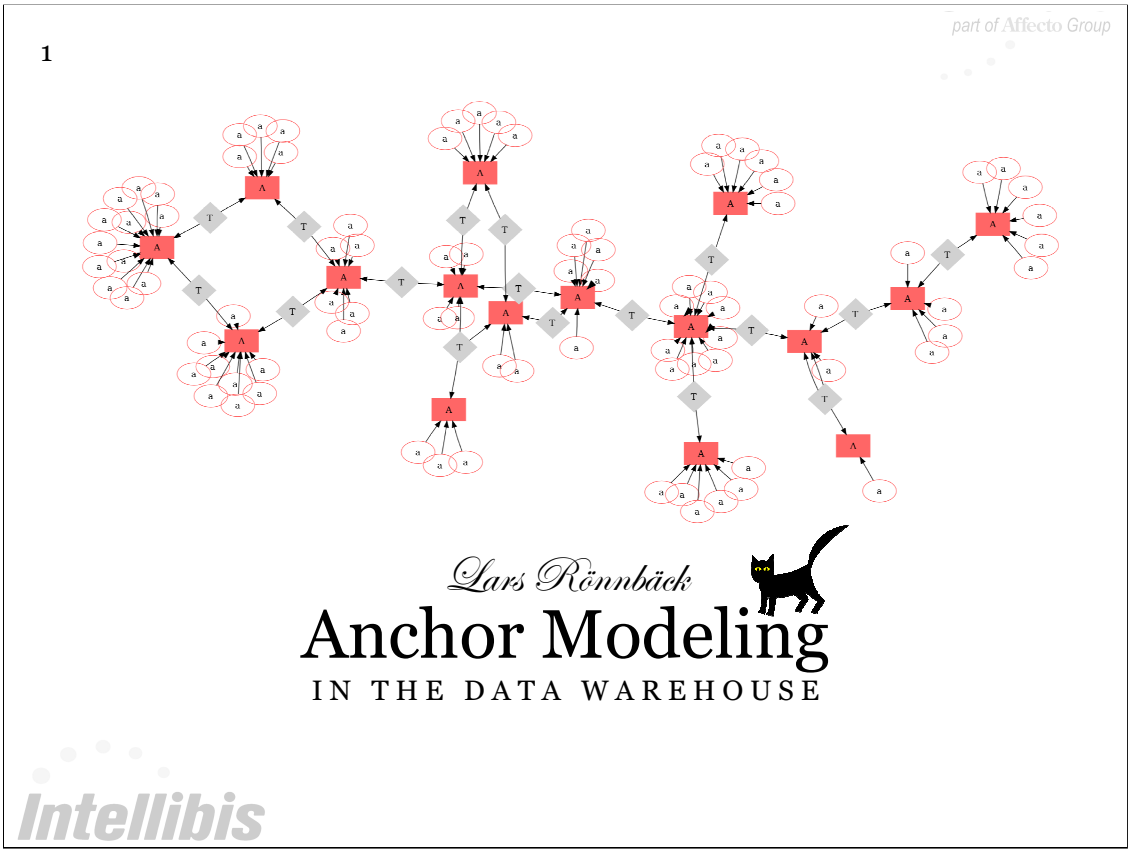
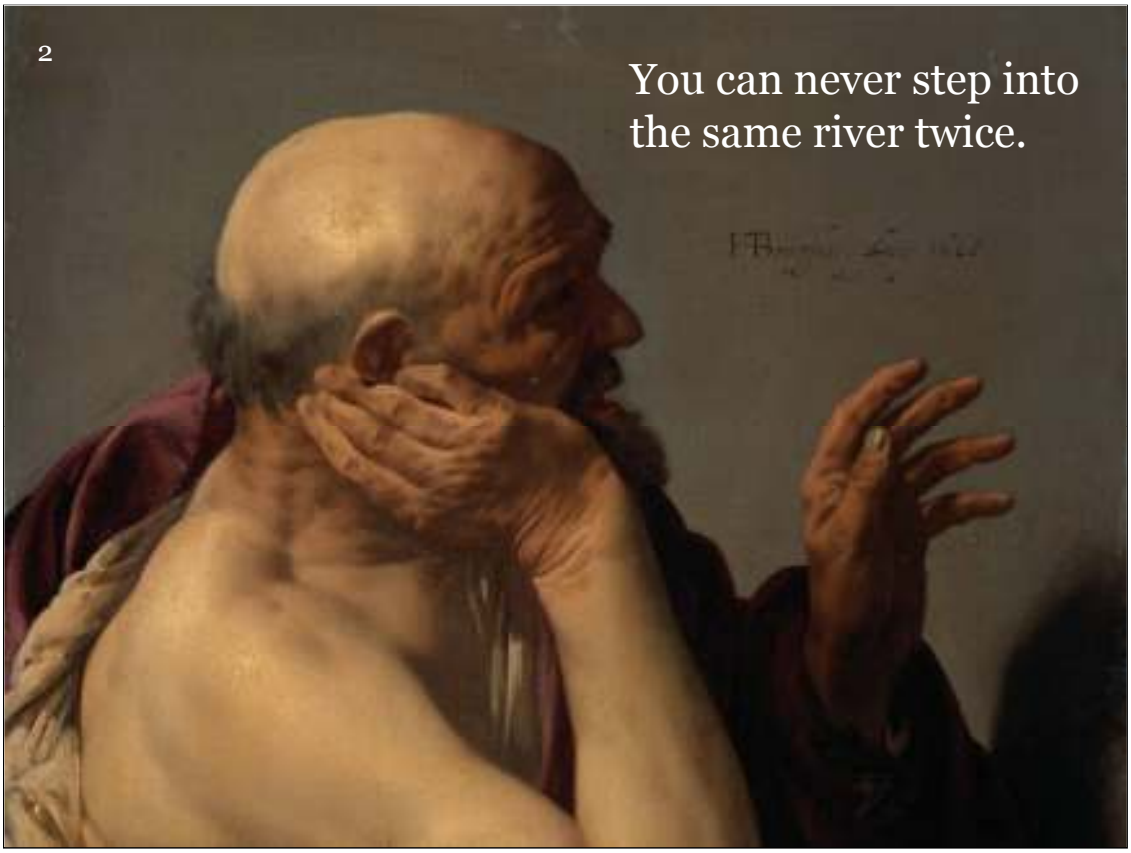


