Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 16 June 2015

Case Number: T 1723/09 - 3.5.07

Application Number: 04793918.6

Publication Number: 1671248

IPC: G06F17/30

Language of the proceedings: EN

Title of invention:
A system, a method and an apparatus for importing text data into a database

Applicant:
Schlumberger Holdings Limited
Services Pétroliers Schlumberger

Headword:
Spreadsheet loader/SCHLUMBERGER

Relevant legal provisions:
EPC Art. 56, 123(2)

Keyword:
Inventive step - main request (no)
Amendments - added subject-matter (yes) - auxiliary request

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern
Boards of Appeal
Chambres de recours

European Patent Office
D-80298 MUNICH
GERMANY
Tel. +49 (0) 89 2399-0
Fax +49 (0) 89 2399-4465

Case Number: T 1723/09 - 3.5.07

D E C I S I O N
of Technical Board of Appeal 3.5.07
of 16 June 2015

Appellant: (Applicant 1)	Schlumberger Holdings Limited P.O. Box 71, Craigmuir Chambers Road Town, Tortola (VG)
Appellant: (Applicant 2)	Services Pétroliers Schlumberger 41, rue Saint Dominique 75007 Paris (FR)
Representative:	Osha Liang 2, rue de la Paix 75002 Paris (FR)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 11 March 2009 refusing European patent application No. 04793918.6 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	R. Moufang
Members:	M. Rognoni
	P. San-Bento Furtado

Summary of Facts and Submissions

I. The applicants (appellants) appealed against the decision of the Examining Division to refuse European patent application No. 04793918.6.

II. In the decision under appeal the Examining Division considered that claim 1 according to the main request filed in electronic form on 16 January 2009 did not involve an inventive step with respect to the following prior art:

D3: Durbin J. et al., "ORACLE 8 Utilities", Release 8.0, December 1997.

Furthermore, the Examining Division held that claim 1 according to the auxiliary request, also filed on 16 January 2009, did not comply with Articles 123(2) and 84 EPC, and that its subject-matter did not involve an inventive step with respect to document D3.

III. With the statement of grounds of appeal, the appellants submitted a new main request and a new auxiliary request, and requested that the decision under appeal be set aside and a patent be granted on the basis of the new main request or, alternatively, on the basis the new auxiliary request.

IV. With letter dated 18 December 2014, the appellants were summoned to oral proceedings to be held on 16 June 2015.

V. In a communication pursuant to Article 15(1) RPBA dated 2 April 2015, the Board expressed the preliminary view that the subject-matter of claim 1 according to the main request did not appear to involve an inventive

step within the meaning of Article 56 EPC and that claim 1 according to the auxiliary request appeared to comprise subject-matter extending beyond the content of the application as filed (Article 123(2) EPC).

VI. In reply to the Board's communication, the appellants, with letter dated 9 June 2015, informed the Board that they would not attend the oral proceedings.

VII. Oral proceedings were held on 16 June 2015 in the absence of the appellants. At the end of the oral proceedings, the chairman pronounced the Board's decision.

VIII. Claim 1 according to the main request reads as follows:

"An apparatus for loading data representing a well in the exploration and production domain from a spreadsheet dataset (20) into a database (35), comprising:

a) a control file (30) containing a set of rules (100) wherein each rule describes how to parse the data and includes a condition (125) which is a logical state that must evaluate to true if a rule is to be processed;

b) a spreadsheet loader (10) having as inputs the spreadsheet dataset (20) having data representing the well in the form of one or more records and the control file (30), wherein each rule (100) in the control file (30) is evaluated for each record (21) to determine if the condition is true for that record,

wherein the control file includes rules further comprising

i) a parsedescriptor (130, 220) used to parse the record by breaking it into either a

single or an array of tokens if the condition (125) is true;

ii) a value clause (135, 230) that references which of the tokens parsed using the parsedescriptor is assigned as a value,

the parsed, valued data being loaded into the database (35) by the spreadsheet loader (10)."

