Internal distribution code:

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

Datasheet for the decision of 12 March 2021

Case Number: T 2739/17 - 3.5.07

Application Number: 14182687.5

Publication Number: 2990960

IPC: G06F17/30

Language of the proceedings: EN

Title of invention:
Data retrieval via a telecommunication network

Applicant:
SAP SE

Headword:
Data retrieval/SAP

Relevant legal provisions:
EPC Art. 83, 84

Keyword:
Claims - clarity - main and auxiliary request I (no)
Sufficiency of disclosure - auxiliary request II and modified auxiliary request II (no)

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

Case Number: T 2739/17 - 3.5.07

D E C I S I O N
of Technical Board of Appeal 3.5.07
of 12 March 2021

Appellant: (Applicant)	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf (DE)
Representative:	Richardt Patentanwälte PartG mbB Wilhelmstraße 7 65185 Wiesbaden (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 5 July 2017 refusing European patent application No. 14182687.5 pursuant to Article 97(2) EPC**

Composition of the Board:

Chair	R. de Man
Members:	P. San-Bento Furtado
	J. Geschwind

Summary of Facts and Submissions

I. The appeal lies from the decision of the examining division to refuse European patent application No. 14182687.5 for lack of inventive step in the subject-matter of claim 1 of each of a main request and first to fourth auxiliary requests. In an *obiter dictum*, the examining division expressed the opinion that certain features were unclear.

II. In the statement of grounds of appeal, the appellant requested that the decision under appeal be set aside and that a patent be granted on the basis of the main request or of one of the auxiliary requests I to III, all four requests filed with the grounds of appeal and corresponding to the first to fourth auxiliary requests considered in the appealed decision.

III. In a communication annexed to a summons to oral proceedings, the board expressed its preliminary opinion that claim 1 of each of the requests did not meet the requirements of Article 84 EPC, that claim 1 of the main request and auxiliary request I did not meet the requirements of Article 56 EPC and that auxiliary request II infringed Article 83 EPC.

IV. With a letter of 21 January 2021, the appellant maintained its main request and auxiliary requests I and II and filed a set of amended claims as "modified auxiliary request II".

V. Oral proceedings were held as scheduled. At the end of the oral proceedings, the Chair pronounced the board's decision.

VI. The appellant's final requests were that the decision under appeal be set aside and that a patent be granted on the basis of the claims of one of the main request, auxiliary request I or II, all three requests filed with the grounds of appeal, or modified auxiliary request II filed with letter of 21 January 2021.

VII. Claim 1 of the main request reads as follows (itemisation added by the board):

(a) "A method for efficient data retrieval via a telecommunication network (118), the network interoperatively coupling a mobile telecommunication device (110) with a sender device (120), the sender device hosting an application (124) and being interoperably coupled via the telecommunication network to an in-memory database management system (136) hosting an in-memory database (134), the method comprising:

(b) - creating the application (124) by automatically refactoring a legacy application (402), the legacy application being configured for receiving data retrieval requests and generating one or more database queries which receive the requested data directly from tables of a legacy database (302, 304) managed by a disk-based database management system;

(c) - automatically transferring data stored in the legacy database (302, 304) to one or more tables of the mentioned database (134), the transferred data comprising first attribute values;

(d) - supplementing (702) the database with multiple database views (148; V1, V2, V3) respectively representing a data model (M1-M3), wherein each of said multiple database views is configured for retrieving instances of the data model it represents from one or more tables (T1, T2, T3) and/or from one or more other views (V1.1, V1.2) of the database upon being called, wherein each of the multiple database views (148; V1, V2, V3) comprises a first column (FC1, FC2, FC3) and a second column (SC1, SC2), the first column representing a first attribute of the data model (M1) represented by said database view, the second column representing a second attribute of the data model (M1) represented by said view;

(e) - providing (704) an adapter module (126) hosted by the sender device;

(f) wherein refactoring the legacy application (402) comprising [sic] executing a refactoring module (602), said execution comprising:

(f1) ° analyzing, by the refactoring module, a log (604) of the legacy application (402) and analyzing, by the refactoring module, the source code of the legacy application;

