Trials@uspto.gov
Tel: 571-272-7822

Paper 11
Entered: June 11, 2024

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CLOUDERA, INC.,
Petitioner,

v.

R2 SOLUTIONS LLC,
Patent Owner.

IPR2024-00303
Patent 8,190,610 B2

Before KARL D. EASTHOM, KRISTEN L. DROESCH, and BRIAN P. MURPHY, *Administrative Patent Judges.*

MURPHY, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

Cloudera, Inc. ("Petitioner") filed a Petition pursuant to 35 U.S.C. § 311 requesting institution of *inter partes* review of claims 1–46 of U.S. Patent No. 8,190,610 B2 (Ex. 1001, "the '610 patent"). Paper 1 ("Pet."). R2 Solutions LLC ("Patent Owner") timely filed a Patent Owner Preliminary Response. Paper 8 ("Prelim. Resp."). Pursuant to authorization from the Board, Petitioner filed a Reply to Patent Owner's Preliminary Response addressing Patent Owner's argument for discretionary denial under 35 U.S.C. § 325(d), and Patent Owner filed a Sur-Reply. Papers 9, 10.

We have jurisdiction under 35 U.S.C. § 314, which provides that an *inter partes* review may not be instituted "unless . . . there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." Under § 314, the Board may not institute review on fewer than all claims challenged in the petition. *SAS Inst., Inc. v. Iancu*, 138 S. Ct. 1348, 1359–60 (2018). If the Board institutes a review, it will institute "on all of the challenged claims and on all grounds of unpatentability asserted for each claim." 37 C.F.R. § 42.108(a). Upon consideration of the Petition, the Preliminary Response, and the evidence of record, for the reasons set forth below, we conclude Petitioner does not demonstrate a reasonable likelihood that it would prevail in showing the unpatentability of at least one challenged claim of the '610 patent. Accordingly, we do not institute an *inter partes* review of claims 1–46 of the '610 patent.

A. Real Parties in Interest and Related Matters

Petitioner, Cloudera, Inc., identifies itself as the real party in interest. Pet. 79. Patent Owner, R2 Solutions LLC, identifies itself as the real party in

interest. Paper 4, 2. Patent Owner further states it is a subsidiary of Acacia Research Group LLC, whose parent is Acacia Research Corporation. *Id.*

Petitioner identifies the following related matters:

Name	No.	Court	Filed
R2 Solutions LLC v. Databricks, Inc.	4:23-cv-01147	E.D. Tex.	Dec. 28, 2023
R2 Solutions LLC v. Cloudera, Inc.	1-23-cv-01205	W.D. Tex.	Oct. 5, 2023
American Airlines, Inc. v. R2 Solutions LLC	IPR2023-00689	PTAB	Mar. 7, 2023
R2 Solutions LLC v. Hilton Domestic Operating Co. Inc.	3-22-cv-02761	N.D. Tex.	Dec. 12, 2022
FedEx Corporate Services, Inc. v. R2 Solutions LLC	IPR2022-01405	PTAB	Aug. 15, 2022
Allegiant Travel Company v. R2 Solutions LLC	2-22-cv-00828	D. Nev.	May. 24, 2022
R2 Solutions LLC v. American Airlines, Inc.	4-22-cv-00353	E.D. Tex.	Apr. 28, 2022
R2 Solutions LLC v. CVS Health Corporation et al	4-22-cv-00354	E.D. Tex.	Apr. 28, 2022
R2 Solutions LLC v. Hilton Worldwide Holdings Inc. et al	4-22-cv-00356	E.D. Tex.	Apr. 28, 2022
R2 Solutions LLC v. Citigroup Inc.	4-22-cv-00357	E.D. Tex.	Apr. 28, 2022
R2 Solutions LLC v. FedEx Corporate Services, Inc.	4-21-cv-00940	E.D. Tex.	Nov. 29, 2021

R2 Solutions LLC v. State Farm Mutual Automobile Insurance Company	4-21-cv-00941	E.D. Tex.	Nov. 29, 2021
R2 Solutions LLC v. Booking.com BV	4-21-cv-00942	E.D. Tex.	Nov. 29, 2021
R2 Solutions LLC v. Booking.com Transport Limited	4-21-cv-00943	E.D. Tex.	Nov. 29, 2021
R2 Solutions LLC v. Agoda Company Pte. Ltd.	4-21-cv-00944	E.D. Tex.	Nov. 29, 2021
R2 Solutions LLC v. Expedia Group, Inc.	6-21-cv-00628	W.D. Tex.	Jun. 17, 2021
R2 Solutions LLC v. iHeartMedia, Inc.	6-21-cv-00552	W.D. Tex.	Jun. 01, 2021
R2 Solutions LLC v. JPMorgan Chase & Co.	4-21-cv-00174	E.D. Tex.	Mar. 02, 2021
R2 Solutions LLC v. The Charles Schwab Corporation	4-21-cv-00122	E.D. Tex.	Feb. 09, 2021
R2 Solutions LLC v. Fidelity Brokerage Services LLC	4-21-cv-00123	E.D. Tex.	Feb. 09, 2021
R2 Solutions LLC v. Deezer SA	4-21-cv-00090	E.D. Tex.	Jan. 29, 2021
R2 Solutions LLC v. Walmart Inc.	4-21-cv-00091	E.D. Tex.	Jan. 29, 2021
R2 Solutions LLC v. Target Corporation	4-21-cv-00092	E.D. Tex.	Jan. 29, 2021
R2 Solutions LLC v. Workday, Inc.	4-21-cv-00093	E.D. Tex.	Jan. 29, 2021

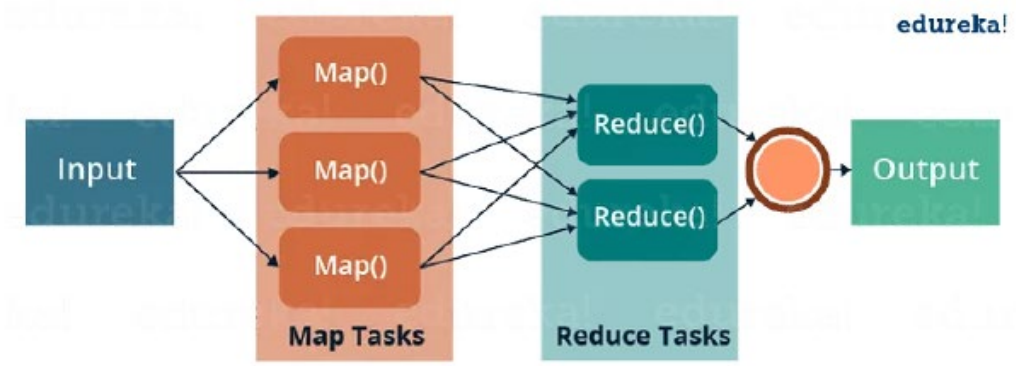
Paper 7, 2–4. Petitioner and Patent Owner both represent that the only currently pending matters are the first two district court litigations listed above: *R2 Solutions LLC v. Databricks, Inc.* (E.D. Tex.) and *R2 Solutions LLC v. Cloudera, Inc.* (W.D. Tex.). *Id.* at 2; Paper 4, 2–3. Petitioner further represents that the two IPR proceedings listed above "were both terminated prior to any decision on institution." Pet. 79; *see also* IPR2022-01405, Paper 9 (PTAB Dec. 9, 2022) and IPR2023-00666, Paper 21 (PTAB Oct. 4, 2023) (granting joint motions to terminate prior to a decision on institution).

B. The '610 Patent

1. Background Regarding MapReduce Processing

The '610 patent, titled "MapReduce for Distributed Database Processing," is directed to a claimed enhancement of MapReduce data processing. "MapReduce is a programming methodology to perform parallel computations over distributed (typically, very large) data sets." Ex. 1001, 1:5–7. Conventional map and reduce processing can be performed by multiple computers, thereby "enabl[ing] the use of massive clusters of commodity computers to provide a simplified programming and execution model for processing large sets of data in parallel." *Id.* at 3:9–12; *see also* Ex. 1005 ¶ 36 (citing Ex. 1001, 3:9–12).

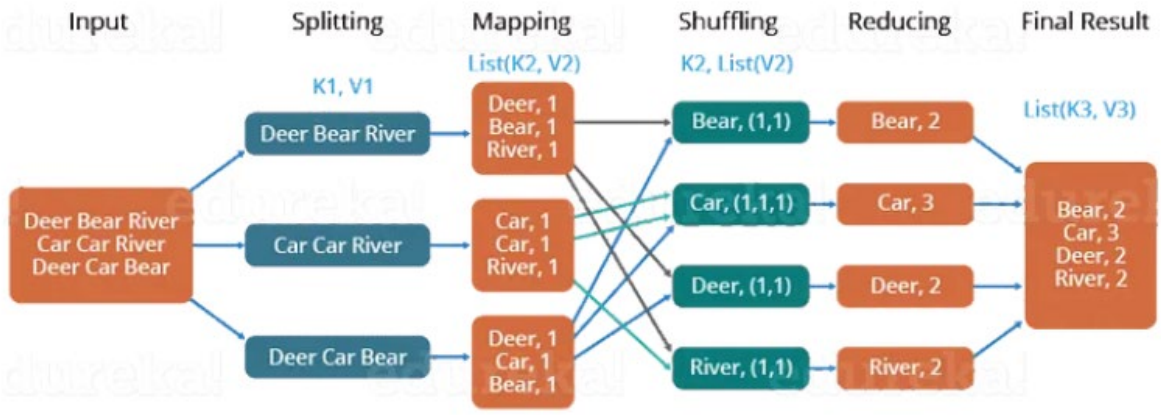
MapReduce involves a "map" function followed by a "reduce" function, as depicted in the figure below:



What is MapReduce - MapReduce Tutorial

Ex. 2002, 4. Each Map Task or map function, shown above, "maps [input] key-value pairs to new (intermediate) key-value pairs." Ex. 1001, 1:17–18. The new intermediate key-value pairs generated by the map function are referred to as "intermediate data" that is input to Reduce Tasks (reduce functions). *Id.* at 4:4–13; *see also* Ex. 1005 ¶ 36 (citing Ex. 1001, 1:6–27). Each reduce function "combines the intermediate data into smaller data sets or lists of values." Ex. 1005 ¶ 36 (citing Ex. 1001, 1:6–27).

MapReduce, for example, can be used to count how often a word occurs in a large collection of documents. In the example reproduced below, "the input data consists of a collection of words (Deer, Bear, River, and Car)":



MapReduce Example - MapReduce Tutorial

Prelim. Resp. 3 (citing Ex. 2002, 5). In the above example, MapReduce divides the input into three splits ("Splitting"), each one supplied to a separate map function for parallel processing. *Id.* Each map function generates a list of intermediate key-value pairs, where each word is a "key" and the "value" for each occurrence of a word is 1 (e.g., "Deer, 1" or "Car, 1"). *Id.*[1] All intermediate key-value pairs having a common key (same word) are grouped together ("Shuffling") so that intermediate data having "the same key are sent to the corresponding reducer." Ex. 2002, 5. "Each reducer counts the values … present in that list of values" and generates new "reduced" key-value pairs that reflect the total count for each word (e.g., "Bear, 2," "Car, 3," "Deer, 2" or "River, 2"). *Id.* at 5.