Did you know that some of the earliest prerequisites for data warehousing were set over 2500 years ago.

It is called an anchor model since the anchors tie down a number of attributes (see picture above).

All EER-diagrams have been made with Graphviz.

All cats are drawn by the author, Lars Rönnbäck.



The great greek philosopher Heraclitus said ”You can never step into the same river twice”.

What he meant by that is that everything is changing. The next time you step into the river other waters are flowing by.

Likewise the environment surrounding a data warehouse is in constant change and whenever you revisit them you have to adapt to these changes.

Image painted by Henrik ter Bruggen courtsey of Wikipedia Commons (public domain).

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

Value – most important, even a very poorly designed data warehouse can survive as long as it is providing good business value.

If you fail in providing value, the warehouse will be viewed as a money sink and may be cancelled altogether.

Maintainability – you should be able to answer the question: How is the warehouse feeling today? Could be healthy, could be ill! Detect trends, e g in loading times.

Usability – must be simple and accessible to the end users. Should not take a university degree in computer science to get the information you want.

Performance – demands may vary depending on the user. Analysts may be satisfied waiting 10 minutes for a query, while users looking at dynamical reports may require sub second response times.

Finally flexibility, which will be the main subject of todays presentation.

Flexibility in Anchor Models

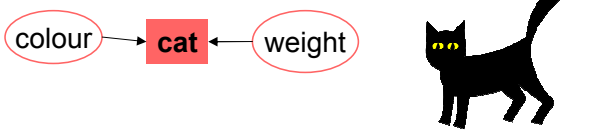
- Resilient to changes in the environment surrounding the data warehouse
- The model simplifies
  - historization
  - null-handling
  - orphans
  - separation of concerns
  - prototyping
- Achieves performance gains

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

Anchor modeling is nothing new. The ideas and theories have been around since the 70ies, but has only recently been adopted by us and used in practice.

Background – 6NF

- A table is in **sixth normal form** if and only if it satisfies no non-trivial join dependencies at all.
- C. J. Date, Darwen, and Lorentzos
  - Temporal Data and the Relational Model
- Esteban Zimányi
  - Temporal Universal Quantification



Note that sixth normal form is not the same as Domain/Key normal form, although stated so in some resources online.

Basically 6NF puts each non-key attribute in a separate relation. A 6NF table is a key plus at most one other column.

