
InfoStar

A Meta-tool with an Orientation to ETL-Warehouses

infolytics ag

The Initial Position, the Task & the Requirements

InfoStar – the Conception

InfoStar – the Technology

*Infolytics AG, Köln
www.infolytics.com*

The Initial Position, the Task & the Requirements

infolytics ag

The Initial Position

Requirements for analysis of corporate business data come from many sides – marketing, selling, accounting, revision and legal regulations – and increase for many different reasons.

The need to create reports, statistics and summarizations of different kinds is a demand that challenges almost any enterprise.

The answer to this need is the installation of a Data Warehouse or the creation of a data supply layer. But today, most enterprises – or the individual divisions of an enterprise – tailor themselves their own suit. Thus a multiplicity of separately developed applications exist, which originate partially from the same data pool. This often leads to data redundancy.

In addition, most larger enterprises have a heterogeneous application landscape, due to “old” applications on Mainframes and “new” in Client-Server technology with different application packages. Often the task of integration is even more complicated due to firm purchases or fusions that have to be mastered...

The Task

The conception of enterprise-spanning databases, an idea of the early times of relational databases, did not carry out itself. For good reasons the data stock for operational and dispositive tasks are held separately today, whereby the latter is derived to a large extent from the operational data.

There is a trend to introduce a standardized procedure for the entrance layer of a Data Warehouse or a data supply layer. In this connection the term “mirror layer” is often used. The data of the operational system is reflected in the entrance layer, which is often called Staging Area.

The metaphor *mirror* says that the goal is a copy of the operational system data in the mirror layer. However, that is not the whole truth: There is data, which is used only for the application control within the operational system, and other data, which is irrelevant for evaluations. So there is a need for an *individual mirror*, which only reflects the data that is relevant for analysis.

The volume of data grows steadily. The extraction of these large amounts of data and their changes are a burden for operational systems. Their import into a Data Warehouse is subject to usually narrow temporal requirements.

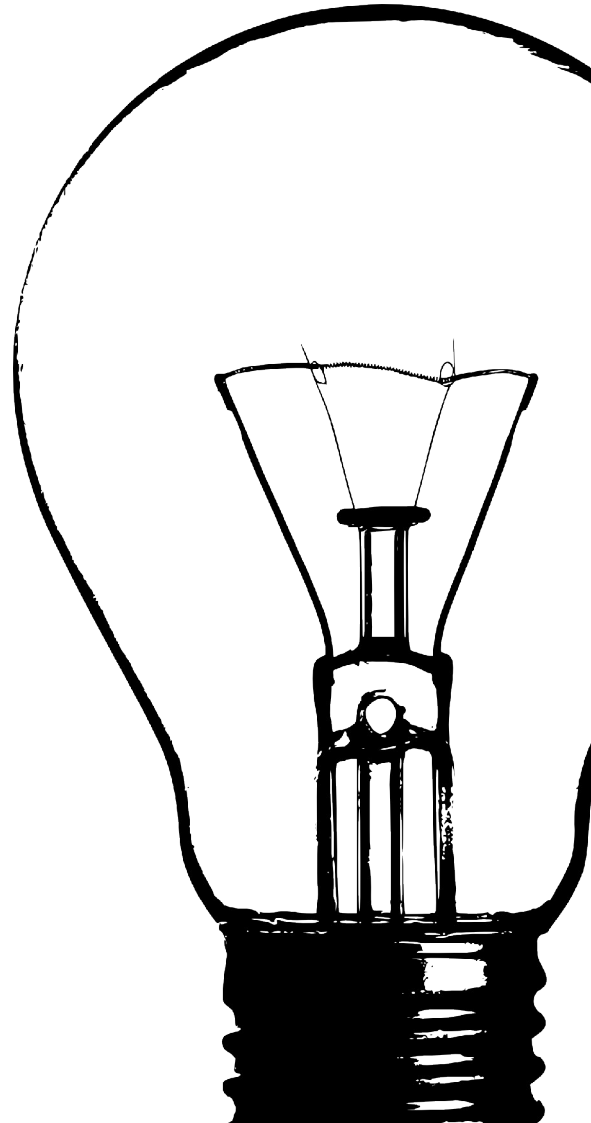
The Requirements

From the initial position and the task given we can summarize the requirements for

- *the Warehouse concept:*
The concept has to be of general nature. It must be applicable to all data delivering systems.
- *the imbedding into a heterogeneous environment:*
The procedure must adapt to the conditions met, and not reverse.
- *the data export:*
The meta description of any interface should be conform to a standard form. The data export should not become a burden for the operational system.
- *the data quality:*
A constant level of data quality must be ensured. Export deficits have to be balanced.
- *the runtime behavior:*
The optimal runtime behavior that is possible in the environment must be achieved.
- *the realization concept:*
The solutions developed should be transferrable to other applications -- without complexity.
- *the realization time:*
From the conception to the product introduction there should lie weeks -- and not many months or even years.
- *the realization expenditure:*
The costs must be obligatorily assessable and accordingly low. The same applies to the follow-up costs.
- *the transparency of the procedure:*
The procedure must be transparent. It should integrate into daily work without great effort.
- *the introduction into test operation and production:*
The introduction into different system or running conditions must be supported.
- *future security:*
Only components may be used that are freely available now and in the future. Maintenance and further development must be ensured.

InfoStar – the Conception

infolytics ag



A New Solution

It seems hardly possible to fulfill these demands. InfoStar follows two ways to provide a solution:

On the one hand a *general procedure* is used, with which a very large Data Warehouse was successfully established. The techniques and experiences from this project were generalized and flowed into the development of InfoStar.

On the other hand you need a tool that allows you to create *solution steps* with the help of definable rules. These steps can appear in a variety of forms and source codes – like SQL scripts, shell scripts, job workflows, programs, or commands for other tools. The generation rules are so general that the work for the transfer to other delivery systems is reduced to a few definitions.

InfoStar offers a broad spectrum of use
InfoStar can be used everywhere, where formal, rule-based texts – even complex – are used for the processing of data, for example 3GL-languages, SQL and scripts. The use as ETL tool for Data Warehouses is thus only one of many possible applications.

Here are some examples of further possible fields of application for InfoStar:

InfoStar can support *data migrations* from DB2, so for example from DB2 to Oracle, SAPDB, MySQL, Sybase and vice versa.