(f2) ° using, by the refactoring module, the result of said analyses for automatically replacing legacy code sections (606, 608) of the legacy application by new code sections (610, 612), thereby creating the application (124);

(f3) ° wherein at least some (606) of the replaced legacy code sections comprise data retrieval statements acting directly on tables of the legacy database (302, 304) for retrieving the first attribute values;

(f4) and wherein the new code sections (610) comprise a specification of a call to the adapter module for triggering the adapter module to create a single SQL query, the single SQL query being configured for retrieving at least some of said first attribute values by calling the one (V1) of the multiple database views of the database (134) that comprises a first column having assigned said first attribute values;

(g) - receiving, by the application (124), a data retrieval request from the mobile telecommunication device and forwarding the data retrieval request to the adapter module;

(h) - receiving (706), by the adapter module, the forwarded data retrieval request (RE) from the application,

(i) - whereby at the moment of receipt, the first column (FC1, FC2, FC3) of each of the multiple database views has already assigned one or more of the first attribute values for the first attribute represented by said first column and each second column has assigned a routine (R1, R2) for dynamically calculating second attribute values for the second attribute represented by said second column;

(j) - evaluating (708), by the adapter module, the data retrieval request for identifying the at least one (V1) of the multiple database views (V1-V3) that is capable of retrieving, upon being called, data specified in the data retrieval request;

(k) - generating, by the adapter module, the single SQL query by transforming the data retrieval request and one or more parameters contained in said request into a single SQL query;

(l) - calling (710), by the adapter module, the identified at least one database view (V1) with the single SQL query, thereby retrieving instances of the data model (M1) represented by the identified at least one database view, the instances being retrieved via the telecommunication network, the single SQL query comprising a first select criterion directed at the first column (FC1-FC3) of said at least one identified database view and a second select criterion directed at the second column (SC1, SC2) of said at least one identified database view, wherein the retrieving of the data model instances comprises calculating the second attribute values selectively for data model instances having been dynamically identified by means of the first select criterion;

(m) - forwarding (720), by the adapter module, the retrieved instances to the application via the telecommunication network; and

(n) - receiving (722) and processing, by the application, the forwarded instances of the identified data model (M1) and returning a result (FR) of said processing to the mobile telecommunication device via the telecommunication network."

VIII. Claim 1 of auxiliary request I differs from claim 1 of the main request in that the following text has been added after feature (f4) (itemisation by the board):

(o) " - wherein the legacy database comprises the first attribute values in one or more source tables and lacks the database views (V1-V3) representing the data models, wherein legacy SQL queries contained in the log comprise column names of columns of the source tables;

(p) - wherein the using of the result of the log analysis for automatically replacing the legacy code sections (606) comprises:

(p1) ° selecting, by the refactoring module, one of the column names comprised in the log;

(p2) ° identifying, by the refactoring module, one or more of the legacy code sections (606) which directly access the source tables of the legacy database via said selected column name;

(p3) ° evaluating, by the refactoring module, a first mapping, the first mapping comprising an assignment of the column names contained in the log with the first attributes of the data models, for identifying one of the first attributes mapped to the selected column name; whereby said new code section (610) used for replacing the legacy code sections (606) is configured to call the adapter module for causing the adapter module to generate the single SQL query, said single SQL query being configured to call the one of the database views that represents the data model comprising the first attribute having been identified by evaluating the first mapping;"

IX. Claim 1 of auxiliary request II differs from claim 1 of auxiliary request I in that the following text was added after feature (p3) (itemisation by the board):

(q) " - wherein the log is descriptive (L608) of legacy procedures (608) of the legacy application having been executed for calculating parameter values of one or more parameters, the parameter values being absent from the legacy database,

(r) the execution further comprising:

(r1) ° selecting, by the refactoring module, one of the parameters comprised in the log;

(r2) ° identifying, by the refactoring module, one or more further legacy code sections (608) which are configured to calculate parameter values for said selected parameter;

(r3) ° evaluating, by the refactoring module, a second mapping, the second mapping comprising an assignment of the parameters contained in the log and one of the second attributes of the data models, for identifying one of the second attributes mapped to the selected parameter;