Full names: Christopher J. Date, Hugh Darwen, Nikos A. Lorentzos

In the 70's most academic people were investigating the definitions and algorithms for the normal forms of relations. Papers on higher normal forms were published rapidly, until someone wrote a paper claiming he had found the infinity:th normal form. Arguing that it did not make any sense to pursue higher normal forms, the academia settled on the fact that forms higher than 3rd or 4th would not have practical significance.

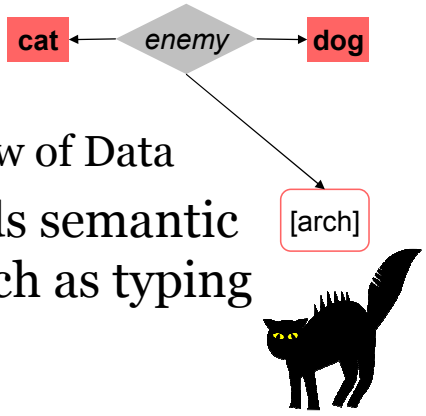
The cat has two attributes; colour and weight. It would be natural to assume that its weight will change over time, whereas the colour will not. Unless you decide to dye it for some reason. In the data warehouse we typically want to keep a history over such changes, hence the keyword 'Temporal' used in the titles.

6

part of Affecto Group

Background – EER

- The **entity-relationship model** adopts the natural view that the real world consists of entities and relationships.
- Peter P. Chen
  - Towards a Unified View of Data
- Enhanced entities adds semantic modeling concepts such as typing



Intellibis

Has a wider scope than just databases, is used throughout all of Information Technology.

Was actually developed alongside relational models in the 70's.

Based on set theory and relation theory it can be used as a framework from which the relational data model can be derived.

The relation *enemy* can be typed, indicating that among all dogs that a certain cat has as enemies, one is the [arch]-*enemy*.

A relation may also change over time. Let's say the cat and dog started out as enemies, but later became friends. Again, we want to be able to track the history of how the relation has evolved in the data warehouse.

Anchor modeling

- Almost 6NF + EER
  - a practical approach rather than purist
- Three table types (ER) are sufficient
  - Anchors
  - Attributes
  - Ties
- Using a fourth table type (EER) we can simplify models
  - Knots

All modeling problems can be solved using only three table types, but that will require more tables in an already table-excessive modeling technique.

This is why we turn to the enhanced-ER, which will help us reduce models by tieing together tables of the three types into a fourth type; the knot.

It is a kind of multi-purpose table table, although it comes with the price of some limitations.

Anchor modeling in practice

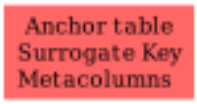
- The originator is Olle Regardt (Intellibis)
  - started using anchor models in 2002 based on experiences with informational models
- Formalization by Lars Rönnbäck
  - white papers late 2007/early 2008
- Large scale implementations in use
  - leading insurance company
  - largest bus logistics company

4TB data at the leading insurance company. Three data warehouses and one master data system.

Informational models foremost related to the insurance industry.

Anchors

- Contents of an anchor
  - The surrogate key of the entity
  - Metacolumns
    - Batch information
    - File information



Metacolumns should answer the questions: WHEN? WHERE? HOW?

The surrogate key is a technically generated identity based on the natural key.

What is the natural key for a cat? Perhaps its name + birthdate. This would be the base when generating surrogate keys such that every unique combination of a name and a birthdate would yield a new key.

Attributes

- Contents of an attribute
  - The foreign key of the belonging anchor
  - An attribute value
  - Historization columns
  - Metacolumns

If we have a cat named ’Kitty’ born in 2005 that is black, then the colour black is an attribute. We would keep it in its own table with a reference to the ’Kitty’-key.

Ties

- Contents of a tie
  - The foreign keys of the related anchors
    - which may be an *n*-tuple
  - Historization columns
  - Metacolumns

A tie may tie together an arbitrary number of anchors.

Some cats and dogs may have changed their relation several times. Friend, enemy, friend, enemy, friend...

Historization

- Zero Update Strategy [ZeUS]
  - Only insert/select for normal use
  - Deletes only to revert faulty data
- Consequences
  - Only `FromDate` and never `ToDate`
  - Anchors are eternal

Later FromDate replaces earlier version, so the latest FromDate is the latest version.

Eternal in the effect that everything else might change, but not the identity.

