JS-6

Link: 262

UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

WESTERN DIVISION

ENFISH, LLC,

Plaintiff,

v.

MICROSOFT CORPORATION; FISERV, INC.; INTUIT, INC.; SAGE SOFTWARE, INC.; and JACK HENRY & ASSOCIATES, INC.,

Defendants.

Case No. 2:12-cv-7360-MRP-MRWx

ORDER GRANTING DEFENDANTS' MOTION FOR SUMMARY JUDGMENT ON NONINFRINGEMENT AS TO CLAIM 17

I. Introduction

Plaintiff Enfish, LLC ("Enfish") has sued Defendants Microsoft Corporation, Fiserv, Inc., Intuit, Inc., Sage Software, Inc., and Jack Henry & Associates, Inc. (collectively, "Defendants") for infringement of two patents: U.S. Patent Nos. 6,151,604 ("the '604 Patent") and 6,163,775 ("the '775 Patent"). In an order issued March 31, 2014, the Court concluded that claims 1, 2, and 16 of the '604 patent are single-means claims invalid under 35 U.S.C. § 112. *See* 9 F. Supp. 3d 1126 (C.D. Cal. 2014). In a separate order issued March 31, 2014, the Court invalidated claims 31, 32, 46, and 47 of the '604 patent and claims 31, 32, and 47 of the '775 patent as anticipated under 35 U.S.C. § 102. *See* No. 2:12-cv-7360, 2014 U.S. Dist. LEXIS 46523 (C.D. Cal. Mar. 31, 2014). In an order issued November 3, 2014, the Court found all asserted claims unpatentable under 35 U.S.C. § 101. *See* No. 2:12-cv-7360, 2014 WL 5661456 (C.D. Cal. Nov. 3, 2014).

Defendants move for summary judgment on noninfringement as to claim 17 of the '604 patent on the basis that no version of ADO.NET performs all the recited limitations. For the reasons set forth in this order, the Court grants the motion.

II. Background

The '604 patent recites claims directed to storing computer memory in a logical table. The patent employs a flexible, self-referential table to store data. The table is composed of rows and columns. Each column and each row has an object identification number ("OID"). Rows correspond to records and columns correspond to attributes. The intersection of a row and column comprises a cell, which may contain information for a particular record relating to a particular attribute. A cell also may point to another record. Columns are entered as rows in the table. The record corresponding to a column contains information about the column, rendering the table self-referential. The invention includes an index structure to allow for searching. A key word index contains text from each cell in the table. This index is itself stored in the table. Text cells in the tables contain

pointers to entries in the index, and the index contains pointers to the cells, which provides for extended inquiries. *See* '604 Patent, 2:66–3:6.

Claim 17 of the '604 Patent is a means-plus-function claim directed to a data storage system. It claims:

> A data storage and retrieval system for a computer memory, comprising:
>
> means for configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and
>
> means for indexing data stored in said table.

The Court construed the required structures for "means for configuring said memory according to a logical table" and "means for indexing data stored in said table" in its claim construction order. *See* Claim Construction Order, Dkt. No. 86.

III. Legal Standard

A. Summary Judgment

The Court shall grant summary judgment if there is no genuine dispute as to any material fact, as supported by facts on the record that would be admissible in evidence, and if the moving party is entitled to judgment as a matter of law. Fed. R. Civ. P. 56.; *see Celotex Corp. v. Catrett*, 477 U.S. 317, 322 (1986); *Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 250 (1986). In order to grant summary judgment, the Court must identify material facts by reference to the governing substantive law, while disregarding irrelevant or unnecessary factual disputes. *Anderson*, 477 U.S. at 248. If there is any genuine dispute about a material fact such that a reasonable jury could return a verdict for the nonmoving party,

summary judgment cannot be granted. *Id.* The Court must view facts and draw reasonable inferences in favor of the nonmoving party. *Scott v. Harris*, 550 U.S. 372, 378 (2007). If the party moving for summary judgment does not bear the burden of proof as to a particular material fact, the moving party need only give notice of the absence of a genuine issue of material fact so that the non-moving party may come forward with all of its evidence. *See Celotex*, 477 U.S. at 325.

B. Infringement

Patent infringement is governed by 35 U.S.C. § 271. Subsection (a) of this section provides:

> Except as otherwise provided in this title, whoever without authority makes, uses, offers to sell, or sells any patented invention, within the United States or imports into the United States any patented invention during the term of the patent therefor, infringes the patent.

