

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

SERVICENOW, INCORPORATED,  
Petitioners

v.

INQUISIENT INCORPORATED,  
Patent Owner

---

Case IPR2023-01072  
Patent 8,219,585

---

**PATENT OWNER'S PRELIMINARY RESPONSE**

## TABLE OF CONTENTS

|      |                                                                                                                                                                                            |    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| I.   | INTRODUCTION .....                                                                                                                                                                         | 1  |
| II.  | SUMMARY OF THE '585 PATENT .....                                                                                                                                                           | 5  |
| III. | SUMMARY OF THE CITED REFERENCES.....                                                                                                                                                       | 11 |
| A.   | Althoff (EX1006) .....                                                                                                                                                                     | 11 |
| B.   | Moore (EX1007) .....                                                                                                                                                                       | 16 |
| IV.  | CLAIM CONSTRUCTION .....                                                                                                                                                                   | 18 |
| A.   | Module.....                                                                                                                                                                                | 18 |
| V.   | INSTITUTION SHOULD BE DENIED UNDER 37 C.F.R. § 42.104(B)(3)<br>BECAUSE THE PETITION DOES NOT APPLY WHAT PETITIONER<br>CONTENDS TO BE THE PROPER CONSTRUCTION OF THE TERM<br>“MODULE” ..... | 19 |
| VI.  | GROUND 1 BASED ON ALTHOFF IN VIEW OF MOORE IS<br>DEFICIENT (CLAIMS 1, 4-11, 14-17, 20-27, 30-32, 33, 36-43, 46-47) ....                                                                    | 23 |
| A.   | The Petition Fails to Identify a “Class Module” And Its Methods That<br>Defines and Stores At Least One Class in The Data Repository<br>(Limitations 1[e], 17[c], 33[c]) .....             | 24 |
| 1.   | Althoff’s object-oriented classes .....                                                                                                                                                    | 26 |
| 2.   | “class Class 301” is a class (a data type) and not an object in the<br>user’s object database .....                                                                                        | 28 |
| 3.   | Althoff’s database translator translates each of the user’s class<br>objects into relational database tables, but “class Class 301” is<br>not translated .....                             | 29 |
| 4.   | Table 3-2 in the specification should not be confused with the<br>alleged “class Class 301 table” .....                                                                                    | 35 |
| 5.   | The alleged “class module” and cited class definitions in<br>Althoff are in the wrong place .....                                                                                          | 36 |

|       |                                                                                                                                                                                                               |    |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| B.    | The Alleged “Attribute Module” And “Class Attribute Module” In Althoff (And Their Corresponding Methods) Are Not in The Data Repository As Claimed (Limitations 1[f], 1[g], 17[d], 17[e], 33[d], 33[e]) ..... | 37 |
| C.    | Moore Does Not Disclose the Named Text Module or Methods (Limitations 1[i], 17[g], 33[g]).....                                                                                                                | 39 |
| D.    | Moore Does Not Disclose the Tuple Module or Methods That Generate a Hierarchy as Claimed (Limitations 1[j], 17[h], 33[h]) .....                                                                               | 43 |
| E.    | The Dependent Claims .....                                                                                                                                                                                    | 47 |
| VII.  | GROUND 2 BASED ON ALTHOFF IN VIEW OF MOORE IN FURTHER VIEW OF NYE IS DEFICIENT (CLAIMS 2-3, 18-19, 34-35) .....                                                                                               | 48 |
| VIII. | GROUNDS 3 BASED ON ALTHOFF IN VIEW OF MOORE IN FURTHER VIEW OF SURJANTO IS DEFICIENT (CLAIMS 12-13, 28-29, 44-45).....                                                                                        | 48 |
| IX.   | SOTERA STIPULATION.....                                                                                                                                                                                       | 48 |
| X.    | CONCLUSION.....                                                                                                                                                                                               | 48 |

## I. INTRODUCTION

The Board should deny institution of the Petition for *Inter Partes* Review of U.S. Patent No. 8,219,585 (“the ’585 Patent”) because the ’585 Patent claims a novel and non-obvious arrangement of modules and methods for relational database systems absent from the prior art. The Petition’s attempt to gloss over material gaps in the prior art should be rejected.

The ’585 Patent, entitled “Database Data Dictionary,” is a continuation of U.S. Patent No. 7,979,468, which is the subject of a co-pending IPR petition (IPR2023-01071). Both patents concern an extension to a relational database platform with a unique set of modules that permit database creators to configure a relational database in a manner that is independent of the data model of the system and to manage workflows without needing to write dedicated code. Rather than having to manually define tables and their relations, a database creator enters the database information into predefined modules (table-based structures), leading to an improved experience.

The ’585 Patent explains that in the prior art, a “conventional data storage system implements its own data model according to the user interface and the business rules specification for that system.” ’585 Patent, 1:22-24. “Such systems depend upon a specific data model and lack flexibility and portability.” *Id.*, 1:28-29. For example, if a database creator wished to design a database, the designer

would build it from scratch—deciding what tables and their relationships would be optimal. Any changes to the table organization could be highly problematic and require a reworking of the data model.

The '585 Patent provides a novel and non-obvious modification to prior art relational databases. For example, claim 1 recites a system that contains eight specific database modules: an “element module,” an “element relation module,” a “class module,” an “attribute module,” a “class attribute module,” a “type definition module,” a “named text module,” and a “tuple module.” The claim further recites details concerning each of these modules and how they interact. As explained below, the modules comprise database tables or columns of database tables. This same set of modules is used for each database the database designer wishes to design, thus simplifying the database creation process.

Nothing comparable to this is disclosed in the prior art of record.

The Petition goes awry from the first sentence in the '585 Patent Overview, where Petitioner asserts that the patent “is directed to object-oriented (‘OO’) data modeling.” Petition, 1. This is incorrect. The term “object-oriented” does not even appear in the '585 Patent. The Petition takes this approach to gloss over the many details captured in the claims that distinguish the invention over the prior art. For example, the Petition briefly discusses Figure 2 of the '585 Patent, but this figure is only a subset of an embodiment of the invention, and thus referencing that

figure does not give an accurate portrayal of the invention claimed in the '585 Patent. Petition, 2. A more useful figure that actually contains an embodiment of the claims is found in Figures 1A and 1B. Figures 1A and 1B include, among other claimed modules, an “element table 160” (an embodiment of the element module), “named text table 140” (an embodiment of the “named text module”), and “tuple table 144” (an embodiment of the “tuple module”). Each of these is missing from Figure 2.

Moreover, the Petition asserts that “[t]he '585 patent discloses basic OO concepts, *e.g.* objects, classes, attributes, and relationships.” Petition, 1. However, the Petition ignores the fact that the claims do not purport to cover “basic OO concepts,” but rather cover specific arrangements of modules and methods. The claimed modules and methods are not “basic OO concepts.”

This Reply lays out five independent reasons why this proceeding should not be instituted.<sup>1</sup> The first is procedural and the remaining are substantive. First, the Petition is deficient because a petition must present arguments based on what the

---

<sup>1</sup> Given the early stage of the proceedings, the word limits, and the need to read the evidence in the light most favorable to the Petitioner, Patent Owner does not provide arguments for each missing limitation, but rather highlights a few deficiencies in the Petition.

Petitioner contends to be the correct claim construction. Petitioners may not waste the Board's time with arguments based on a broad claim construction Petitioner believes to be incorrect. Here, the Petition relies on a construction of the term "module" that the Petitioner stated at the time of the filing of the Petition was incorrect. *Compare* Petition, 4 with Ex. 1011, 2. Petitioner further alleges that the construction used in the Petition is allegedly Patent Owner's construction. It is not. It is simply an incorrect claim construction. As such, institution can be denied on this basis alone.

Second, the Petition fails to locate the "class module" of the claims in the prior art. In fact, Althoff generates a relational database that has no need for a "class module" and Althoff in fact does not generate such a class module or provide its functionality in the relational database. The Petition relies on a non-existent "class Class 301 table" and is thus deficient.

Third, the Petition attempts to identify an alleged "attribute module" and an alleged "class attribute module" in Althoff, but the structures identified are outside the data repository. Because the claims require these modules and methods to be in the data repository, the Petition's mapping fails.

Fourth, the Petition relies on Moore for the named text module and its functionality. The Petition suggests a few alternate mappings, but each fails to meet the claim limitations. The functionality identified is largely outside the data

repository.

Fifth, the Petition relies on Moore for the tuple module and its functionality. However, the claims require a “link” that is used to “generate a hierarchy.” The Petition identifies an alleged link, and identifies an alleged hierarchy, but does not show that the identified link is used to generate the hierarchy.

Accordingly, for each of these reasons, the Petition is deficient and should not be instituted.

