Internal distribution code:

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

Datasheet for the decision of 8 November 2024

Case Number: T 2076/22 - 3.5.07

Application Number: 19175959.6

Publication Number: 3572952

IPC: G06F16/24

Language of the proceedings: EN

Title of invention:
UNIFIED OPTIMIZATION OF ITERATIVE ANALYTICAL QUERY PROCESSING

Applicant:
SAP SE

Headword:
Unified optimisation/SAP SE

Relevant legal provisions:
EPC Art. 83
RPBA 2020 Art. 12(2), 12(4), 12(6), 13(2)
EPC R. 116(2), 137(3)

Keyword:
Sufficiency of disclosure - main request, first, second, fourth and fifth auxiliary requests (no)
Admissibility - third auxiliary request (no)
Admissibility - fifth auxiliary request (yes)

Decisions cited:
G 0001/03, G 0001/19, T 0068/85, T 0922/17, T 2773/18

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

Case Number: T 2076/22 - 3.5.07

D E C I S I O N
of Technical Board of Appeal 3.5.07
of 8 November 2024

Appellant: (Applicant)	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf (DE)
Representative:	Müller-Boré & Partner Patentanwälte PartG mbB Friedenheimer Brücke 21 80639 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 26 April 2022 refusing European patent application No. 19175959.6 pursuant to Article 97(2) EPC**

Composition of the Board:

Chair	J. Geschwind
Members:	C. Barel-Faucheux
	M. Jaedicke

Summary of Facts and Submissions

I. The appellant (applicant) appealed against the examining division's decision refusing European patent application No. 19175959.6, published as European application EP 3572952 A1.

II. The examining division decided that the subject-matter of the independent claims of the main request did not meet the requirements of Article 83 EPC. The same objection applied to claim 1 of the first to third auxiliary requests. The examining division decided not to admit the fourth auxiliary request into the proceedings, since this request was filed late, during the oral proceedings before the examining division, and did not, *prima facie*, overcome the objections under Article 83 EPC (Rule 137(3) EPC).

The examining division cited *inter alia* the following documents in the decision:

D2 US 6 026 240 A, published on 15 February 2000

D4 Kisung Park et al., "Iterative Query Processing based on Unified Optimization Techniques", SIGMOD '19, pp. 54 to 68, 5 July 2019

III. With the statement of grounds of appeal, the appellant requested that the decision under appeal be set aside and that the application be granted on the basis of one of the main request, the first to second and fourth auxiliary requests, all requests being the subject of the decision and resubmitted with the statement of

grounds of appeal, or an amended third auxiliary request filed with the statement of grounds of appeal.

IV. In a communication annexed to the summons to oral proceedings, the board informed the appellant that it was of the preliminary opinion that the application did not meet the requirements of Article 83 EPC since it did not disclose the invention as defined in claim 1 of all the requests in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art. The board also stated that the examining division's decision not to admit the fourth auxiliary request suffered from an error in the use of discretion and the board considered the fourth auxiliary request to be admissible, while it was inclined not to admit the (amended) third auxiliary request into the appeal proceedings. The board was of the preliminary opinion that claim 1 of all the requests did not meet the requirements of Article 84 EPC.

V. In a letter dated 13 September 2024, the appellant filed a fifth auxiliary request and provided arguments. It also filed an excerpt from the following book:

Abhijit Dasgupta "Set Theory With an Introduction to Real Point Sets", Birkhäuser, 2014, front page, bibliographic page and pages 2 and 3

VI. In a phone consultation following a call by the appellant to the board's registry, the board informed the representative that it was preliminarily not convinced by the appellant's arguments regarding the objections under Article 83 EPC for all the requests, that the third auxiliary request appeared not to be admissible, and that the fifth auxiliary request would be admitted but did not appear to overcome the Article 83 EPC objections.

VII. Oral proceedings were held as scheduled and the appellant was heard on relevant issues. At the end of the oral proceedings, the Chair announced the decision.

VIII. 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 set of claims of the main request or of any of the first to fourth auxiliary requests, all requests as resubmitted or filed with the statement of grounds of appeal, or the fifth auxiliary request filed by letter of 13 September 2024.