There are two steps for determining infringement. "First, the asserted claims must be interpreted by the court as a matter of law to determine their meaning and scope." *Southwall Techs., Inc. v. Cardinal IG Co.*, 54 F.3d 1570, 1575 (Fed. Cir. 1995). Then, "[i]n the second step, the trier of fact determines whether the claims as thus construed read on the accused product." *Id.* The Court construed certain claim terms in its July 15, 2013 order. *See* Claim Construction Order, Dkt. No. 86. An accused product must meet the "all elements" test, under which "an accused product or process is not infringing unless it contains each limitation of the claim, either literally or by an equivalent." *Freedman Seating Co. v. Am. Seating Co.*, 420 F.3d 1350, 1358 (Fed. Cir. 2005).

An accused product can infringe a claim literally or under the doctrine of equivalents. Under a theory of literal infringement, "every limitation set forth in a claim must be found in an accused product, exactly." *Southwall*, 54 F.3d at 1575. For a product to infringe under the doctrine of equivalents, "any differences between the claimed invention and the accused product must be insubstantial."

Brilliant Instruments, Inc. v. GuideTech, LLC, 707 F.3d 1342, 1346 (Fed. Cir. 2013). The plaintiff may demonstrate insubstantiality "by showing on a limitation by limitation basis that the accused product performs substantially the same function in substantially the same way with substantially the same result as each claim limitation of the patented product." *Crown Packaging Tech., Inc. v. Rexam Beverage Can Co.*, 559 F.3d 1308, 1312 (Fed. Cir. 2009).

Courts must construe means-plus-function claims according to 35 U.S.C. § 112 ¶ 6, which states that such a claim "shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof." Therefore, "[l]iteral infringement of a claim limitation in means-plus-function format 'requires that the relevant structure in the accused device perform the identical function recited in the claim and be ***identical*** or ***equivalent*** to the corresponding structure in the specification.'" *Welker Bearing Co. v. PHD, Inc.*, 550 F.3d 1090, 1099 (Fed. Cir. 2008) (emphasis added) (quoting *Applied Med. Research Corp. v. U.S. Surgical Corp.*, 448 F.3d 1324, 1333 (Fed. Cir. 2006)). An accused device is equivalent under § 112 ¶ 6 if the differences between it and the disclosed structure are insubstantial. *Welker*, 550 F.3d at 1099. An equivalent under § 112 ¶ 6 may "perform[ ] the same function as the disclosed structure, in substantially the same way, with substantially the same result." *Regents of the Univ. of Minn. v. AGA Med. Corp.*, 717 F.3d 929, 941 (Fed. Cir. 2013). Analysis for § 112 ¶ 6 equivalence resembles analysis for the doctrine of equivalents. In fact, "when the accused technology was known at the time of patenting and the functions are identical, the structural equivalence inquiry under § 112 and the structural equivalence portion of the doctrine of equivalents are coextensive." *Ring & Pinion Serv. Inc. v. ARB Corp. Ltd.*, 743 F.3d 831, 835 (Fed. Cir. 2014).

Infringement is a question of fact. *See Brilliant Instruments*, 707 F.3d at 1344; *Odetics, Inc. v. Storage Tech. Corp.*, 185 F.3d 1259, 1268 (Fed. Cir. 1999)

("Whether an accused device infringes a § 112, ¶ 6 claim as an equivalent is a question of fact.").

IV. Discussion

Defendants argue that ADO.NET does not infringe claim 17 because it meets neither the "means for indexing" limitation nor the "means for configuring" limitation. To begin with, the Court notes there are genuine issues of material fact as to the "means for configuring" limitation.[1] However, based on the Court's construction of the "means for indexing" limitation, ADO.NET does not literally infringe claim 17. ADO.NET also does not infringe claim 17 under the doctrine of equivalents.

A. Construction of the "Means for Indexing" Limitation

The parties first dispute the meaning of the Court's construction for the "means for indexing" limitation. The Court construed the required structure for this means-plus-function limitation as "the structure defined in Exhibit 21 [of Enfish, LLC's Opening Claim Construction Brief] and equivalents thereof." *See* Claim Construction Order at 9. Exhibit 21 assembles various parts of the specification to describe a structure that performs steps listed under a subheading "Algorithm":

1. Extract key phrases or words from the applicable cells in the logical table.

2. Store the extracted key phrases or words in an index, which is itself stored in the logical table.

3. Include, in text cells of the logical table, pointers to the corresponding entries in the index, and include, in the index, pointers to the text cells.