Claim 1 according to the auxiliary request reads as follows:

"An apparatus for loading data representing a well in the exploration and production domain from a spreadsheet dataset (20) into a database (35), comprising:

a) a control file (30) containing a set of rules (100) wherein each rule describes how to parse the data and includes a condition (125) which is a logical state that must evaluate to true if a rule is to be processed;

b) a spreadsheet loader (10) having as inputs the spreadsheet dataset (20) having data representing the well in the form of one or more records and the control file (30), wherein each rule (100) in the control file (30) is evaluated for each record (21) to determine if the condition is true for that record,

wherein the control file comprises a mixture of variable rules (110) and entity rules (120), wherein said variable and entity rules each comprise

i) a parsedescriptor (130, 220) used to parse the record by breaking it into either a single or an array of tokens if the condition (125) is true;

ii) a value clause (135, 230) that references which of the tokens parsed using the

parsedescriptor is assigned as a value, the parsed, valued data being loaded into the database (35) by the spreadsheet loader (10) and wherein

the variable rule (110) is used to read a record from an input text file and stores the parsed, valued data in a variable to be used later in an entity rule or in a condition of another rule and

the entity rule (120) further comprises an attributelist (145) with a list of attributes (200) that refer to an object to be created and are defined on the database (35) into which the spreadsheet loader will load the parsed valued data, wherein the entity rule further describes how the attributes are set using the parsed, valued data stored in the variable rule or a record read from an input text file."

Both requests comprise a further independent claim directed to a "method for loading data representing a well in the exploration and production domain from a spreadsheet dataset (20) into a database (35)". As the independent method claims are not relevant to the present decision, their wording need not be specified.

IX. The appellants' arguments, submitted with the statement of grounds of appeal, may be summarised as follows:

The apparatus defined by claim 1 according to the main request was novel over document D3 first of all because data in the spreadsheet dataset represented a well in the exploration and production domain. Furthermore, as it appeared from the example of the ORACLE SQL loader utility described on pages 5.44 and 5.45 of document D3, the INTO TABLE statements, which corresponded to

the rules in the control file according to the invention, did not include a combination of a "parsedescriptor" and a "value clause" as specified in claim 1 of the main request.

As shown by the control file for example 1 in paragraph [00034] of the published application, the parsedescriptor defined how to read the records, consisting of tokens, in the text file and, in particular, specified the character used to separate the tokens. Once the tokens were read, the value clause for each attribute identified the token (*i.e.* the value) in the incoming data record to be assigned to that attribute. For instance, in the given example token 2 was selected for the attribute "UBHI" and token 1 for the attribute "Driller_PBTD".

Thus, the apparatus according to claim 1 required firstly that each record was evaluated and parsed into one or more tokens if the condition specified by the WHEN clause was true, and secondly that a value clause identified which of the parsed tokens was assigned as a value, so that only the parsed, valued data was loaded into the database by the spreadsheet loader. This differed from the disclosure of document D3 where each record, capable of being correctly parsed, was parsed and loaded into the database.

The specific features which distinguished the present invention from the Oracle SQL loader described in document D3 allowed to load any text data in spreadsheet format, including row-based data and column-based data. In fact, the SQL loader disclosed in document D3 could only read row-based data since records which met the WHEN conditional clause were directly loaded into the database and could not be

selectively put on hold using the combination of parsedescriptor and value clause of the present invention. As the prior art did not mention this technical effect, the apparatus according to claim 1 involved an inventive step.

The subject-matter of claim 1 according to the auxiliary request also involved an inventive step over the cited prior art for the same reasons given with respect to the main request and furthermore because the control file included a mixture of variable and entity rules.

As explained in paragraph [00032] of the published application, a variable rule was used to read a token from data in the form of an input text file and store it as a variable after it had been parsed and assigned as a value using a corresponding value clause. Then, the stored parsed data was used in either an entity rule or in a condition of another rule. The use of the entity rule was to describe the object to be created and list its attributes. In particular, the entity rule described how the attributes were set using the parsed, valued data stored in the variable rule, or using directly a token which was read from the input text file.

The technical effect of the control file having a mixture of variable and entity rules was to enable the loading of any format, including row-based and column-based formats, and the loading of the data in the database.

Hence, the subject-matter of claim 1 of both requests satisfied the requirements of Article 56 EPC.

Reasons for the Decision

1. The appeal is admissible.

2. As specified in paragraph [0002] of the published application, the present invention is concerned with the problem of loading text data into a database.

According to paragraph [0004], data to be loaded (*i.e.* a *"dataset"*) is usually encoded in binary or text (ASCII) format and contained in an external storage unit. Also in the oil and gas industry it is known to use a software program (a "loader") to read, decode and load sets of data written in various formats into database tables, whereby custom loaders are written to read specific industry standard formats and load data in a specific repository.

However, it is noted in the description (*ibid.* paragraph [0005]) that there is a large amount of data in text files for which no standard exists and which may contain data in any format.