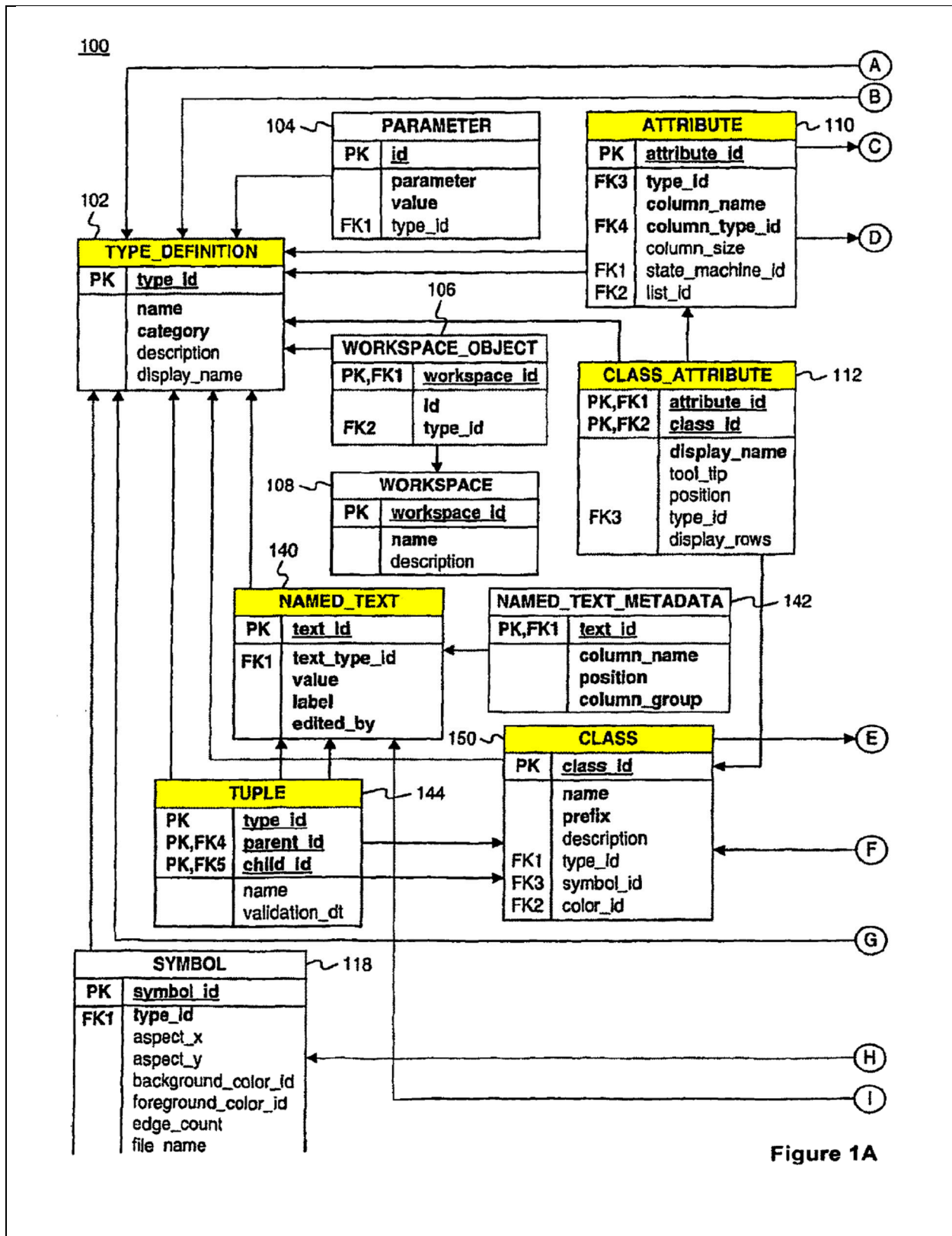
## **II. SUMMARY OF THE '585 PATENT**

The '585 Patent is directed to a novel extension to relational databases. The novel construction and configuration of the '585 Patent concerns the various interrelated modules that work together to provide an improved way to work with data sets and workflows. '585 Patent, 1:66-2:59. The modules are table-based structures and may be implemented as one or more tables or columns of tables. One benefit of the invention is that the user can develop a database structure without concerning his or herself with the data model needed to implement the system. Rather, the user just needs to add the data to the pre-defined table structures (the modules) defined in the '585 Patent. Another benefit is that a user may easily store text information within the data repository to manage queries and generate data hierarchies. '585 Patent, 1:22-29, 1:46-2:59, 10:11-50.

A relational database is a type of database that stores and provides access to

data, which the database maintains in tables having rows and columns. In a relational database, one or more columns of a table may serve as an index, and the indices may be used to create relationships among the tables. The specification of the '585 Patent uses relational databases in a new way and discloses an embodiment in which various interrelated modules are implemented as table-based structures to provide increased flexibility, portability, and ease of use. '585 Patent, 3:3-6, 3:55-5:22.

Figures 1A and 1B illustrate one embodiment of the invention, which contains the data structures corresponding to the eight claimed modules of claim 1 and others too. In the reproduced figures below, the eight modules required by claim 1 (implemented as individual tables in this embodiment) are highlighted:



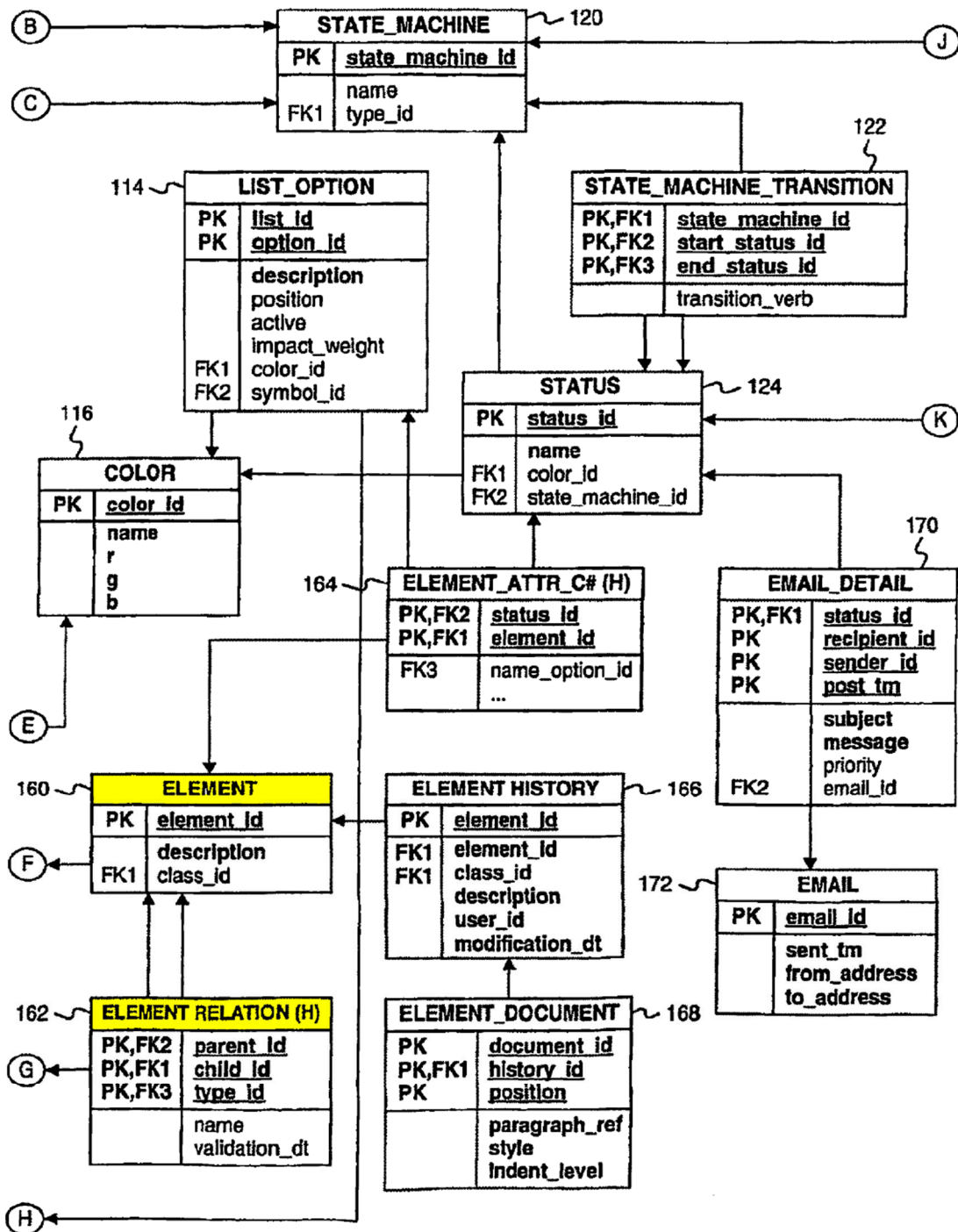


Figure 1B

In prior art relational databases, the database creator would have to manually design a table structure and table relations within a relational database system. The '585 Patent provides a new way to work with a relational database. Instead of having to manually design a table structure, the database creator using the invention would enter the database information into the specified modules. '585, 3:55-4:55, 6:3-45. For example, the database creator would decide which of the data items are the items of interest and enter those into the class module. These could be, for example, the employees and projects. '585 Patent, 4:20-21, 8:4-10. If the database creator decided at a later date to track company departments or maybe assigned company cars, those too could be added to the class module.

The database creator would then define the attributes of interest and list those in the attribute module. These might be "employee name," "project name," "employee birthday," and "project duration," among others. The database creator would add the individual employee and project information into the element module. The database creator would continue this process and enter information into the remaining modules. For example, if employees are to be assigned to projects, then that information would be entered into the element relation module. If the database creator decided at a later date to track employee telephone numbers, employee addresses, or project leads, that information can be added as attributes to the attribute module. '585 Patent, 5:23-43.

In addition, to customize the display of information, the invention can store SQL queries in a “named text” module within the relational database, thus simplifying the management and organization of the data. “Unlike a conventional database, e.g., Oracle® or Microsoft SQL Server®, that stores user-generated SQL as a ‘view,’ the EMR stores user-generated SQL as a named text entry.” ’585 Patent, 11:3-6. One benefit of this approach is that the content of the relational database can be dynamic content. ’585 Patent, 11:11-25. For example, the database can track dynamic tasks by storing SQL text in the database instead of just the tasks themselves. After a task is assigned, an “administrator may assign a task to someone else, so the task may be deleted or moved to a completed state and the user’s folder would only contain a current task set that matches the query as defined by a named text of that folder. This allows for an extensible method of organizing data of interest and allowing the user to easily find and retrieve data of interest in a large, complex system.” ’585 Patent, 11:32-39.

A database designer’s task is thus simplified and enhanced by using the modules defined in the ’585 Patent to create the desired database rather than the prior art relational databases.

\* \* \*

The Petition ignores these details and declares that the patent “is directed to object-oriented (‘OO’) data modeling, known in application software development

since the 80s.” Petition, 1. The Petition misreads the ’585 Patent, which is directed to relational databases and not object-oriented data modeling. To be sure, the ’585 patent uses terms that are also found in object-oriented programming, such as the terms “class” and “attribute.” However, the class and attribute modules are not object-oriented structures, but rather are relational database structures. That is, they are table-based structures.

Thus, the ’585 Patent uses a combination of relational database modules to simplify the process of designing databases. A user need not manually design a complex table structure. Instead, the user can enter the categories of information to be tracked into the class module, create attributes and their properties and store that information in the attribute module, and store the individual elements in the element module. The user can further control how data is retrieved and presented to the user by storing text strings, such as SQL text, in the named text module. In one exemplary embodiment the storage of text strings in the data repository is used to generate a hierarchy of one or more data entries, such as a dynamic folder hierarchy. ’585 Patent, 11:3-39.