2. The claimed "data group" enhancement to MapReduce

The '610 patent states that "conventional MapReduce implementations do not have facility to efficiently process data from heterogeneous sources." Ex. 1001, 3:15–17. The '610 patent describes an enhancement where "map processing is carried out independently on two or more related datasets (e.g., related by each being characterized by a schema with a key in common)."[2] *Id.* at 1:34–36. Input data is "partitioned" into two or more "data groups" where "data sets within the same group are characterized by the same schema; and data sets within different groups are characterized by different schemas." *Id.* at 3:52–55.

---

[1] The input data also may be provided as a list of key-value pairs, in which case the map function generates new key-value pairs as intermediate data. Prelim. Resp. 4–5 (citing Ex. 2003, 8).

[2] The word "schema" (short for logical schema) is a term of art in computer science meaning "[t]he encoding of the data model of a database in the relevant database language. It is sometimes simply referred to as the **schema** of a database." Ex. 3001 (A Dictionary of Computer Science, "*logical schema*" 166 (Oxford Univ. Press., 7 ed.) (2016)). Patent Owner also argues that "[s]chema can refer to, e.g., 'fields' in a particular collection of data." Prelim. Resp. 6.

The different "data group" schemas may reflect data originating from heterogenous data sources, such as the Employee table and Department table exemplified in the '610 patent. *Id.* at 3:15–28. Thus, the fundamental organizing principle of the claimed invention is to provide at least two input "data groups" characterized by different schemas or data types for separate MapReduce processing.

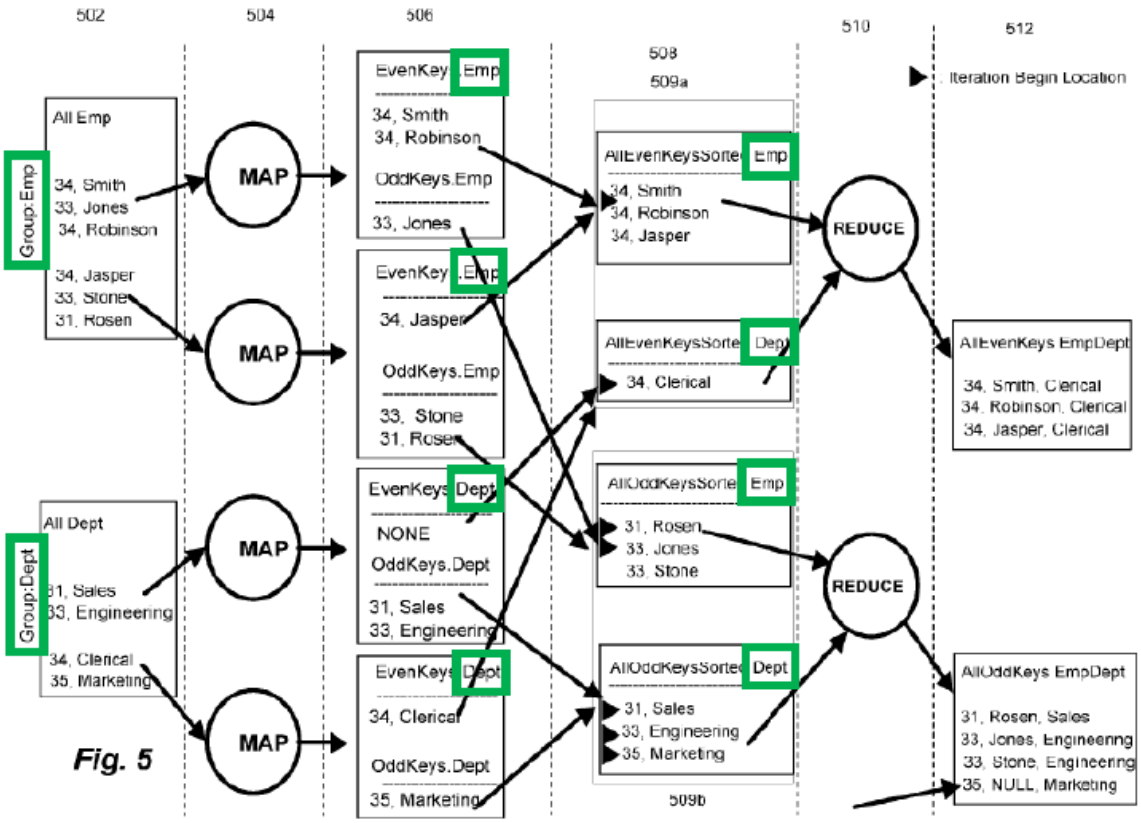
Importantly, the claimed method also provides a mechanism to identify data being processed as corresponding to its original input "data group." *Id.* at 4:4–22, 6:29–48, 7:40–58. A common key, in contrast, permits data from different "data groups" to be related and reduced in a MapReduce process. The '610 patent describes the claimed "data group" improvement as follows:

> In general, partitioning the data sets into data groups enables a mechanism to associate *(group) identifiers with data sets, map functions and iterators* (useable within reduce functions to access intermediate data) and, also, to produce output data sets with *(group) identifiers*. It is noted that the output group identifiers may differ from the input/intermediate group identifiers.

Id. at 3:58–64 (emphasis added). As an example, Figure 5 of the '610 patent depicts an embodiment of claim 1 where different "data group" schemas (the Employee table of employee names and department IDs and the Department table of department names and department IDs), their associated "data group" identifiers ("Emp" and "Dept"), and common keys (the department ID numbers) are used in MapReduce to join the data in the Employee table and Department table. *Id.* at 3:19–34, 8:14–19, Figs. 3, 5.

Patent Owner provides a helpful summary of the invention supported by the

record in an annotated version of Figure 5, reproduced below:



Prelim. Resp. 9. Starting from the left in annotated Figure 5, above, the "full data of the Employee table and the Department table" is the input data partitioned into two "data groups" labeled Group:Emp (502, green box at top) and Group:Dept (502, green box below). Ex. 1001, 8:15–24; *see also* Prelim. Resp. at 9–10 (citing Ex. 1001, 8:15–24). Each input "data group" is "characterized by its own schema," e.g., the data in the Group:Emp group is of a different type (employee names) and source (Employee table) than the data in the Group:Dept group (Department table with department names), such that the two "data groups" have

different attributes.[3] Ex. 1001, 3:66–4:1; *see also* Ex. 1005 ¶ 56 ("The '610 patent therefore specifies a different data format or 'schema' in different input data groups.").

The Group:Emp data group has key-value pairs of DeptID[key]-**LastName**[value] for each employee, i.e., 34, Smith; 33, Jones; 34, Robinson etc. *Id.* at 3:23–28. The Group:Dept data group has key-value pairs of DeptID[key]-**DeptName**[value] for each department, i.e., 31, Sales; 33, Engineering; 34, Clerical etc. *Id.* The two input "data groups" share common keys—DeptID numbers 31, 33, 34, and 35—but their different values—LastName v. DeptName—and sources reflect different schemas.

Annotated Figure 5 depicts each data group as being further partitioned into two "data partitions" (502), and each "data partition" provides a set of key-value pairs to separate MAP functions (504). Ex. 1001, 8:14–23, Fig. 5. For example, the Group:Emp "data group" has a first "data partition" for key-value pairs: 34, Smith; 33, Jones; and 34, Robinson; which are input into one MAP function. The Group:Emp "data group" has a second "data partition" for key-value pairs: 34, Jasper; 33, Stone; and 31, Rosen; which are input into a separate MAP function. *Id.* The independent claims 1, 17, 33, and 40 of the '610 patent recite "*a plurality of data groups*" and "*partitioning the data of each one of the data groups into a plurality of data partitions that each have a plurality of key-value pairs*," or similar language, exemplified in annotated Figure 5 above.

MAP functions (504) generate intermediate data in the form of new key-value pairs (506), which are identified with their corresponding input "data group"

[3] "[T]he schema of each data set … includes a set of attributes (such as DeptID, LastName, DeptName) and their properties (such as their data types: integer DeptID, string LastName, string DeptName)." Ex. 1001, 3:37–41.

by "being emitted into files with an 'emp' extension [that] identifies the data with the 'emp group," or "being emitted into files with a 'dept' extension" that identifies the data with the "dept" group. *Id.* at 7:40–58, 8:24–37. In annotated Figure 5, Patent Owner highlights the "data group" identifiers "Emp" and "Dept" (green boxes) that "follow[] the data through the mapping phase 504 and into the reducing phase 510." Prelim. Resp. 10, 12. Providing a mechanism to identify processed data as corresponding to its input "data group" is a critical component of the claimed enhancement to MapReduce data processing. The independent claims recite a map function that forms "*corresponding intermediate data for that data group and identifiable to that data group*" (claims 1 and 17), or similar language.

The description of the claimed invention concludes as follows:

> [T]he MapReduce concept may be utilized to carry out map processing independently on two or more related datasets … even when the related data sets are heterogenous with respect to each other, such as *data tables organized according to different schema*. The intermediate results of the map processing (*key value pairs) for a particular key can be processed together in a single reduce function by applying a different iterator*[4] *to intermediate values for each group*. In this way, operations on the two or more related datasets ["data groups"] may be carried out more efficiently or in a way not even possible with the conventional MapReduce architecture.

Ex. 1001, 8:46–57 (emphasis added). Thus, in the example of Figure 5, heterogenous data sets (Employee and Department tables) are organized (partitioned) into separate "data groups," where data in one "data group" has the same schema (same type of data or data format) that is different from the schema

[4] "Iterator" is defined as a "self-contained software object[] that accepts[] a stream of rows from null or n-ary data sets. The role of an iterator is to process many iterations of a data set across many nodes in serial or parallel. . . . For example, scanning rows from a table on disk can be the behavior of one type of iterator." Ex. 1003 ¶ 61; *see also* Ex. 1005 ¶ 76 (citing same).

characterizing the data in the other "data group." Each "data group" is further partitioned into a plurality of "data partitions" that are separately map-processed to form intermediate data (key-value pairs), and the map-processed intermediate data is identified as corresponding to one of the input "data groups" using an "emp" or "dept" file extension. Such "data group" identification permits intermediate data having a common key, e.g., key-value pairs having the same DeptID number but originating from different "data groups"—such as 34, Smith; 34, Robinson; 34, Jasper (Employee data group); and 34, Clerical (Department data group) (*see* Fig. 5, 508, 509a, above)—to be reduced (combined) in a single reduce function by applying a separate iterator to each corresponding "data group."

C. Illustrative Claim

Claims 1, 17, 33, and 40 are the independent claims. Claim 1, reproduced below, illustrates the claimed subject matter:

> 1. A method of processing data of a data set over a distributed system, wherein *the data set comprises a plurality of data groups*, the method comprising:
>
> *partitioning the data of each one of the data groups* into a plurality of data partitions that each have a plurality of key-value pairs and providing each data partition to a selected one of a plurality of *mapping functions* that are each user-configurable to independently output a plurality of lists of values for each of a set of keys found in such map function's corresponding data partition *to form corresponding intermediate data for that data group and identifiable to that data group*, wherein the data of a first data group has a different schema than the data of a second data group and the data of the first data group is mapped differently than the data of the second data group so that different lists of values are output for the corresponding different intermediate data, wherein the different schema and corresponding different intermediate data have a key in common; and

> reducing the intermediate data for the data groups to at least one output data group, including *processing the intermediate data for each data group in a manner that is defined to correspond to that data group*, so as to result in a merging of the corresponding different intermediate data based on the key in common,
>
> wherein the mapping and reducing operations are performed by a distributed system.

Ex. 1001, 8:60–9:19 (emphasis added).