For newly planned interfaces or database systems, InfoStar can produce *artificial data for module and performance tests*.

For *automatic correction*, data can be output directly from the Staging Area in the batch format of an operational system.

Charakter of InfoStar

InfoStar is a meta application which enables the development of tools – this corresponds to the requirement formulated above: to make applications possible that are not “off the peg”, but are custom-made.

In other words: InfoStar doesn't force the user into a corset – there are no obligations or fixed rules. The user does not receive a rigid, but he gets a flexible tool, that is adaptable to the specific business needs.

Warehouse concept

The concept comprises a general procedure for establishing and running a Staging Area or a mirror layer of a Warehouse. The goal is to put historized data in continuous quality at the user's disposal. The user can freely decide upon the type of data historicization.

In the first step, an agreement between the operational and the dispositive system has to be made. This agreement regulates responsibilities, data extent, quality, quantities, times, tests and many other aspects. There is a questionnaire with guidelines, in order to lay down these subjects in a structured manner.

In the second step accurate interface definitions are stipulated. Thereby it is possible (e.g. with SAP R/3) to call on the meta data of the delivery system.

It is realistic to expect quality deficits in the supplied data. Test mechanisms will automatically correct some of these errors, others will be documented at least.

The procedure is designed in such a way that it can process both *full* and *delta* data supplies. Temporally consecutive full supplies have the effect of delta supplies. Depending upon specification the absence of records in a delivery can also be understood as a deletion (i.e. the record will be historized).

For a definable period all new, changed and deleted records are stored, so that analyses can be made for the current date or for any point in time.

After defining the interface declarations the standard sequence can be generated and implemented.

If changes are made to the standard technique, then these can be implemented into all delivery systems (and their interfaces) in very short time.

Integration into a heterogeneous environment

No tool can control all components which can be found in large installations today: different operating systems, application packages, databases, development tools, script and job control languages.

Therefore the basic approach for InfoStar is: generate programs and scripts in languages that are already in use at the enterprise. The advantage: the languages are well known on-site.

An example: in an environment there was a demand for examination of imported data. However, for that purpose there was only one meta language, from which COBOL code was generated. On the basis of the interface definitions instructions in this language were produced. Now a test program can be generated for each interface.

In this way environment-specific solutions can be developed.

This results in the following advantages:

- A well-known language is spoken.
- You get directly executable and not interpreted code.
- No further tools have to be used.
- The definition is made on a meta level, so that the solution applies both to all current and any future interfaces.
- For graphic presentation, all data definitions as well as their foreign key relations can be imported easily into an Entity Relationship Modeling Tool.

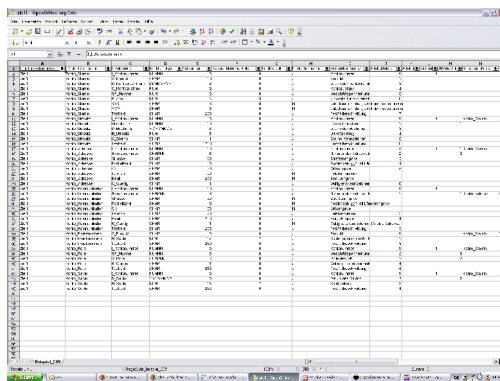
The image shows a screenshot of a spreadsheet application, likely Microsoft Excel, displaying a large table of data. The table has multiple columns and rows, with some cells containing text and others containing numbers. The data appears to be organized in a structured format, consistent with the caption's description of interface definitions in CSV format. The spreadsheet is titled 'InfoStar' and has a standard menu bar and toolbar at the top.

Figure 1: interface definition in csv format

Data export

Usually the employees of the delivery system are responsible for the data export. They know both the environment and the application and therefore they can write efficient extraction programs.

In many cases they also belong to the team which uses the dispositive data. Once the same extraction algorithms can be used for different interfaces, it is possible to generate the corresponding applications.

For R/3 these functions already exist in Java, PHP and Perl, so that in many cases there is no need to write ABAP programs and generate extractors.

But as soon as large amounts of data or extreme load of the operational system make scalability a necessity, the generative approach is put into work, in order to generate parallel working extractors and synchronize them during job flow.

Data quality

Delivered data often has deficits which cannot be repaired by the extractors. For example, there are incorrect data types, double records, missing field values, undefined key values and records not fitting into the hierarchy, to name only the most frequent errors.

These errors must be detected and, if possible, corrected and logged. The principle must be that no data may be lost.

For each type of error there is a separate process step. The error can be discovered in the test phase.

Example: a date field contains February 30 as value – in finance corporations this can

be found frequently. On the one hand, this can be an error in the source system and must be corrected there. On the other hand, it can be an artificial date for calculation of interest. Then the data type in the target system (the Staging Area) has to be changed: from DATE to CHAR (8).

Only a few errors can be corrected during runtime.

Example: when there are multiple records with identical contents all can be deleted up to one. But for records with identical primary key, but different other contents no automatism fits. In this case clarification must be provided in the delivery system.

For these and for other cases verification code is generated: upstream test programs or specific SQL instructions for data processing in the Data Warehouse.

Runtime behavior

Runtime behavior is crucial for the successful operation of a Data Warehouse.

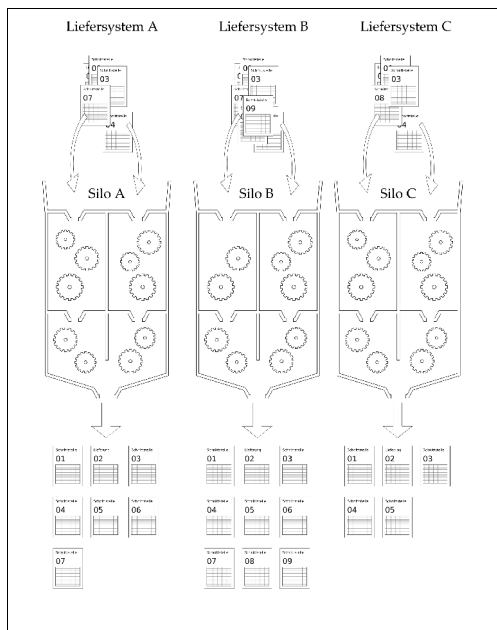


Figure 2: Each delivery system has its own silo for data processing. It is possible to run an unlimited number of silos

In InfoStar's concept a delivery system and its interfaces correspond to one or more "silos" in the Warehouse. The silos are independent from each other, so they can be operated parallelly.

