together with strong typing our data is stored very efficiently.

We have done comparisons with fourth normal form models with generalized types (weak typing).

The weakly typed model was twice as large.

The weakly typed model took almost three times longer to query because of more data to scan + type conversion being made.

Benefits: Null values

There are no null values in an anchor model, but there may be null values in the collapsing views.

An attribute always belong to an anchor, but the reverse is not true. An anchor does not have to have all its attributes present.

Missing values = missing rows in the attribute.

If only 10% of the identities in an anchor have a certain attribute, then you will find rows only for these in that table.

Benefits: Orphaned relations

Let's say that it is of the utmost importance for our business managers to be able to see the revenue from a performance immediately after it has been held.

Say a performance is uniquely identified by when and where it was held.

What if this is the only information we get, and that who took part in the performance and what was played always arrive a week later?

No problem! We can store what we know now and add the completing information later.

Benefits: Temporal querying

Really complex questions can be formulated as simple queries using the collapsing views:

```
select
  pAC.ACNAM_ActorName,
  max(pACPR.RAT_Rating)
from
  pAC_Actor ('1995-01-01') pAC
join
  pACPR_Actor_Program ('2005-01-01') pACPR
group by
  pAC.ACNAM_ActorName
```

What best ratings did our actors from 1995 have in 2005?

We decide how we want to view the history!

Benefits: Simplicity

Thanks to its few building blocks and the simple logic behind the design patterns, anchor modeling is easy to learn and hard to make modeling errors in.

Because of its structured nature, we can also automatically generate scripts that create tables and all the different collapsing views.

A compact XML defining the model is the only input.

Further reading

Please visit the site www.anchormodeling.com where we will be adding new material that we produce.

Anchor Modeling has an Open Source-like license:
Creative Commons Attribution-Share Alike 3.0 Unported

You can also contact us at:
info@anchormodeling.com