In summary, the claims are directed to a method for enhanced MapReduce data processing by organizing (partitioning) input data into "*data groups*" having different schemas. The data in each "*data group*" is further partitioned for separate processing in MapReduce to generate intermediate data ("*mapping*") and reduced data ("*reducing*") "*identifiable*" to a "*corresponding*" "*data group*." The claimed method, therefore, enables MapReduce processing on a "*data group*" basis.

D. Asserted Grounds of Unpatentability

Petitioner asserts claims 1–46 are unpatentable on the following grounds:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/ Basis
1–46 (Ground 1)	103(a)[5]	Pike[6]
1–46 (Ground 2)	103(a)	Pike, Chowdhuri[7]
3, 4, 9–11, 19, 20, 26–28, 37–39, 44–46 (Ground 3)	103 (a)	Pike, Chowdhuri, MacLeod[8]

[5] The Leahy-Smith America Invents Act, Pub. L. No. 112-29, 125 Stat. 284 (2011) ("AIA"), includes revisions to 35 U.S.C. § 103 that became effective after the filing of the application that led to the challenged patent.
[6] US Patent No. 7,590,620 B1 issued Sep. 15, 2009, filed Sep. 29, 2004 (Ex. 1002, "Pike").
[7] US 2006/0218123 A1, Pub. Sep. 28, 2006 (Ex. 1003, "Chowdhuri").
[8] US Patent No. 6,343,295 B1, Jan. 29, 2002 (Ex. 1024, "MacLeod").

Pet. 2.[9] Petitioner relies on the Declaration testimony of Jimmy Lin, Ph.D. in support of the asserted grounds of obviousness. Ex. 1005.

II. ANALYSIS

A. Level of Ordinary Skill in the Art

Petitioner describes a person of ordinary skill in the art ("POSITA") as someone having "at least a bachelor's degree in computer science or similar field, and approximately two years of industry or academic experience in a field related [to] performing data analytics and/or related data processing tasks." Pet. 9 (citing Ex. 1005 ¶¶ 67–69; *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995)). Petitioner further asserts that work experience can substitute for formal education and additional formal education can substitute for work experience. *Id.* Patent Owner does not contest Petitioner's definition for purposes of its Preliminary Response. Prelim. Resp. 14.

In determining the level of ordinary skill in the art, various factors may be considered, including the "type of problems encountered in the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and educational level of active workers in the field." *In re GPAC, Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995) (citing *Custom Accessories, Inc. v. Jeffrey-Allan Indus., Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986)).

---

[9] The Petition analyzes dependent claims 3, 4, 19, and 20 with reference to Ground 3, although claim 20 is not listed in the Unpatentability Grounds. *Compare* Pet. 2 *with* Pet. 55, 56. Similarly, the Petition includes substantive analysis of MacLeod (Ground 3) in connection with only limitation 9[b] regarding dependent claims 9–11, 26–28, 37–39, 44–46, but Petitioner's list of Unpatentability Grounds omits claim 28 from Ground 3. *Compare* Pet. 2 *with* Pet. 64–66, 78–79.

"These factors are not exhaustive but are merely a guide to determining the level of ordinary skill in the art." *Daiichi Sankyo Co. v. Apotex, Inc.*, 501 F.3d 1254, 1256 (Fed. Cir. 2007). There is uncontested evidence in the record before us that reflects the knowledge and experience of a POSITA.

We determine no express finding is necessary, on this record, and that the level of ordinary skill in the art is reflected by the prior art of record. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001); *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995); *In re Oelrich*, 579 F.2d 86, 91 (CCPA 1978).

B. Claim Construction

We construe claim terms using "the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b)." 37 C.F.R. § 42.100(b). In this context, claim terms "are generally given their ordinary and customary meaning" as understood by a POSITA at the time of the invention. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–13 (Fed. Cir. 2005) (en banc). "Importantly, the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Id.* at 1313. The "specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id.* at 1315 (citation omitted). Accordingly, when construing a disputed claim limitation, "we look principally to the intrinsic evidence of record, examining the claim language itself, the written description, and the prosecution history, if in evidence." *DePuy Spine, Inc. v. Medtronic Sofamor Danek, Inc.*, 469 F.3d 1005, 1014 (Fed. Cir. 2006) (citing *Phillips*, 415 F.3d at 1312–17).

Petitioner and Patent Owner both apply the same district court construction of "*data group*" to support their respective arguments. Pet. 9–10 (citing Ex. 1007,

13–31; Ex. 1008, 10–19); Prelim. Resp. 10 n.1, 17 (citing Pet. 10; Ex. 1007, 24–25); *see also* Paper 7, 2 (Updated Mandatory Notices).[10] The parties each provided proposed constructions of "data group" and related arguments to the court. *Id.* The district court construed "*data group*" as "a group of data and a mechanism for identifying data from that group (e.g., a group identifier)." Ex. 1007, 23–25.

Having reviewed the district court's construction of "*data group*," and given the parties' application of that construction here, we apply the district court's construction of "*data group*" for purposes of this Decision. *See Wellman, Inc. v. Eastman Chem. Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011) (claim terms "need only be construed 'to the extent necessary to resolve the controversy'") (quoting *Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999)).

C. Asserted Obviousness of Claims 1–46 Over Pike or Pike in View of Chowdhuri (Grounds 1 and 2)

Petitioner argues that independent claim 1 of the '610 patent would have been obvious over Pike alone (Ground 1) or Pike in view of Chowdhuri (Ground 2). Pet. 15–51. Given the similarities in claim language, Petitioner addresses independent claims 17, 33, and 40 by providing brief cross-citations to its argument and evidence regarding claim 1, with some limited additional analysis provided for claim 33. *See* Pet. 51–52 (claim 17), 71–77 (claim 33), 77 (claim 40). All four independent claims recite similar language for map and reduce processing "*a plurality of data groups*" in a manner "*corresponding*" and/or "*identifiable*" to each "*data group*" or to a data partition of each "*data group*." We apply the district court's claim construction of "*data group*" and address whether Petitioner

[10] Petitioner filed Updated Mandatory Notices on February 27, 2024. Paper 7. Petitioner further advised that the co-pending litigation between Patent Owner and Petitioner pending in the Western District of Texas "was stayed pending *inter partes* review on February 15, 2024." *Id.* at 2.

has provided sufficient evidence and argument to establish a reasonable likelihood that Pike or Pike-Chowdhuri renders obvious the claimed MapReduce data processing method recited in claim 1 of the ’610 patent.

A patent claim is unpatentable under 35 U.S.C. § 103(a) if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the art at the time the invention was made. *See KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007) (quoting 35 U.S.C. § 103(a)). Obviousness is resolved based on underlying factual determinations, including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) objective evidence of nonobviousness, i.e., secondary considerations. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). A party who petitions the Board for a determination of obviousness must show that “a skilled artisan would have been motivated to combine the teachings of the prior art references to achieve the claimed invention, and that the skilled artisan would have had a reasonable expectation of success in doing so.” *Procter & Gamble Co. v. Teva Pharms. USA, Inc.*, 566 F.3d 989, 994 (Fed. Cir. 2009) (quoting *Pfizer, Inc. v. Apotex, Inc.*, 480 F.3d 1348, 1361 (Fed. Cir. 2007)). We assess Petitioner’s evidence and argument according to the above-referenced standard and in view of Patent Owner’s argument and evidence.

We begin with a discussion of Pike and Chowdhuri.

1. The Pike Reference (Ex. 1002)

Pike’s MapReduce system is designed to “efficiently handle the analysis of data records” in a large-scale, distributed data processing system that “includes allocating groups of records to respective processes of a first plurality of processes executing in parallel.” Ex. 1002, Abstract (code 57), 1:32–41. Pike discloses a

MapReduce method to process groups of data records "in a parallel and distributed processing environment." *Id.* at 1:22–25. Pike's system utilizes a set of "application-specific map(), reduce() and, optionally, partition() operators," and the work queue master "determines the number of map tasks and reduce tasks to be performed to process the input data." *Id.* at 5:40–44.

Petitioner provides a color-coded form of Pike Figure 3, reproduced below:

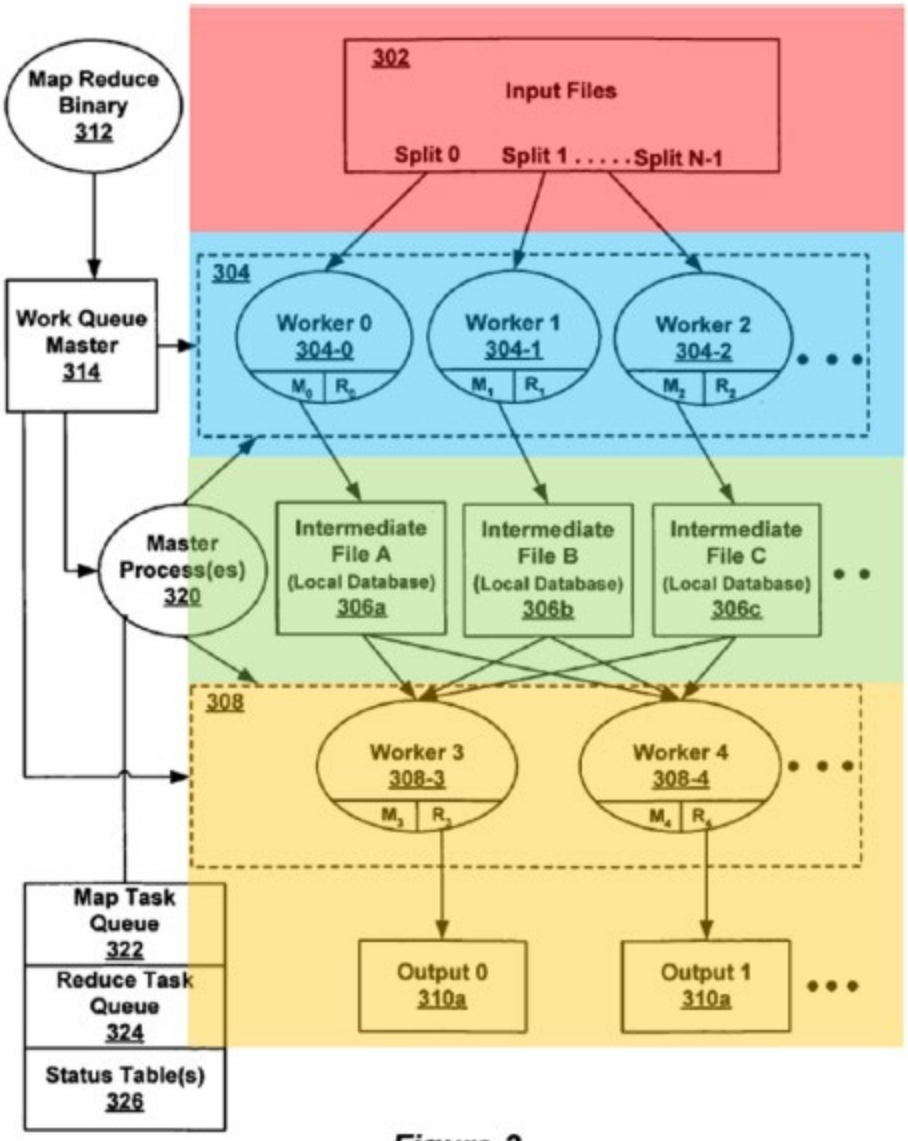


Figure 3

See Pet. 10–11. In Pike Figure 3, above, input files (302, highlighted in red) "'can include a variety of data types,' including 'text files' and 'tables.'" *Id.* at 10 (citing Ex. 1002, 9:56–10:7). The input files are split into "data blocks" (Split 0, Split-1 . . . Split N-1) "of either a specified or predefined size (e.g., 64 MB). Alternately, in some embodiments the input files have a predefined maximum size (e.g., 1 GB), and the individual files are the data blocks." Ex. 1002, 4:54–60; *see also id.* at 9:58–62 (user may "control the size of the data blocks"). Pike further discloses that a data block may comprise "a portion of an input file (e.g., where the portion comprises a data block) to be processed … [or] portions of two o[r] more input files." *Id.* at 6:59–62. In summary, Pike discloses a MapReduce parallel processing system where a "data block" to be processed is either i) an input file, e.g., a data table, having a predetermined maximum file size, ii) a portion of one input file that has been split into two or more data blocks of a specified or predefined size, or iii) comprised of portions of two or more input files of a specified, predefined, or maximum size.