Parallel processing of data is also possible within a silo. However borders are set to that: if there are dependencies between interfaces, then a defined sequence has to be met. The job flow will take this into account.

In the concept and in the realization of the standard technique only mass SQL instructions are used. Instructions for individual record are taboo. In connection with updated Database-Optimizer-statistics an optimized runtime behavior is guaranteed.

Each database system possesses one or several tools for loading data. In each case the optimal load procedure will be used.

Since the optimization algorithms of the databases are different, the processing time of an SQL instruction differs depending upon the database. In order to obtain the shortest processing time, InfoStar uses database-specific SQL instructions. This ensures the fastest way to the desired result.

Training

The training course expenditure for InfoStar is very small: The basic principles and the standard techniques only require an one-day training.

For training into more complex projects (with platform adjustments and extensions) the training course expenditure amounts to about five days.

Realization concept, -expenditure and -time

If it is possible to use the standard technique, the expenditure for the concept is pares down to an one-day training.

Even if you only want to use a part of the standard procedure, the training course expenditure of one day will be worthwhile, in order to profit from the experiences of others.

A reduction of the function range can be accomplished within a few days.

Extensions and platform adaptations require a greater expenditure. In a typical concept phase, text samples (loader and SQL instructions, program calls, instructions for job control, shell scripts) for a small interface will be written and tested with a small volume of data (module test).

In order to be able to generate the defined texts on the basis of the interface

definitions, generation rules have to be written. InfoStar carries a stock of rules, which can be used for standard tasks. In many cases it is possible to change existing rules so that they adapt to the defined task. The rules – for some tasks a combination of several rules can be necessary – are combined into a template, which serves as the basis for a generation module.

This module must be supplied with interface data. There is an extensive data collection for this purpose, which is sufficient for many tasks. In exceptional cases supplying routines must be written (this was the case in the generation of COBOL code mentioned above). The expenditure for this is very small.

A training course that covers all of these details should require five days.

According to a rule of thumb coordination and design including the text samples will take 60% of the total expenditure. Realization and introduction will need 20% each.

A simple application with adjustments can be realized within two to three months. A complex application with several delivery systems that has to comply to specific conditions will need six months – from the first draft to the introduction into the production environment.

Transparency

Transparency in the sense of seeing-through means that *three areas* need consideration.

The *development environment* is completely open. The system is delivered in source code. In addition there is a user manual,

in which each aspect of InfoStar is documented. InfoStar is commented up to the last routine. The definition data is stored in an open repository. The model is present in DDL format and is described in detailed reports of the modelling tool.

The *runtime environment* consists of scripts and is available in the languages used on-site. It is completely commented. The data model of the application can be imported into any modelling tool.

Since definitions and generation results are present in source code, the total procedure is saliently suitable for a source code management system.

The job flow for *operation* is described in a generated operating manual. During operation the execution of each script and each instruction will be logged. The log files are stored in the repository. Since the format is familiar, it is possible to create reports.

Introduction into test operation and production

InfoStar separates development, testing and operation. For the latter two there is a distribution and installation procedure, which exports only the required programs and scripts. The user is free to use his own versioning tools and procedures.

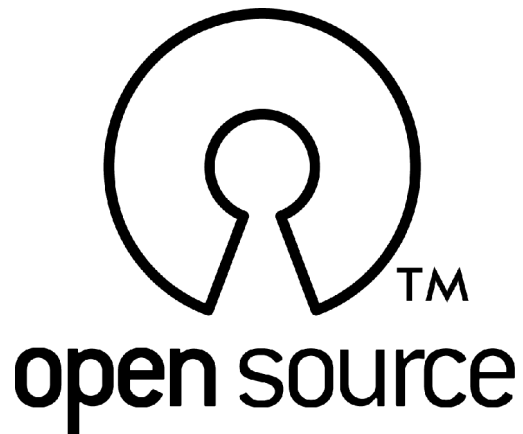
For both test and operation job sequences can be generated. Generated job sequences can be imported into the job control system if a batch import facility is available.

Guaranteed Future

Only open source products are used for the development of InfoStar. These products are widely used: Perl, XML, standard SQL – there is no need to think about an uncertain future. By today InfoStar did not run into any error while utilizing these products.

InfoStar is delivered with documented source code.

InfoStar generates the actual application, which is designed and optimized for the existing hardware and software platform. The generative approach ensures that each application can be adapted to changing conditions.



Generative Approach versus Standard Tool – a Comparison

<i>Key Point</i>	<i>Generative approach</i>	<i>Standard tool approach</i>
<i>Transparency</i>	completely transparent	limited
<i>Price</i>	reasonable	high
<i>Functionality</i>	custom-made for the requirements of the user	high, possibly only partially used
<i>Transferability</i>	rule-based, transferable solutions	limited
<i>Reproducibility</i>	ensured	repeatable tasks hardly definable
<i>Adaptability</i>	complete adaptation and optimization to existing systems is possible	badly adaptable optimized SQL for the different database systems is not possible
<i>Elimination of errors</i>	generative approach permits correction of errors on meta level (command and/or script template). Errors are thus corrected for all interfaces.	interface-specific elimination of errors (takes place only in one single place)
<i>Area of application</i>	broad spectrum of use: all applications which can be defined with formal, rule-based texts	covers only few areas
<i>Job control</i>	The job control is manageable from the outside	The job control is totally enclosed within the database system
<i>Training investment</i>	Three-part model: - Definition - Rule design (Rule and supplying method) - Change of the meta tool	multi-level, Change of the meta tool is not possible
<i>Expandability</i>	Own applications can be integrated	

Benefits

- *Low project investment*
The investment for a production-ready project is very small (about three to six man-months).
- *Minimal training required*
The investment for training amounts to a maximum of five days.
- *Moderate resource requirements: "Your existing resources aren't outdated"*
InfoStar uses the existing resources of your enterprise – both employees and existing tools. A smooth integration into the existing environment is ensured.
- *No dependencies*
The entire procedure is open and transparent. Thus, no dependencies exist.
- *Once custom tailored, and thereafter off-the-peg-solutions*
The application is adapted once to the requirements of the job, i.e. it is "custom-made".

"The rest is off-the-peg"

Then a simple adaptation for similar cases is possible.
- *"That much can be made out of nearly nothing"*
A few definitions are sufficient in order to generate complex systems.
- *Sample solutions*
Your business profits from existing solutions. InfoStar already offers solutions for the most complex cases of data manipulation and data cleansing as a standard technique.