Enfish contends that the Court adopted its proposed structure for the means-plus-function claim, which was "[t]he computer programmed to perform ***one or***

[1] Defendants argue that Enfish never disclosed its theory in its infringement contentions that ADO.NET's DataColumnCollection was a column-defining row for purposes of claim 17's "means for configuring" limitation. The Court finds that Defendants were on notice of this theory when Enfish disclosed that the DataColumnCollection was a column-defining row for purposes of claim 1 of the '604 Patent, which contains the identical "means for configuring" limitation. *See* Dkt. No. 84-2 at 9.

more of the algorithms disclosed in the specification and described in Exhibit 21." Enfish LLC's Opening Claim Const. Br. at 20, Dkt. No. 73 (emphasis added). Microsoft insists that the exhibit discloses three steps for a single algorithm and that a product infringes the claim only if it performs all three steps.

This confusion arises because Exhibit 21 describes only one algorithm, while Enfish's proposed construction suggests the exhibit contains multiple algorithms. The Court now clarifies that Exhibit 21 discloses only one algorithm with multiple steps. The Court construed the structure for the "means for configuring" limitation in the same manner, even though Enfish had proposed that the required structure need only perform one step listed in Exhibit 19. *Compare* Enfish's Claim Const. Br. at 7 (proposing that structure for "means for configuring said memory according to a logical table" was "[t]he computer programmed to perform one or more of the algorithms disclosed in the specification and described in Exhibit 19"), *with* Claim Construction Order at 9 ("As such, the scope of the claim term is the structure defined in Exhibit 19 and equivalents thereof."), *and* 2014 U.S. Dist. LEXIS 46523 at *25 ("The algorithm required for the 'means for configuring' has four steps."). The Court sees no reason to construe the "means for indexing" limitation and Exhibit 21 differently. Therefore, to meet the "means for indexing" limitation, an accused structure must perform all three steps of the algorithm.

B. ADO.NET Does Not Meet the "Means for Indexing" Limitation

ADO.NET fails to perform steps two and three of the algorithm for the "means for indexing" limitation. First, ADO.NET does not store extracted key phrases or words in an index in the logical table. Second, ADO.NET does not "include, in text cells of the logical table, pointers corresponding to the entries in the index, and include, in the index, pointers to the text cells." Therefore, ADO.NET does not perform the "means for indexing" limitation and does not infringe claim 17.

i. ADO.NET Does Not Store Extracted Key Phrases or Words in an Index

ADO.NET fails to meet the second step of the algorithm because it does not store extracted key phrases or words. This step requires the index structure to hold a copy of the extracted key phrase or word. There can be no reasonable dispute about the meaning of this step. As the specification makes clear, "[e]ach key phrase is extracted from a cell and stored in a list format" and "the list may be alphabetized, providing for very rapid searching of a particular name." '604 Patent, 12:12–15. Figure 11 also demonstrates that the index structure holds a copy of the indexed word or phrase in a list. *See* '604 Patent, fig. 11; *see also* '604 Patent, 12:22–26 ("For example, the word 'Ventura' occurs in cells **252**, **254** and **256** that correspond to different rows and different columns. The word 'Ventura' in the list **250** contains a pointer, or cell identification number, to cells **252**, **254**, and **256**."). Thus, to satisfy step two, ADO.NET must store copies of key phrases or words in an index.

Both parties agree that ADO.NET does not store copies of words in an index. But Enfish contends that ADO.NET satisfies step two because ADO.NET's index references the locations of text cells. *See* Enfish's Opp'n to Defs.' Mot. for S.J. at 16 ("ADO.NET stores these keys by reference in the index, which is itself stored in the logical table."); *see also* Defs.' SOF ¶ ¶ 32, 34, Dkt. No. 263-1. But storing the location of a text cell is not the same as storing a copy of the text, as required by step two. Enfish's argument conflates step two and step three of the algorithm. Enfish's argument is essentially that the index "stores" key phrases and words by having pointers to text cells. *See* Enfish's Opp'n at 17 (arguing "ADO.NET accesses the stored keys in its index using a pointer"); Decl. of H.V. Jagadish ¶ ¶ 101–105, 129–32 (under seal); Black Tr. at 170:9-18, Dkt. 282-10. But step three explicitly requires the index to have pointers to text cells. Enfish's interpretation of step two is incorrect because it renders some of step three's

language superfluous. In light of the specification, the only reasonable reading of step two is that the index must store a copy of the text itself. Because ADO.NET does not have an index that stores copies of text, ADO.NET does not meet the algorithm's second step.