### **III. SUMMARY OF THE CITED REFERENCES**

#### **A. Althoff (EX1006)**

Althoff discloses a method and system for translation between an object-oriented database and a relational database. Thus, Althoff has little relevance to

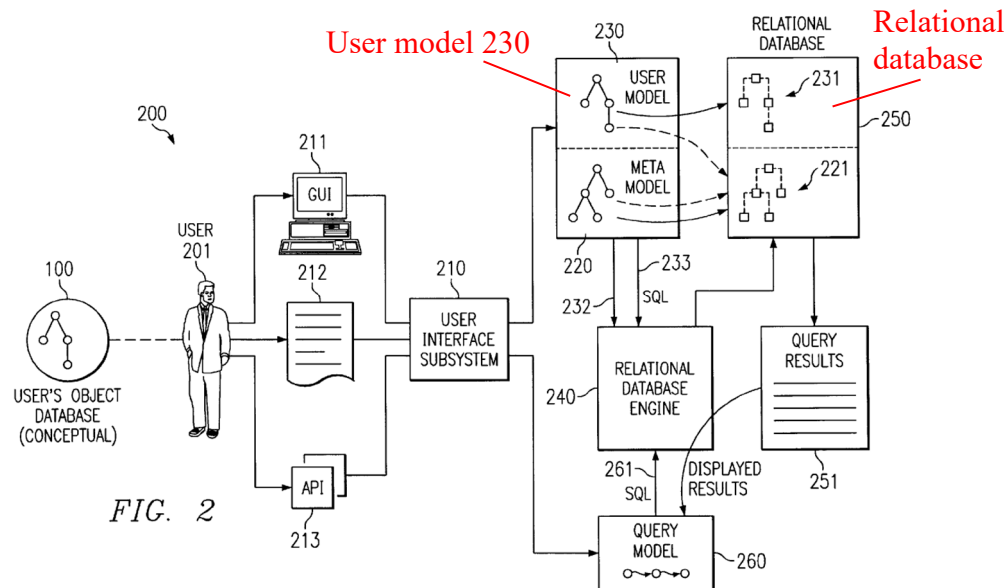
the '585 Patent and is directed to a different problem. Moreover, Althoff's disclosure of the details of its relational database is thin, leaving many gaps that the Petition cannot fill.

Althoff states that “object-oriented database” or “OODB” applications have grown in prominence. However, Althoff explains that “one problem which has arisen in the art is that many database systems have been designed to operate with **another, different** database technique, known as ‘relational database’ management (RDBM).” Ex. 1006 (Althoff), 1:41-57.<sup>2</sup> Using the system of Althoff, a user “may create, edit, and manipulate a user’s object database” and the user’s object database is “dynamically translated into a set of relational database structures.” *Id.*, Abstract.

Figure 2 of Althoff clearly distinguishes between the two databases within Althoff's system. On the one hand there is “user model 230,” which contains the user’s object database. On the other hand, there is the “relational database 231,” which is created by the system via the database translation process.

---

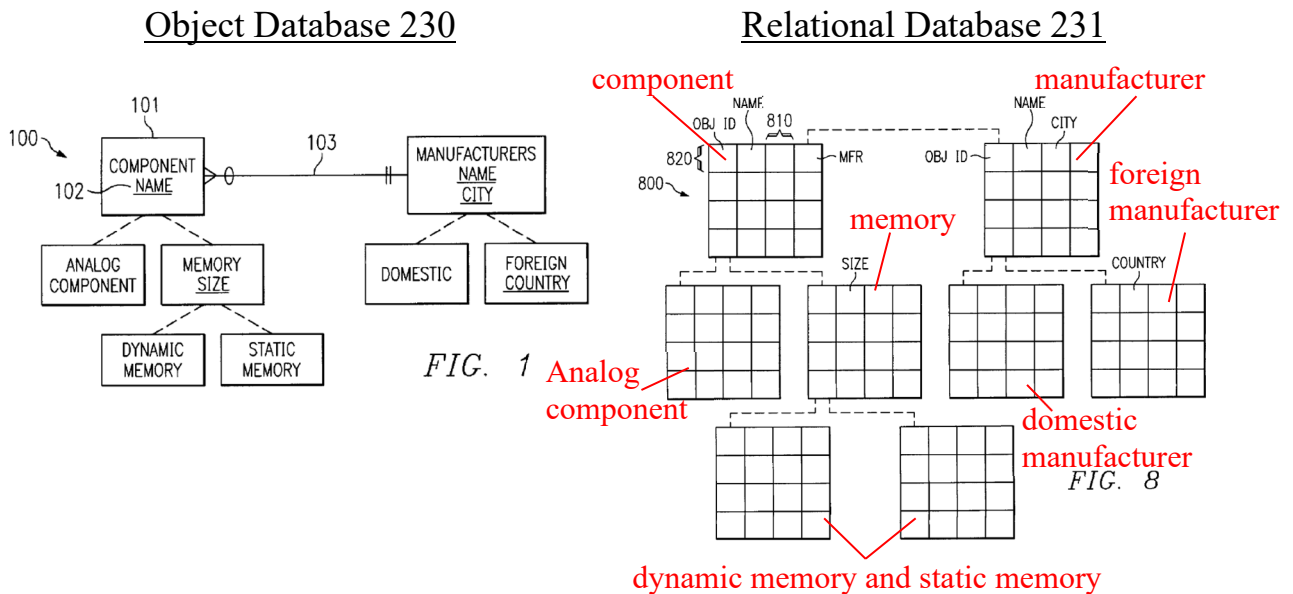
<sup>2</sup> Emphasis added throughout unless otherwise noted.



Ex. 1006, Fig. 2. “In response to a triggering event, the System 200 translates the user model 230 into a user relational database 231, using a set of SQL commands 232 for specifying and creating relational structures in the user relational database 231.” Ex. 1006, 8:18-21.

Althoff provides an example of the translation from one database to another in Figures 1 and 8. Figure 1 depicts an object database stored in an object-oriented model. Each box in Figure 1 represents a class (and each class definition is stored as a distinct object). The database would be further populated with additional objects that would be instances of one of these classes. For example, if “Acme Corporation” was added to the object database as a domestic manufacturer, an object of class “domestic” would be created that holds the relevant information concerning Acme Corporation.

Figure 8 depicts a relational database created by the database translator from the object-oriented database shown in Figure 1. *Id.*, 4:58-59, 5:8-10.



As illustrated above, each class created in the object database has a corresponding table in the relational database. Although Figure 8 does not name each table, it is possible to identify tables by their columns. For example, the component table has a “name” column which corresponds to the “name” property in the object database and the manufacturer table has “name” and “city” columns which correspond to the “name” and “city” properties in the object database.

The component and manufacturer tables are related using “keys” as was standard in relational databases. The component table has a column to hold foreign keys to match rows in the manufacturer table, allowing each component to be associated with a manufacturer. This column is labeled “MFR,” presumably to

refer to “manufacturer.” Ex. 1006, Fig. 8.

Althoff Figure 8 further illustrates via the vertical dotted lines the “inheritance relationships” between related classes. Ex. 1006, 3:61-63. Althoff explains that an object may be broken apart with pieces stored in different relational tables: “An object member of the derived class is entered in both the table for the base class and the table for the derived class, and it is given the same object-ID in both tables.” Ex. 1006, 3:64-67. For example, a static memory component would have a row in each of the static memory table, the memory table, and the component table. Each table would have a row identified by the same object ID (because each row concerns the same object). Likewise, a domestic manufacturer would have a row in the domestic manufacturer table and a row in the manufacturer table. Each of these two tables would have a row with the same object ID. *Id.*

To retrieve all the properties related to a static memory component, for example, one must read the rows with the relevant object-ID from multiple tables. That is, one must extract and combine the information from the static memory table, the memory table, and the component table. From those three tables, the static memory object may presumably be recreated (although these details are absent from Althoff).

Taking the earlier example of the “Acme Corporation” object, the object

database would contain an object called “Acme Corporation” that would contain all the information relevant to “Acme Corporation.” However, when that object is translated for insertion into the relational database, rows will be generated in the “manufacturer” table, including the name and city for Acme Corporation and additional rows will be generated in the “domestic” table, which would include additional information specific to domestic corporations and not relevant to foreign corporations. *See* Ex. 1006, 3:64-67.

**B. Moore (EX1007)**

Moore is a Microsoft patent that provides “a system and method for filtering and organizing items from computer memories.” Ex. 1007, Abstract. “The system utilizes virtual folders. The virtual folders expose regular files and folders to users in different views based on their metadata instead of actual physical underlying file system structure on the disk.” *Id.*

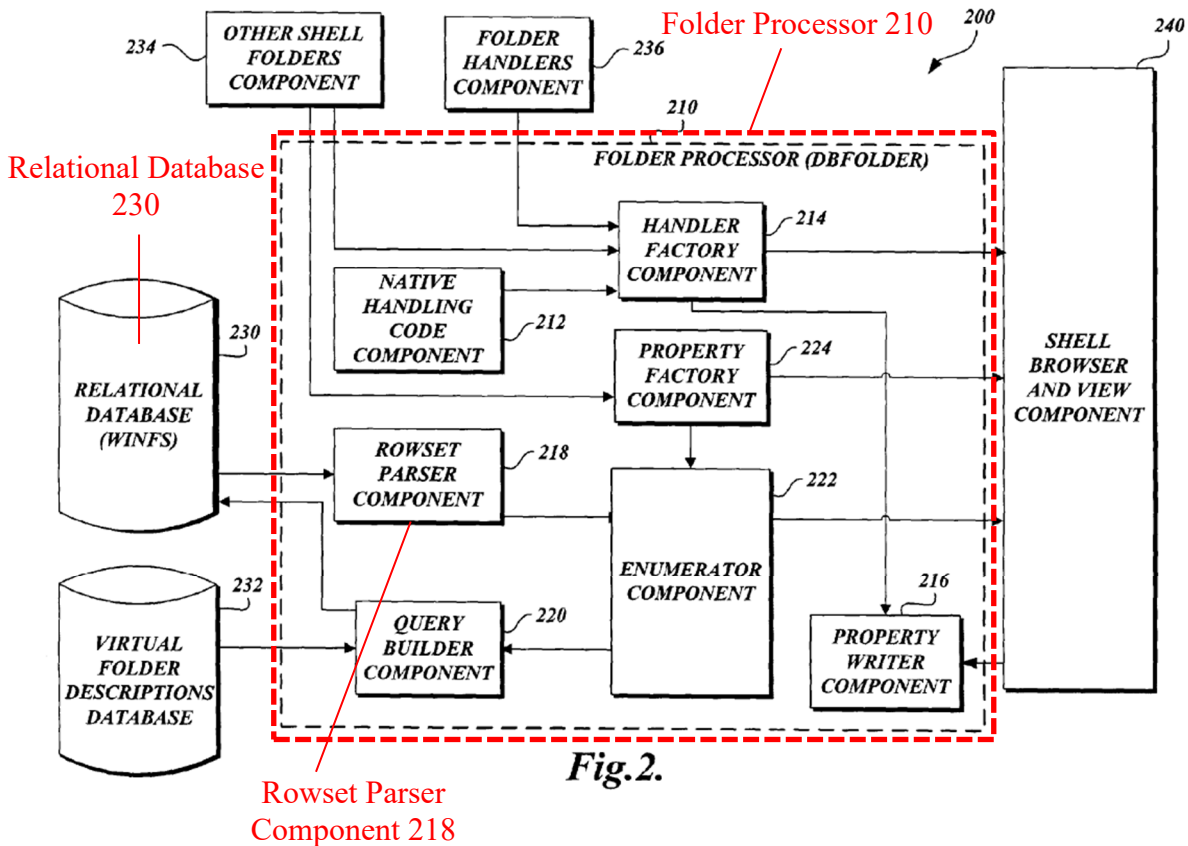
To accomplish the task of presenting a user with virtual folders instead of file system folders, Moore provides a “folder processor” to convert the file system folder structure into a virtual folder structure. Ex. 1007, 10:5-12. The “folder processor” accesses two database to carry out its task: “relational database 230” and “virtual folder descriptions database 232.” Ex. 1007, 8:5-14, FIG. 2. In Moore, the “relational database 230 stores properties about all files in the system.” Ex. 1007, 8:15-16. In response to a query, the relational database sends the

property information to “rowset parser component 218” inside the folder processor.