Pike's input data blocks, moreover, "may in some embodiments be treated as key/value pairs," and each worker process (304-0, 304-1 . . . , highlighted in blue) "applies the application-specific map( ) operator to the respective input data block so as [to] generate intermediate data values." *Id.* at 9:65–66, 10:4–7; *see also* Pet. 10 (citing and partially quoting same). The intermediate data values "are collected and written to one or more intermediate files" (306a, b, and c, highlighted in green). Ex. 1002, 10:8–9. Each worker process (308-3, 308-4 . . . , highlighted in yellow) "sorts the intermediate data values . . . in accordance with the key of the key/value pairs in the intermediate data [and] also merges or otherwise combines the sorted intermediate data values having the same key … [using] an application-specific reduce( ) operator." *Id.* at 11:28–38; *see also* Petition 10–11 (citing same).

Pike further discloses a Task Status Table "for keeping track of the status of map and reduce tasks" in Figure 7A, reproduced below:

Task Status Table

Task ID	Status	Process	Input Files	Output Files	...
Map0000	Completed	P0000	12340	12341	...
Map0001	Completed	P0001	12344	12345	...
⋮	⋮				
Map0103	In Progress	P0010	12030	12031	...
Map0104	Failed	P0011	10101	12102	...
Map0103b	In Progress	P0010	12030	12031	...
⋮	⋮				
Red0000	In Progress	P0033	12340, ...	14000	...
Red0001	Waiting	-	-	-	...

Figure 7A

Id. at 6:55–56, Fig. 7A. Figure 7A, above, depicts "each task (e.g., map, reduce) is assigned [a] task ID, a status, a process, and one or more input files and output files." *Id.* at 6:57–58. In Pike's system, a map or reduce "process retrieves data from one or more input files (or the one or more input file portions) identified in the input field and writes the results . . . to one or more output files identified in the output field." *Id.* at 6:65–7:2. The Task Status Table shows the current status of the task being performed by the assigned process (Completed, In Progress, Failed, Waiting) and the input and output files for each Task ID are identified by number. *Id.* For example, Task ID Red0000 "is assigned to process P0033, which is still in progress" and "retrieves data blocks from input file 12340 **. . .** and writes the results of the task to output file 14000." *Id.* at 7:2–6.

2. The Chowdhuri Reference (Ex. 1003)

Chowdhuri discloses a system of "parallel query optimization using semantic-based partitioning" to process data, such as relational database tables. Ex. 1003, Abstract (code 57). Chowdhuri uses "a divide and conquer technique for improving query performance, by partitioning data into smaller fragments and operating on those fragments concurrently (i.e., in parallel)." *Id.* ¶ 120. Chowdhuri uses "data and index partitions" and iterators[11] to process "different data sets located across different storage units," depicted in Figure 3B, reproduced below:

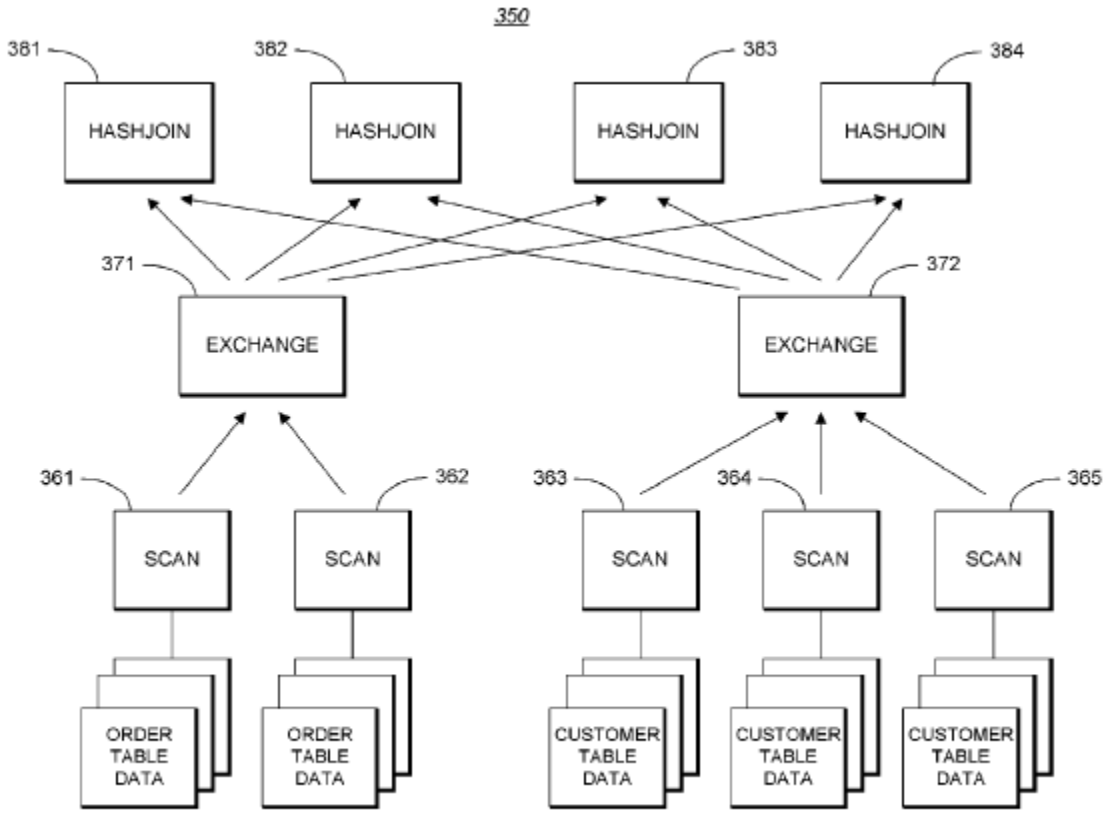


FIG. 3B

[11] "Iterators" are "self-contained software objects that accept a stream of rows from one, two, or multiple data sources," e.g., "scan" and "hashjoin." Ex. 1003 ¶ 130.

Id. ¶¶ 126, 135, Fig. 3B.

Chowdhuri Figure 3B, shown above, depicts a parallel data processing operation where the desired output is customer orders by state. *Id.* ¶¶ 126–127. "[A]n order table (data) is scanned in parallel by two threads, 361 and 362, while the customer table is scanned in parallel by three threads 363, 364, and 365, respectively." *Id.* ¶ 135. Chowdhuri's parallel scanning of each data table reflects a form of data partitioning. *Id.* ¶ 128 ("if the customer table is partitioned across several disk devices … the database can initiate multiple scans to be performed in tandem"); *see also* Ex. 1005 ¶ 102 ("data from the customer and order tables are each partitioned and scanned by individual scan iterators"). The scanned data from the order and customer tables is "repartitioned on-the-fly . . . by a routing function [Exchange operator]" that redistributes the data to four separate hashjoin iterators operating in parallel. *Id.* The hashjoin iterators perform a "join" operation to match the orders that belong to a particular customer (join the appropriate rows of data from each table), after which the customer orders are grouped by state, the order amounts summed for each state, and sorted to "appear in the order of the sum." *Id.* ¶¶ 127, 131.

3. "data group" analysis of Pike (Ground 1) under 35 U.S.C. § 103

a. Petitioner's "data group" arguments

Petitioner argues that Pike renders claims 1–46 obvious under 35 U.S.C. § 103. Pet. 2, 15–79. After acknowledging the district court's construction of "*data group*" for a second time, Petitioner asserts that Pike (and Pike-Chowdhuri) teaches or renders obvious the "*data group*" feature of the claimed invention "under both its plain meaning and the district court's construction." Pet. 16–17. Petitioner, however, does not present an alternative "plain meaning" of "*data group*" to that provided by the district court. Therefore, we analyze Grounds 1–3

based on the district court's construction of "*data group*"—a group of data and a mechanism for identifying data from that group (e.g., a group identifier)—in the context of claim 1 and consistent with specification of the '610 patent.

Petitioner argues that because Pike's input files include "text files" and "tables," a POSITA would have "recognized that Pike's input files teach or render obvious" the use of a "*data group*" identifier, such as a file name. *Id.* at 17 (citing Ex. 1005 ¶¶ 88–91). Petitioner, citing the Linn Declaration, asserts that a file name is an "identifier associated with the data in the file, and typically includes information about the content, type, and structure of the data in each file." *Id.* (citing Ex. 1005 ¶ 91). Referencing the Employee and Department tables described in the '610 patent, but not in Pike, Petitioner asserts that a POSITA would have understood that such input files would include a name—such as "Employee"—and a file extension of the type described in the '610 patent, but not in Pike. *Id.* at 17–18 (citing Ex. 1005 ¶ 92). Petitioner argues that a file extension indicates the "file type or structure (e.g., an extension such as '.csv' indicating a 'comma separated file'), a common format for large-scale data processing." *Id.* Petitioner reasons, again by analogy to the Employee and Department tables described in the '610 patent, but not in Pike, that Pike's input files would have "similar identifiers." *Id.* at 17–18.

Petitioner also argues that, at a minimum, a POSITA would have found it obvious to provide a "*data group*" identifier for Pike's input files "because it would have allowed for locating that data group in Pike's system and understanding the data in that file." *Id.* at 18 (citing Ex. 1005 ¶ 93). Petitioner contends such data group identifiers would have "enabled a MapReduce programmer to identify input files and write 'application-specific' map and reduce

functions on those files." *Id.* Petitioner concludes by highlighting the phrase "Input Files" in Pike Figure 3 without further explanation. *Id.*

Petitioner later asserts that Pike discloses the claim limitation of map processing each "*data group*" "to form *corresponding* intermediate data for that *data group* and *identifiable* to that *data group*." Pet. 30 (citing Ex. 1002, 13:9–21, Fig. 3 (item 320); Ex. 1005 ¶ 115). Petitioner argues that Pike discloses a "master process" that maintains a Task Status Table (Fig. 7A) to track processing of "each input file using identifiers through mapping and reducing," where the "map and reduce tasks are tracked according to 'Input Files' (i.e., Pike's 'data groups')." *Id.* at 30–31 (citing Ex. 1002, 13:7–21, Figs. 3, 7A; Ex. 1005 ¶¶ 115–116). Referring to Pike Figure 7A, Petitioner points out that a map task and reduce task are both "associated with input file 12340," the intermediate data from the map task is stored in output file 12341, and the output of the reduce task is stored in output file 14000. *Id.* at 31–32 (citing Ex. 1005 ¶¶ 116–117); Ex. 1002, Fig. 7A. Petitioner reasons that a "POSITA would have understood" that Pike's input file "'12340' is an identifier that associates intermediate data (e.g. output file '12341') with a particular input file." *Id.* at 32 (citing Ex. 1005 ¶ 117).