ii. ADO.NET Does Not Have Pointers from Text Cells to the Index

ADO.NET also fails to meet the third step of the algorithm, which requires the structure to "include, in text cells of the logical table, pointers corresponding to the entries in the index, and include, in the index, pointers to the text cells." Enfish's primary argument is that under step three of the "means for indexing" algorithm, a text cell does not need to contain a pointer to the index. In other words, Enfish argues the first half of step three is optional. This argument is wrong. Under the Court's adopted construction, each part of every step is essential. This means-plus-function limitation covers only structures that meet all parts of all three steps of the algorithm.

In the alternative, Enfish argues that there is a genuine dispute about whether ADO.NET's text cells contain pointers to the index. Enfish concedes that ADO.NET does not contain pointers directly from the text cells to the index. Instead, Enfish shows the presence of multiple pointers which, via a circuitous route, lead from the StringStorage object eventually to the Red-Black Tree index (represented by the numbered circles):

//

//

//

//

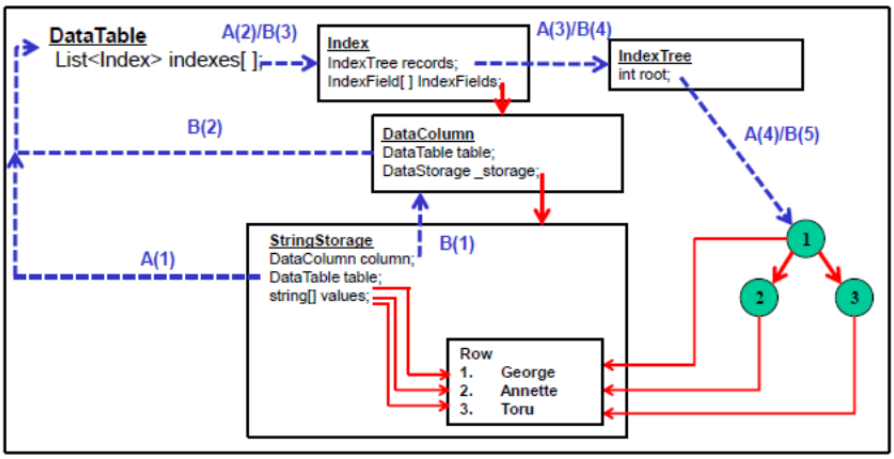
//

//

//

//

Figure 5



Enfish Opp'n at 23. Again, there is no factual dispute between the parties, who both agree on how ADO.NET operates. The dispute remains over claim construction. The question is whether the string of pointers depicted above satisfies the third step of the algorithm. Based on the specification of the '604 patent, a string of pointers cannot satisfy the third step. The specification shows only structures with text cells containing pointers that point directly to entries in the index. *See* '604 Patent, Fig. 14, Fig. 15, 14:10–67. The specification does not depict a string of pointers that eventually leads to the index. The specification discloses only structures where a single pointer corresponds to an index entry.

In light of this fact, it is clear that ADO.NET does not meet the algorithm's third step. As depicted above in Figure 5, the text cells "George," "Annette," and "Toru" contain no pointers directly to index entries. In fact, the text cells contain no pointers at all. As Defendants correctly observe, the dotted blue arrows do not begin in specific text cells, as required by the third step, but rather in the StringStorage object. The StringStorage object represents ***multiple text cells***, as confirmed by Enfish's own expert. *See* Decl. of H.V. Jagadish ¶ ¶ 112, 113, 117. Because the StringStorage object is not a text cell itself, a pointer starting in the StringStorage object cannot satisfy the algorithm's third step.