Ex. 1007, 8:20-22, FIG. 2. The “rowset parser component 218” takes the property information and transforms the data “into an ID list” which is “a well-known shell structure which is used to reference items in a namespace.” Ex. 1007, 9:30-36.

Moore explains: “Doing this allows virtual folders to be just like any other namespace to the rest of the shell. Also, caching this data helps keep database access, which can be costly, to a minimum.” *Id.* “Thus, the system is able to take a property that is stored in the database and represent it as a container that is like a folder.” Ex. 1007, 3:39-41. As shown in Figure 2, the rowset parser component 218 is part of the “folder processor 210” and is not part of the relational database 230.

There is no suggestion in Moore that relational database 230 is anything other than a prior art well-known relational database without virtual folder capabilities.



Ex. 1007, FIG. 2.

#### IV. CLAIM CONSTRUCTION

##### A. Module

Contrary to Petitioner's arguments, the claim term "module" should be treated no differently than the other claim terms. Namely, it should be understood according to its "ordinary and customary meaning as it would have been understood by a POSITA at the time of the alleged invention, in the context of the '585 Patent and its prosecution history." *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc)." See Petition, 3.

In the context of the '585 Patent, which concerns relational databases, the term “module” would be understood to be a table-based structure. That is, a module comprises one or more columns within one or more tables. Accordingly, Patent Owner submits that no construction is necessary, but to the extent that the Board issues a formal claim construction, Patent Owner offers the following:

The term “module” within the context of the '585 Patent means “a table or, one or more columns of/within one or more tables.”

However, the PTAB need not resolve this potential dispute in this proceeding because the Petition **is deficient even under the construction stated in the Petition.**

**V. INSTITUTION SHOULD BE DENIED UNDER  
37 C.F.R. § 42.104(B)(3) BECAUSE THE PETITION DOES NOT  
APPLY WHAT PETITIONER CONTENDS TO BE THE PROPER  
CONSTRUCTION OF THE TERM “MODULE”**

The Petition first states that a person of ordinary skill in the art “would understand ‘module’ to be a portion of a data repository.” Petition, 4. Next, the Petition asserts that Patent Owner defined “module” to mean “a table or a portion of a table” and then states: “Petitioner applies **Patent Owner’s** own definition of ‘module’ throughout this Petition.” Petition, 4.<sup>3</sup>

---

<sup>3</sup> Emphasis added throughout unless otherwise stated.

Petitioner is, in fact, incorrect, as Patent Owner contends that the term “module” should take on its ordinary meaning in the context of the ’585 Patent, which is a table or, one or more columns of/within one or more tables.

This dispute need not be resolved here because the Petition is deficient under either construction, and because Patent Owner will apply the construction offered in the Petition in this Preliminary Response. *See Realtime Data, LLC v. Iancu*, 912 F.3d 1368, 1375 (Fed. Cir. 2019) (“The Board is required to construe ‘only those terms . . . that are in controversy, and only to the extent necessary to resolve the controversy.’”).

As a threshold matter, however, the Board need not reach the substance of the dispute because the Petition fails to comply with the basic requirements under the procedural rules. The Petition must set forth “how the challenged claim is to be construed” and “how the construed claim is unpatentable.” 37 C.F.R. § 42.104(b)(3)-(4). If the Petition asserts unpatentability solely under a construction the Petitioner believes is incorrect, then the Petition is deficient and it should not be instituted. *Toyota Motor Corp. v. Blitzsafe Tex., LLC*, IPR2016-00422, Paper 12, 26 (P.T.A.B. Jul. 6, 2016) (“The ‘construction’ referred to by 37 C.F.R. § 42.104(b)(3) is the construction proposed by the Petitioner, one that Petitioner **believes is the correct construction** under applicable law and should apply in the involved proceeding . . . .”) (denying petition); *see also Carefusion Corp. v. Baxter*

*Int'l, Inc.*, IPR2016-01456, Paper 9, 7 (P.T.A.B. Feb. 6, 2017) (“Petitioner states that it relies on constructions advanced by Patent Owner in the related litigation; remarkably, however, Petitioner also states that it does not agree that those constructions are in fact correct.”) (denying petition).

Here, Petitioner does not assert that the construction of “module” used in the Petition is correct. In fact, Petitioner is clear to point out that the construction it applies is allegedly one offered by Patent Owner, and not one that it accepts as correct. Petition, 4. Although not discussed in the Petition, Exhibit 1011 provides what Patent Owner contends is the correct construction, and that construction is narrower than the construction relied upon in the Petition. In the Petition, Petitioner relies on multiple tables from Althoff (and sometimes no table at all) to identify what it contends is a “module.” Petition, 19 (relying on “class Class Link 302” as a module), 25 (relying on “class Pointer Valid Class 305” as a module), 28 (relying on a database and folder processor “together” as a module), 35 (relying on a “value” as a module). However, in the District Court, Petitioner contended (and continues to contend) that a module must be of/within a **single** table.<sup>4</sup> The single

---

<sup>4</sup> Ex. 1011 contains Petitioner’s construction before the District Court, which is: “a table or, one or more columns of/within a single table.” The Petition does not use this definition and does not explain why it chose to use a different definition in the

table requirement is incorrect, but nonetheless, on the basis of Petitioner's inconsistency alone, the Petition should be denied.

Indeed, since the revised § 42.100(b) has taken effect, the Board has repeatedly declined institution where a petitioner, like Petitioner here, maintains inconsistent claim construction positions. *See Orthopediatrics Corp. v. K2M Inc.*, IPR2018-01546, Paper 10, 10-12 (P.T.A.B. Feb. 14, 2019) ("Petitioner's failure to provide a claim construction is . . . further compounded by the fact that Petitioner takes an inconsistent position before the District Court . . . by failing to reconcile its proffered claim construction here with its very different construction proffered in District Court . . . Petitioner fails to satisfy [its] burden") (denying institution); *see also Samsung Elecs. Co., Ltd. v. Ancora Techs., Inc.*, IPR2020-01184, Paper 11, 27 (P.T.A.B. Jan. 5, 2021) ("Significantly, Petitioner's [claim construction] argument . . . seems to be inconsistent with its position advanced in the parallel litigation and the Federal Circuit's interpretation . . .") (denying institution). Thus, a petitioner may not take inconsistent claim construction positions between district court and IPR proceedings without justification, and by failing to address those

---

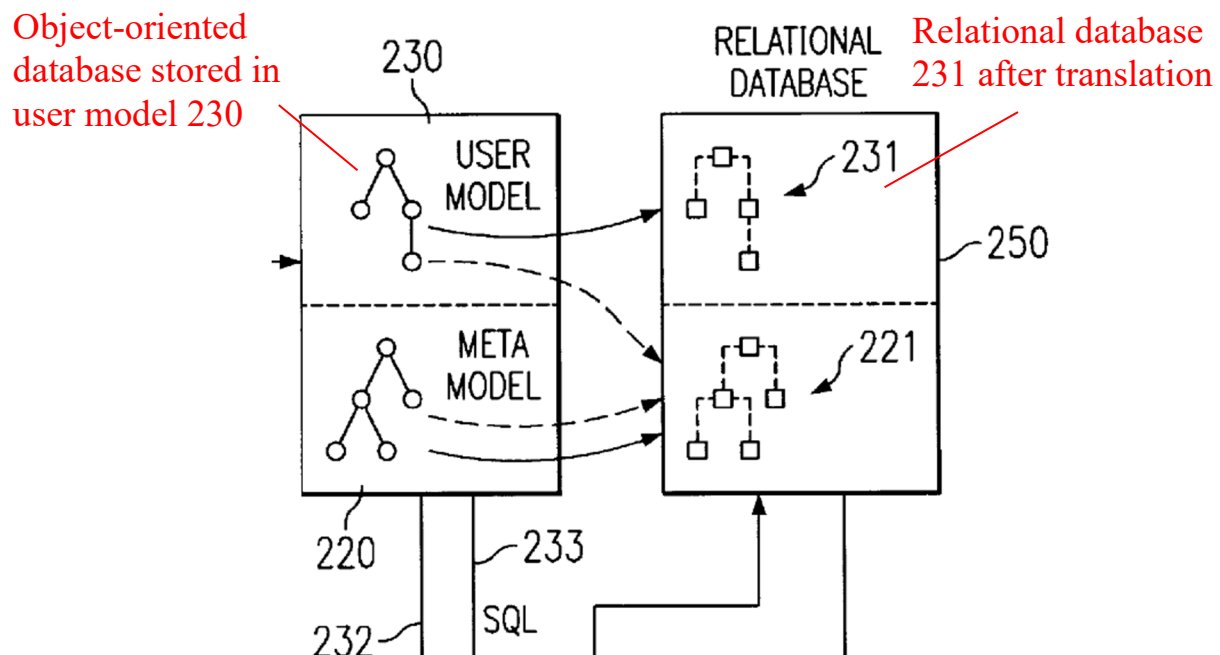
Petition. For example, in the Petition, a module may be any portion of a table whereas in the District Court the portion must be one or more columns of a single table.

inconsistencies, “Petitioner fails to satisfy [its] burden” for institution purposes.

*Orthopediatrics*, IPR2018-01546, Paper 10, 12.

**VI. GROUND 1 BASED ON ALTHOFF IN VIEW OF MOORE IS DEFICIENT (CLAIMS 1, 4-11, 14-17, 20-27, 30-32, 33, 36-43, 46-47)**

Tellingly, the Petition begins its discussion of Ground 1 by referring to non-existent “OO relational databases of Althoff.” Petition, 9. But there are no “OO relational databases” in Althoff. On the contrary, Althoff discloses an “object database 100,” depicted in Figure 1, that is stored in “user model 230,” and that is **translated into** “relational database 231,” depicted in Figure 8. Ex. 1006, FIGS. 1, 2, 8, 8:18-28, 23:62-24:31.