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

A A computer-implemented method comprising:

B receiving an intermediate representation (120, 310, 510, 1010, 1310) of an input procedure (110) comprising a plurality of statements (815, 835, 915, 1715, 1735),

C wherein the plurality of statements (815, 835, 915, 1715, 1735) comprises a plurality of query statements, and a plurality of imperative statements comprising a loop;

D enumerating a plurality of query execution plan candidates (145, 330, 590) for the input procedure (110) via the intermediate representation (120, 310, 510, 1010, 1310) of the input procedure (110),

E wherein the enumerating comprises performing at least one sink operation on a query statement,

F wherein the at least one sink operation moves the query statement inside a loop boundary while preserving program correctness,
and

G wherein the enumerating comprises considering a plurality of query statements as sink operation candidates for a loop;

H based on a data dependency graph (530, 1030, 1330) representing at least one data dependency relationship for the plurality of statements (815, 835, 915, 1715, 1735), excluding at least one of the sink operation candidates from consideration;

I performing query inlining that combines at least two query statements while preserving program correctness,

J wherein performing query inlining comprises performing query inlining on at least one sinked query statement;

K estimating computing execution resource demands (370) for respective of the plurality of query execution plan candidates (145, 330, 590); and

L determining an optimal query execution plan (175, 390) for the input procedure (110),

M wherein the determining comprises finding a candidate query execution plan having a lowest estimated computing execution resource demand (370);

N compiling and executing the input procedure (110) according to the optimal execution plan (175, 390).

X. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that the following feature has been added between features H and I:

H1 generating a plurality of alternative query execution plan candidates (145, 330, 590), wherein the generating comprises performing various different permutations of sink operations on the initial query execution plan candidate (145, 330, 590) for different of the plurality of alternative query execution plan candidates (145, 330, 590);

XI. Claim 1 of the second auxiliary request differs from claim 1 of the first auxiliary request in that the following feature has been added between features H1 and I:

H2 building a sink subgraph (1390, 2000) for the loop from a data dependency graph (530, 1030, 1330) representing at least one data dependency relationship for the plurality of statements (815, 835, 915, 1715, 1735), wherein the sink subgraph (1390, 2000) represents sink dependencies between a plurality of statements in the loop;

XII. Claim 1 of the third auxiliary request differs from claim 1 of the second auxiliary request by the addition of the following two features:

H3 moving a query statement in the intermediate representation (120, 310, 510, 1010, 1310) into a different loop; (between feature H2 and feature I)

H4 , whereby the optimal query execution plan (175, 390) integrates query motion across iterative constructs (between features M and N but before the ";" character at the end of feature M)

XIII. Claim 1 of the fourth auxiliary request differs from claim 1 of the main request in that the text ", where possible while preserving program correctness" has been added at the end of feature J (before the ";" character).

XIV. Claim 1 of the fifth auxiliary request reads as follows (itemisation by the board, parts in *italic* were added with respect to claim 1 of the main request):

A A computer-implemented method comprising:

B' receiving an intermediate representation (120, 310, 510, 1010, 1310) of an input procedure (110), *the input procedure (110)* comprising a plurality of statements (815, 835, 915, 1715, 1735),

C' wherein the plurality of statements (815, 835, 915, 1715, 1735) comprises a plurality of *SQL* query statements, and a plurality of imperative statements comprising a loop, *and the intermediate representation (120, 310, 510, 1010, 1310) being a directed graph comprising nodes for basic blocks of the input procedure and comprising edges between the nodes representing control flow*;

D' enumerating a plurality of query execution plan candidates (145, 330, 590) for the input procedure (110) via the intermediate representation (120, 310, 510, 1010, 1310) of the input procedure (110),
wherein a query execution plan contains both operations directed to accomplishing processing for the plurality of SQL query statements and operations directed to accomplishing processing the plurality of imperative statements,

E' wherein the enumerating comprises performing at least one sink operation on a*n SQL* query statement,

F' wherein the at least one sink operation moves the *SQL* query statement inside a loop boundary while preserving program correctness, *wherein the program correctness is preserved when there is a statement inside the loop boundary that has dependency on the SQL query statement and when there are no statements outside the loop boundary that affect the results of the SQL query statement if it is moved inside the loop boundary,*
and

G' wherein the enumerating comprises considering a plurality of *SQL* query statements as sink operation candidates for a loop;

H' based *only* on *flow dependencies of* a data dependency graph (530, 1030, 1330) representing at least one data dependency relationship for the plurality of statements (815, 835, 915, 1715, 1735), excluding at least one of the sink operation candidates from consideration, *wherein a flow dependency is a read-after-write dependency*;