2.1 In view of the above scenario, apparently typical in the oil and gas industry, the present invention seeks to provide an apparatus and a method for loading data, especially in non-standard text files, into a database (cf. paragraph [0006]). The data considered in the application is defined as a *"spreadsheet dataset"* and may be *"row-based"* or *"column-based"*.

2.2 The gist of the present invention consists essentially in providing a control file containing a set of rules,

which describe how to parse and map the data in the dataset.

Main request

3. Claim 1 according to the main request relates to an apparatus for loading data representing a well in the exploration and production domain from a spreadsheet dataset into a database.

The apparatus comprises:

(a) a control file containing a set of rules wherein

(x) each rule describes how to parse the data and

(xi) (each rule) includes a condition which is a logical state that must evaluate to true if a rule is to be processed;

(b) a spreadsheet loader having as inputs:

(x) the spreadsheet dataset having data representing the well in the form of one or more records and

(xi) the control file,

(xii) wherein each rule in the control file is evaluated for each record to determine if the condition is true for that record,

(xiii) wherein the control file includes rules further comprising:

(xiv) a parsedescriptor used to parse the record by breaking it into either a single or an array of tokens if the condition is true;

(xv) a value clause that references which of the tokens parsed using the parsedescriptor is assigned as a value;

(c) the parsed, valued data being loaded into the database by the spreadsheet loader.

4. Claim 1 according to the main request on file differs from claim 1 of the main request considered by the Examining Division essentially in that it specifies that the control file includes rules comprising features (b)(xiv) and (b)(xv), and in that the expression *"the parsed record"* in feature (c) has been replaced with *"the parsed, valued data being"*.

In fact, instead of features (b)(xiv), (b)(xv) and (c), the former main request recited that *"if the condition is true, the record is parsed into one or more tokens and the parsed record is loaded into the database (35) by the spreadsheet loader (10)"*.

4.1. In order to better understand some of the terminology used to define the claimed invention, it is useful to refer to some passages of the description.

According to paragraph [0009] of the application as published, the *"invention may also include a user interface used to create the control file. The user interface may include a control file information section, having an object tree, attributes tab, and control file preview area, and a data file information section, having a spreadsheet data area reflecting the spreadsheet dataset, [sic] Using the user interface, selecting a business object in the object tree opens a list of one or more attributes for the business object in the attributes tab. Attributes from the open list of attributes may be dragged and dropped into columns in the spreadsheet data area; and the spreadsheet dataloader uses the attributes dropped into the*

spreadsheet data area to map the columns to the business objects and to create the control file from the mapping. The control file may be previewed in a control file preview area in the user interface. Attributes in the attribute tab that are mandatory for the selected business object may distinguished [sic] from attributes that are not mandatory, such as with a color, shading, or use of a special font. The spreadsheet data area of the user interface may include editing functions. The object tree may display the business objects using a two-tier hierarchy, with a list of sub models (being groups of business objects), and with business objects under each sub model" (underlining added).

As specified in paragraph [00030], the spreadsheet dataset generated as explained above "*may be read ... in two ways: as a sequence of rows ("row-based data") or as a sequence of columns ("column-based data"). Referring to FIG. 8, each entry in the spreadsheet dataset is called a record 21*" (underlining added).

5. The Examining Division referred to examples 1 and 2 on pages 8 and 9 of the description, and interpreted the term *"a spreadsheet dataset"* in the light of the description as an input text file which was either character delimited or fixed length.

5.1 Furthermore, in the opinion of the Examining Division, the wording *"the record is parsed into one or more tokens and the parsed record is loaded into the database"* meant that input text files were read and a number of physical records, delimited by the end-of-the-line signs, were identified. These records were parsed into a number of strings or numbers, referred to

as tokens in the claim, which were used for populating the database tables.

5.2 The Examining Division arrived then at the conclusion that the subject-matter of claim 1 differed from the disclosure of document D3 only in that data in the spreadsheet dataset represented a well in an exploration and production domain. However, the content of the data stored in the database did not provide per se any technical effect. Consequently, in the Examining Division's opinion, the fact that a spreadsheet dataset contained data representing a well in an exploration and production domain did not contribute to an inventive step (Article 56 EPC).