Ex. 1006, FIG. 2 (excerpt). These two databases are distinct. Any implication that Althoff discloses an “object-oriented relational database” is incorrect. In fact,

Althoff's purpose is to build a translator between the two types of databases specifically because they are incompatible: "One problem which has arisen in the art is that the tools for creating, implementing, and manipulating object-oriented databases are generally inconsistent with the tools for creating, implementing, and manipulating relational databases." Ex. 1006, 2:9-14. Accordingly, Althoff proposes its translator between two distinct databases. Figure 1 of Althoff depicts an exemplary object database 100 and Figure 8 depicts an exemplary counterpart relational database. "Accordingly, it would be advantageous to provide a method of compilation and translation between object-oriented and relational database structures...." Ex. 1006, 3:4-7; *see also* Ex. 1006, Field of Invention, 1:14-18 ("The invention [of Althoff] relates to modeling of object-oriented database structures, translation to relational database structures, and dynamic searches thereon.").

As explained in more detail below, numerous arguments in the Petition fail because the Petition ignores the distinction between object database 100 in user model 230 and relational database 231.

**A. The Petition Fails to Identify a "Class Module" And Its Methods That Defines and Stores At Least One Class in The Data Repository (Limitations 1[e], 17[c], 33[c])**

Claim 1 requires the data repository to contain a "class module configured to define at least one class of the plurality of elements and store the at least one

class.” ’585 Patent, 16:32-34. The remaining independent claims similarly require defining and storing a class in the data repository. The Petition maps the data repository onto Althoff’s relational database: “Althoff discloses the data repository as a ‘relational database ... implemented using a relational database engine’. A POSITA would understand that a relational database is a specific implementation of a data repository.” Petition, 14 (emphasis in original). Therefore, Petitioner needed to identify a module (using its construction of module) in Althoff’s relational database that is configured to define at least one class. Petitioner failed to do so.<sup>5</sup>

Instead, Petitioner asserts that Althoff “discloses a class module as a ‘class Class 301’ table.” Petition 21 (emphasis removed). However, there is no “class Class 301 table” in Althoff, and on this basis alone, the Petition fails.

As will be explained, “class Class 301” is a data type defined by the object

---

<sup>5</sup> The Petition also discusses the object database in the next paragraph, but does not assert that the object database maps onto the “data repository” limitation. Because the databases are distinct and incompatible according to Althoff (a translator must translate one into the other), it would be improper to call the combination of the two databases a “data repository.” Moreover, Althoff does not explain how the objects in the object database are stored prior to translation. Finally, the object database of Althoff has no “tables” and thus has no “modules.”

meta-model and that can be used by the user's object database as a data type when the user creates objects. "class Class 301" is not an object of the user's relational database and there is no "Class 301 table" disclosed in Althoff.

### **1. Althoff's object-oriented classes**

To understand Althoff, it is helpful to understand the basics of object-oriented programming. As Petitioner's expert explains, software engineering developed object-oriented programming to provide "higher level abstractions" to "specify the software" in a more useful manner. Ex. 1003, ¶33. "For example, software for the automotive domain might have a 'Car' class with operations for 'startEngine' and 'stopEngine.' An automotive domain expert can then more easily understand how the high-level 'Car' class and its 'startEngine' operation map to automotive requirements that the expert cares about." *Id.* Petitioner's expert further explains that "[a] class is an abstraction that can be roughly thought of as a 'type' that describes a set of data and operations related to that data. For example, a 'Car' could be a class that describes the concept of a car and the attributes and operations common to a car, such as its 'color' attribute and operations for 'startEngine' and 'stopEngine'." Ex. 1003, ¶34.

As Petitioner's expert explains, a class is a data type, much like "integer" and "string" are data types. Ex. 1006, 1:21-35. In object-oriented programming, once a class is defined, the software can create different objects that are members

of the class. Thus, if one defines the class Car (a class called “Car”), then different objects can be created to represent different cars. And if the class “Car” has a color attribute, then each object of class “Car” can have a different color. Thus, as Petitioner’s expert explains: “For example, my car is one instance and my wife’s car is another instance. In the OO programming realm, my car and my wife’s car would each be a separate object with their own values for the color attribute.” Ex. 1003, ¶35.

Althoff provides an exemplary object-oriented database in Figure 1. In this figure, the database has eight classes: (1) Component, (2) Analog Component, (3) Memory, (4) Dynamic Memory, (5) Static Memory, (6) Manufactures, (7) Domestic [manufacturer], and (8) Foreign [manufacturer]. Ex. 1006, FIG. 1, 5:33-6:29.

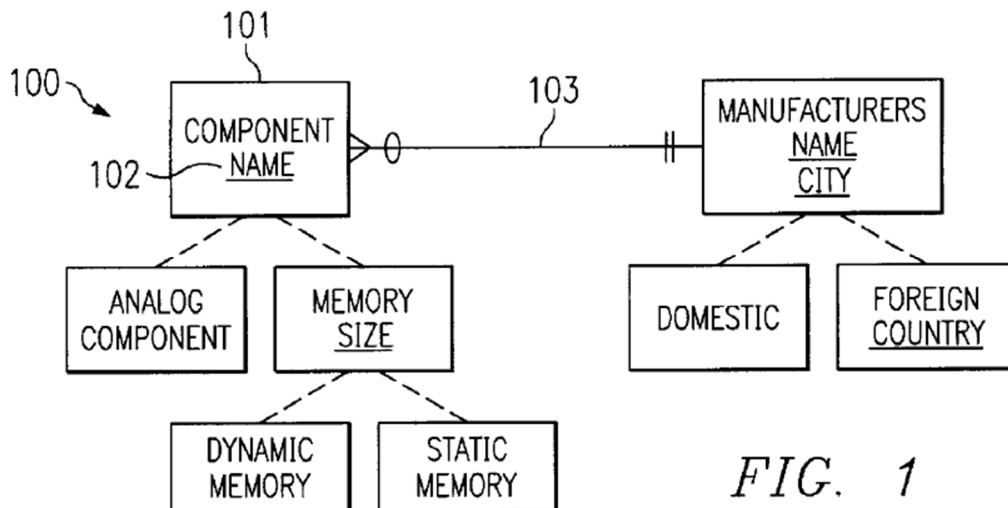


FIG. 1

Ex. 1006, FIG. 1. Each of these classes, like the class “Car,” would have objects

that are members of (are “instances of”) the class. For example, a database may have an object called “Widget #1” of class “Component” and may have an object called “Acme Corporation” of class “Manufacturer.”

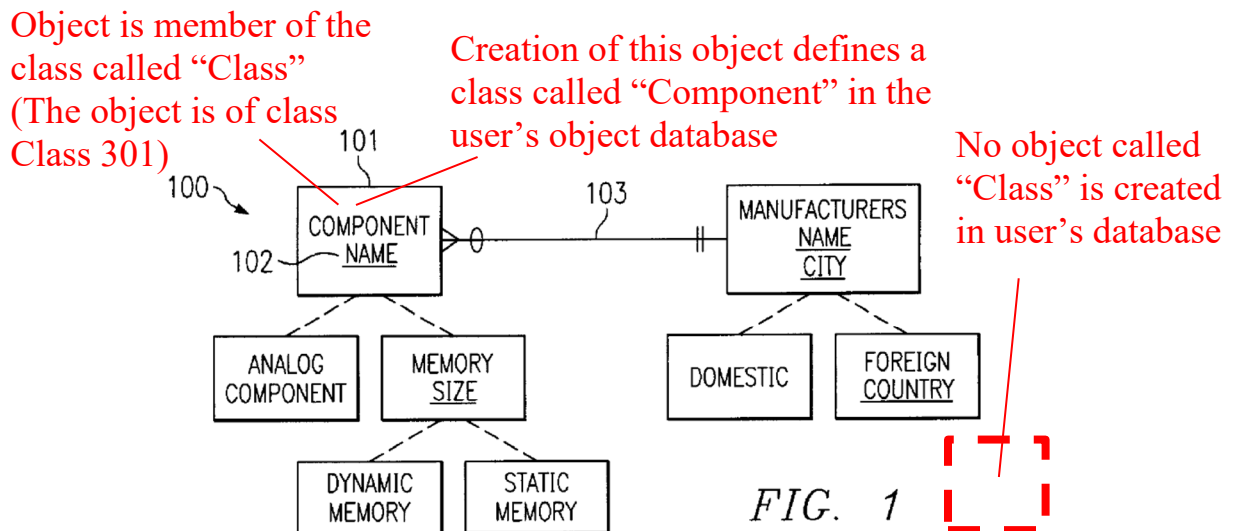
In Althoff, a database designer defines classes by creating objects that are identified as having the type “Class.” Thus, each of the classes in Figure 1 is defined in Althoff’s object database by creating objects of class Class (that is, objects that are members of the class called “Class”). Ex. 1006, 10:41-46.

For example, “Widget #1” is an object of class Component, and Component is an object of class Class. The user’s object database will thus have an object for each class created by the database designer (such as Component and Manufacturer) and will have objects for each instance of each class (such as Widget #1 and Acme Corporation). But the user’s object database has no object called “Class.”

**2. “class Class 301” is a class (a data type) and not an object in the user’s object database**

In Althoff, “class Class 301” is a class (a data type) defined in Althoff’s meta-model. Ex. 1006, 10:41-46. Objects assigned to that class are thus identified as classes in the user’s object database. It exists in Althoff’s meta-model so that a user may define classes (such as “Component” and Manufacturer”) by building objects having the data type “Class” (object of class Class 301). As shown in Figure 1, the exemplary user’s object database has eight objects that define the

eight classes of the user's object database. Each object is member of the class Class 301. Ex. 1006, 11:12-14. Critically, there is no object called "Class" in the user's object database:

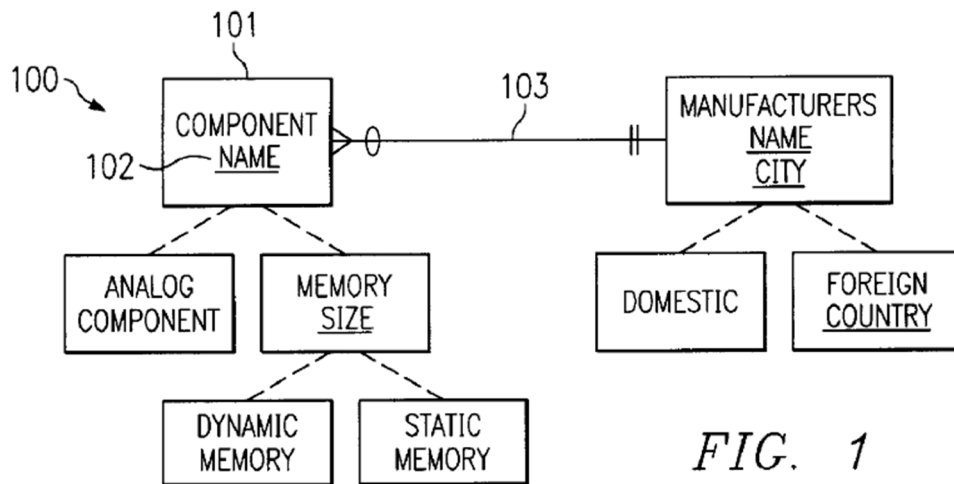


**3. Althoff's database translator translates each of the user's class objects into relational database tables, but "class Class 301" is not translated**