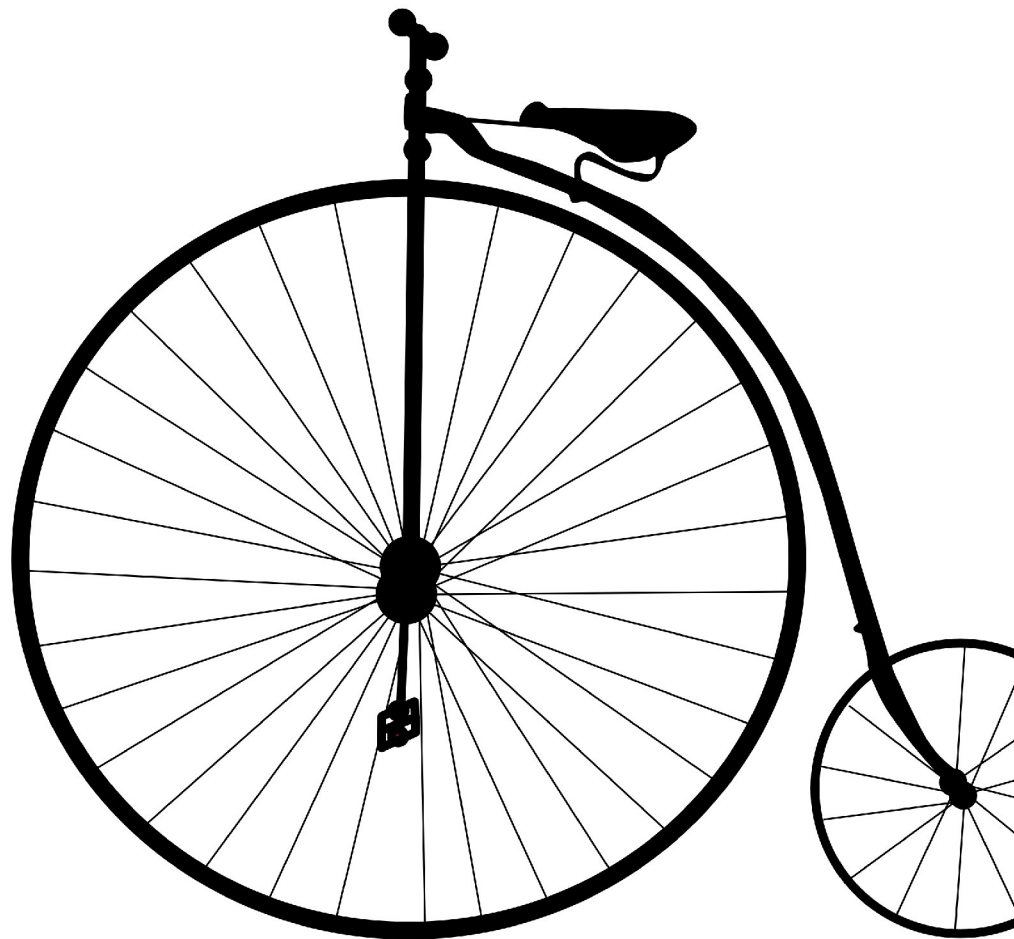
The functionality of the standard procedure can be reduced at any time, depending on the requirements of the task. A reduction of the procedure results in a slimmer application, i.e. that parts of application are not only turned off. The generative approach makes it possible to generate only those parts of the procedure that are actually needed.
- *Individual procedures:*
"Intelligence should be a part of the architecture and not of the furnishings".
User procedures can be realized with little investment – the essential work lies in the design of the solution. With InfoStar, the realization of the solution is not costly.



Once a procedure is developed it can be transferred easily to similar cases. The reproducibility is ensured – only an application with a generative approach can offer such a flexibility.

InfoStar – the Technology

infolytics ag



Modelling

Interface definition

Essential prerequisite for the procedure is the formal description of a delivery system with interface, field, data type and dependency-declarations in tabular format.

Checking and completion

The declarations are checked automatically, and database and platform-dependencies will be added.

Standard technique

There is a proven standard technique for taking over, checking, cleansing and supplying of delivery system data.

Changing the standard technique

The user is able to change the standard procedure and can add functions. Large parts of the application can be omitted, if the goal is a very fast data supply.

Script design

There are ready-to-use templates. Based upon the templates and the interface definitions, InfoStar will automatically generate scripts (transactions). The execution of a script is defined as a database transaction.

Designing commands and instructions

Generated scripts consist of two parts: comments and commands -- platform-dependent commands for data transfer and SQL instructions. Both can be arranged in a free manner. SQL-BNF (Backus-Naur form) is used as language for definition.

Flow control

InfoStar supplies a given operational sequence in the format of job control or input tables for flow control tools. They

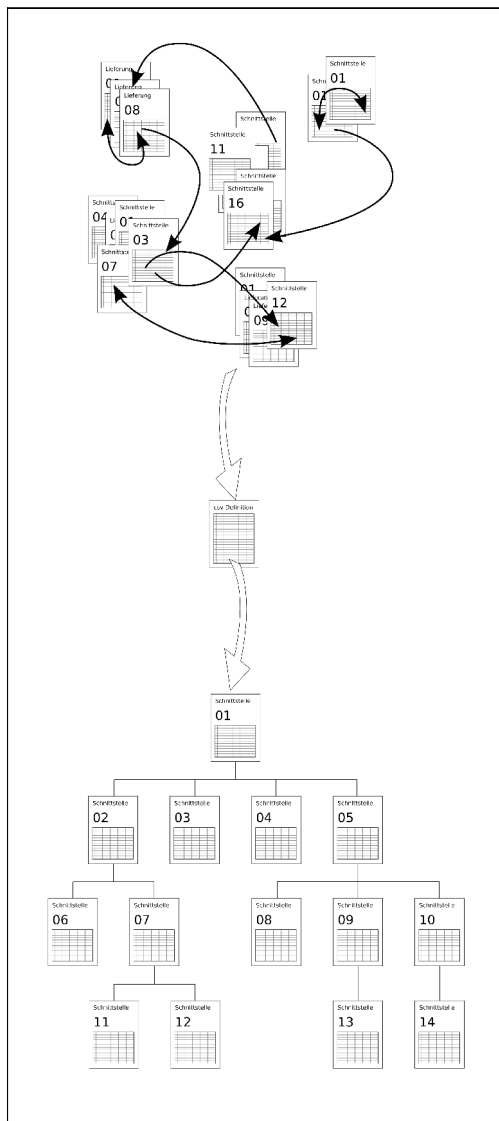


Figure 3: A csv-file creates a mirror of a delivery system. The file contains definitions for all interfaces of the delivery system

can be adapted to the specific requirements of the operator at will.