6. In the statement of grounds of appeal, the appellants did not contest the Examining Division's interpretation of the claim wording, but essentially argued that document D3 did not describe at least the combination of features (b)(xiv) and (b)(xv) recited in claim 1. These features made it possible to load any text data whose layout was in spreadsheet format, including row-based data and column-based data, as respectively shown in examples 1 and 2 of paragraph [00030].

Interpretation of claim 1

7. Claim 1 relates to an apparatus for loading data, which represents *"a well in the exploration and production domain"*, from a spreadsheet dataset into a database. As to the meaning to be given to the term *"spreadsheet dataset"*, the Board agrees with the Examining Division that it can be interpreted as an input text file which is either character-delimited or fixed length.

An example of a character-delimited dataset read as a sequence of rows (*"row-based"*) is given on page 8 of the application. As explained in paragraph [00030], data in each row represents *"one or more business objects"*, *i.e.* an *"entity"* such as a well in the exploration and production domain. Each column represents an "attribute" used to describe the "business object", whereby attributes in a row are separated by special characters. In example 2 relating to the column-based data given on page 9, data in a column is said to be a "business object" and typically to represent "bulk channel data in the exploration and production domain". In fact, the first column in example 2 contains depth values and each of the subsequent columns the values for a particular parameter measured at the depths specified in the first column. In other words, each row contains logging measurements taken at a particular depth. The values of each column are written in fields with a fixed length.

7.1 According to feature (a) (see the Board's itemisation of claim 1 at point 3.), the claimed apparatus comprises a "control file" containing a set of rules, whereby each rule describes how to parse the data and includes a condition for the rule to be processed, i.e. for a corresponding data item to be extracted from the spreadsheet dataset.

7.2 Feature (b) identifies the inputs of the spreadsheet loader, *i.e.* data representing the well and the control file, and specifies that each rule in the control file is evaluated for each record.

7.3 An example of a control file comprising a "parsedescriptor" (feature (b)(xiv)) and a "value clause" (or "Values clause" - as the terminology is not

consistent throughout the application - cf. feature (b)(xv)) is given in paragraph [00034] of the description (cf. <ParseDescriptor> and <Values> in the control file 30).

As explained by the appellants (statement of grounds of appeal, page 4, last paragraph and page 5, first paragraph), the parsedescriptor tells the system how to read the data "tokens" (i.e. the strings or numbers). In particular, it specifies the delimiter of each token or the character position of a token in the data record.

As to the "value clause", it identifies for a particular attribute one of the tokens parsed by the "parsedescriptor" (cf. paragraph [00036]).

7.4 In the example given in paragraph [00034], the entity "XBorehole" is identified by an attribute list comprising different attributes. One of the attributes is the "Unique Borehole Identifier" (UBHI). The corresponding "ParseDescriptor" identifies two values separated by a delimiter. The "Values clause" identifies the token at the second position "1" as the value to be selected by the loader and loaded into the database.

7.5 It is understood that the function of the "Values clause" depends on the fact that a corresponding parsedescriptor may extract from a record more than one string of characters or numbers ("token"), and that one token in an array of extracted tokens is to be identified as attribute value for a certain "object" (cf. paragraph [00036]).

According to claim 1, however, the parsedescriptor may also parse the physical record by breaking it into a single token. In this particular case, the value clause would merely assign the parsed value to a variable or attribute.

7.6 In the Board's opinion, the functionality provided by features (b)(xiv) and (b)(xv) is linked to the particular data structure selected for the spreadsheet and the way the parsedescriptor operates. If the parsedescriptor extracts not one, but an array of items or "tokens" from the dataset as a possible "attribute value" for an "entity", it is necessary to define a "rule" for identifying and selecting the desired token in the array of extracted tokens.

8. As observed in the contested decision (page 4, first full paragraph), document D3 discloses the use of a WHEN clause which defines conditions to be fulfilled before data from the dataset can be loaded (cf. features b(xiv) and b(xv) of the Board's itemisation of claim 1 at point 3.).

8.1 Case 5 (starting from page 4-18) of document D3 demonstrates, *inter alia,* the use of Oracles's SQL*Loader *"to break down repeating groups in a flat file and load the data into normalized tables".* Multiple INTO TABLE clauses are used "*to load data into different tables*", "*extract multiple logical records from a single input record*" and "*distinguish different input record formats*" (see D3, page 5-43, "Using Multiple INTO TABLE Statements"). In particular, a first INTO TABLE clause identifies the physical records for the table "emp" by specifying for each of the corresponding attributes ("empno", "ename", "deptno" and "mgr") the position of the physical record in the

input datafile. Three similar INTO TABLE clauses identify the physical records for the table PROJ. As explained in the "Notes" on page 4-19 of document D3, the WHEN clause expresses the condition for loading a row into the table PROJ only if there is a value in a specified position, whereby the position is expressed in terms of the columns occupied by the corresponding data field in the input datafile (*i.e.* "spreadsheet dataset").

