Internal distribution code:

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

Datasheet for the decision of 1 February 2016

Case Number: T 1667/11 - 3.5.06

Application Number: 02076952.7

Publication Number: 1267263

IPC: G06F9/445, G06F17/30, G06F9/46

Language of the proceedings: EN

Title of invention:
Hot migration through incremental roll-over using migration plug-ins for conversion during upgrade

Applicant:
Océ-Technologies B.V.

Headword:
Hot migration/OCE

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step - (no)

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 1667/11 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 1 February 2016

Appellant: (Applicant)	Océ-Technologies B.V. St. Urbanusweg 43 5914 CC Venlo (NL)
Representative:	Vanoppen, Ronny R.J. Océ-Technologies B.V. Corporate Patents Postbus 101 5900 MA Venlo (NL)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 11 February 2011 refusing European patent application No. 02076952.7 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Sekretaruk
Members:	M. Müller
	S. Krischer

Summary of Facts and Submissions

I. The appeal lies against the decision of the examining division, with reasons dispatched on 11 February 2011 to refuse European patent application No. 02 076 952.7 for lack of an inventive step over the documents

D1: EP 0 738 980 A2 and

D3: Lautemann, S-E, "An Introduction to Schema Versioning in OODBMS", Proc. 7th International Conference and Workshop on Database and Expert Systems Applications, IEEE Press, 1996, pages 132-139, XP010200865.

A number of clarity objections were also raised in a section entitled "Obiter Dicta".

II. Notice of appeal was filed on 11 April 2011, the appeal fee being paid on the same day. A statement of grounds of appeal was received on 21 June 2011. The appellant requested that the decision be set aside and that a patent be granted based on claims 1-13 as filed with the grounds of appeal, the other application documents being description pages 1-10 and drawings sheets 1/7-7/7 as originally filed.

III. In an annex to a summons to oral proceedings, the board informed the appellant of its preliminary opinion, *inter alia* that claim 1 lacked an inventive step, Article 56 EPC 1973.

IV. In response to the summons, the appellant did not file either amendments or arguments. In a letter dated 18 December 2015 it only reiterated its belief "that the application is deemed allowable" and indicated that no

one would be present at the scheduled oral proceedings. Moreover, it withdrew its request for oral proceedings.

V. The board then cancelled the oral proceedings.

VI. Claim 1 reads as follows:

"A method for executing a hot migrate operation from a first version of a service (511) using a first data model (N), to a second version of the service (512) using a second data model (N+1) that is modified with respect to said first data model, the service comprising a client application, a data manager and a data repository, said migrate operation being effected on a server facility (SA, SB) that accommodates multiple processes to be running in parallel, said method comprising the steps of:

- installing second version client applications
- installing a second version data manager operating according to the said second data model, and
- installing a second version data repository arranged according to the said second data model and cooperating with the second version data manager;

characterized in that
- the second version data manager is provided with a first migration plug-in;
in that, the second version service is controlled to retrieve and remove data from the first version data repository, and thereupon to convert and store these data to the second version data repository, whereafter the second version data manager will retrieve converted data from the second version data repository;
in that the first version service is controlled to retrieve data from the first version data repository,

but if unsuccessful, to retrieve and convert data from the new version data repository without storing thereof, where said conversion takes place with a second migration plug-in coupled to the first data manager; and in that an incremental roll-over process is effectuated, wherein in successive steps data is converted from said first version data repository to said second version data repository by means of the first migration plug-in until all data will have been converted."

Reasons for the Decision

1. The following reasons are based on the board's analysis presented in the annex to the summons to oral proceedings, on which the appellant chose to to comment.

The invention

2. The application relates to the migration of databases to a new format - or, as the claims specify, to a new "data model" - without interrupting the operation of the database system (see Figures 1 and 2). The migration of both "user entities" (e.g. applications) and of data "entries" is disclosed as being executed 'in a "rolling-over" manner', so that the database continues to operate even when only some "user entities" have been "upgra-d[ed] or migrat[ed]" and even when the data migration is still incomplete (see description, page 1, last paragraph, page 2, first paragraph, and Figure 3). This is referred to as "hot migration". The central features of the claimed invention are illustrated in Figure 5 and on page 6 of the description. Data is held in data repositories (503, 507) and applications access data

repositories through "data managers" (501, 508). Before and after data migration from version N to version N+1, only one data manager and one data repository is needed (see Figure 5, situations A and C). Migration is performed in an incremental manner and is driven by new version N+1 applications, which operate on data in version N+1. When a version N+1 application accesses data which has not yet been migrated and is only available in version N, a so-called "migration plugin" converts the data into version N+1. The converted data is stored in the version N+1 data repository and removed from the version N data repository (see page 6, lines 8-14). On the other hand, when a version N application accesses data which has already been migrated, another migration plugin will convert version N+1 data into version N. Eventually, when all data is converted, the version N data manager, data repository and migration plugins can be dispensed with.