Petitioner argues in the alternative that, to the extent "Pike does not expressly disclose outputting corresponding intermediate data for that data group and identifiable to that data group (Pike does)," it would have been obvious for a POSITA to "include an identifier associated with Pike's input files to track them through the map and reduce tasks." *Id.* at 32 (citing Ex. 1005 ¶ 118). Petitioner argues a POSITA would have been motivated to modify Pike to "track data from particular input files through map and reduce processing" because Pike's task status tables "suggest doing so." *Id.* Petitioner further posits that the proposed modification was a "known technique" yielding a "predictable result" that would

have been used "to track which output data from the map and reduce processes was derived from a particular file." *Id.* at 32–33 (citing Ex. 1005 ¶ 119). Petitioner reasons that a POSITA "could" have used the tracking information in a "post-processing operation" to improve the efficiency of the system by allowing selective reduction of a data set "without requiring the desired subset of input files to be re-mapped." *Id.* at 33 (citing Ex. 1005 ¶ 119).

Petitioner further argues that Pike's worker processes render obvious the claim limitation of processing the intermediate data "for each *data group* in a manner that is defined to *correspond to that data group*" ("reducing" step). *Id.* at 48–49. Petitioner argues that Pike processes intermediate data by using "an application-specific reduce( ) operator." *Id.* (citing Ex. 1002, 11:28–55). Petitioner reasons a POSITA "would have understood" that reducing the intermediate data "based on the input file of Pike" would have been "an obvious implementation of Pike's application-specific data processing operators." *Id.* (citing Ex. 1005 ¶ 148).

b. Patent Owner's "data group" arguments

Patent Owner argues that Pike does not render obvious the claimed "*data group*," because Pike does not teach "'a mechanism for identifying *data*' from any particular input file." Prelim. Resp. 37 (quoting district court claim construction, with emphasis). Patent Owner argues that Pike discloses using the input file name only to identify "the location of the data to be processed," not to identify the *data* in a file in order to execute map and reduce processing based on the file name (the asserted "*data group*" identifier). *Id.* (citing Ex. 1002, 4:34–40). Patent Owner further argues that Pike does not discuss the use of file extensions to identify input data files "at all." *Id.* at 38 n.6.

Patent Owner notes that Pike's work queue master "'assign[s] tasks to processes'" and tracks the status of the map and reduce tasks to distribute the processing load. *Id.* at 39 (citing Ex. 1002, 4:7–11). Patent Owner argues that what Pike does not do is disclose the use of identifiers associated with different "*data group*" schemas to accomplish map and reduce processing on a "*data group*" basis, because "Pike is unconcerned with the *type* of data being processed." *Id.* Patent Owner further argues that Petitioner, who asserts a POSITA would have found it obvious to use input files as "*data groups*" and write "application-specific" map and reduce functions based on data identifiable to those input files, "is using the claims as a roadmap." *Id.* Patent Owner emphasizes that Pike is not focused on the type of data being processed, because Pike is not applying map and reduce operators to "data of different schema to merge heterogenous data" as described and claimed in the '610 patent. *Id.* Patent Owner further emphasizes Pike's disclosure regarding application-specific programming is limited to teaching an operator "to process the input data one record at a time," or "to generate its output data in the form of key/value pairs." *Id.* at 39–40 (quoting Ex. 1002, 7:31–36).

Patent Owner also argues that Pike's task status table (Figure 7A) does not disclose or render obvious the claim limitation of map processing input data "to form *corresponding* intermediate data for that *data group* and identifiable to that *data group*." *Id.* at 44. Patent Owner asserts that although Pike "tracks which files are allocated to and generated by particular processes, there is no indication in Pike that the 'intermediate *data*' itself is 'identifiable' to a given 'input file.'" *Id.* (emphasis in Prelim. Resp.). Patent Owner further asserts that such an identification is inconsistent with Pike's disclosure and teaching regarding the task status table of Figure 7A, because a given map process "retrieves data from ***one or***

more input files (or the one or more input file portions) identified in the input field and writes the results of the task ID to ***one or more output files*** identified in the output file field." *Id.* at 45 (quoting Ex. 1002, 6:65–7:2) (emphasis in Prelim. Resp.). Patent Owner reasons that, because Pike teaches a given map process receives data originating from different input files and then outputs the intermediate data "into an 'output file' with a file name that is *different* from that of the 'input files,'" the intermediate data loses its identification with the input files. *Id.* at 45–46 (citing Pike Fig. 7A) (emphasis in Prelim. Resp.).

Patent Owner further draws our attention to an asserted inconsistency in Petitioner's argument that "it would have been obvious to include an identifier associated with Pike's input files to track them through the map and reduce tasks," because "Pike's status tables [] suggest doing so." *Id.* at 46 (quoting Pet. 32). Patent Owner argues:

> But this is an about-face—indeed, Petitioner previously alleged that Pike already discloses such an identifier, i.e., Pike's "file name," which is the purported "mechanism for identifying data.". . . . Petitioner does not allege here that the "file name" is the "identifier" to track data "through the map and reduce tasks" and instead argues that it would just be obvious to include one. That is because the "file name" *cannot* serve as this mechanism for two reasons: (1) Pike's "data blocks" (purported "partitions") can be entire "input files," parts of "input files," or parts of multiple "input files," meaning that a file name from just one of these input files cannot identify all of the data within a data block; and (2) Pike's "intermediate data" *loses* its identification with its respective "input files" as discussed above, meaning that the original "file names" of the "input files" are of no moment to the actual data processing taking place in Pike.

Prelim. Resp. 46 (citation omitted). Patent Owner concludes that Petitioner is relying on improper hindsight gleaned from the '610 patent itself, because Pike's "status tables only reveal that Pike tracks which 'output file' is generated from

processing given 'input files,' not that any data itself in that file is identifiable to any original 'input file'." *Id.* at 47.[12]

c. Analysis of Pike's input files as asserted "data groups"

Pike is concerned with controlling MapReduce processing based on input file size, location, and processing status, rather than identifying and processing different input data types or "*schemas*," i.e., "*data groups*" as construed by the district court and described and claimed in the '610 patent. Prelim. Resp. 37–47 (citing Pike Ex. 1002, 4:1–11, 4:34–40, 4:54–59, 6:55-7:2, 7:31–36, Figs. 2, 3, 7A). As Dr. Linn acknowledges, in Pike "the user may set an optional flag, or specify a parameter, so as to control the size of the data blocks into which the input files are divided." Ex. 1002, 9:56–10:7 (cited in Linn Dec. Ex. 1005 ¶ 72); *see also* Ex. 1002, 4:54–56 ("input files are split into multiple data blocks . . . of either a specified or predefined size (e.g., 64 MB)"). Although Pike teaches that input files for MapReduce processing can include a variety of data types, such as text files and tables, Pike teaches organizing input files by size, not by identifying different types, origins, or schemas of input data. *See* Prelim. Resp. 48–49 ("Pike does not even use the term 'schema';" "schema of the data processed by Pike's architecture is irrelevant").[13] Pike teaches controlling input files by size to distribute the processing load and "fully utilize the aggregate storage space and disk bandwidth of the data processing system." Ex. 1002, 4:54–64; *see also*

---

[12] Patent Owner relies on similar reasoning to argue Pike does not disclose or render obvious the "reducing" claim limitation for processing intermediate data "in a manner that is defined to *correspond to that data group*." Prelim. Resp. 53–55.

[13] The fact that Pike's MapReduce system can "accept various data sets having different formats or schemas[] from different sources" (Linn Dec. Ex. 1005 ¶ 75), without more, does not amount to a teaching, suggestion, or motivation to form data groups characterized by different schemas and provide a mechanism for identifying the data from each group for separate map and reduce processing.

Prelim. Resp. 24 (Pike focused on "parallelizing of mapping and reducing operations to distribute the processing load.").

i. Pike's posited input file names as "data group" identifiers

The'610 patent describes two examples for providing the claimed "*data group*" identifiers. One example describes the use of programming code so the identifiers "can be passed as a parameter to a mapping function and thereby travel through to the resulting intermediate data, followed by a reduce function that employs the 'data group' identifier to call particular programming objects (e.g., iterators) associated with specific 'data groups.'" Prelim. Resp. 11–12 (citing Ex. 1001, 6:29–33, 6:39–48). Patent Owner highlights the "emp" and "dept" names as "*data group*" identifiers in the '610 patent's programming code example for map and reduce functions, reproduced below:

```
map(string group, string key, string val) {
    // group: "emp" or "dept"
    // key: DeptID
    // val: LastName for "emp", DeptName for "dept"
    DeptID = key;
    emit_to_group(group, DeptID, val);
}
```

```
reduce(hashtable iterators) {
    // iterators: one iterator for "emp", another for "dept"
    DeptID = iterators.get_key( );
    emp_iter = iterators["emp"].get_iter( );
    dept_iter = iterators["dept"].get_iter( );
    for each LastName in emp_iter do {
        for each DeptName in dept_iter do {
            emit_to_group("emp_dept", DeptID, (LastName,
            DeptName));
        }
    }
}
```

Id. The programming code, above, illustrates each "*data group*" identifier—"emp" and "dept" names—for controlling the map and reduce functions.

Another example from the '610 patent provides execution instructions to indicate the "*data group*" identifiers "such that different 'data groups' can be mapped differently and/or processed in a way that corresponds to a particular 'data group'." *Id.* at 12 (citing Ex. 1001, 7:45, 7:55, 7:64–65, Figs. 3–5). We reproduce the execution instructions below, in highlighted form by Patent Owner:

map("emp", 34, "Smith")	map("dept", 34, "Clerical")

reduce((34, ("emp", ("Smith", "Robinson", "Jasper")), ("dept", ("Clerical"))))

Id. As annotated by Patent Owner above, the "*data group*" identifiers (underlined in red) are "emp" and "dept." The "*data group*" identifiers in the '610 patent identify input data sets with file names for different schemas or data types, which allows a programmer to program and execute MapReduce functions that correspond to a particular "*data group*" when joining two different types of input data based on a common key. *See* Ex. 1001, Fig. 5.

Petitioner does not cite a comparable disclosure, teaching, example, or suggestion in Pike (or another MapReduce reference) to use an input file name as a "*data group*" identifier for programming MapReduce functions. Petitioner instead refers to the '610 patent itself for the use of file names to identify Pike's otherwise unidentified "tables" as input file "*data groups*":

> As an example, for an input file in Pike's system, *such as the "Employee table" as contemplated by the '610 patent*, a POSITA would have understood the *input file would include, for example*, *a name (e.g., "Employee")* and an extension indicating its file type or structure (e.g., an extension such as ".csv" indicating a "comma separated file," a common format for large-scale data processing). . . . Pike's input files correspond to input tables in the '610 patent because a *POSITA would have understood that the Employee and Department tables of the '610 patent* would be stored as files and have *similar identifiers*. **EX-1001 ['610 patent]**, 4:1-5:18, 7:18-23, 8:15-46, Figs. 4, 5; EX-1005 ¶92. Pike therefore teaches a data set comprising data groups.

Pet. 17–18 (emphasis added). Dr. Linn's Declaration paragraph 92 (Ex. 1005) also cites the '610 patent for the asserted teaching of "*data group*" identifiers in Pike.

Petitioner's argument explicitly and implicitly relies on the teachings of the '610 patent, not Pike or some other MapReduce reference or teaching, for the proffered motivation that a "POSITA would have understood," "recognized," or "found it obvious to implement" Pike's MapReduce system to include input file names as the "*data group*" identifiers in claim 1 of the '610 patent. *See* Pet. 17–18.