8.2 In other words, the INTO and WHEN clauses described in document D3 express the condition for parsing the physical record and for loading data values into the database. Thus, as far as their functions are concerned, they correspond to features (b)(xiv) and (b)(xv) of claim 1 (see the Board's itemisation), in particular, when the record is broken into a single token.

9. The appellants essentially argued that the parsedescriptor and the value clause specified in features (b)(xiv) and (b)(xv) of claim 1 could be used to read row-based data or column-based data as shown in example 2 of paragraph [00030]. In other words, each physical record was read using the parsedescriptor. Then, the value clause identified, among all the parsed data associated with that record, the data string or "token" to be loaded into the database.

9.1 In the appellants' view, the SQL loader disclosed in document D3 could only read row-based data since the records, if they met the WHEN conditional clause, were directly loaded into the database, and could not be selectively *"put on hold, as in the invention, using the combination of the parsedescriptor and the values*

clause" (statement of grounds of appeal, page 5, last paragraph).

10. As to the latter argument, the Board notes that according to claim 1 the value clause merely references which of the tokens parsed is assigned as a value (for a particular attribute) and that the parsed, valued data is loaded into the database. No feature of claim 1 can be linked to the function of selectively putting on hold data records extracted from the spreadsheet datafile.

10.1 As to the alleged limitation of the SQL loader to reading only row-based data, the Board has difficulty in following the appellants' argument. The example given in the application (example 2 on page 9) of column-based data identifies as a "business object" the "channel data" in a column, whereas each row identifies the depth at which "channel data" are collected.

10.2 Similarly in the datafile for "Case 5" shown on page 4-19 of document D3, the last three columns containing data of project 1, project 2 and project 3 can be seen as a "business object" in the sense of the present application. As shown on page 4-18 of document D3, the control file contains instructions to load data into the table "PROJ" and does so by addressing the first column and the last three columns of the spreadsheet file on page 4-19.

10.3 According to Case 5 in document D3 (page 4-18), each physical record (*i.e.* data value) to be loaded into a table row is identified by specifying the exact positions it occupies in the row of a flat file. In the example given on page 8 of the present application, the tokens for the same "attribute" (for instance "uwi") do

not occupy the same position in each row because the first tokens ("well_name") in the rows have different lengths and records are separated by a special character. The Board accepts that the example shown on pages 4-18 to 4-23 of document D3 would not be directly applicable to the case of variable format records separated by symbols.

10.4 However, document D3 deals also with variable-length data (see for instance page 3-8) and gives an example of a loader for variable-length data (see "TERMINATED BY" clause in case 1, page 4-5).

Furthermore, document D3 explains how the position of a data field is specified using the "POSITION" keyword. According to page 5-41, the data field in the "logical record" is defined by the starting and the ending positions, or by an integer indicating the number of characters to be skipped from the last data field before reading the value for the current field. On page 5-42, the use of "POSITION" with multiple table loads is further illustrated. Finally, document D3 on page 5-43 shows how "POSITION" is used to extract multiple logical records.

10.5 In summary, the loader described in document D3 has the same functionality provided by the parsedescriptor and the value clause specified in claim 1 of the main request, whereby the parsedescriptor breaks a physical record into a number of tokens and the value clause identifies the token representing the data value to be loaded into a table of the database.

10.6 The Board understands that the approach suggested by the present application may be particularly suitable for variable length record formats where the column

position of the physical records in a flat datafile cannot be directly specified, as for instance in Case 5 of document D3.

However, in the Board's opinion, a person skilled in the art, faced with the problem of applying the teaching disclosed in D3 to the kind of spreadsheet datasets considered in the present application, would have been aware that physical records of variable length could be easily broken into an array of tokens simply by specifying the special characters which separated them. The skilled person would also have realised that in this case it was necessary to take some measures in order to identify a token in an array of tokens. Consequently, it would have been obvious to the skilled person to add to the control file of the loader a rule for defining which value in an array of parsed values was to be selected for a certain data field (*i.e. "attribute"*). In so doing, the skilled person would have arrived at an apparatus falling within the terms of claim 1 of the appellants' main request.