(r4) ° replacing said one or more further legacy code sections (608) by a further new code section (612), the further new code section (612) being configured to call the adapter module for causing the adapter module to generate a further single SQL query, the further single SQL query being configured to call the one of the database views that represents the

data model comprising the identified second attribute;

(r5) said further single SQL query triggers the execution of a routine assigned to a second column of the database view representing said data model for calculating said parameter values, the calculated parameter values act as the second attribute values of one of the second attributes of said data model;"

X. Claim 1 of modified auxiliary request II differs from claim 1 of auxiliary request II in that the text of features (c), (d), (f4), (p3), (r2), (r4), (r5), (i), (k) and (l) has been amended as explained in the following:

(c) "to one or more tables of the mentioned database (134)" has been replaced with "to one or more tables of the in-memory database (134)";

(d) "supplementing (702) the database with multiple database views (148; V1, V2, V3)" and "from one or more other views (V1.1, V1.2) of the database" have been replaced with "supplementing (702) the in-memory database with multiple database views (148; V1, V2, V3)" and "from one or more other views (V1.1, V1.2) of the in-memory database", respectively;

(f4) "the single SQL query being configured for retrieving [...] by calling the one (V1) of the multiple database views of the database (134)" has been replaced with "said single SQL query being configured for retrieving [...] by calling the one (V1) of the multiple database views of the in-memory database (134)";

(p3) "for causing the adapter module to generate the single SQL query" has been replaced with "for causing the adapter module to generate said single SQL query";

(r2) the text "wherein the one or more further legacy code sections (608) comprise a parameter name of said selected parameter and a function-ID, wherein said parameter name and said function-ID are stored in said log" has been added at the end of the text;

(r4) "further" has been deleted in the text "for causing the adapter module to generate a further single SQL query", and "the further single SQL query" has been replaced with "said single SQL query";

(r5) "further" has been deleted from "said further SQL query";

(i) the text has been replaced with "whereby at the moment of receipt, the first column (FC1, FC2, FC3) of each of the multiple database views has already assigned one or more of the first attribute values, which are already stored in the in-memory database at the moment of receipt, for the first attribute represented by said first column and each second column has assigned a routine (R1, R2) for dynamically calculating second attribute values, which are not stored in the in-memory database at the moment of receipt, for the second attribute represented by said second column;"

(k) the text has been replaced with "generating, by the adapter module, a single SQL query by transforming the data retrieval request and one or more parameters contained in said request into said single SQL query, said single SQL query being configured to call the at least one a [sic]

database view whose represented data model instances comprise the requested data in form of the first and second attribute values;"

(l) the two occurrences of "the single SQL query" have been replaced with "said single SQL query".

XI. The appellant's arguments, where relevant to this decision, are addressed in detail below.

Reasons for the Decision

1. The invention concerns modifying a legacy system for retrieving data from a legacy database with the purpose of improving efficiency of data retrieval. The functioning of the legacy system and the system according to the invention is illustrated in Figures 4 and 5 of the application, which are reproduced below.