Petitioner relies on conclusory reasoning grounded in hindsight to assert a POSITA would have found it obvious to include file names to identify the data in each of Pike's input files "because it would have allowed for locating that data group in Pike's system and understanding the data in that file." *Id.* at 18 (citing Ex. 1005 ¶ 93).[14] Petitioner then makes the leap that, by understanding the data in an input file, a MapReduce programmer would have had reason "to identify input files and write 'application-specific' map and reduce functions on those files." *Id.* (citing Ex. 1002, 10:1–7, 11:28–47, 14:12–27; Ex. 1005 ¶ 93). The difficulty with Petitioner's position is that neither Pike nor other cited references disclose, teach, suggest, or otherwise provide motivation to write application-specific map and reduce functions that process each input file by identifiable schema or data type, i.e., by "*data group*" as recited in claim 1. Pike teaches application-specific programming "to process the input data one record at a time" or "to generate its output data in the form of key/value pairs," nothing more. Ex. 1002, 7:31–36; *see* Pet. 27 (citing same); Prelim. Resp. 39–40, 49–50, 55 (citing same).

Petitioner's posited file names for Pike's input files, moreover, could not serve as the claimed "*data group*" mechanism to identify the data in each input file by schema (or data type) for separate map and reduce processing. Each of Pike's data blocks can include data from more than one input file, a portion of one input

[14] Linn Declaration paragraph 93 is substantially identical to Petitioner's argument and does not add any further insight or analysis on this point.

file, or an entire input file. Ex. 1002, 6:55–7:6; *see also* Prelim. Resp. 46 ("Pike's 'data blocks' (purported 'partitions') can be entire 'input files,' parts of 'input files,' or parts of multiple 'input files'"). Therefore, a file name for one input file could not uniquely identify all of the data within a given data block, which could include data from different input files. *Id.* We agree with Patent Owner's reasoning and argument on this point as well.

In sum, Petitioner fails to show sufficiently that a POSITA would have had reason or motivation to use input file names in Pike as "*data group*" identifiers, as recited in claim 1 of the '610 patent. *See KSR*, 550 U.S. at 418 (obviousness requires "some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness").

ii. Pike's Task Status input files as "data group" identifiers

Pike's concern with controlling file size and distribution of the parallel processing load helps explain why Pike uses task status file identifiers to identify input file location and processing status. The location and status information allows Pike's work queue master to distribute the processing load and improve efficiency. *See, e.g.*, Ex. 1002, 5:46–52 ("the work queue master module generates a task status table having entries representing all the tasks to be performed, and then begins assigning those tasks to idle processes … based on a resource allocation scheme"), 9:26–28 ("the work queue master may undertake remedial measures to repair, restart or replace a failed worker process"). Pike's Track Status Table (Fig. 7A), reproduced below, uses numerical identifiers to locate and distribute input files for processing and "for keeping track of the status of map and reduce tasks":

Task Status Table

Task ID	Status	Process	Input Files	Output Files	...
Map0000	Completed	P0000	12340	12341	...
Map0001	Completed	P0001	12344	12345	...
⋮	⋮				
Map0103	In Progress	P0010	12030	12031	...
Map0104	Failed	P0011	10101	12102	...
Map0103b	In Progress	P0010	12030	12031	...
⋮	⋮				
Red0000	In Progress	P0033	12340, ...	14000	...
Red0001	Waiting	-	-	-	...

Figure 7A

Ex. 1002, 6:55–56 (yellow highlight added). Pike's Task Status Table tracks which (Input and Output) files are allocated to and generated by particular processes (under Task ID) and the status of map and reduce functions (e.g., Completed, In Progress, Failed, Waiting). Pike is silent as to whether map-processed intermediate data stored in an output file is identifiable to an original input file.

Petitioner argues that Pike's Task Status Table discloses the claim limitation of map processing each "*data group*" partition "to form *corresponding* intermediate data for that *data group* and *identifiable to that data group*." Pet. 30–33. Contrary to Petitioner's argument, Patent Owner persuasively argues there is no indication that Pike tracks map-processed intermediate *data* as "*corresponding*" and "*identifiable*" to the data in a particular input file. Prelim. Resp. 44–46. Pike discloses that "[t]he *input files field* [Fig. 7A] may specify a portion of an input file (e.g., where the portion comprises a data block) to be processed by the task, or this field may specify portions of *two o[r] more input files*." Ex. 1002, 6:59–62,

Fig. 7A (emphasis added). Pike further discloses that a map or reduce process "'retrieves data from ***one or more input files*** . . . and writes the results of the task ID to ***one or more output files***.'" Prelim. Resp. 45 (quoting Ex. 1002, 6:65–7:2) (emphasis in Prelim. Resp.). An "input file" field that includes data from multiple input files does not teach or suggest the claimed "*data group*," because it includes data from different sources. Pike's task status system does not identify and partition input file *data* based on a characteristic schema or data type, which means it cannot generate map-processed intermediate data "*corresponding*" and "*identifiable*" to a particular schema of data in an input file ("*data group*"), as claimed in the '610 patent. Pike's Task Status Table only tracks the *status* of map and reduce tasks and the processing *location* of input and output files by file number.

Pike further teaches outputting processed data into one or more output files having different file numbers from the numbers assigned to the input files. Therefore, map-processed data "*loses* its identification with its respective 'input files' because all 'intermediate data' generated by a given process, regardless of where it originated from, is simply entered into a single 'output file' with a new file name." Prelim. Resp. 45–46 (emphasis in Prelim. Resp.). Pike's lack of attention to identifying map-processed data as "*corresponding*" and "*identifiable*" to particular input file *data* is consistent with Pike's express disclosures and emphasis on controlling file size, location, and processing status. We agree with Patent Owner's argument on this point.

We are persuaded also by an inconsistency in Petitioner's conclusory argument that it would have been obvious "to include an identifier associated with Pike's input files to track them through the map and reduce tasks" because "Pike's status tables [] suggest doing so . . . [as] a known technique of tracking data . . . to

yield a predictable result." Pet. 32 (citing Ex. 1005 ¶ 118). We agree with Patent Owner it is inconsistent for Petitioner to assert first that the input *file names* (with file extension) are the claimed "*data group*" identifiers (Pet. 17 (citing Ex. 1005 ¶¶ 88–92)), but then argue the use of *file numbers* in Pike's Task Status Table (Fig. 7A) was a known technique to "track data from particular input files through map and reduce processing." Pet. 32–33 (citing Ex. 1005 ¶¶ 117–119); Prelim. Resp. 46. Petitioner fails to explain how an input file *name* and file extension identifies the data in a "*data group*" and tracks the data through the map and reduce processes depicted in Pike's Task Status Table (Figure 7A). *Id.*; *see also* Prelim. Resp. 46 ("Petitioner does not allege here that the 'file name' is the 'identifier' to track data 'through the map and reduce tasks' and instead argues that it would just be obvious to include one."). We are left with Petitioner's unsupported assertion that Pike somehow uses input file *names* as a mechanism to identify and track different groups (schemas) of input file data through separate map and reduce processes, which is inconsistent with Pike's use of file *numbers* in the Task Status Table.

We further agree with Patent Owner that Petitioner's asserted motivation for tracking processed data "derived from a particular [input] file" to improve efficiency "by enabling a data set to be selectively reduced after it is mapped" (Pet. 33 (citing Ex. 1005 ¶ 119)), is speculative hindsight-based reasoning. Petitioner's posited motivation is grounded in benefits residing in the '610 patent, not in the prior art or knowledge of a POSITA. Prelim. Resp. 47. Petitioner's evidence—Dr. Linn's naked opinion that a POSITA "could" have used the identification of processed output file data as "derived from a particular file" to selectively reduce intermediate data (Ex. 1005 ¶ 119)—is unpersuasive for the reasons given above and legally insufficient. *See Belden Inc. v. Berk-Tek LLC*, 805

F.3d 1064, 1073 (Fed. Cir. 2015) ("[O]bviousness concerns whether a skilled artisan not only *could have made* but *would have been motivated to make* the combinations or modifications of prior art to arrive at the claimed invention.").

For the reasons given above, we determine Petitioner has not established that Pike's Task Status Table discloses, teaches, suggests, or otherwise provides motivation to use an input file name or number as a "*data group*" identifier—a mechanism for identifying data from that group—as recited in claim 1 of the '610 patent.

iii. Conclusion

In sum, we are not persuaded by Petitioner's argument and evidence that there is a reasonable likelihood Pike discloses, teaches, suggests, or otherwise renders obvious a mechanism for identifying the data in each of "*a plurality of data groups*" in order to execute "mapping functions … to form *corresponding* intermediate data for that *data group* and *identifiable to that data group* …; and reducing the intermediate data … for each *data group* in a manner that is defined to *correspond to that data group*," as recited in claim 1 of the '610 patent. Because Petitioner relies on the same analysis in support of its challenge to independent claims 17, 33, and 40 based on Pike, our determination applies equally to those independent claims and the dependent claims. Therefore, we determine Petitioner has not established a reasonable likelihood of prevailing in its challenge to the patentability of claims 1–46 of the '610 patent as obvious over Pike.

4. Analysis of Pike-Chowdhuri (Ground 2) under 35 U.S.C. § 103

a. Petitioner's "data group" arguments regarding Pike-Chowdhuri

Petitioner argues that "Chowdhuri's order and customer tables teach data groups because Chowdhuri's table names provide a mechanism for identifying the

data in the table (e.g., order and customer data in table format)." Pet. 18 (citing Chowdhuri Ex. 1003 ¶¶ 126–128, 131, 135, Figs. 3A, 3B; Linn Dec. Ex. 1005 ¶ 94). Petitioner contends that the combination of Pike-Chowdhuri "would have involved nothing more than the simple substitution of Chowdhuri's input tables for Pike's input files, with predictable results." *Id.* at 19 (citing Ex. 1005 ¶ 95). Petitioner also contends that "Pike's suggestion of processing input 'tables'" would have motivated a POSITA to use Pike's MapReduce system to process Chowdhuri's order and customer data tables. *Id.* at 19–20 (citing Ex. 1002, 10:1–4, 2:22–26; Ex. 1003 ¶¶ 126–127, 131–135; Ex. 1005 ¶¶ 95–96 ("A POSITA also would have been motivated to map and reduce Chowdhuri's tables in Pike's system 300").

Petitioner also argues that a POSITA would have been motivated to combine Pike and Chowdhuri "because Chowdhuri teaches and suggests performing similar MapReduce operations on its data tables—scanning and merging its customer and order tables to identify orders from a specific 'state.'" *Id.* at 19–20 (citing Ex. 1005 ¶ 96; Ex. 1003 ¶¶ 126–127, 131–35). Petitioner posits that Chowdhuri's scan and hashjoin functions "correspond to Pike's map and reduce tasks," and that a POSITA "would have looked to Pike to map and merge Chowdhuri's tables using Pike's distributed system to, for example, leverage Pike's distributed system and its fault tolerant task scheduling." *Id.* at 20 (citing Ex. 1002, 4:28–5:33, 6:37–54, 6:55-7:25, 11:57-13:3; Ex. 1005 ¶¶ 96–97). Petitioner then argues that a Pike-Chowdhuri combined system also would have been used to process Chowdhuri's order and customer data tables by further using "Chowdhuri's scanning, hashjoin, GroupBy, and sort iterators as map and reduce tasks in Pike, while employing Pike's fault-tolerant architecture to do so." *Id.*

Petitioner further asserts that Chowdhuri discloses the claim limitation of map processing each partitioned "*data group*" "to form *corresponding* intermediate data for that *data group* and *identifiable* to that *data group*." Pet. 34–35. Petitioner argues that Chowdhuri discloses "[i]dentifiers 'o' and 'c' [that] correspond to Chowdhuri's order and customer tables, respectively, and identify the data from each table through the scan, join, and group processes." *Id.* at 34 (citing Ex. 1005 ¶ 121). Petitioner argues that a POSITA would have understood that the output of Chowdhuri's scan iterator, which Petitioner equates to a map function, is "identifiable to its input tables because the results are logically merged by customer state and customer order based on different input data to yield orders by state." *Id.* (citing Ex. 1003, ¶¶ 126–127, 131, 135; Ex. 1005 ¶¶ 121). Petitioner reasons the "customer state and customer order are therefore identifiable to each input table." *Id.* (citing Ex. 1005 ¶ 121).