10.7 Hence, the subject-matter of claim 1 according to the main request lacks an inventive step within the meaning of Article 56 EPC.

Auxiliary request

11. Claim 1 according to the auxiliary request differs from claim 1 of the main request in that feature (b)(xiii) of the latter has been replaced with

(xvi)wherein the control file comprises a mixture of variable rules and entity rules,

(xvii) wherein said variable and entity rules each comprise

and in that it further comprises the following features:

(d) the variable rule is used to read a record from an input text file and stores the parsed, valued data in a variable to be used later in an entity rule or in a condition of another rule and

(e) the entity rule further comprises an attributelist with a list of attributes that refer to an object to be created and are defined on the database into which the spreadsheet loader will load the parsed valued data,

(x) wherein the entity rule further describes how the attributes are set using the parsed, valued data stored in the variable rule or a record read from an input text file.

12. In the statement of grounds of appeal, the appellants indicated that claim 1 according to the auxiliary request had been limited with the features of dependent claim 2 of the main request. According to the appellants, these features found support in paragraphs [00032] and [00034] to [00036] of the application.

13. Features (xvi), (xvii), (d), (e) and (e)(x) correspond indeed to a large extent to the features recited in claim 2 of the main request. However, claim 1 of the auxiliary request further specifies that *"said variable and entity rules each comprise"* a parsedescriptor and a value clause as recited in features (b)(xiv) and (b)(xv).

13.1 Paragraph [00032] of the application relates to Figure 2, which illustrates the mixing of variable and entity rules. For more details on these rules, paragraph [00032] refers to Figures 3 and 4.

14. According to paragraph [00035], lines 3 to 6 of the description, the "variable rule" is constructed as a sequence of a "condition", a "parsedescriptor" and a "values clause". It may additionally include and "endcondition" and an "attributelist" (see Figure 3). The same is specified in claims 3 to 5 of the original application. Paragraph [00038] is concerned with Figure 5 which shows the loader's components used for assigning a value to an attribute.

14.1 As to the entity rule, it is specified in paragraph [00032] that it *"describes an entity, an object that is to be created. The entity rule **120** lists attributes of the object and describes how the attributes are set, using a variable rule **110** or the token, which is read from the input text file"*. Similarly, it is shown in Figure 4 and explained in paragraph [00037] that an entity rule comprises a "condition", *i.e.* the logical expression that must evaluate to true if the variable or entity is to be processed, an "attributelist", *i.e.* a list of attributes which can be assigned values for the given entity, and optionally an "endcondition" and a "referenceentity" (Figure 4). This corresponds to claims 6 to 9 of the application as filed.

14.2 As shown in Figure 4, the entity rule 120 comprises an attributelist and the value of an attribute is determined by the parsedescriptor and a value rule (see Figure 5). Thus, it could be argued that, via the attributelist, also the entity rule comprises a parsedescriptor and a value clause. In this case, a

value clause related to the entity rule would be used to assign a value to an attribute of the entity.

14.3 On the other hand, claim 1 states explicitly that, apart from a list of attributes, the entity rule comprises a parsedescriptor and a value clause which are apparently distinct from the parsedescriptor and value clause used to assign a value to an attribute. In other words, the apparatus according to claim 1 appears to comprise value clauses for assigning a "value" to an entity and additionally value clauses for assigning values to the attributes of the entity.

However, it is clear from the whole disclosure that entities (such as "well" or "borehole") are objects to be created in the relational database, whereas attributes are "aspects" of business objects expressed in terms of values read from the text file (see for instance paragraphs [00037] and [00038]). For instance, in the example 1 on page 11 of the description, the entity rule for the entity "XWell" comprises a "Condition", an "EndCondition" and an "AttributeList" with a corresponding variable rule which attributes, *inter alia*, the name to the "XWell". In other words, in the application as filed, a parsedescriptor and a value clause are always part of a variable rule.

14.4 As the combination of features recited in claim 1 of the auxiliary request does not find support in the original application and the evident inconsistencies between the claim and the disclosure cannot be easily overcome by interpreting the claim wording, it must be concluded that claim 1 according to the auxiliary request comprises subject-matter extending beyond the content of the application as filed and thus violates Article 123(2) EPC.

15. In summary, the Board comes to the conclusion that none of the appellant's requests is allowable. Thus, the appeal has to be dismissed.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

I. Aperribay R. Moufang

Decision electronically authenticated