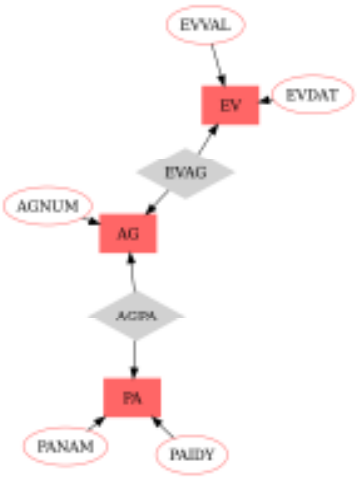
When the kitten was born it weighed 105 grams. We record this in the data warehouse. Later on (maybe much later, seeing how large it has become) we weigh the cat again and record the weight in the data warehouse. We do not update the existing data. Instead we insert a new row, with a later FromDate. The latest FromDate is the lastest information we have about the cat. It is the current version of the cat.

13

part of Affecto Group

A sample model

- Anchors
  - PA_Part
  - AG_Agreement
  - EV_Event
- Ties
  - AGPA_Agreement_Part
  - EVAG_Event_Agreement
- Attributes
  - PANAM_PartName
  - PAIDY_PartIdentity
  - AGNUM_AgreementNumber
  - EVVAL_EventValue
  - EVDAT_EventDate



Intellibis

We will now move away from the cats and dogs and into a customer case example from the insurance industry.

This is a LOGICAL MODEL, represented as an entity-relationship diagram. The symbols used for anchors, attributes and ties are the standard ones used when creating ER-diagrams.

As you can see, there's a hint of a naming convention if you look at the names of the different tables.

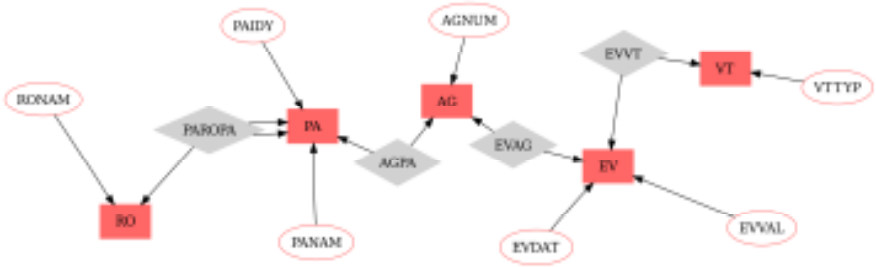
Naming conventions

- Anchors have a two letter prefix
  - `AG_Agreement`
- Attributes have a five letter prefix
  - `AGNUM_AgreementNumber`
  - first two letters taken from anchor
- Ties have a four (or 2*n*) letter prefix
  - `AGPA_Agreement_Part`
  - letters taken from adjoining anchors
- Avoids bad models
  - 'cause you'll get in trouble naming your objects

Note that ties have underscore between the anchor names.

Prefixes must be unique in the model.

An extended sample model



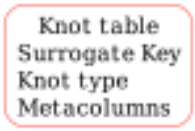
- PAROPA_Part_Role_Part
- RO_Role
- RONAM_RoleName
- EVVT_Event_ValueType
- VT_ValueType
- VTTYP_ValueTypeName

This is what the model would look like if we were ’purists’.

Every event value has a currency, represented by the value type. A ”business rule” guarantees that every value must have a type.

Knots

- Contents of a knot
  - The surrogate key for the knotted entity
  - An attribute value
    - representing the type of the knot
  - Metacolumns
- Uses a three letter prefix



The symbol for a knot table does not exist in standard ER notation. However, the squarish shape hints that it is something of a cross between an anchor and an attribute.

Note that a knot:

-is fundamentally a collapsed anchor

-may never change over time

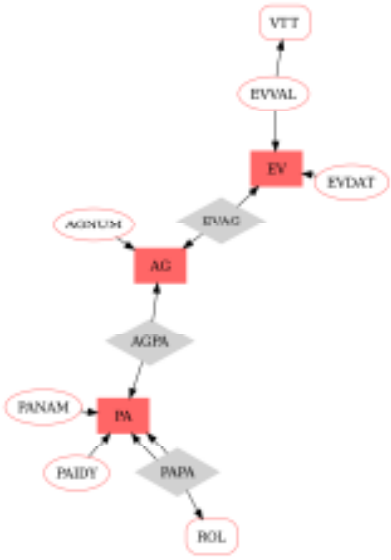
-is often used to ”type” a relation or attribute

-can also be used for domain values or state tables (i e if something is to be marked as deleted)

A simplified sample model

- Anchors, attributes and ties can be tied into knot tables
- Every `EVVAL` has a `VTT`

- Every `PAPA` has a `ROL`



Note that normalization is preserved.