Petitioner further contends it would have been obvious for a POSITA "to implement Chowdhuri's table identifiers through Pike's map and reduce processes." *Id.* (citing Ex. 1005 ¶ 122) (emphasis added). The combined system, according to Petitioner, would permit merging Chowdhuri's customer and order tables "based on a common key (e.g., by "customer_id"), as Chowdhuri suggests." *Id.* Petitioner contends that "tracking input tables as Chowdhuri suggests" would have allowed for selective postprocessing operations to improve processing efficiency. *Id.* at 34–35 (citing Ex. 1005 ¶ 122). Petitioner concludes that a POSITA would have scanned and merged Chowdhuri's "input tables in Pike, as Chowdhuri suggests." *Id.*

Petitioner also argues that Chowdhuri's "joining, grouping, and sorting by customer and order 'state'" renders obvious the claim limitation "reducing the intermediate data for the *data groups* … including processing the intermediate data

for each *data group* in a manner that is defined to *correspond to that data group*." *Id.* at 49–50. Petitioner reiterates the argument that a POSITA would have understood Chowdhuri "processes the results of the scanning iterators according to which table the data came from" in order to join, group, and sort each customer's orders by state and amount. *Id.* at 49 (citing Ex. 1005 ¶ 149). Petitioner argues that Chowdhuri's hashjoin, GroupBy, and sort iterators correspond to Pike's reduce functions and are "indistinguishable" from the '610 patent's reduce functions. *Id.* at 49–50 (citing Ex. 1005 ¶ 150). Petitioner concludes that a POSITA would have been motivated to use Chowdhuri's hashjoin, GroupBy, and sort iterators in Pike's MapReduce system "to leverage Pike's distributed system and fault tolerant task management." *Id.* at 50 (citing Ex. 1005 ¶¶ 151–152).

b. Patent Owner's "data group" arguments regarding Pike-Chowdhuri

Patent Owner argues that Chowdhuri does not render obvious the claimed "*data groups*," because Chowdhuri does not disclose, teach, or suggest the use of its order and customer table names as "mechanisms for identifying data from those tables in the course of processing such data." Prelim. Resp. 56. Patent Owner contends "there is no disclosure in Chowdhuri of any sort of identifier (table name or otherwise) being, e.g., passed to specific functions to guide group-specific processing as in the '610 patent." *Id.* at 56–57 (citing Ex. 1001, 6:29–33, 6:39–48, 7:45, 7:55, 7:64–65, Fig. 5 (programming code and execution instructions for "emp" and "dept" group identifiers in map and reduce functions)). Patent Owner emphasizes Chowdhuri's description of the customer "c" and order "o" in the example DSS query as referring to "two tables, the customer table and the order table, are being scanned," not as mechanisms for identifying data in the input customer and order tables for group-specific processing (asserted "*data groups*").

Id. at 57 (citing Ex. 1003 ¶¶ 126–127). Patent Owner cites Chowdhuri's disclosure of its scan function iterator as scanning input data "'rows from a table on disk,'" to distinguish scanning from Pike's map function. *Id.* at 58 (citing Ex. 1003 ¶ 61).

Patent Owner also argues that Chowdhuri does not disclose or render obvious the '610 patent claim limitation of map processing each partitioned "*data group*" "to form *corresponding* intermediate data for that *data group* and *identifiable* to that *data group*." *Id.* at 62–65. Patent Owner first points out the Chowdhuri scan function does not "correspond" to a map function and does not transform input data tables "to form corresponding intermediate data" (new key-value pairs), as asserted by Petitioner. *Id.* at 62–63 (citing Ex. 1003 ¶ 61; Pet. 26). Patent Owner next argues that Petitioner fails to explain how the asserted "c" and "o" "*data group*" identifiers in Chowdhuri "actually identify the purported 'intermediate data' output from the scan iterators" as "*corresponding*" to a particular input table during data processing. *Id.* at 63. Patent Owner emphasizes the "c" and "o" represent the customer and order tables "*within the query itself*, not within the scanners or hashjoin functions." *Id.* at 63–64 (emphasis in Prelim. Resp.).

Patent Owner further emphasizes "Chowdhuri never discloses that 'o' and 'c' are passed as identifiers to any functions that process the data in Chowdhuri, so 'o' and 'c' cannot be 'mechanisms for identifying data' from 'data groups.'" *Id.* at 64. Patent Owner argues that Chowdhuri does not form "*intermediate data*" "*identifiable*" to a particular "*data group*" (data table) as required by the claims of the '610 patent. *Id.* According to Patent Owner, the logical merging of selected rows of data from Chowdhuri's customer and order tables, asserted by Petitioner to disclose processed data "*identifiable*" to each "*data group*," "says nothing of *how* that merger is accomplished, much less that any sort of 'a mechanism for

identifying data' from a particular 'data group' is used for accomplishing such merger." *Id.* (emphasis in Prelim. Resp.).

Patent Owner further argues that "Chowdhuri does not need a 'mechanism' to identify which parent data group intermediate data came from because Chowdhuri accomplishes data consolidation *via the same function*, i.e., the hash join." *Id.* (emphasis in Prelim. Resp.). Unlike the '610 patented method, Patent Owner emphasizes Chowdhuri consolidates data in a "self-contained" hashjoin function that works on "*the original input data*" from the scanners (after distribution by the exchange operators), not on "data that has been transformed into 'intermediate data'." *Id.* at 64–65 (emphasis in Prelim. Resp.). Patent Owner reasons that because Chowdhuri does not generate intermediate data and "avoids movement of hash-mapped data to other function phases entirely," Chowdhuri "needs no such identifier." *Id.* at 65.

c. Pike-Chowdhuri "data group" analysis

Chowdhuri teaches a parallel query optimization method that uses "semantic-based partitioning" to partition and process data in parallel. Ex. 1003 ¶ 6. Chowdhuri's system uses a query execution scheme and functional operators (iterators) to process data, such as by scanning, joining, grouping, and sorting input data from a customer table and order table to produce customer orders by state and amount. Pet. 12–14 (citing Ex. 1003 ¶¶ 126–135). Chowdhuri describes the process as follows:

> In a relational database system, a user query is broken down into a relational algebra that is comprised of algebraic operations. Each of these algebraic operations is thought to be a basic operator (i.e., an execution primitive). Depending upon the data-partitioning scheme, *a query execution engine runs several instances of a given operator to work on different partitions*, *or allows operators to be executed in parallel*. … In the present invention this is achieved through the use

> of *data and index partitions*. The ability to run multiple instances of the operators on different data sets located across different storage units is called *horizontal (or partitioned) parallelism*. … Partitioning of execution is extremely useful in operational DSS[15] queries where large volumes of data are being scanned, joined, and sorted.

Ex. 1003 ¶ 126. Thus, Chowdhuri's system is based on partitioning large volumes of input data and running algebraic operators (e.g., scanning, joining, sorting) in parallel in response to a user's database queries.

Chowdhuri discloses the example of processing two input data tables—customer and order tables—to join heterogenous data in a selected fashion. Each input data table is partitioned into two or more data sets and scanned in parallel by separate scan iterators. We are persuaded by Petitioner that each data table in Chowdhuri bears structural similarities to the Employee and Department tables of the '610 patent to the extent that each data table has a different data type or "*schema*" and is partitioned for parallel processing. Pet. 18–19, 22–23, 40–41 (citing Ex. 1001, 3:23–34, 3:65–4:13, Fig. 3; Ex. 1003 ¶¶ 126–135, Fig. 3B; Ex. 1005 ¶¶ 94–95, 102, 134–136). We are not persuaded, however, by Petitioner's argument that Pike-Chowdhuri discloses, teaches, suggests, or otherwise provides motivation to process each partitioned data table as a "*data group*" having a "mechanism for identifying data from that group (e.g., a group identifier)," as construed by the district court and recited in the '610 patent claims.

i. Distinctions between Pike's MapReduce architecture and Chowdhuri's parallel query system

Petitioner argues that a POSITA "would have understood that Chowdhuri's iterators perform mapping and reducing tasks on data groups having different schema and track those input tables through the map and reduce phases." Pet. 14

[15] Decision support system. Ex. 1003 ¶¶ 11, 67.

(citing Ex. 1005 ¶¶ 76–80). Throughout the Petition, Petitioner relies on an asserted, sometimes assumed, correspondence or identity between Pike's map and reduce functions and Chowdhuri's scan, hashjoin, GroupBy, and sort functions. In contrast to Pike's MapReduce architecture, however, Chowdhuri does not disclose, teach, or suggest its scan function is "performing any sort of 'mapping'," as recognized by both Patent Owner and Petitioner. Prelim. Resp. 58 (citing Ex. 1003 ¶ 61; Pet. 26). Petitioner acknowledges that Chowdhuri's "scan iterators 'scan[] database rows from a database on disk' or 'a table or index.'" Pet. 26–27 (citing Ex. 1003 ¶¶ 61, 130 ("Scanning rows from a table on disk can be the behavior of one type of iterator.")). And rather than use map and reduce functions as in Pike, Chowdhuri's system uses scan and hashjoin functions to join customer and order data that has been scanned and distributed by exchange iterators.

Dr. Linn, moreover, acknowledges that Chowdhuri's iterators (scan, hashjoin, GroupBy, sort) operate "without knowing 'the source of the data stream.'" Ex. 1005 ¶ 76 (citing Ex. 1003 ¶ 130. "Iterators do not know what the source of a data stream is; whether it is the contents of a disk file, a network connection, or even another iterator."). The quoted disclosure suggests Chowdhuri is not concerned with tracking a data source through data processing functions, much less with uniquely identifying and processing data in groups that correspond to a particular input table or schema, as argued by Petitioner. Dr. Linn then describes Chowdhuri's scanning, joining, grouping, and sorting functions and applies a color code to those functions in Figure 3B, which mirrors the color code Petitioner applies to the '610 patent Figures 4 and 5 (*see* Pet. 7–8), but without explaining precisely why, in his opinion, Chowdhuri's parallel query method "teaches mapping and reducing using its iterators." Ex. 1005 ¶¶ 79–80. We are not persuaded by the cited declaration testimony or Petitioner's argument that

Chowdhuri's parallel query optimization system and iterators are comparable to Pike's MapReduce architecture or the MapReduce system described and claimed in the '610 patent.