I' performing query inlining that combines at least two *SQL* query statements *having a flow dependency* while preserving program correctness,

J' wherein performing query inlining comprises performing query inlining on at least one sinked *SQL* query statement, *where possible while preserving program correctness, wherein the program correctness is preserved when the statement inside the loop boundary that has dependency on the at least one sinked SQL query statement is the only one depending on the at least one sinked SQL query statement*;

K estimating computing execution resource demands (370) for respective of the plurality of query execution plan candidates (145, 330, 590); and

L determining an optimal query execution plan (175, 390) for the input procedure (110),

M wherein the determining comprises finding a candidate query execution plan having a lowest estimated computing execution resource demand (370);

N compiling and executing the input procedure (110) according to the optimal execution plan (175, 390).

Reasons for the Decision

Application

1. The application relates to database processing, and more particularly to optimising procedures with query statements in iterative scenarios (paragraph [0001] of the application as filed).

Claim 1 of main request and Article 83 EPC

2. **Insufficient disclosure of the main request**

2.1 The claimed method is about optimising an input procedure comprising query statements and statements of an imperative programming language comprising a loop. The optimisation attempted by the method of claim 1 is to generate alternative execution plans by moving a query statement inside a loop in the input procedure, i.e. by performing a so-called "sink" operation. If an execution plan based on the sunk (SQL) statement has a lower estimated computing execution resource demand than other execution plans, this plan can be selected as the best plan by the method and therefore the query execution could potentially be optimised (provided that the cost estimates are sufficiently accurate). Evidently, moving an (SQL) statement inside a loop has to preserve program correctness, since otherwise the input procedure is not optimised but produces erroneous

results. As discussed in the following and as argued by the appellant, the claimed method relies on a data dependency graph to determine whether an (SQL) statement can be moved into a loop while preserving program correctness.

2.2 Paragraphs [0128], [0129] and [0131] of the description disclose (emphasis added by the board): "*At 1430, each SELECT statement S in the loop is checked, to identify whether there is a statement T that is outside the loop that S uses, either directly or indirectly. At 1440, it is determined for statement T identified in 1430 whether there are any interfering statements, and if not, then T is added to the set of sinkable statements S'.[…] In some cases, identifying a statement T outside the loop boundary that is used by at least one SELECT statement inside the loop boundary for which there are no interfering statements comprises determining that moving the statement T inside the loop boundary will not affect the operation of the statement, such as by changing its output as a result of the existence of one or more interfering statements (e.g., statements on which T also has dependence)*".

Therefore the description discloses that a criterion used to move the statement T inside the loop boundary, i.e. to sink it, is that there are no statements on which T has dependence or, in other words, no "interfering statements". In its letter of reply, the appellant referred to paragraph [0185] of the application as originally filed: "Data dependency refers to how the execution of a statement affects the execution result of another statement".

2.3 Moreover, the board notes that paragraph [0122] (together with Figure 12) enumerates two conditions for

a statement to be a "sinkable statement", i.e. a statement which can be sunk, "[i]n general", "without breaking program correctness":
- there is a statement inside the loop that has dependence on the statement outside the loop, and
- there are no "interfering statements" (i.e., other statements outside the loop that might affect the results of the statement outside the loop if it is moved before the consuming statement inside the loop).

Figure 13 illustrates an example system 1300 of identifying whether a given statement or set of statements is sinkable. A sink identifier 1350 can take as input a data dependency graph 1330 (DDG) and can use data dependencies to identify one or more sinkable statements. The sink identifier may enumerate and return as output one or more sink subgraphs 1390A-N reflecting different potential combinations of sinkable statements that may be sunk within the loop (paragraphs [0124] and [0126]).

2.4 In its communication, the board was of the preliminary opinion that the skilled person did not find sufficient information in the application as filed to determine all relevant data dependencies for arbitrary imperative code and arbitrary query statements.

For example, when one query updates a table that is read by another query, the set of data records updated and the set of data records read may or may not be non-overlapping, depending possibly on the run-time values of variables that are manipulated using imperative statements.

Difficulties in determining data dependencies could also arise for so-called "active" databases, where an

SQL statement can trigger database updates in addition to those specified by the SQL statement itself by means of triggers, for example.