Modeling practices

- Find the business entities [*anchors*]
- Generalize [*anchors, ties*]
- Balance and specialize [*attributes, knots*]
- Historize [*attributes*]
- Find relations [*ties, knots*]
- Historize [*ties*]

This is a general approach on where to start and what steps to take when you are creating an anchor model.

An example of a generalization is:

To have FinancialEvent as anchor with typing, rather than actual, budget, forecast values as their own anchors.

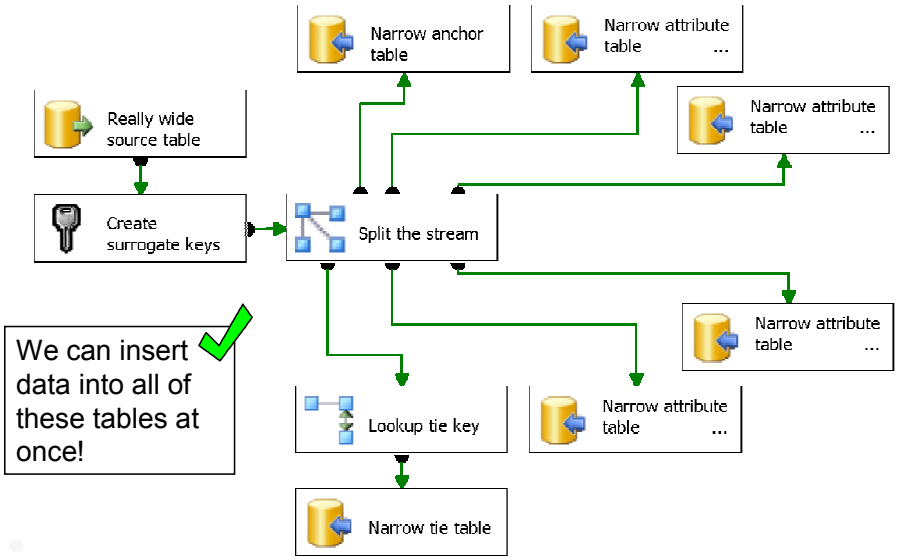
Avoid when modeling relations: fan traps, chasm traps.

Common Questions

- There is no support for multiple assignment statements in SQL. How can we insert data into all of these tables at once?
- Will not the SQL needed to query the implemented model be extremely complex?
- The performance must be horrible with all the joins needed!

Let us look at how we have answered these questions!

Stream based ETL-tools



The popular ETL tools available today all tend to be stream based rather than set based, like SQL.

We can, with only one scan of the source table, insert data into as many tables we want simultaneously. Loading performance is not an issue.

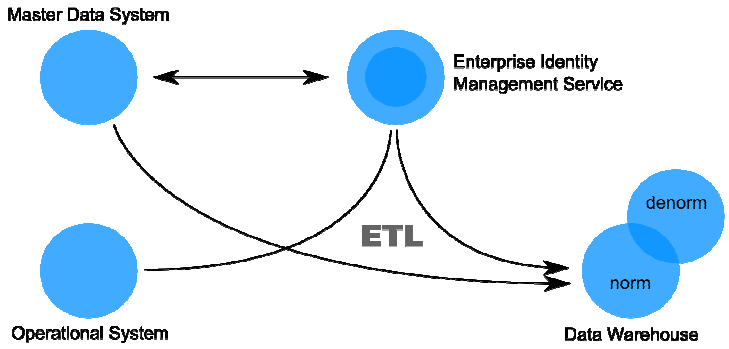
Image taken from SQL Server Integration Services.

The surrogate key component is custom made and exchanges information with an identity management service.