ADO.NET also does not meet the third step's requirement that a pointer ***directly*** correspond to an index entry. Figure 5 above does not show a single pointer

corresponding to an index entry. Instead, Figure 5 shows a pointer that corresponds to the DataTable object, which has a pointer that corresponds to the Index, which has a pointer that corresponds to the Index Tree, which has a pointer that corresponds to the Red-Black Tree. Although the string of pointers ends in the Red-Black Tree, there is no single pointer directly from a text cell to an index entry. Under the proper construction of "means for indexing," no reasonable jury could find that this string of pointers satisfies the algorithm's third step.

iii. ADO.NET Is Not an Equivalent Structure Under § 112 ¶ 6

Although ADO.NET is not an identical structure for the "means for indexing" limitation, the Court must also decide whether ADO.NET is an equivalent structure under § 112 ¶ 6. As a matter of law, it is not. Defendants correctly observe that 112 ¶ 6 equivalence does not require the accused device to have an equivalent to each algorithm step. *See Odetics*, 185 F.3d at 1268. The question then is whether the differences between ADO.NET and the disclosed structure are insubstantial. Because ADO.NET's indexing structure operates in a substantially different way from the disclosed structure, ADO.NET is not an equivalent under § 112 ¶ 6.

When performing equivalence analysis, courts must consider the context of an invention in determining whether an accused structure is substantially different. *See IMS Tech., Inc. v. Haas Automation, Inc.*, 206 F.3d 1422, 1436 (Fed. Cir. 2000). The differences between ADO.NET and the disclosed structure are substantial because ADO.NET lacks important and beneficial elements of the disclosed structure. With regard to the invention's key word index, the specification repeatedly mentions the benefit of storing key phrases in an index. *See* '604 Patent, 12:10–15 ("The present invention includes an indexing system that provides for rapid searching of text included in any cell in the table **100**. Each key phrase is extracted from a cell and stored in a list format according to a predefined hierarchy. For example, the list may be alphabetized, providing for very rapid searching of a particular name."). The specification provides no evidence

that referencing the location of a text cell achieves the same benefits. More significantly, pointers from the text cells to the index are critical for extended queries. Extended queries are an important benefit of the invention. *See* '604 Patent, Abstract ("The table includes an index structure for extended queries."); '604 Patent, 14:13–18 ("The associations between the list of records with text and the list of key phrases is two-way since the cells that include text point to the key words. . . . Each record can point to multiple key phrases, and each key phrase can point to multiple records."). ADO.NET's lack of such pointers is a crucial difference because these pointers are critical for a benefit of the invention.

These key differences show that ADO.NET does not perform its function in substantially the same way as the disclosed structure. When an accused device lacks elements that represent the disclosed structure's significant benefits, the accused device should not be a 112 ¶ 6 equivalent.

C. ADO.NET Does Not Infringe Under the Doctrine of Equivalents

ADO.NET does not infringe under the doctrine of equivalents. As discussed above, ADO.NET cannot meet the "means for indexing" limitation because it is neither a corresponding nor equivalent structure. Therefore, ADO.NET fails the all-elements test and cannot infringe claim 17 under the doctrine of equivalents. *See Ring*, 743 F.3d at 835 ("[W]hen the accused technology was known at the time of patenting and the functions are identical, the structural equivalence inquiry under § 112 and the structural equivalence portion of the doctrine of equivalents are coextensive.").

D. Enfish Has Shown No Basis for Distinguishing Between Versions of ADO.NET

Enfish argues that the Court, at most, should grant summary judgment only as to the ADO.NET version discussed in Defendants' motion—4.5.1. But Enfish misapprehends burdens at the summary judgment stage. If the moving party does not have the burden of proof at trial, "the burden on the moving party may be

discharged by 'showing' -- that is, pointing out to the district court -- that there is an absence of evidence to support the nonmoving party's case." *Celotex*, 477 U.S. at 325. Enfish has not shown any material differences between ADO.NET versions. Enfish makes only a conclusory statement that "Microsoft's code and documentation show[ ] that substantive differences exist between different versions." Enfish Opp'n at 25; *see* Decl. of H.V. Jagadish ¶ ¶ 137–40. Despite having the opportunity, Enfish has not pointed to any differences between versions of ADO.NET that would change a noninfringement analysis. Because Enfish has failed to show any genuine issue of material fact, the Court's analysis applies to all accused versions of ADO.NET.

V. Conclusion

Because ADO.NET fails to meet the "means for indexing" limitation, the Court grants Defendants' Motion for Summary Judgment on Noninfringement as to Claim 17 of the '604 Patent.

IT IS SO ORDERED.

DATED: November 20, 2014

Hon. Mariana R. Pfaelzer
United States District Judge