2.5 In its letter of reply to the board's communication, the appellant argued that a person skilled in the field of computer science, in particular in programming and query optimisation, had sufficient knowledge to determine how the execution of an arbitrary statement affected the execution result of another arbitrary statement, i.e. whether there were data dependencies between two statements. Exemplarily, this could be achieved by analysing the code of the input procedure and/or by compiling and executing the code under different, controlled conditions to infer the dependencies.

2.6 In its letter of reply to the board's communication, the appellant also argued that the claimed plurality of statements, comprising both query statements and imperative statements, was part of an input procedure, which comprised a specific sequence of statements. It was therefore possible to establish whether reading was performed before or after updating the table, which determined the direction of the dependency between the reading statement and the updating statement. This was also discussed in paragraph [0203] of the application. Furthermore, even considering hypothetical cases in which it might not be possible to univocally determine dependencies before run-time, the DDG would simply contain all potential edges, in order to conservatively assess dependencies and thus avoid the risk of violating program correctness.

2.7 The board is not convinced by the appellant's arguments for the following reasons. It is not possible for the

skilled person to determine, over the whole scope of the claim and without undue burden, how the execution of an arbitrarily chosen SQL statement affects the execution result of another SQL statement.

For example, consider the following two SQL statements on a table named "parts":
1) UPDATE parts p SET p.price = p.price + 10 WHERE p.part_category = 'cars'
2) SELECT * FROM parts p WHERE p.part_key = input_variable

The first SQL statement updates the price for parts in the category cars, and the second selects the data for a part with a key value that is obtained from an input variable. Evidently, to determine whether the two statements have a data dependency it is necessary to know (1) the value of the input variable (which will in general only be known at run-time, not at compile-time) and (2) whether in the database there exists a part with a key having the value of the input variable which is in the category 'cars'. Since the contents of the database change over time, it is not possible to know whether or not condition (2) is true before run-time, even if one were to rely on testing. Even worse, the input variable could be further processed by imperative statements, for example. This would give rise to further difficulties in determining data dependencies.

Since it is in general necessary to take the actual database content at run-time into account, the appellant's argument that the skilled person could somehow determine the DDG manually by testing is not convincing either. Manual testing at run-time was not disclosed in any of the passages cited by the appellant, nor is the method of claim 1 limited to

query optimisation at run-time of the query or to the use of a manually-determined DDG. The cited paragraph [0203] also relies on the DDG ("*If the statements s_1 and s_2 are the select statements, a path exists between the two nodes corresponding to s_1 and s_2 in the data dependency graph.*").

2.8 Regarding the appellant's argument that it was possible to use a "conservative model" of dependencies, the board observes that claim 1 is not limited to any "conservatively" defined DDG. Consequently, this argument fails to overcome the Article 83 EPC objection.

2.9 During the oral proceedings, the appellant argued that the data dependency graph was not part of the method (see feature H: "*based on a data dependency graph [...] representing at least one data dependency relationship for the plurality of statements [...]*").

2.10 The appellant stated that it could be that it was not trivial to determine "all kinds of data dependencies for arbitrary imperative code and arbitrary query statements". Even if, for very specific cases, it might be difficult, that did not completely prevent the skilled person from determining the data dependencies. But this was a question relating to the generation of a DDG, which generation was not part of the claim. In the reply to the board's communication, the appellant had cited the following Wikipedia page, https://en.wikipedia.org/w/index.php?title=Dependency_graph&oldid=836892639, which is a version of the Wikipedia page published before the priority date of the application.

The concept of a DDG was well known to the skilled

person. The appellant referred as evidence to document D2, column 2, lines 38 to 45, which reads: "*For loops, the intermediate code representation generally includes data structures that either represent, or can be used to create, data dependency graphs (DDGs). DDGs embody the information required for an optimizer to determine which statements are dependent on other statements. The nodes in the graph represent statements in the loop and arcs represent the data dependencies between nodes*".

The appellant also cited column 5, line 66 to column 6, line 11, of document D2, which reads: "*A Data Dependency Graph (DDG) is a data structure in the computer memory that represents how statements within a loop depend on other statements. These graphs include nodes that represent computer operations and arcs that represent dependencies between the nodes. These dependencies include flow dependencies, data dependencies and anti-dependencies. Data structures within the compiler that represent data dependency graphs are used to determine loop-invariant and omega-invariant statements within a loop. These data structures are often represented by diagrams using circles for nodes corresponding to statements and arcs between the nodes representing dependencies. FIG. 4 illustrates an example DDG and is described below.*".