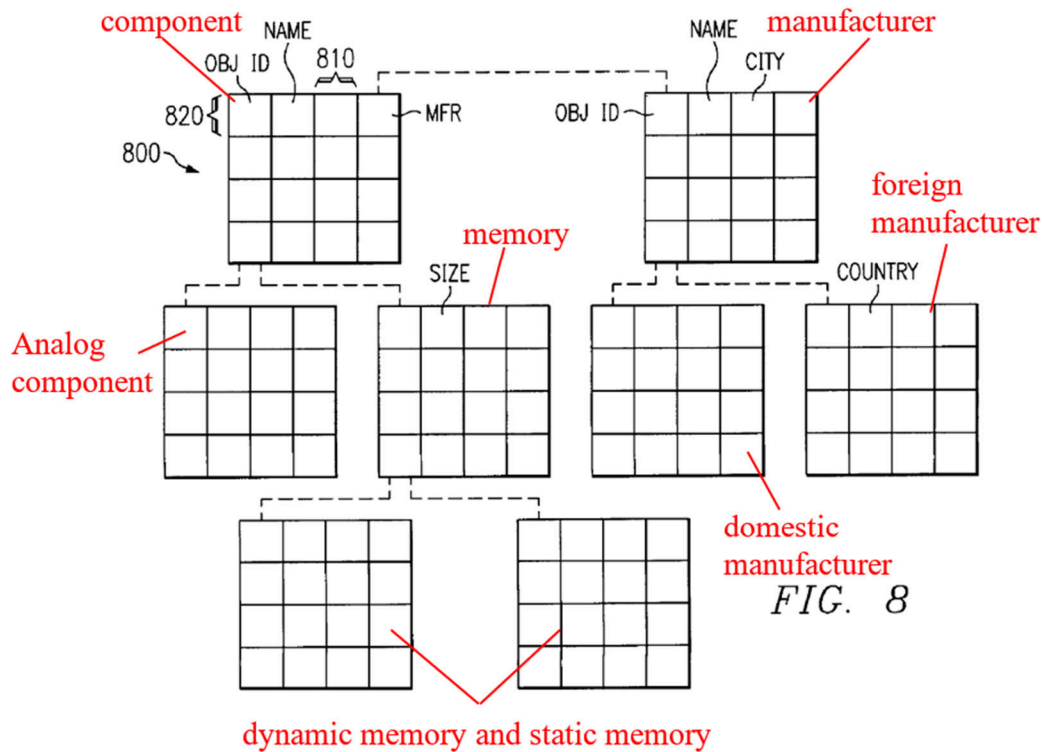
Althoff's translation process is described in connection with Figure 7.

Althoff explains that the system "selects a class 101 in the user's object database 100." Ex. 1006, 23:6-7, *see also* 5:39-40 ("An example user's object database 100 [in FIG. 1] comprises a set of classes 101"). In the exemplary object database of Figure 1, the system would select each of the eight classes created in the user's object database: (1) Component, (2) Analog Component, (3) Memory, (4) Dynamic Memory, (5) Static Memory, (6) Manufactures, (7) Domestic [manufacturer], and (8) Foreign [manufacturer]. Ex. 1006, FIG. 1, 5:33-6:29. The

system then “creates a table in the corresponding relational database 203 to correspond to the selected class 101.” *Id.*, 23:11-13. Thus, when translating the exemplary database shown in Figure 1, the translator will generate eight tables, one corresponding to each object created by the user of type “Class” (object of class Class 301). Thus, the object database of Figure 1 is translated into the relational database of Figure 8:



Ex. 1006, FIG. 1 is translated into:



Ex. 1006, FIG. 8, 4:58-59, 5:8-10.

Because there is no class in the user's object database called "Class," the translator will not select this non-existent class for translation into relational database tables. There is no disclosure in the citations to Althoff in the Petition supporting Petitioner's assertion that the translator selects Class 301 and translates that into a table.

In fact, because Althoff generates a separate table for each class in the user's object database, Althoff would have **no reason** to generate a "class Class 301 table" as Petitioner states. Althoff's relational database preserves the class information by creating separate tables for each class. For example, there would

be a “Component” table and a “Manufacturers” table. Thus, the specific table structure of Althoff preserves the class information.

The Petition cites to Althoff, 3:45-46 for support of its “class Class 301 table” assertion. The Petition misstates the Althoff disclosure by cropping critical language. The quote actually states: “In a preferred embodiment, each class **in the user’s object database** is modeled by a relational database table . . . .” The omitted phrase “in the user’s object database” is critical. The user’s object database does not contain any class called “Class 301.” As stated previously, “Class 301” is a class defined in the meta-model that is used by the user’s object database **as a data type** to identify the classes in the user’s database. In other words, “Class 301” is a class defined by the system, and is used to identify the classes in the user’s database. The user’s database classes would be, *e.g.*, “Component,” “Manufacturers,” “Memory,” etc. Ex. 1006, FIG. 1. There is no class called “Class 301” in the user’s object database and there is no “Class 301” table in the relational database. This is confirmed by Figure 8 of Althoff. There, the eight classes created by the user in Figure 1 generate eight tables in Figure 8.

The Petition doubles down on its assertion, stating that “Althoff teaches that these classes are defined and stored in a ‘class Class 301’ table (*i.e.* class module).” Petition. 21 (emphasis removed). However, the citations offer no support for this incorrect statement. The Petition and expert declaration are identical for this issue,

and neither identifies any teaching in Althoff to support this assertion. *Compare* Petition, 21 *with* Ex. 1003, ¶93. The cited passages from Althoff state that classes in the user's object database (such as "Component" and "Manufacturers") are modeled as objects of type "Class" ("object of class Class 301"). *Id.* This is precisely what has just been explained in this Preliminary Response. Nothing in the citations mentions a "class Class 301 table" or otherwise suggests that the objects are stored in a "class Class 301 table" within the object database.<sup>6</sup> Indeed, if objects were already stored as tables in the object database, there would be no need for Althoff to build a translator to translate objects into tables.

The Petition again misstates the teachings of Althoff in the next paragraph. There, the Petition states that an object in Althoff's object database is a "row" of the fictitious "class Class 301 table." Petition, 21 (internal quotations omitted). The Petition partially quotes Althoff, 11:12-16 in support, but the Petition radically alters the meaning of the quoted passage via its partial quotation, ellipsis, and insertion. Petition, 21-22.

---

<sup>6</sup> Objects can be complex data structures, as explained by Petitioner's expert. In paragraphs 35-37 of Exhibit 1003, Petitioner's expert provides examples of objects. Nowhere in that description does he suggest that objects are stored in tables. On the contrary, he discusses **translating** such objects into tables in paragraph 40.

What Althoff actually says is:

The user's object database 100 comprises a set of classes 101. In the meta-model 220, each class of the user's object database 100 is modeled by an object of class Class 301. In a preferred embodiment, each object of class Class 301 comprises information in table 3-2.

Ex. 1006, 11:12-16. Thus, as the first sentence makes clear, the discussion here concerns the user's object database and not the relational database. The second sentence begins by referring to the "meta-model 220"—again, an object database platform, and not the relational database—a detail excised by the Petition's partial quoting. The second sentence further states that object database classes are "modeled by an object of class Class 301." Nothing in this sentence states that a "class Class 301" table is created. In fact, Althoff says something quite different. The passage confirms that "class Class 301" is a data type and not an object in the user's object database.<sup>7</sup> See Ex. 1006, FIG. 1; 5:33-36.

Moreover, the citations to Althoff nowhere state that class definition information is stored in a "row" of an alleged "class Class 301 table" as asserted in

---

<sup>7</sup> The nomenclature can sound confusing. An "object of class "Class" means that the object is a member of the "Class" class. It does not mean that "Class" (or the definition of "Class") is stored as an object.

the Petition. Petition, 21. There is no “class Class 301 table” and each object of type “Class 301” would be distinct in the object database. Again, if the class objects in Figure 1 were already stored in a table, then Althoff would have no need to describe a translation process to convert objects into tables.

Figure 1 suggests the opposite—it depicts independent objects. The text of Althoff confirms this: “At step 614, the system 200 creates a new object of class Class 301 in the meta-model 220.” Ex. 1006, 21:39-40. “At step 618, the user 201 selects an object of class Class 301 in the meta-model 220, and proceeds to delete that object. The system 200 removes the object from the meta-model 220....” *Id.*, 21:53-55. Contrast these descriptions with the descriptions of the relational database: “At a step 730, the system 200 creates a table in the corresponding relational database 203 to correspond to the selected class 101.” Ex. 1006, 23:10-12. Clearly, there are tables in the relational database, but not in the user’s object database.

In sum, Althoff’s disclosure of a class called “Class 301” in the object meta-model (which is used as a data type for objects in the user’s object database) in no way discloses the claimed “class module” of the ’585 Patent.

**4. Table 3-2 in the specification should not be confused with the alleged “class Class 301 table”**

The Petition refers to “Table 3-2” in the specification of Althoff. It is

important to note that Table 3-2 is merely a descriptive device in the specification of Althoff that describes in a two-column format the information stored within an object of type “Class” in the user’s object database. For example, it lists the information that the object named “Component” in Figure 1 would have. “The user's object database 100 comprises a set of classes 101.... In a preferred embodiment, each object of class Class 301 [each class in Figure 1] comprises information in table 3-2.” Ex. 1006, 11:12-16. Nowhere does Althoff state that the information in Table 3-2 is stored as a class table or as a row in a “class table” in the object database. The tables in the written description simply present information in a form more easily accessible to the reader than a narrative.

**5. The alleged “class module” and cited class definitions in Althoff are in the wrong place**

Setting aside whether “class Class 301” is a table or not (it is not), the description of “class Class 301” cannot in any event meet the claim requirement that the module must “define at least one class of the plurality of elements and store the at least one class in the data repository.” ’585 Patent, Claim 1 (similarly in claims 17 and 33). “class Class 301” defines a class called “Class” in the meta-model that used by objects as a data type in the user’s object database to identify those objects as classes. Ex. 1006:11-16. “class Class 301” is not used by Althoff’s relational database, which is what the Petition maps to the data

repository. Petition, 14. Missing from the Petition's analysis is any suggestion (or more importantly evidence) that "class Class 301" is a structure in the relational database that stores information about classes. It is not.