Selection procedure: Interface-Template-Relation

In addition to the standard technique, which generates all scripts for all interfaces, InfoStar offers a selection procedure. This enables the user to define groups of interfaces and groups of script templates. Subsequently it is possible to individually assign groups to each other. InfoStar will create only the selected scripts for each group of interfaces.

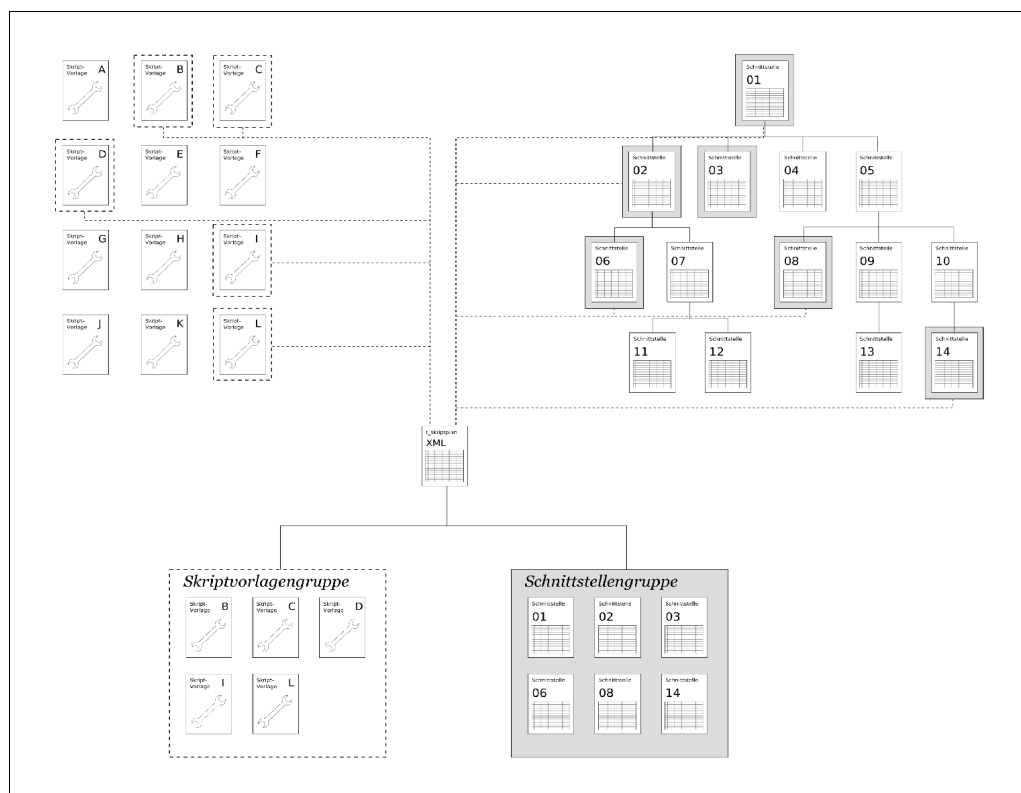


Figure 4: The selection procedure allows the user to define a XML-file with groups of interfaces and templates. For script generation each template-group can be assigned to each interface group.

Standard Technique

Data import with conversion

The data of the delivery system is provided as files in fix- or csv-format. The load procedure depends on the database platform. In cooperation with the customer the most efficient format will be chosen.

There are already generated alternatives, which can be used. After importing data automatic conversions can be made, e.g. number- and date-formats. However this depends on the database platform.

Temporary storage

During a definable period the imported data can be stored in the original format (optional).

Data examination

The data will be checked in several steps for contents, double or empty records and referential integrity.

Cleansing

All deviations are logged in tables. Here it is essential that InfoStar differentiates between new records and already existing records.

Data supply with historization

All records are stored with a validity period. Only correct records are taken over.

By means of Views it is possible to survey uncorrected data.

Historization of data allows the user to define any analysis period. Usually the current and the previous month are covered. The period of historization can be defined freely by the operator.

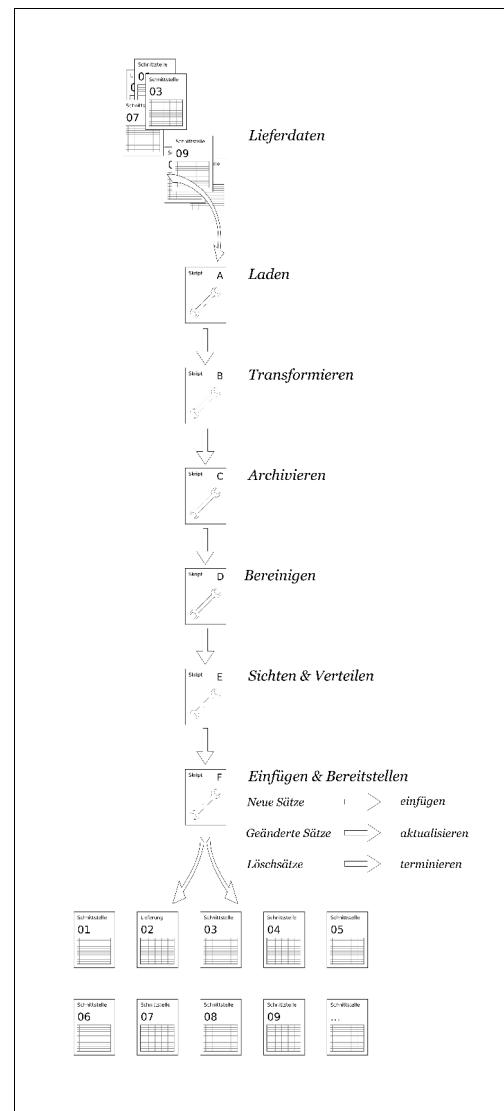


Figure 5: Data processing steps of the standard technique

Custom Procedures

Changes to the standard technique

InfoStar provides an established procedure for importing delivery system data, examining and correcting it and supplying it to a staging area or a Data Warehouse. In each case, the developer or infolytics ag will model an individual optimized procedure for the current environment.

instructions are then added to this template.