2.11 According to the appellant, the invention was built "on top of" a well-known DDG: using the DDG, it determined which statements were dependent on other statements. For example, the DDG could be 99% accurate. If there were some difficulties in determining the dependencies, this information could be provided in the DDG. The improvement relied on the optimiser itself. The DDG had to be correct and complete for the program to be

correct. Every optimiser had to apply the same trust in the correctness of the DDG. The DDG was a different representation of the "intermediate representation".

2.12 The board notes that paragraph [0085] of the application as originally filed states that the query execution plan enumerator 520 can generate a data dependency graph 530 representing dependencies between the plurality of statements. In this case, the "enumerating" step of feature D would (implicitly) comprise the generation of the DDG. Moreover, step 620 of Figure 6 of the application is a step of generating a data dependency graph (see also paragraphs [091], [0146] and [0236] of the description as originally filed). Moreover, paragraph [0163] reads: "*In order to find all sink subgraphs for a given IR, one can find all the loop body blocks by using in-order traversal and build a data dependency graph*".

2.13 Moreover, the claim wording does not exclude the DDG being generated by the method of claim 1. For example, claim 1 does not state that the DDG is received as a (manually generated) input. Rather, claim 1 also encompasses methods generating the DDG (see feature A: "method comprising" certain steps, not a method of consisting only of features B to N).

2.14 The board does not contest that the concept of a DDG was well known. The objection raised by the board was rather how this DDG can be practically obtained for an input procedure according to claim 1. In this regard, none of the cited passages of document D2 disclose a solution for determining a DDG in the context defined by claim 1.

2.15 During the oral proceedings, the appellant cited decision T 2773/18, reasons 3.2, which reads: "*[...] Whether claims, description and figures provide the skilled person with sufficient information to carry out an invention, is a purely technical question, that is separate from that of what reasonably falls within the ambit of claim wording. In the Board's view if the skilled person upon consideration of the entire disclosure possibly using common general knowledge can infer what will and what will not work, a claimed invention is sufficiently disclosed, even if a broad construction might also encompass what doesn't work [...]*".

2.16 According to the present claim 1, the sink operation is performed "while preserving program correctness" (see feature F). However, the board is not convinced that the program correctness is preserved when sinking SQL statements in all cases or for an incomplete or incorrect DDG. Nor did the appellant explain where the application as filed provides sufficient information to the skilled person on what will and what will not work or on what common general knowledge the skilled person could rely for this purpose.

2.17 In its letter of reply to the board's communication, the appellant argued that the feature "while preserving program correctness" was a functional feature and that, according to decision T 68/85 (see headnote), functional features were allowed if such features could not otherwise be defined more precisely without restricting the scope of the invention, and if these features provided instructions which were sufficiently clear for the expert to reduce them to practice without undue burden, if necessary with reasonable experiments.

The appellant argued that the feature limited the way the sink operation or query inlining could be performed in a manner that could be easily verified by the skilled person without undue burden, namely by running the program before and after the sink operation (or query inlining) and checking whether (1) the program could be run at all after the modification and (2) the results of the runs were the same.

2.18 The board is however of the opinion that neither the claim nor the application as originally filed enable the skilled person, without undue burden, to preserve program correctness when sinking a statement. Even if the skilled person were to run the program before and after each sink operation, this would constitute an "undue" burden in particular in view of the potentially very large number of combinations of possible sink operations. Furthermore, testing would not solve the problem of preserving program correctness since, at least in the circumstances of the case at hand, testing can only demonstrate the presence of an error resulting from a sink operation, but not verify that a sink operation preserves program correctness. For example, the results of testing depend on the database content at run-time (see above, point 2.7).

2.19 During the oral proceedings, the board pointed out that the inventor's post-published paper D4, section 2.3, second paragraph on page 57, reads: "*We assume that all statements have no hidden side-effects. In other words, all data dependencies through Read and Write performed by a statement are captured in the DDG.*".

2.20 Therefore the inventors at least indirectly acknowledged that it was difficult for a DDG to be

completely accurate under all circumstances.