Wrapping the package in a transaction to ensure consistency

Identity Management

- Keys as globally unique identifiers
  - "Global" is more than local to the DW

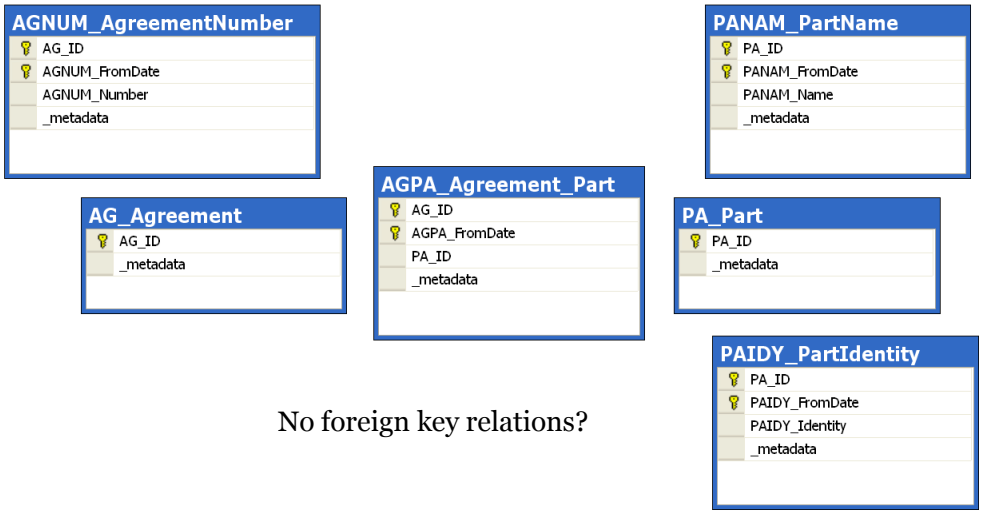


The EIMS is a service that keeps track of all entity identities in your organization - a master key storage.

Even if you have no such service, it is better to create locally unique keys to your data warehouse in the ETL process.

Surrogate keys are never propagated to the presentation layer (may be denormalized).

Physical implementation



No foreign key relations?

This is a part of the example implemeted as a physical model in SQL Server 2005.

Note that all tables reference the metadata table (for maintainability purposes).

Why are there no relations in the picture? Because foreign key constraints do not belong in the data warehouse (performance hogs). The ETL process should ensure referential integrity. Same goes for primary keys. We are further generating identities in the ETL process rather than letting the warehouse do this.

The anchor is the keeper of the cardinality.

We ensure later that no duplicates can be entered into any tables by having unique constraints (primary key).

Collapsing views – regular

```
create view vAG_Agreement as
select
        AG.AG_ID,
        AGNUM.AGNUM_Number,
        AGNUM.AGNUM_FromDate
from
        AG_Agreement AG
left join
        AGNUM_AgreementNumber AGNUM
on
        AGNUM.AG_ID = AG.AG_ID;
```


Note that collapsed views can/will include NULL values.

Collapsing views – latest

```
create view lAG_Agreement as
select
        AG.AG_ID,
        AGNUM.AGNUM_Number
from
        AG_Agreement AG
left join
        AGNUM_AgreementNumber AGNUM
on
        AGNUM.AG_ID = AG.AG_ID
and
        AGNUM.AGNUM_FromDate = (
                select
                        max(AGNUM_FromDate)
                from
                        AGNUM_AgreementNumber sub
                where
                        sub.AG_ID = AG.AG_ID)
```


Important that we use AG.AG_ID in the join condition for the subselect and not AGNUM.AG_ID, since the latter will result in a self-join.

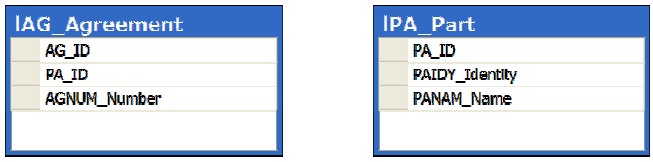
Collapsing views – knotted

- Incorporating the tie into the view

```
left join
        AGPA_Agreement_Part AGPA
on
        AGPA.AG_ID = AG.AG_ID
and
        AGPA.AGPA_FromDate = (
                select
                        max(AGPA_FromDate)
                from
                        AGPA_Agreement_Part sub
                where
                        sub.AG_ID = AG.AG_ID)
```


PA_ID from the EER can be added to the latest view if the circumstances allow, i e there is a one to many relation.

Implemented views



Looks like third normal form!

The SQL need not be any more complicated than when using other modeling techniques!

All we have to do is make sure we get good performance when querying the views.