The collection of templates is supplemented with new or altered XML or BNF declarations. The specification is compiled and stored in the repository. At the same time Perl interface calls are generated.

Knowledge needed for creating individual solutions with InfoStar

The infolytics ag will train the computer personnel of the customer how to use the technology of InfoStar. Main goal of the initial project is the independent use and, if necessary, further development of InfoStar through the customer.

A generation procedure (which has to be defined) supplies the interfaces. The procedure contains the necessary meta data in its call.

Integration of hand written scripts

Hand written scripts can be integrated at any point of the job control flow.

Prerequisite is some basic knowledge in XML, BNF (Backus-Naur form) and Perl. For simple text generation, which only needs to substitute placeholders, a XML format is sufficient. For more complex formats such as DDL and DML, which may specify variable column lists with names and data, primary keys, constraints and tablespaces or partitioning, annotated BNF will be used.

Perl is used as programming language, whereby the functional range is essentially limited to 3GL-expressions. This range is easily understandable for any software developer.

Execution

For execution a new script with a unique name must be added to the script template collection. New and/or existing

Extraction Procedures

Analyses

In the standard technique the input interfaces description is the base for the generation. It is also possible to start from output interfaces and generate scripts which produce outputs.

Historization

The analysis results can be historized if these outputs take place periodically, or if there is a requirement for documentation or the need to access data of the past.

Object view

Tools for further processing (like e.g. the SAP Bank Analyzer) often have an object view on the data and require the complete object when changes occur. Since the data is stored relationally, the object dependencies must be defined and considered for output. InfoStar has a standard technique which provides this functionality.

Database Structure

Structuring according to delivery systems

Each delivery system is represented by its own user and its own database schema.

One silo per delivery system

A delivery system can be regarded as an independent data silo. All silos share the same structure, but with delivery system-specific characteristics.

Working layers

Each silo has separate working layers for data transfer, cleansing and analysis.

Factual and dimensional data

Each silo is additionally separated into areas for factual and dimensional data. Dimensional data define enumerated data types. InfoStar compares the contents of data fields with the specified dimensional data.

Views for analysis

Different Views are made available for further processing. These Views are generated with the authorization defined by the developer.

Repository

The repository contains the meta data, the generated code and logging data of each delivery system. Factual and dimensional data is generally stored into the staging area. It's also possible establish the repository in a separate database of another RDBMS manufacturer.

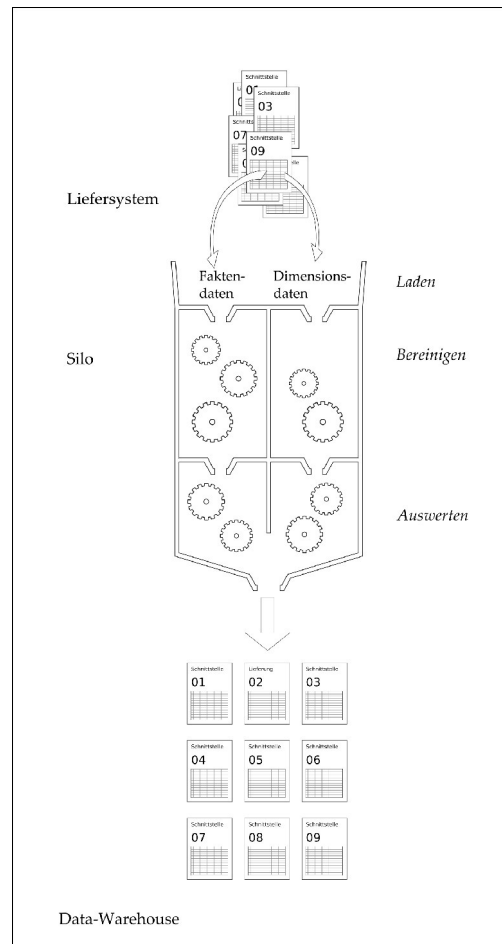


Figure 6: A data silo has separate layers for loading, clearing and analysis. Each silo has two wells for separate processing of dimensional and factual data

Generation

Script templates

For each script of the standard technique there is a template, which determines contents and sequence.

Determination of sequences

The sequences are arranged according to groups:

- Data definition and authorization (DDL)
- Load procedures (LDR and CTL)
- Data manipulation (DML)
- Analyses (SEL)
- Scripts (SH)

- Footer (FTR)

For each interface of a delivery system a standard sequence is defined within its group.

The structure can be extended by any type of generation.

Templates for instructions

Each script template has a generic name and contains one or more instructions. The patterns for these commands have a XML format, they are stored in pattern tables within the repository.

Generation procedures

For each script template there is an appropriate generation procedure. The procedure is supplied with the meta data and makes it possible to generate the interface-specific scripts.

Storage

During the generation process all scripts are written into the repository. A delivery system with 35-40 interfaces needs approx. 1200 generated scripts. The exact extent depends on the complexity of the chosen supplying procedure.

In addition all scripts are exported redundantly into a delivery-system-specific directory structure. From here they can be re-imported into the database, e.g. after an approval which is based on the exported scripts.

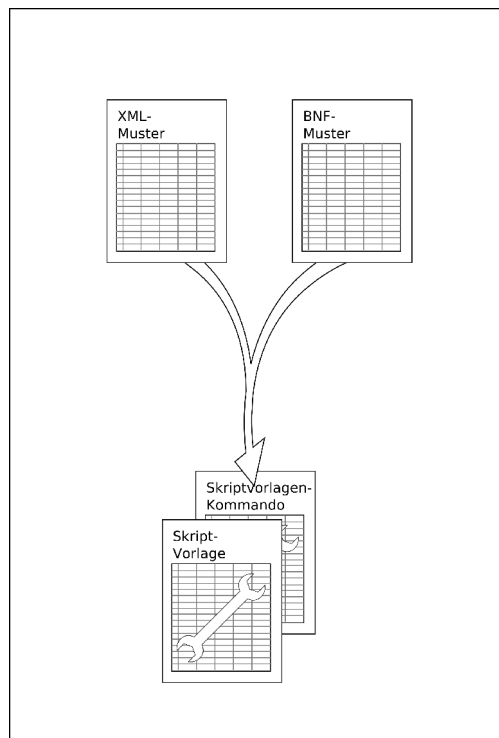


Figure 7: Script templates can be defined with XML- and BNF-patterns

Generation of job flow

At generation time a global process sequence in tabular format is produced, from which job control flow procedures for different job control platforms can be derived.