2.21 In this context, the board also refers to decision G 1/03, Reasons 2.5.2, according to which the inclusion of non-working embodiments in claimed subject-matter is of no harm if there are a large number of conceivable alternatives and the specification contains sufficient information on the relevant criteria for finding appropriate alternatives over the claimed range with reasonable effort. However, if this is not the case and there is lack of reproducibility of the claimed invention and an effect is expressed in a claim, there is lack of sufficient disclosure.

In the present case, the claim expresses the effect of preserving program correctness and, in the board's view, it is not possible to find working embodiments, with reasonable effort, over substantial parts of the claimed range. This issue is even more important in view of point 84 of decision G 1/19, which states that a prerequisite for meeting the requirement that the claimed invention be inventive over the whole scope of the claim is that it also be technical over the whole scope. This implies that a computer-implemented invention specifying an effect that is a precondition for a credible technical effect also has to be sufficiently disclosed over the whole scope of the claim. The reason is that if there are non-working embodiments in a claimed subrange, then there can also be no technical effect linked to the claimed effect in that claimed subrange.

2.22 Therefore the application as filed does not disclose the invention as defined in claim 1 of the main request in a manner sufficiently clear and complete for it to

be carried out by a person skilled in the art (Article 83 EPC).

Claim 1 of first auxiliary request and Article 83 EPC

3. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that feature H1 (see point X. above) has been added between features H and I.

3.1 This amendment constitutes a further limitation of claim 1 of the main request, but is not related to the DDG. Therefore it does not overcome the above objection under Article 83 EPC against the main request.

Claim 1 of second auxiliary request and Article 83 EPC

4. Claim 1 of the second auxiliary request differs from claim 1 of the first auxiliary request in that feature H2 (see point XI. above) has been added between features H1 and I.

5. Since, in claim 1 of the second auxiliary request, a sink subgraph is built from the DDG, and since the board came to the conclusion that the crucial point was that the application does not enable the skilled person to generate the DDG over substantially the whole scope of the claim, the second auxiliary request cannot overcome the board's objection under Article 83 EPC against the main request.

Claim 1 of third auxiliary request - admissibility

6. Claim 1 of the third auxiliary request has been amended with respect to claim 1 of the previous third auxiliary request (considered in the decision under appeal) in

that the text "or moving a query statement in the intermediate representation (120, 310, 510, 1010, 1310) into a loop already having another query statement" after feature H3 has been deleted.

6.1 In the statement of grounds, the appellant stated that one alternative had been removed from the independent claim(s).

6.2 Article 12(4) RPBA stipulates that any part of a party's appeal case which does not meet the requirements in Article 12(2) RPBA is to be regarded as an amendment, unless the party demonstrates that this part was admissibly raised and maintained in the proceedings leading to the decision under appeal. Any such amendment may be admitted only at the discretion of the board.

The party shall clearly identify each amendment and provide reasons for submitting it in the appeal proceedings. In the case of an amendment to a patent application or patent, the party shall also indicate the basis for the amendment in the application as filed and provide reasons why the amendment overcomes the objections raised.

The board shall exercise its discretion in view of, *inter alia*, the complexity of the amendment, the suitability of the amendment to address the issues which led to the decision under appeal, and the need for procedural economy.

6.3 The appellant neither provided reasons for submitting the amendment to the third auxiliary request, or the (amended) third auxiliary request, in the appeal

proceedings, nor indicated the basis for the amendment in the application as filed.

6.4 In its letter of 13 September 2024 in reply to the board's communication, the appellant stated that the amendment was supported by paragraph [073] of the description as originally filed. Generally, the deletion of an alternative did not introduce added subject-matter, since by definition two alternatives could not be inextricably linked.

The appellant also stated that the amendment to claim 1 was a simple deletion of one of two alternatives. This amendment actually reduced the complexity of the claim, since only one alternative had to be considered instead of two.

Furthermore, the deletion of the alternative did not introduce any clarity issues and narrowed the scope of the claims and hence could not give rise to new novelty or inventive-step objections. Accordingly, no additional issues were raised by the amendment to claim 1.

Even with the deletion of the alternative "moving a query statement [...] into a loop already having another query statement", the third auxiliary request remained convergent with the second auxiliary request.

The appellant was of the view that, since almost all the criteria for admitting the amended third auxiliary request were satisfied, the board should admit this request. It was not a "fresh case" since the alternatives were on file before the examining division.

6.5 However, the appellant did not present arguments in favour of allowing the amended third auxiliary request with its statement of grounds of appeal. Moreover, in its letter of 13 September 2024, the appellant still only argued that the same arguments as presented for the main request also applied to the third auxiliary request. There is therefore a lack of specific arguments in favour of allowing the third auxiliary request.