The prior art

3. D1 discloses a method for converting data on external storage from an old format to a new format which does not affect system availability (column 1, lines 3-6), i.e. a "hot data migration" technique. More specifically, the technique enables the co-existence of processors running code which operates on the data in the new format and processors running code which operates on data in the old format (column 2, lines 6-50). Processors are "migrated" one by one (column 4, lines 5-8; column 5, lines 45-55). The processor running "new" code will access the data stored in the old format and convert it into the new format for the new code. Computational results produced in the new format will be converted (back) into the old format and stored on the external storage (see column 4, lines 8-10; column 5,

line 55 - column 6, line 5). Only when "the new code is loaded on all of the processors" will the data on the external storage be converted and permanently stored in this form (see column 4, lines 10-13; column 6, line 8-11). Data accessed during conversion may be in the old format, the new format or "being converted" (see column 6, lines 12-14 and 49-52, and column 7, lines 11-31; Figure 3). If the new "code" accesses old data which is not yet converted, the conversion is performed "on the fly" for that access (*loc. cit.*). A corresponding "backward" translation "on the fly" of data in the new format for "old" code is not disclosed.

4. D3 discusses "schema versioning in OODBMS" (see title), i.e. the modification of database schemas (see page 134, left column, items 1-3) in object-oriented database management systems. D3 distinguishes between schema "evolution" and schema "versioning". Schema evolution is the conversion of the data and the code from from one schema version to another one (see Figure 1), whereas schema versioning (Figure 5) allows the coexistence of data (objects) in several versions at the same time. To avoid system down-time it is disclosed to use a "lazy mechanism" for schema evolution which converts an object from the old to the new schema only when it is actually accessed (page 133, left column, 2nd paragraph). It is then argued in D3 that schema evolution would require the instantaneous "evolution" of old applications as well (see *loc. cit.*, 3rd paragraph). Schema versioning is proposed as a solution to that problem. In the context of schema versioning, it is disclosed to use "conversion functions" between data in different formats, and it is expressly disclosed that conversions may be provided in both directions (see section 7.4 and Figure 5). By concatenating the conversion functions,

propagation between several different versions becomes possible (page 137, right column, last paragraph).

Inventive step

5. In the decision under appeal, D1 is used as the starting point for the assessment of inventive step.

5.1 The board agrees with the appellant that D1 does not disclose an incremental data conversion process during simultaneous presence of "old" and "new" code. As long as both old and new code is present, data is only temporarily converted into the new format and converted back for storage. That is, the "old" data in external storage is not permanently converted into the new format, so that the old data cannot be removed. Complete and final conversion of the data in external storage only starts once all the old code has disappeared. In both cases, there is no need to provide a conversion function - i.e. a "plugin" as claimed - for the old code.

5.2 Therefore, the board finds D3 a more suitable starting point for the assessment of inventive step.

6. In the board's view, the lazy conversion mechanism of D3 implies all the features of claim 1 except for the ones specified in claim 1, lines 20-23. In particular, the component performing the on-need conversion qualifies as a "migration plugin" to a "data manager" as claimed, taking account of the fact that no detail about either the "plugin" or the "data manager" is specified explicitly in the claims or implied by these terms alone. Moreover, the lazy mechanism is disclosed in D3 as an alternative to the conversion of a database as a whole. The board considers that the skilled person would understand conversion to be "destructive", i.e. that the

converted data would take the place of the original data, thereby removing the old version data. For completeness' sake, the board also notes that it considers the removal of old - and thus out-dated - data to be an immediately obvious option for the skilled person. The lazy mechanism of D3 therefore also constitutes an "incremental roll-over process" which terminates only once all the data has been converted.

7. The subject-matter of claim 1 thus differs from the disclosure of D3 in that

i) a distinction is made between first and second data repositories,

ii) the data manager of a first version client application initially tries to access the data in the first version repository, and

iii) the data manager, if data is not accessible from the first version repository because it has already been converted, uses a "second migration plug-in" to access (retrieve and convert) the data in the new format.

7.1 The board takes the view that it would have been obvious for the skilled person to store converted data elsewhere (i.e. in a "second version repository") rather than convert it in-place (difference i). In fact, this might even be necessary, e.g. if the new format required more space than the old format and the converted data did not fit into its original place.

7.2 D3 teaches that, for schema evolution to work, "applications have to be adapted to modified schemas" (page 133, left column, 3rd paragraph, lines 5-7), and states

that this may be "unacceptable" or impossible "simply for practical reasons or because the source code is not available" (lines 12-17). That it may often be "unacceptable", however, does not mean that it always is. Therefore, the board takes D3 to suggest that, in principle, adaptation of old applications to schema updates is an option, if a less preferable one.

7.3 In the board's understanding, difference iii) represents such an adaptation: the "second migration plugin" enables an old application to access converted data. Moreover, difference ii) modifies this adaptation to work with the lazy conversion mechanism.

7.4 The board thus finds that differences ii) and iii) solve the problem of adapting old applications to be compatible with the lazy conversion mechanism.

7.5 In the board's view, the solution provided by differences ii) and iii) was obvious. It would have been obvious for the skilled person to provide an old application with some suitable code for converting data in the new format back into the old format so that the old application could operate on data that had already been converted. It would further have been obvious to provide such code as a separate "module" or "plugin" to the old application.

7.6 In passing, the board notes that D3 already discloses the necessary conversion functions, albeit in the context of "schema versioning" (see Figure 5). Therefore, the board cannot see that the skilled person would have to exercise an inventive step in order to provide the claimed "migration plugins".

7.7 The board therefore concludes that the subject-matter of claim 1 lacks inventive step over D3 (Article 56 EPC 1973).

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

L.Malécot-Grob W. Sekretaruk

Decision electronically authenticated