Data model generation

For each silo a data model is generated, which can be imported into modelling tools. This permits the representation of the model in the accustomed on-site form.

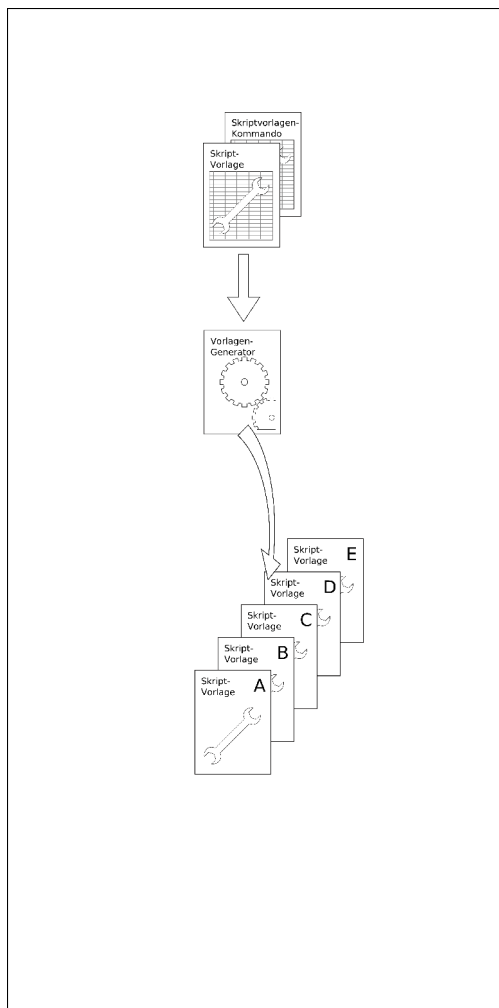


Figure 8: The template generator reads the script definitions and creates the various script templates.

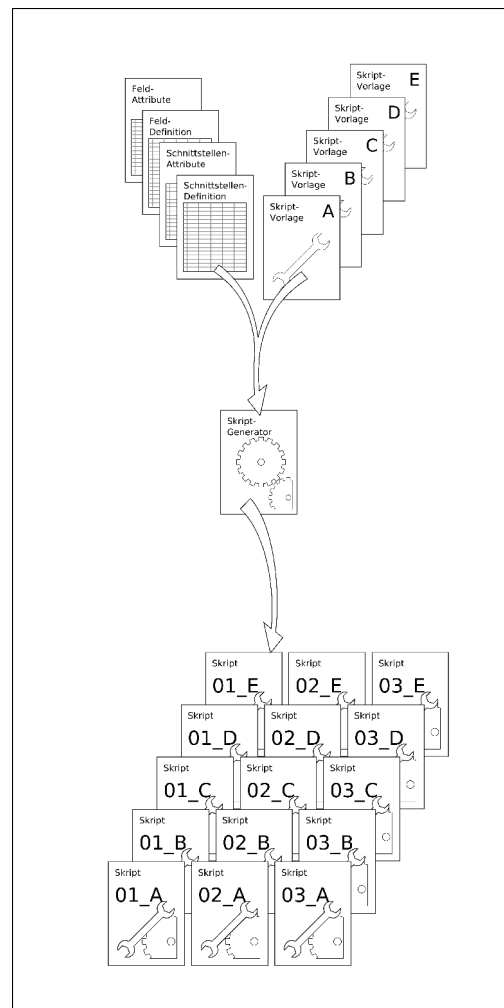


Figure 9: The individual scripts are created on the basis of script templates and interface definitions

Job Control Flow

Process sequence

The process sequence can be divided into:

- *Data declaration/definition (DDL)*
The data declaration takes place only once. All tables and views are created that are needed for the delivery system and its interfaces.
- *Loader declaration (LDR/CTL)*
The loading procedure depends on the platform and the data format. Usually a load command and an corresponding data definition (LDD) is needed for each interface. The loading procedure for the complete dimensional and factual data takes place at the beginning of the data import, in order to recognize data errors at smallest expenditure. It is up to the operator to structure the job nets according to the requirements.
- *Data manipulation (DML)*
The actual processing is made by mass SQL instructions (DML). Dimensional data is processed in the first place. The subsequent transactions take place per interface, whereby hierarchical dependencies are considered for examination of referential integrity. Parallel processing is possible as long as it pays attention to the interface hierarchy.
- *Complete and delta runs*
As a basic principle initial and delta runs are differentiated.
- *SEL*
Reports

Transactions

For each script a transaction (in database terminology) is formed during data processing. The transaction completes with a COMMIT or a ROLLBACK-statement.

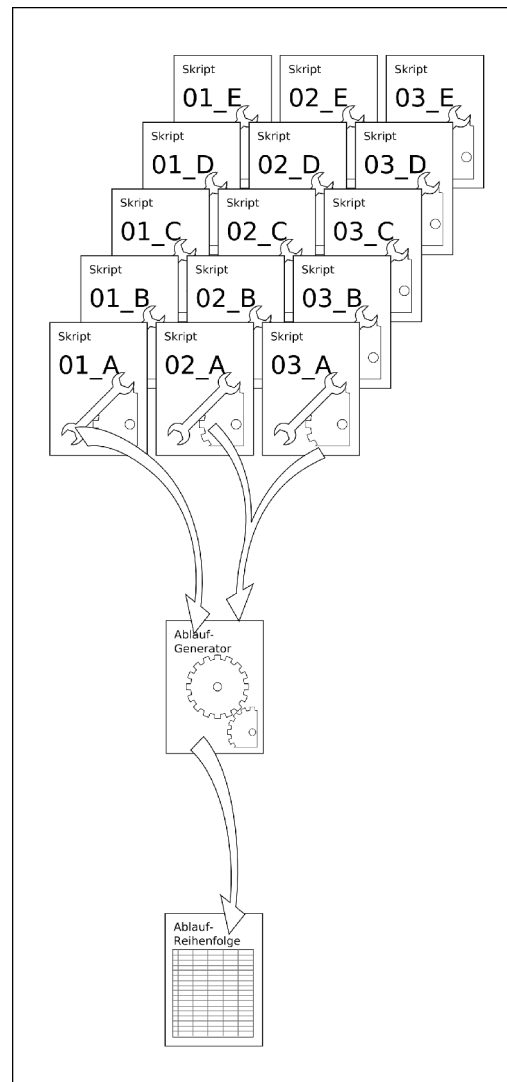


Figure 10: The sequence generator creates the process sequence

Miscellaneous drivers

A transaction is called via the respective job control. For the development and test environment additional process scripts are generated which are used as driver scripts.