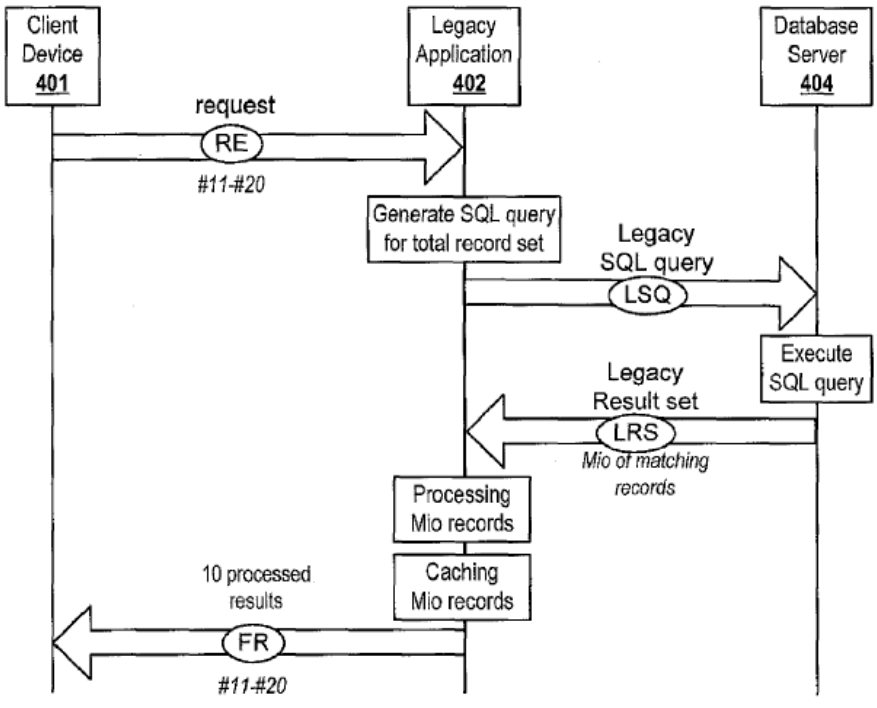

FIG. 4 PRIOR ART DATA RETRIEVAL SYSTEM

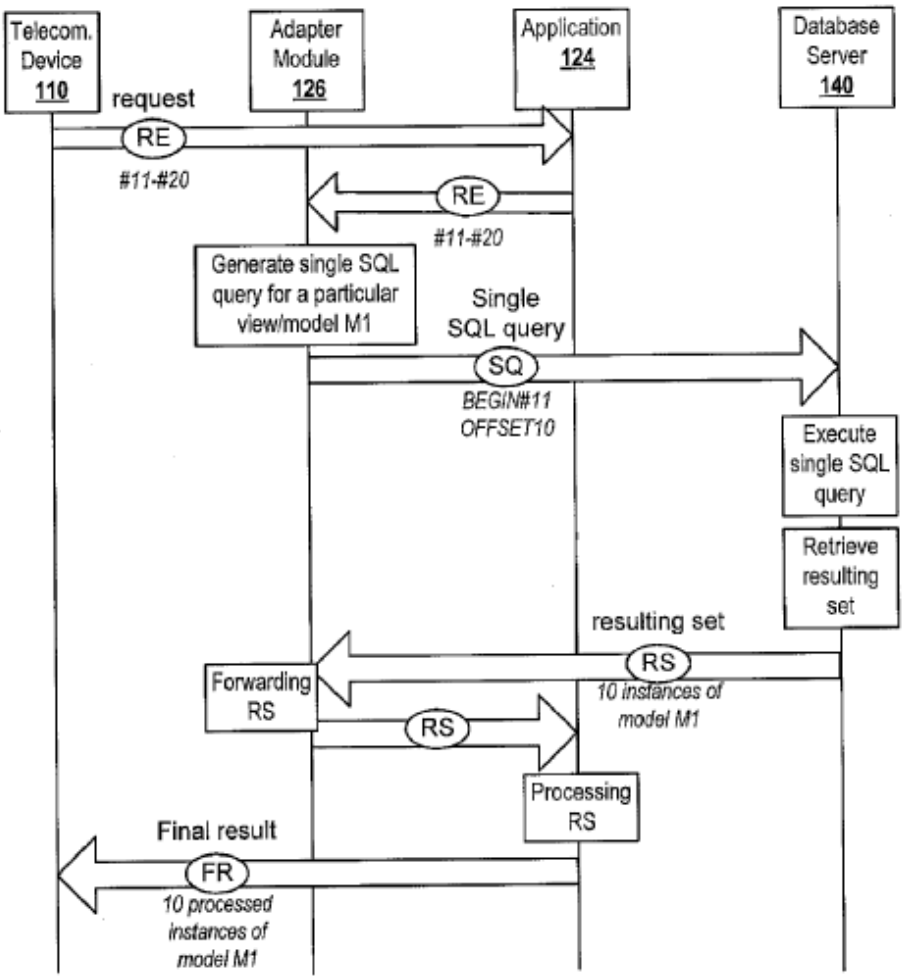

FIG. 5

Main request

Clarity - claim 1

2. Claim 1 of the main request concerns a "method for efficient data retrieval" via a telecommunication network in a system obtained from a legacy system. In the following, the board refers to the system obtained from the legacy system as the "new system".