Thus, the Petition fails to identify any class module (or any other structure) that defines and stores classes in the data repository (which the Petition maps to the relational database of Althoff). For this reason as well, the Petition is deficient.

**B. The Alleged "Attribute Module" And "Class Attribute Module" In Althoff (And Their Corresponding Methods) Are Not in The Data Repository As Claimed (Limitations 1[f], 1[g], 17[d], 17[e], 33[d], 33[e])**

The Petition relies on the relational database of Althoff for the data repository. Petition, 14. Thus, limitations 1[f] and 1[g] must be found in that relational database. The Petition is deficient because it relies exclusively on aspects of Althoff's object database for these limitations. For example, for the "class attribute module" limitation, the Petition relies on "relationships 103," which the Petition concedes is part of the "object-oriented programming database." Petition, 23-24. The Petition alternatively cites to "class Pointer Valid Class 305" as a class attribute module, but this too is a class in the object-oriented meta-model. Petition, 25 (citing to Ex. 1006, 13:7-14, 14:34-36). The Petition does not explain how, if at all, this information allegedly makes its way into the user's relational database. The Petition is thus deficient. The Board and Patent Owner should not

have to guess as to what mapping Petitioner intended. *See Olaplex, Inc. v. L'Oreal USA, Inc.*, 855 F. App'x 701, 712 (Fed. Cir. 2021) (“Judges are not like pigs, hunting for truffles buried in the record”).

Finally, the Petition makes one mention of a relational database when discussing the alleged “class attribute module.” The Petition states: “Althoff teaches relational database structures comprising ‘a set of classes 101, a set of searchable properties for each class 101, and a set of relationships 103 between classes 101.’ *Id.*, 10:12-16.” Petition, 24. However, Althoff does not say this. Rather, Althoff is clearly discussing the object database and not the relational database in that passage. The heading preceding the passage states: “Meta-Model for Modeling the User’s Object Database.” Ex. 1006, 10:7-8. And, the omitted text from the quoted passage states that meta-model 220 contains the set of classes 101, etc. There is no mention of a relational database in this passage or any statement that each of these classes is somehow converted into a table. *Id.*, 10:12-16.<sup>8</sup>

---

<sup>8</sup> Although Petitioner does not rely on column 7 of Althoff, the Board may note that this column states that the meta-model is also represented by a relational database 221. Ex. 1006, 7:58-62. No information is provided as to how the “meta-model” is stored therein. There is no suggestion in Althoff that the definition of

Because the Petition relies exclusively on aspects of the object meta-model and fails to explain how those aspects are carried over—if at all—into the relational database, the Petition has failed to show a “data repository” that contains all the recited limitations of the challenged claims.

**C. Moore Does Not Disclose the Named Text Module or Methods (Limitations 1[i], 17[g], 33[g])**

Claim 1 of the '585 Patent recites a “named text module” that is configured to “store textual representations of query, control, and display mechanisms” and a “tuple module” that is configured to store tuples that link folders and filters “to generate a hierarchy of one or more data entities.”<sup>9</sup>

In one embodiment, the '585 Patent stores SQL queries in the `named_text` table as elements in the database. '585 Patent, 10:64-11:10. The Patent explains: “SQL query is a powerful notion for relational databases and may be implemented

---

“class Class 301” is stored as a module (as opposed to, for example, a record in a table, outside of any table, or some other format). Moreover, regardless of how it is stored, the definition of “Class 301,” would have no information about the classes shown in Figure. 1. That information would be in the objects within the user’s database. Finally, relational database 221 is a different database than relational database 231.

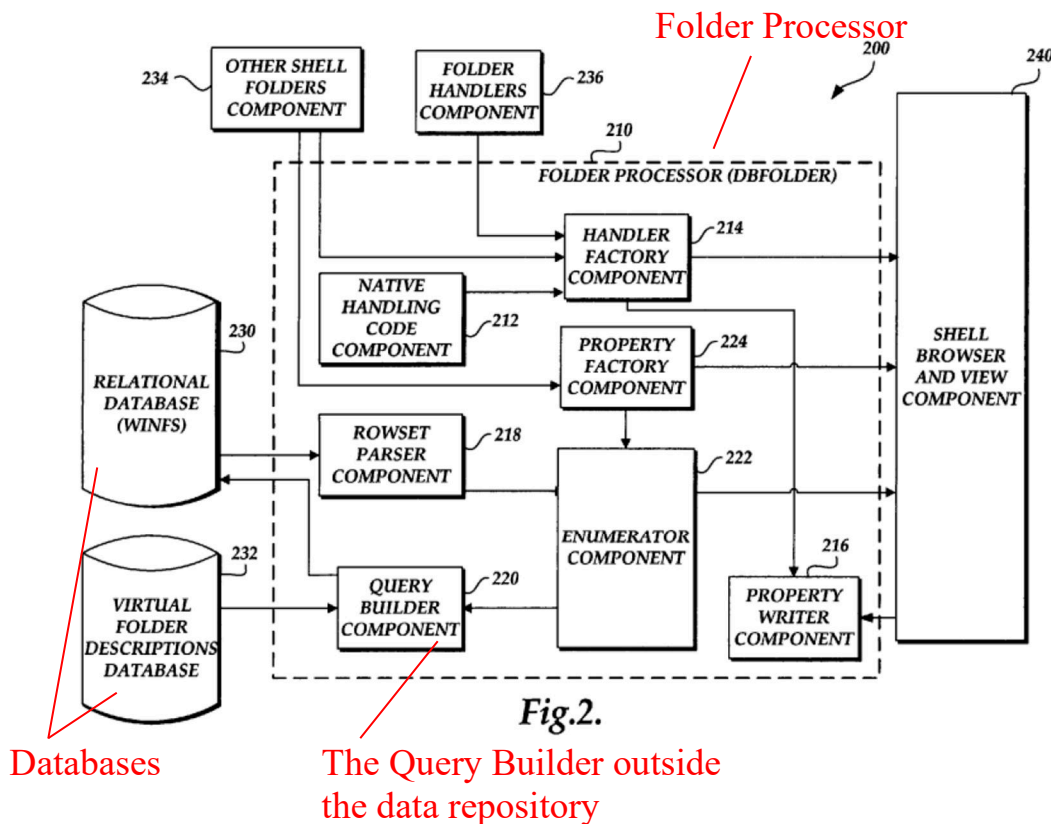
<sup>9</sup> Claims 17, 33, and 48 have similar requirements and the Petition does not provide separate arguments for these claims.

as a named text entry. Unlike a conventional database, *e.g.*, Oracle(R) or Microsoft SQL Server(R), that stores user-generated SQL as a ‘view,’ the EMR stores user-generated SQL as a named text entry.” ’585 Patent, 11:1-6.

The ’585 Patent further explains that the named text module and tuple module can work together: “The combination of named\_text table 140 and tuple table 144 provides to the EMR the capability to create a recursion between named\_text types that represent folder to other named\_text types that represent folders in a hierarchy.” ’585 Patent, 10:46-49.

The Petition relies on Moore for the “named\_text” and “tuple” limitations, but the Petition glosses over details of the claims that distinguish the claims from Moore.

**First**, Moore does not disclose the storage of “textual representations of query, control, and display mechanisms” (such as SQL statements) in the data repository (the relational database). As the Petition admits, Moore’s SQL statements are generated by Moore’s “Query Builder Component 220” within “Folder Processor 200,” which is **outside** Moore’s two databases, as the Petition itself depicts on Page 29.



The Petition attempts to avoid this clear fact by drawing a wide circle around the **combination of** the “folder processor” and the “virtual folder descriptions database.” Petition, 28. However, the Petition nor the expert declaration provides any support from Moore suggesting that the “virtual folder descriptions database” stores SQL queries. On the contrary, Moore clearly states: **“The query builder component 220 builds SQL queries....** The query builder component 220 also receives data from the virtual folder descriptions database 232, including a list of the types to display in the folder, the initial filter, and the physical location to show results from (the scopes).” Ex. 1007, 9:19-25. Because the query builder **“builds** SQL queries,” the builder would not receive SQL queries

from the “virtual folder descriptions database.” Instead, as Moore states, it receives “a list of the types,” the “initial filter,” and the “physical location” for display. *Id.*

**Second**, even if the “query builder component 220” were in the data repository (it is not), Moore does not “**store**” queries in the query builder component. Rather, Moore builds the queries upon request from the “enumerator component 222.” Ex. 1007, 9:10-18. And the “enumerator component 222 operates in response to a navigation to a virtual folder.” *Id.*, 9:37-38. Recognizing this fact, the Petition argues: “To the extent Patent Owner argues that the SQL query is not stored **because it is dynamically generated**, alternatively, the ‘virtual folder descriptions database 232 includes virtual folder descriptions,’ *i.e.*, textual representations.” Petition, 31 (emphasis modified). Note that the Petition does not deny that the SQL query is dynamically generated. Thus, the Petition is deficient because it does not find storage of SQL text (the text it relies upon) within the databases or anywhere else.

**Third**, the Petition’s alternative mapping of the “virtual folder descriptions database 232” is also deficient. The Petition acknowledges that the “virtual folder descriptions database 232” stores “a list of types to display in the folder” and “the initial filter.” Petition, 31. The Petition further acknowledges that this information “**is converted into**” the “query, control and display mechanisms” by the query

builder. *Id.*, 30-31. But that is not good enough. The claims require the “query, control, and display mechanisms” to be stored in the data repository and not merely inputs to a query builder. Moreover, the Petition is focused on the wrong database. The “query, control, and display mechanisms” must be stored in the **relational database**, which is what the Petition maps to the data repository.

Moore evidently had a reason to separate the “relational database 230” from the “virtual folder descriptions database 232.” The Petition does not suggest that a skilled artisan would deviate from Moore’s teaching when making the combination. On the contrary, in the proposed combination (assuming one were motivated to do so), a “virtual folder descriptions database” would be added to the Althoff system, and thus even under the Petition’s analysis, the mapping fails. The Petition points to nothing in Moore suggesting the storage of query, control, and display mechanisms in the relational database.

**D. Moore Does Not Disclose the Tuple Module or Methods That Generate a Hierarchy as Claimed (Limitations 1[j], 17[h], 33[h])**

The Petition attempts to map the “virtual folder descriptions database” onto the “tuple module.” The mapping is deficient. As the Petition acknowledges, the virtual folder descriptions database stores “a list of types to display in the folder, the initial filter, and the physical locations to show results from.” Petition, 32 (*citing* Ex. 1007, 8:23-28). Thus, as the Petition concedes, the information in the

database provides information concerning how a virtual folder will look when displayed. Nothing in this passage suggests that the information links folders and filters “to generate a hierarchy of one or more data entries.” Petition, 32.