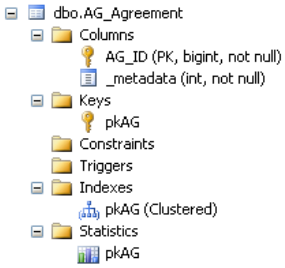
27

part of Affecto Group

Creating an anchor

```
create table AG_Agreement (
        AG_ID bigint not null,
        _metadata int not null,
        constraint pkAG primary key (
                AG_ID asc
        )
);
```

Will also result in a clustered index over the primary key.



Intellibis

In data warehousing scanning large portions of tables is common practice.

A clustered index is an ordering of the data on disk, implying that scanning can be done sequentially, without ”disk trashing”.

Bigint ID:s from a volume perspective. Total number of rows in all tables larger than an int can hold.

28

part of Affecto Group

Creating an attribute

```
create table AGNUM_AgreementNumber (
        AG_ID bigint not null,
        AGNUM_FromDate smalldatetime not null,
        AGNUM_Number int not null,
        _metadata int not null,
        constraint pkAGNUM primary key (
                AG_ID asc,
                AGNUM_FromDate desc
        )
);
```

The physical order on disk will be aligned so that the latest record always is the first row for every id.

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

Intellibis

We want to align the clustered indexes with eachother, so joining can be done as smoothly as possible.

The FromDate order is reversed, since we are likely to be interested in the latest versions more often than the earliest ones.

29

part of Affecto Group

Creating a tie

```
create table AGPA_Agreement_Part (
        AG_ID bigint not null,
        AGPA_FromDate smalldatetime not null,
        PA_ID bigint not null,
        _metadata int not null,
        constraint pkAGPA primary key (
                AG_ID asc,
                AGPA_FromDate desc
        )
);
```

Assumes one-to-many relation between agreements and parts.

Intellibis

For a many-to-many tie, or a tie that ties together several anchors the primary key would have to be extended with additional foreign keys to ensure uniqueness.

30

part of Affecto Group

A sample query

```
select
        lPA.PANAM_Name,
        count(distinct lAG.AGNUM_Number)
from
        lPA_Part lPA
join
        lAG_Agreement lAG
on
        lPA.PA_ID = lAG.AG_ID
group by
        lPA.PANAM_Name
```

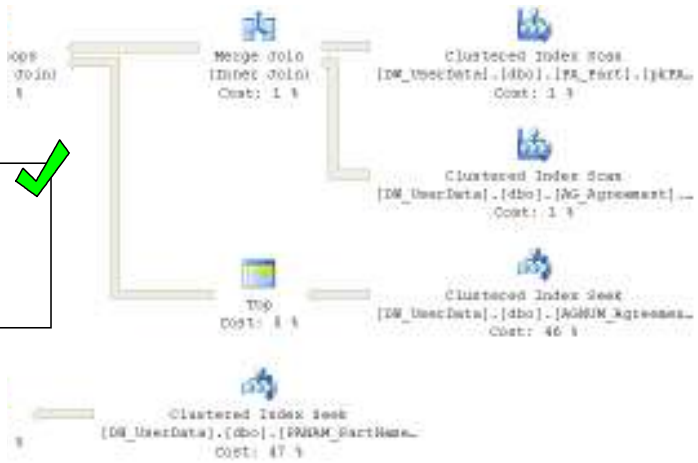
Intellibis

This is more true to the type of queries you would see in a data warehouse.

Will produce table scans.

Does not use all attributes.

Actual execution plan



The performance will be good and sometimes even better than with other models!

Note that it will not bring in the table PAIDY_PartIdentity, even though it is joined into the view.

Optimal execution plan

- Would involve only three tables
  - AGNUM_AgreementNumber
  - PANAM_PartName
  - AGPA_Agreement_Part
- Since we are starting with the anchors in the collapsing views and left joining we have prohibited the optimizer from figuring out the best plan.

The query optimizer cannot figure this out since we are implicitly disregarding the cardinality of the anchors.

33

part of Affecto Group

Temporal functions

```
create function fAG_Agreement (@date smalldatetime)
returns table return select
        AG.AG_ID,
        (select top 1
                PA_ID
        from
                AGPA_Agreement_Part AGPA
        where
                AGPA.AG_ID = AG.AG_ID
        and
                AGPA_FromDate <= @date
        order by
                AGPA_FromDate desc) as PA_ID,
from
        AG_Agreement AG
```

Intellibis

There is no performance difference between using top or max – they both result in the same execution plan.

Note that this might differ depending on your database vendor, so please examine your execution plans to find the optimal way to do temporal queries.