The claim specifies how the legacy system is converted into the new system (mainly features b to f4) and how a

data retrieval request is processed in the new system (essentially, features g to n).

In the legacy system (see Figure 4), a legacy application 402 receives data retrieval requests, generates database queries and receives the requested data directly from tables of a legacy database managed by a disk-based database management system (feature b).

In the new system (see Figure 5), a mobile telecommunication device is coupled with a sender device, the sender device hosts an application 124 and is interoperably coupled to an in-memory database management system via the telecommunication network (feature a). The application receives a data retrieval request from a mobile telecommunication device, processes the request with the help of an adapter module hosted by the sender device, and returns the results (features e, g and n).

3. The new system is obtained from the legacy system essentially by:

- creating the application 124 by automatically refactoring the legacy application 402 (feature b);
- automatically transferring data from the legacy database to an in-memory database (feature c);
- supplementing the (in-memory) database with views, where each of the views comprises
  a first column (FC1, FC2, FC3) representing a first attribute (feature d) and
  a second column (SC1, SC2) representing a second attribute, where the second column has assigned a routine (R1, R2) for dynamically calculating second attribute values which are not stored in the database when a request is received (features d and i);

- providing the adapter module which processes data retrieval requests received from the application (see features e and h to m).

3.1 Refactoring comprises analysing a log and the source code of the legacy application and replacing code sections of the legacy application which act directly on tables of the legacy database for retrieving first attribute values with new code sections for calling the adapter module. The adapter module creates a single SQL query for retrieving first attribute values using the views of the in-memory database (features f to f4).

3.2 When the adapter module receives a data retrieval request from the application, it identifies at least one of the database views capable of retrieving the data specified in the request and generates the single SQL query comprising first and second select criteria directed at the first and second columns/attributes of the identified database view(s), wherein retrieving the data instances comprises calculating the second attribute values selectively for instances having been dynamically identified by means of the first select criterion. It calls the identified database view(s) with the single SQL query and forwards the retrieved instances to the application (features h to m).

3.3 It is however not clear from claim 1 how the data processing functionality of the new code sections, the adapter module and the database views relates to the original functionality of the legacy application and the legacy database. In particular, the claim does not specify how the second columns/attributes of the views relate to the functionality in the legacy application. Nor does it explain for which purpose the second attribute values are selectively calculated for

instances having been dynamically identified by means of the first select criterion in feature (l).

3.4 According to the appellant, the purpose of the invention is to improve the performance of data retrieval of the legacy system. This presupposes that the result of a data retrieval request in the new system is the same as in the legacy system. This is also in line with the appellant's statement that the calculation of second attribute values is "pushed down" from the application layer to the database layer. But this is not apparent from claim 1, since the claim does not clearly specify the relationship between the second attributes and the legacy database or legacy application.

3.5 Since claim 1 does not clearly define how the second attributes relate to the legacy system, the claim is unclear. Furthermore, the claim does not specify the features necessary to ensure that the functionality of the new system is the same as that of the legacy system, which are essential features of the invention.

3.6 It follows that claim 1 of the main request does not meet the requirements of Article 84 EPC.

Auxiliary request I

4. *Clarity - claim 1*

4.1 Claim 1 of auxiliary request I further specifies in feature (o) that the legacy database comprises the first attribute values in one or more source tables, wherein the legacy SQL queries contained in the log comprise column names of the source tables. It also recites features (p1) to (p3), which specify how the result of the log analysis is used for automatically

replacing the legacy code sections. However, none of the additional features (o), (p), (p1), (p2) and (p3) refers to the second attributes or contributes to clarifying how the second attributes of the database views relate to the functionality of the legacy application and the data retrieval requests supported by the legacy system.

4.2 Therefore, claim 1 of auxiliary request I does not overcome the deficiencies described above for claim 1 of the main request and does not meet the requirements of Article 84 EPC.