6.6 Therefore the board does not admit the (amended) third auxiliary request into the appeal proceedings (Article 12(4) RPBA).

Claim 1 of fourth auxiliary request - admissibility

7. The board notes that the communication annexed to the summons to oral proceedings before the examining division raised the objections that claims 1 and 14 did not meet the requirements of Article 84 EPC in that the matter for which protection was sought was not defined. The claims attempted to define the subject-matter in terms of the result to be achieved. Moreover, the subject-matter of claims 1 to 14 did not involve an inventive step within the meaning of Article 56 EPC, and the requirements of Article 52(1) EPC were therefore not met.

8. The final date for filing written submissions in reply to the summons in the first-instance proceedings was 9 February 2022. In response to the summons, the appellant submitted a new main request and new first to third auxiliary requests. In a communication of 21 February 2022 the examining division informed the appellant of its objections under Article 56 EPC and Article 84 EPC against the newly-filed requests and

cited two prior-art documents. The fourth auxiliary request was submitted during the oral proceedings on 9 March 2022.

9. Since the late-filed request did not clearly resolve the objection under Article 83 EPC, the division used its discretion under Rule 137(3) EPC not to admit it into the procedure.

10. The appellant argued that, since the objection under Article 83 EPC had been raised for the first time after the summons to oral proceedings, and the introduction of a new, main ground for refusal represented a change of the subject of the proceedings, the request should have been admitted (Rule 116(2) EPC).

11. In view of the fact that the examining division had raised a completely fresh objection under Article 83 EPC during the oral proceedings (and even introduced new prior-art documents in the communication of 21 February 2022), in the special circumstances of the current case the examining division should have admitted the appellant's new claim request filed in response, i.e. the fourth auxiliary request, all the more so since the examining division could have raised at least some Article 83 EPC issues earlier.

12. According to Article 12(6) RPBA, the Board shall not admit requests, facts, objections or evidence which were not admitted in the proceedings leading to the decision under appeal unless the decision not to admit them suffered from an error in the use of discretion or unless the circumstances of the appeal case justify their admittance.

12.1 The board considers that the decision not to admit the fourth auxiliary request suffered from an error in the use of discretion. Therefore it considers the fourth auxiliary request to be admissible under Article 12(6) RPBA 2020.

Claim 1 of the fourth auxiliary request and Article 83 EPC

13. Claim 1 of the fourth auxiliary request differs from claim 1 of the main request in that the text ", where possible while preserving program correctness" has been added at the end of feature J (before the ";" character).

14. The board is not convinced that this amendment overcomes the objection under Article 83 EPC against the main request since it does not further define or limit the DDG and only further emphasises the need to preserve program correctness.

15. Feature I specifies that the query inlining combines at least two query statements "while preserving program correctness", while the amended feature J specifies that the query inlining is performed on at least one sunk query statement "where possible while preserving program correctness".

16. The application fails to disclose what conditions are necessary and/or when it is possible for the query inlining to be performed on at least one sunk query statement to preserve "program correctness".

17. Consequently, the application does not meet the requirements of Article 83 EPC since it does not disclose the invention as defined in claim 1 of the fourth auxiliary request in a manner sufficiently clear

and complete for it to be carried out by a person skilled in the art.

Claim 1 of the fifth auxiliary request - admissibility

18. In the communication accompanying the summons to oral proceedings, the board raised Article 84 EPC objections that had not been raised in the decision (even if some of them had been raised in communications of the examining division, but see T 0922/17, reasons 3.1), as well as new objections or remarks relating to Article 83 EPC.

18.1 The board considers that these circumstances are "exceptional" under Article 13(2) RPBA.

18.2 Therefore, the board admits the fifth auxiliary request into the proceedings, under Article 13(2) RPBA.

Claim 1 of fifth auxiliary request - Article 83 EPC

19. The amendments made in claim 1 of the fifth auxiliary request fail to overcome the Article 83 EPC objection concerning the DDG and raised against claim 1 of the main request (Article 83 EPC). In particular, the limitation of the claimed method to use flow dependencies of a DDG does not help, since the objection under point 2.7 also applies to flow dependencies.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chair:

S. Lichtenvort J. Geschwind

Decision electronically authenticated