For the hierarchy requirement, the Petition states that “Moore teaches that a virtual folder can ‘further include[] virtual folders.’” This fact is unremarkable, as virtually any folder system allows for sub-folders. Moore provides examples of folder hierarchies, such as Figure 7, which is reproduced in the Petition at page 33:

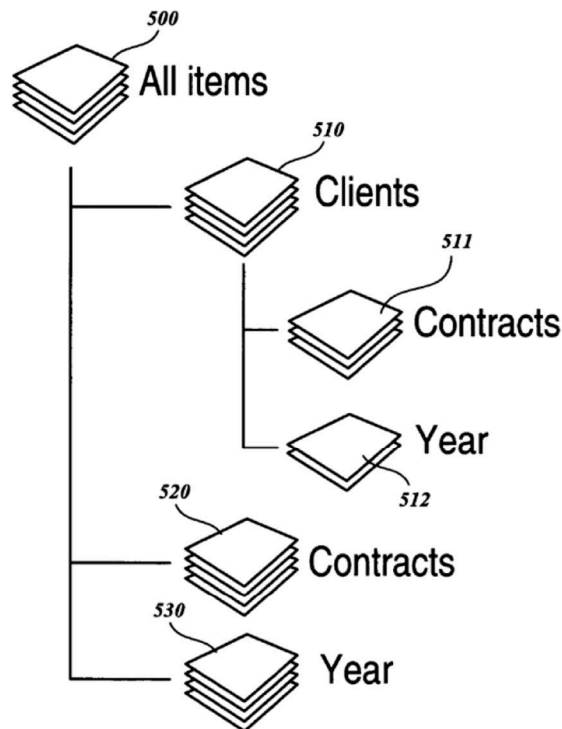


**Fig.7.**

Ex. 1007, FIG. 7. Thus, Moore discloses folder hierarchies. That much is correct.

However, missing from the Petition is any suggestion, argument, analysis, or

more importantly, evidence, that the “virtual folder descriptions database 232” (the alleged tuple module) contains linking information to generate these hierarchies, as claimed. Rather, the Petition breaks up the “link” requirement and the “hierarchy” requirement and treats them separately. Petition, 32-33. This is improper and deficient. The alleged “links” must be used to “generate a hierarchy.” However, in Moore, the folder hierarchy is generated by the “folder processor” using complex input that may include both databases and user queries. *See, e.g.*, Ex. 1007, FIG. 4.

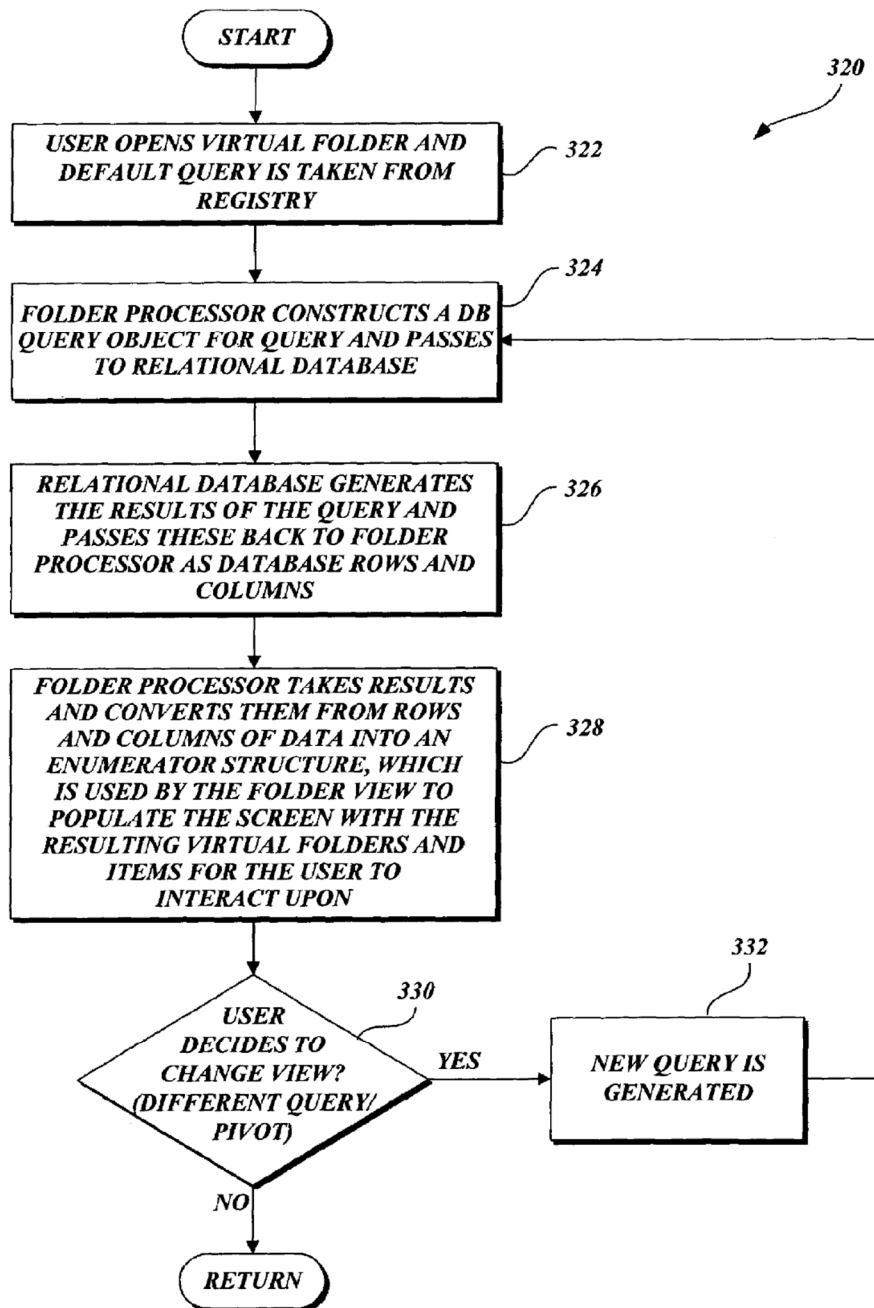


**Fig.4.**

The Petition identifies nothing in Moore suggesting that the hierarchy is generated with the information that allegedly “links the folder descriptions . . . with an initial filter.” Petition, 32.

In addition, similar to the “named text” limitations, the Petition identifies something outside the relational database 230 of Moore (the “virtual folder descriptions database 232”) as corresponding to the tuple module. In the proposed combination, a skilled artisan following the teachings of Moore according to the Petition’s analysis would add such a database to the system of Althoff. The Petition overlooks this plain fact, and does not suggest that a skilled artisan would be motivated to abandon the “virtual folder descriptions database 232” and instead somehow figure out how to move the identified functionality (even assuming it were sufficient to otherwise meeting the claim language) to the “relational database 230.”

Accordingly, the Petition fails to show that the prior art of record discloses the tuple limitations of the challenged claims, each of which requires tuples to link folders and filters to generate a hierarchy and reside in the identified data repository (the relational database).

#### **E. The Dependent Claims**

The dependent claims are deficient for the same reasons as the independent claims. The Petition makes incorrect arguments in defense of the dependent claims that are similar to those made for the independent claims. Because there is no need to reach the dependent claims given the deficiencies exposed for the independent claims, this Preliminary Response does not address the dependent claims

individually. But that silence should not be read as any sort of agreement with the arguments presented for the dependent claims.

**VII. GROUND 2 BASED ON ALTHOFF IN VIEW OF MOORE IN FURTHER VIEW OF NYE IS DEFICIENT (CLAIMS 2-3, 18-19, 34-35)**

This ground concerns dependent claims and fails to fill the gaps identified regarding the Althoff and Moore combination for Ground 1. Accordingly, Ground 2 is deficient for the same reasons as Ground 1.

**VIII. GROUNDS 3 BASED ON ALTHOFF IN VIEW OF MOORE IN FURTHER VIEW OF SURJANTO IS DEFICIENT (CLAIMS 12-13, 28-29, 44-45)**

This ground concerns dependent claims and fails to fill the gaps identified regarding the Althoff and Moore combination for Ground 1. Accordingly, Ground 3 is deficient for the same reasons as Ground 1.

**IX. SOTERA STIPULATION**

In view of the Petitioner's representation that it is providing a "*Sotera*-type stipulation," Patent Owner will not address discretionary denial under *General Plastics*. Petition, 68. The denial of institution can and should be on the merits.

**X. CONCLUSION**

For the foregoing reasons, Patent Owner respectfully requests that the Board deny institution of Grounds 1-3 of the Petition, and thus decline to institute *inter*

*partes* review of any claim of the '585 patent.

Respectfully submitted,

Date: October 18, 2023

/Steven R. Katz/

Steven R. Katz, Reg. No. 43,706  
Andrew G. Pearson, Reg. No. 68,884  
Jason W. Wolff, Reg. No. 43,281  
Fish & Richardson P.C.  
60 South Sixth Street  
Minneapolis, MN 55402

*Attorneys for Patent Owner*

**CERTIFICATION UNDER 37 C.F.R. §42.24(d)**

Under the provisions of 37 C.F.R. §42.24(d), the undersigned hereby certifies that the word count for the foregoing Patent Owner's Preliminary Response to Petition totals 9,086 which is less than the 14,000 allowed under C.F.R. §42.24(b)(1).

Respectfully submitted,

Date: October 18, 2023

/Steven R. Katz/  
Steven R. Katz, Reg. No. 43,706  
Fish & Richardson P.C.  
60 South Sixth Street  
Minneapolis, MN 55402

*Attorney for Patent Owner*

**CERTIFICATE OF SERVICE**

Pursuant to 37 C.F.R. §§42.6(e), the undersigned certifies that on October 18, 2023, a complete and entire copy of this Patent Owner's Preliminary Response were provided, via email to Petitioner by serving the correspondence email address of record as follows:

James M. Glass  
John T. McKee  
Quincy Lu  
Joshua Parzivand  
QUINN EMANUEL URQUHART & SULLIVAN, LLP  
[qe-servicenow-inquisient-ips@quinnemanuel.com](mailto:qe-servicenow-inquisient-ips@quinnemanuel.com)  
[jimglass@quinnemanuel.com](mailto:jimglass@quinnemanuel.com)  
[johnmckee@quinnemanuel.com](mailto:johnmckee@quinnemanuel.com)  
[quincylu@quinnemanuel.com](mailto:quincylu@quinnemanuel.com)  
[joshuaparzivand@quinnemanuel.com](mailto:joshuaparzivand@quinnemanuel.com)

/Crena Pacheco/  
Crena Pacheco  
Fish & Richardson P.C.  
60 South Sixth Street, Suite 3200  
Minneapolis, MN 55402  
[pacheco@fr.com](mailto:pacheco@fr.com)