Auxiliary request II and modified auxiliary request II

5. Claim 1 of auxiliary request II further specifies that the log is descriptive of legacy procedures of the legacy application for calculating parameter values, and that the refactoring module selects one of the parameters, identifies one or more further legacy code sections which calculate the parameter values, and replaces the further legacy code section(s) by a new code section configured to call the adapter module to generate a further single SQL query. The further single SQL query triggers the execution of a routine assigned to a second column of the database view for calculating the parameter values.

6. Modified auxiliary request II clarifies features of auxiliary request II and additionally specifies in feature (r2) that the one or more further legacy code sections comprise a parameter name of said selected parameter and a function-ID, wherein said parameter name and said function-ID are stored in said log.

7. *Sufficiency of disclosure*

7.1 The features added to claim 1 of both requests relate to the transfer of the calculation of parameter values for a parameter from the legacy system to the new system. In the new system, the parameter values are calculated as "second attribute values" of the second attribute/column of a database view (see features d, q, r1, r3 and i). The parameter values are absent from the legacy database of the legacy system (see feature q) and from the in-memory database of the new system (see feature i of both requests), which means that in both the legacy and the new system those values are dynamically calculated on the basis of values retrieved from the database (see also features i and l). Feature (l) further specifies that the parameter values/second attribute values are calculated selectively for data instances dynamically identified by means of the first select criterion on the first attribute.

It follows from claim 1 of both requests that the (further) single SQL query triggers the execution of a routine assigned to the second column of the database view for calculating the parameter values which were calculated by legacy procedures/code sections of the legacy application (features q, r2 and r5). In order to achieve this, the refactoring module automatically analyses the log and the source code of the legacy application to identify legacy code sections which calculate the parameter values (features f1 and r2) and replaces those code sections in the legacy application with new code sections in the application which call the adapter module for causing the adapter module to generate the (further) single SQL query (features f4 and r4).

Therefore, for a parameter selected from the log (feature r1), the (further) single SQL query generated by the adapter module provides the same functionality as the legacy code sections which calculated the parameter values for that parameter. This means that the creation or generation of the (further) single SQL query to calculate the parameter values requires:

(I) analysing the log and the source code of the legacy application,

(II) detecting the legacy code sections which calculate the parameter values for that parameter and replacing them in the application with the new code sections calling the adapter module,

(III) creating the necessary database view in the in-memory database with a second column representing a second attribute mapped to the selected parameter (see features d and r3);

(IV) creating a routine for dynamically calculating second attribute values and assigning the routine to the second column of the database view (see feature i) and

(V) generating a single SQL query with the same functionality as the corresponding legacy code sections, where the single SQL query calls the database view comprising the second attribute/ column and hence triggers the execution of the routine.

Steps (I) to (IV) are performed in the stage of converting the legacy system into the new system. This is expressed e.g. in feature (i), according to which "at the moment of receipt [of the forwarded data retrieval request]", values of the first column are already stored in the in-memory database and the second columns have already been assigned a routine for dynamically calculating second attribute values, which

are not stored in the in-memory database. At least part of step (V) is performed by the adapter module when it receives a data retrieval request (feature k).

In claim 1 of both requests, steps (I) and (II) are performed automatically by the refactoring module (see features f1, r1 and r2) and the generation of the (further) single SQL query in step (V) is performed automatically by the adapter module (see feature k). Even though the claim does not explicitly mention that steps (III) and (IV) are performed automatically, it is implicit from the claim that steps (III) and (IV) are at least in part automatically performed. In particular, claim 1 specifies in feature (r3) that the refactoring module evaluates a second mapping comprising an assignment of the parameters contained in the log and one of the second attributes, which is related to steps (III) and (IV). The routine generated in step (IV) and the (further) single SQL query have to support the functionality of the legacy code sections, which are identified automatically by the refactoring module.

7.2 However, the application does not explain in sufficient detail how the functionality of the legacy code sections can be transferred automatically to the routine and to the (further) single SQL query which is generated by the adapter module.