It is also possible to use a test driver which directly controls the entire process based upon repository entries. The test driver works independently from the job control system.

Hand written scripts

Although the procedure takes an important advantage of the automatic generation of all scripts, hand written scripts can be integrated at any place into the process.

Installation and Operation of the Repository

Installation

The installation is done by means of a few procedures. A demo application is part of the installation. Under Windows systems no registration is necessary. No DLLs get installed or are presupposed on system side.

Uninstalling

Uninstalling is simple and straightforward: you only have to remove the installation directory and the repository-DB-schema.

Batch administration

The administration can be performed via batch-procedures.

Dialog administration

Additionally a generic HTML-dialogue-application can be used.

Distribution Procedure

Preparation

The scripts produced in the development environment are stored in the repository and in the file system. Normally repository contents cannot be used directly for the test or production environment.

Therefore a little preparation is necessary: you have to copy one or several generation directories into a distribution directory. Afterwards distribution procedures are started which create a compressed delivery file.

Installation

During installation the delivery file will be unpacked into the target environment. The installation scripts are differentiated according to their different authorizations.

A database administrator sets up the repository, whereby only the runtime portion of the repository is taken over, and creates (e.g. for DB2 and Oracle) tablespaces for the repository and the silos. Additionally, the user names for the repository and the silos are created. The corresponding scripts are already generated.

The operator who is responsible for test and production imports the generated scripts into the repository and sets up the tables and views for the individual silos. As before, the corresponding scripts are already generated. When this is done, operation state is reached.

In the test operation the process can be controlled via generated shell scripts or the test driver. In the production environment the process has to be integrated into the official job control flow.

System Requirements

Database system

In principle InfoStar can use any established relational database management system. InfoStar is currently available for Oracle, DB2, MaxDB, and MySQL. Other systems are in preparation.

Perl

As language Perl 5.6.1 and higher is used. Perl is portable and easy to install.

Perl-DBI or ODBC

For accessing the repository an ODBC interface or a database-specific DBI (database interface) is required.

Perl-XML

Additionally, a XML library is necessary for the development environment.

Apache

An Apache Webserver must be installed if the available HTML interface for repository administration is to be used.

Job control language (JCL)

A job control system should be used as a driver.

Adaptations

If a database that is not supported yet is to be used, the process has to be examined for compatibility and should be adapted. Additionally an optimization analysis is necessary. For JCL not supported yet templates and generation procedures must be specified.

Documentation

User manual

InfoStar is comprehensively documented. The application contains “more comments than code”.

User manuals are supplied for installation and operation.

Training manual

Beside the user manual there is a training manual, which provides an introduction into the program.

Powerpoint Presentation

A PowerPoint presentation is available. It can also be used for training.

Documentation of the delivery systems

The respective delivery systems including their foreign key relations are documented as DDL. It is possible to create diagrams with the help of CASE tools.

Generation of documentation

InfoStar generates a hierarchically organized HTML documentation with thorough information depth.

Open Procedure

Perl scripts

All Perl scripts are handed over. The entire procedure is transparent for the developer. The generation environment has to be installed only on the development system. In the production environment the Perl DBI interface is sufficient.

Database scripts

The scripts required for installing and operating the repository are a part of the InfoStar delivery.

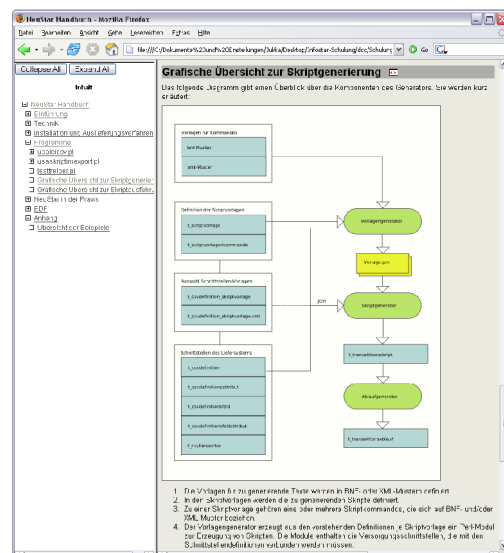


Figure 11: The user manual provides both a “Quick Tour” for a fast introduction and detailed information on every aspect of the application

Content-related Functions of the Procedure

The functions specified on this page are a part of the *standard technique* provided by InfoStar. That means: according to the requirements of the customer they can be transferred as a whole or in part into user defined procedures.

Interfaces

Interface data is stored unchanged (though historized) in the databases. The period of storage of the historized data is defined by the operator.

Treatment of blank records

Records with blank content are recognized and filtered.

Deleting records

Records with a deletion mark are terminated. That means: the timestamp for the end of validity is set to the timestamp of the data supply. Thus the data record is scheduled/terminated, since a new current copy will not be inserted into the database.

Delta delete procedure

Some delivery systems “think” in objects, and not relationally. They supply complete objects, without being able to indicate which part of the object has been changed. The missing of a sub-structure is an indicator for the deletion of a record. For checking this the supplied data must be compared with the historized data. If a record is a part of the stock, but not a part of the object, the record has to be

terminated. The procedure is performed via mass-SQL-instructions.

Double records

Double records in the interfaces do not lead to an abort, since they are filtered and thus are not historized.

Changes

New and edited records are separated. Separate Views, which show either new or existing or all records, are made available.

Referential integrity

Violations of referential integrity are filtered out. Views on these records are provided.

Historization

The historization corresponds to a complete historization. If an attribute changes, the entire record is historized.

On requirement, other techniques like hybrid historization or qualifying-date-dependent or field-related historization can be implemented.

Contact Information

infolytics ag

Jürgen Höhe,
Jukka Höhe

Infolytics AG
Marktstraße 8
50968 Köln

Phone: 0049 (0) 221 – 340 58 40
E-Mail: info@infolytics.com