34

part of Affecto Group

An advanced sample query

```
select
        fPA.PANAM_Name,
        count(distinct fAG.AGNUM_Number)
from
        fPA_Part('2007-01-01') fPA
join
        fAG_Agreement('2006-01-01') fAG
on
        fPA.PA_ID = fAG.AG_ID
group by
        fPA.PANAM_Name
```

Intellibis

What agreements did todays parts have in 2006?

This has the same query plan as seen before, however, now the top sorting will have to look further for each key, since the specified date does not necessary have to be larger than the latest one.

The only challenge?

- To ensure business rules in the database we would need complicated constraint logic that decreases the performance
- Leaving it to the ETL means that rules are found outside of the database
- Other modeling techniques may face the same challenges

Good documentation needed to complement the model. As always.

Anchor Modeling Benefits

- Historization by design
  - slowly changing dimensions
- Eliminates NULL
  - referential integrity
- Handles orphans
  - early arriving facts
- Supports separation of concerns

We will look closer at each one of these.

Historization by design

- Relations as well as attributes can be historized
- All types of historization is handled
  - dependent on the querying method
- Non-historized data can easily be transformed to historized by adding historization columns

Bring back the cat again and its color. Say we actually do decide to dye it, then we can simply add a FromDate column to the attribute, and voila the current cat is violet.

Eliminates NULL

- Cardinality between attributes and the attributed anchor can differ (0-1)
- An entity need only have an identity to be allowed into the data warehouse
- Constraints are not used in the data warehouse due to performance reasons
- Integrity and business rules must instead be guaranteed by proper ETL

Integrity example: Attributes may never be dangling, in the sense that they point to an anchor that does not exist.

Business rule example: All agreements must have an agreement number, i e 1-1 mappings between anchors and attributes.

39

part of Affecto Group

Handles orphans

- The EER model must not be referentially complete
- Either we know enough from a record to create the referred anchor
- Or if we do not, we can leave out the connection by not adding any rows in the relation-table until we have the information

Intellibis

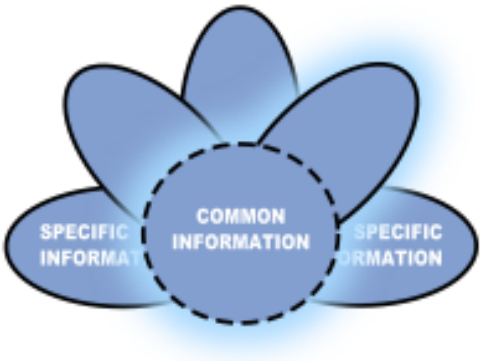
If the relation is historized, we can further add a relation as we think is likely and later change it if we made an erronous assumption.

40

part of Affecto Group

Separation of concerns

- Prototyping
- Project scoping
- Access control
- The federative data warehouse



Grow into your enterprise data warehouse at your own pace.

Intellibis

Five divisions in a corporation. Every petal + the common information can be viewed as a data warehouse of its own.

Each petal can even contain its own attributes to common anchors.

In a mixed environment with legacy warehouses one can start with a small common base and then grow into an enterprise data warehouse.

Quotes from users

”New attributes can be added without affecting the applications already using the warehouse whatsoever.”

DW expert, Intellibis

”The transition was painful, but once the first central anchor had been modeled the rest was easy.”

DW modeler, Länsförsäkringar

”Reuseability is high, the speedup when implementing new or changed data into the model is magnitues faster than with others we’ve used.”

DW maintenance manager, Länsförsäkringar

”I am relieved to have found a model where nulls won’t have to be interpreted.”

DW Performance Manager, Itello

The final NULL in the coffin. (quote from: Darwen, later used by Fabian Pascal)

Lars Rönnbäck
lars.ronnback@intellibis.se

Intellibis Stockholm
Tel +46 8 545 100 90
Kungsgatan 56
111 22 Stockholm
info@intellibis.se

Intellibis Malmö
Tel +46 40 611 78 70
Gustav Adolfs Torg 41
211 39 Malmö
infosyd@intellibis.se

Intellibis Göteborg
Tel +46 31 335 82 20
Fabriksgatan 7
412 50 Göteborg
infovast@intellibis.se

Intellibis Karlstad
Tel +46 8 545 100 90
Expositionshuset
652 26 Karlstad
infokarlstad@intellibis.se