7.2.1 The appellant argued that application refactoring as it related to the calculation of the second attributes was described in detail on page 41, line 12, to page 42, line 19, of the description as originally filed. The log was searched for log entries referring to a calculation of parameter values of parameters not stored in the legacy database, but rather calculated dynamically by the legacy application. Parameters not

stored in the legacy database could be identified. Based on an equality of column names and parameter names or based on a mapping, database views could be identified which comprised attributes that corresponded to the identified parameters. As described on page 41, lines 18 to 21, the functions provided by the source code for calculating such parameters could be identified by a function-ID stored with the name of the respective parameter in the log entry. Thus, with the function-ID provided by the log entry, the source code used for calculating the parameter values of the respective parameter could be identified.

The board does not find these arguments convincing, as they do not address the question of how the refactoring module can create a routine associated with the second column of the database view and the (further) single SQL query that achieve the same functionality as an identified legacy code section or legacy function. The board notes that in order to create such a routine, the refactoring module has to analyse the legacy source code and establish all the dependencies between first values and parameter values. Then the refactoring module has to create a routine which performs the same functionality and can be called automatically by the database management system. Since the claim does not limit the source code of the legacy application to any program structure, without further explanations the skilled person would not have been able to implement a refactoring module that could perform these tasks over the whole scope of the claim.

On pages 38 to 42 the application describes with reference to Figure 6 a method of refactoring a legacy application. In this example, the legacy source code includes first source code sections which directly

retrieve data from the legacy database and second source code sections which calculate parameter values which are not stored in the legacy database. A refactoring module generates a new application by automatically transferring data to the new database and supplementing the database with database views (page 39, lines 11 to 17). On pages 40 to 42 the application further describes the application refactoring as described by the appellant. It mentions a single SQL query and a routine assigned to the second column for calculating the parameter values, which may be a routine R1 executed by a processor of the database server or an external routine R2 executed by a processor of a remote server (page 42, lines 16 to 19). However, the description in these pages does not explain how the routine and single SQL are generated on the basis of the legacy code sections.

7.3 The appellant further argued that the attributes or, more precisely, the columns of the respective database view representing the respective attributes had routines assigned to them for calculating attribute values of the respective attributes. Thus, upon identifying corresponding parameters and attributes based on an equality of names or a mapping as well as identifying the source code sections, e.g. based on function-IDs, it was within the ordinary skills of a programmer to implement a program for automatic replacement of source code sections identified by the function IDs with calls to the adapter module to generate SQL queries calling the database views that comprised the corresponding attributes.

These arguments assume that the program, i.e. the refactoring module, is implemented for one particular legacy system and only once the skilled person has

obtained knowledge of the relevant parameters, attributes and source code sections of the legacy system. The claim, however, requires the refactoring module to be able to refactor any legacy application as described in feature (b). The arguments therefore cannot convince the board.

7.4 The appellant also cited pages 31 and 32 of the description. Those pages provide an example of calculation of a parameter "output-per-year" (corresponding to a second column and a routine) as a function of the factory location and machine type (which correspond to first column attributes). The description explains, on page 33, first two lines, and with reference to Figure 2, that the routines R1 and R2 used for calculating the second attribute values may be part of the program logic 202 external to the DBMS 206. However, this passage does not disclose how the routine is created on the basis of the legacy code sections. It does not describe either how the single SQL query is generated. Moreover, this is only a specific example which cannot be generalised to a broad range of applications, whereas the refactoring module of claim 1 would have to be able to refactor arbitrary legacy applications.

7.5 The board is not convinced that the skilled person would be able, without exercising inventive skills and solely on the basis of the disclosure of the patent application as described above, to write a program for automatically detecting code sections of the legacy application and move them to a routine to be performed by the database server, while preserving the functionality of the program and the system as a whole. In particular, the board is not convinced that the skilled person would be able to write such a program

capable of automatically factorising legacy applications over the whole claimed scope.

7.6 Therefore, auxiliary request II and modified auxiliary request II do not fulfil the requirements of Article 83 EPC.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chair:

S. Lichtenvort R. de Man

Decision electronically authenticated