We agree with Patent Owner's argument and supporting evidence that Chowdhuri's scan function is distinguishable from a "map" function that transforms input data and generates intermediate data in the form of new key-value pairs. Prelim. Resp. 57–58 (citing Ex. 1002, 3:59–61, 7:29–58; Ex. 1003 ¶ 61); *see also* Ex. 1001, 1:17–18 ("a 'map' function maps key-value pairs to new (intermediate) key-value pairs"); Ex. 2002 ("map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs"). Chowdhuri scans selected rows from the data tables into an exchange operator that functions as "a control operator" to "affect [sic, effect] parallelism" by "redistribution of data to multiple servers on a cluster for processing the data." Ex. 1003 ¶¶ 133–134; *see also* Ex. 1005 ¶ 102. "For example, once a scan operator starts processing data the exchange operator immediately repartitions the data and redistributes it to the hash join operator (iterator)." Ex. 1003 ¶ 134. Chowdhuri's scanned data is not "mapped" to form a new set of key-value pairs—intermediate data—for input into a separate reduce function.

We also reject Petitioner's argument that a Pike-Chowdhuri combined system would have been used to process Chowdhuri's order and customer data tables by "using, for example, Chowdhuri's scanning, hashjoin, GroupBy, and sort iterators as map and reduce tasks in Pike." Pet. 20. Petitioner's contention is conclusory, unsupported by the evidence, and self-contradictory given that Pike executes MapReduce and Chowdhuri does not. Petitioner's obviousness argument relies on substituting Chowdhuri's partitioned customer and order tables for Pike's input files and processing Chowdhuri's data in Pike's MapReduce system.

Otherwise, there would be little reason or need to use Pike's system in the first place. In sum, Chowdhuri does not teach a MapReduce architecture.

ii. The "simple substitution" of Chowdhuri's input data tables for Pike's input files

Petitioner argues that a POSITA would have been motivated to make "the simple substitution" of replacing Pike's input files with Chowdhuri's input tables for processing in Pike's MapReduce system (300). Pet. 19 (citing Ex. 1002, 2:22–26, 10:1–4; Ex. 1003 ¶¶ 126–127, 131–135; Ex. 1005 ¶¶ 95, 96). Petitioner posits that processing Chowdhuri's customer and order tables in Pike's MapReduce system would have yielded "predictable results." *Id.* Although not entirely clear, given Petitioner's argument that each of Pike's input files includes a "file name" as an asserted "*data group*" identifier, a "simple substitution" of Chowdhuri's input tables for processing in Pike's system could mean Petitioner is relying on Pike's system to provide a mechanism for identifying the data in each of Chowdhuri's customer and order tables through map and reduce processing. *See* Pet. 41 ("Chowdhuri's order table and customer table would be partitioned and mapped in [Pike's] system 300 to output intermediate data identifiable to these tables and reduced by 'state.'"). To the extent Petitioner is relying on Pike's system to provide "*data group*" identifiers for Chowdhuri's input data tables, we are not persuaded for the same reasons given in Section II.C.3.c., above. *See also* Prelim. Resp. 37–40, 44–47, 54–55.

iii. The asserted "data group" identifiers in Chowdhuri

Taking a similar approach to that used for Pike, Petitioner first asserts that "Chowdhuri's table names provide a mechanism for identifying the data in the table (e.g., order and customer data in table format)." Pet. 18 (citing Ex. 1003 ¶¶ 126-28, 131, 135; Ex. 1005 ¶ 94). As with Petitioner's treatment of Pike,

however, Petitioner fails to explain how Chowdhuri's *table names*—"customer" and "order"—serve to identify the *data* in each of Chowdhuri's input tables. Petitioner also fails to explain how the table names would be carried through the mapping and reducing functions in Pike's system, so that each partitioned data table is mapped and reduced on a "*data group*" basis as recited in claim 1 of the '610 patent. *See* Pet. 18–20. Chowdhuri's table names are no different than Pike's input file names, in that respect, and their use in Pike's MapReduce system would not render the '610 patent claims obvious for the same reasons given in our analysis of Pike. *See* Section II.C.3.c., above.

We also agree with Patent Owner that Petitioner incorrectly asserts Chowdhuri's table names somehow transform Chowdhuri's customer and order tables into the claimed "*data groups*" as construed by the district court. Prelim. Resp. 56 (citing Pet. 18). Chowdhuri does not use table names as mechanisms for identifying the *data* in those tables to guide processing the data in a way that corresponds and is identifiable to a particular data table. *Id.* We agree with Patent Owner that Chowdhuri does not disclose an "identifier (table name or otherwise) being, e.g., passed to specific functions to guide group-specific processing as in the '610 patent." *Id.* at 56–57 (citing Ex. 1001, 6:29–33, 39–48, 7:45, 7:55, 7:64–65, Fig. 5). Unlike the '610 patent's "emp" and "dept" identifiers, for example, Chowdhuri does not disclose, teach, suggest, or exemplify programming code or execution instructions for providing data table names "customer" and "order" (or, e.g., "cust" and "ordr") as a mechanism for identifying data from that group through scan, hashjoin, GroupBy, and sort functions.

Addressing the claim limitation "mapping functions . . . to form *corresponding* intermediate data for that *data group* and *identifiable to that data group*," Petitioner argues as follows:

> Identifiers "o" and "c" correspond to Chowdhuri's order and customer tables, respectively, and identify the data from each table through the scan, join, and group processes. This teaches "form[ing] corresponding intermediate data for that data group and identifiable to that data group." EX-1005 ¶121. A POSITA would have understood that the results of Chowdhuri's scan iterator (e.g., map function) are identifiable to its input tables because the results are logically merged by customer state and customer order based on different input data to yield orders by state. *Id.*; Chowdhuri ¶¶126-27, 131, 135. The customer state and customer order are therefore identifiable to each input table. EX-1005 ¶121.

Pet. 34. There are several problems with this argument.

First, Petitioner does not provide a functional connection between the asserted use of table names ("order" and "customer") to identify the data in each table and the "o" and "c" *query indicators* for scanning the order and customer tables. *Id.* at 34–35. Second, the customer "c" and order "o" in Chowdhuri's example DSS query refers to the fact that "the customer table and the order table[] are being scanned," not used as a mechanism for identifying the original data source during processing. Ex. 1003 ¶¶ 126–127. Third, Petitioner appears to substitute Chowdhuri's scan, join, GroupBy, and sort functions for Pike's map and reduce functions, rather than explain how *Pike's* MapReduce system would use Chowdhuri's "o" and "c" query indicators to identify order and customer data when executing a map function to generate "*corresponding* intermediate data for that *data group* and *identifiable to that data group*." Fourth, we agree with Patent Owner that the logical merging of selected rows of data from Chowdhuri's customer and order tables "says nothing of *how* that merger is accomplished, much less that any sort of 'a mechanism for identifying data' from a particular 'data group' is used for accomplishing such merger." Prelim. Resp. 64. Fifth, Chowdhuri does not generate intermediate data and would not need a "*data group*"

identifier for identifying scanned data being consolidated in a self-contained hashjoin function that does not output data to another operator for further processing. *Id.* at 65.

We agree with the inconsistency identified by Patent Owner, that the "c" and "o" represent the customer and order tables "*within the query itself*, not within the scanners or hashjoin functions" that execute Chowdhuri's parallel query system. *Id.* at 63–64 (emphasis in original). We further agree with Patent Owner that Petitioner does not explain how the asserted "c" and "o" "*data group*" identifiers in Chowdhuri actually identify "*intermediate data*" as "*corresponding*" to a particular data table during map processing. *Id.* at 63. Chowdhuri does not teach or suggest scanning data into separate files identified by "c" or "o" identifiers, particularly given the scan "[i]terators do not know what the source of a data stream is." Ex. 1003 ¶ 130. Even if we were to accept Petitioner's table identification premise, Chowdhuri is not MapReduce, and Petitioner does not explain how *Pike* would use the "c" and "o" identifiers to map process Chowdhuri's data tables to generate "*corresponding* intermediate data for that data group and *identifiable to that data group*" as recited in the '610 patent.[16]

Nor are we persuaded that leveraging "Pike's distributed system and its fault tolerant task scheduling" is a sufficient motivation to modify Pike-Chowdhuri as claimed in the '610 patent. Pet. 20. Chowdhuri's parallel query system already is a distributed system that can revise "a tree of relational operators" and "generate[] a plan for execution of the query based on the revised tree." Ex. 1003 ¶¶ 11, 15. Moreover, we fail to see how an improvement in efficiency gained from Pike's

[16] Linn Decl. ¶¶ 121–122, on which Petitioner relies for support, tracks the unpersuasive arguments made in the Petition.

fault-tolerant *task scheduling* function relates to the use of "*data group*" identifiers as described and claimed in the '610 patent.

In short, Petitioner does not sufficiently demonstrate how or why Chowdhuri's "c" and "o" query indicators (or table names) would identify the data in each partitioned customer and order table through the map and reduce functions in Pike's system, rather than simply indicate "the customer table and the order table[] are being scanned," as expressly stated in Chowdhuri. Ex. 1003 ¶ 127.

iv. Conclusion

In sum, Petitioner fails to show that there is a reasonable likelihood Pike-Chowdhuri discloses, teaches, suggests, or otherwise renders obvious a mechanism for identifying the data in each of "*a plurality of data groups*" in order to execute "mapping functions . . . to form *corresponding* intermediate data for that *data group* and *identifiable to that data group* . . . ; and reducing the intermediate data . . . for each *data group* in a manner that is defined to *correspond to that data group*," as recited in claim 1 of the '610 patent. Because Petitioner relies on the same analysis in support of its challenge to independent claims 17, 33, and 40 based on Pike-Chowdhuri, our determination applies equally to those independent claims and the dependent claims. Therefore, we determine Petitioner has not established a reasonable likelihood of prevailing in its challenge to the patentability of claims 1–46 of the '610 patent as obvious over Pike-Chowdhuri.

D. Asserted Obviousness of Claims 3, 4, 9–11, 19, 20, 26–28, 37–39, 44–46 Over Pike in View of Chowdhuri and MacLeod (Ground 3)

Petitioner contends that certain dependent claims are rendered obvious by the combination of Pike, Chowdhuri, and MacLeod. Petitioner analyzes dependent claims 3, 4 and 19, 20 with reference to MacLeod and Ground 3. *See* Pet. 2, 55, 56. Similarly, the Petition includes substantive analysis of MacLeod in connection

with limitation 9[b] regarding dependent claims 9–11, 26–28, 37–39, and 44–46. *Id.* at 64–66, 78–79. The Petition addresses MacLeod in the context of dependent claim limitations relating to the storage of intermediate data in association with "metadata." *Id.* at 55–56, 64–66, 78–79. The Petition does not address MacLeod in the analysis of the independent claims. Because Pike (Ground 1) and Pike-Chowdhuri (Ground 2) do not render any of the independent claims obvious, the combination of Pike, Chowdhuri, and MacLeod does not render any of the challenged dependent claims obvious for the same reasons given in Section II.C., above.

III. CONCLUSION

For the foregoing reasons, we determine Petitioner has not established a reasonable likelihood of prevailing in its challenge to the patentability of any of claims 1–46 of the '610 patent. Therefore, we deny institution of *inter partes* review of the challenged claims. We need not address Patent Owner's argument for discretionary denial under 35 U.S.C. § 325(d) in view of our decision denying institution on the merits.

IV. ORDER

For the foregoing reasons, it is

ORDERED that the Petition is denied and no *inter partes* review is instituted.

FOR PETITIONER:

Jeffrey Berkowitz
Daniel Tucker
Alexander Boyer
Taylor Stark
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER LLP
jeffrey.berkowitz@finnegan.com
daniel.tucker@finnegan.com
alexander.boyer@finnegan.com
taylor.stark@finnegan.com

FOR PATENT OWNER:

Carder W. Brooks
Brent N. Bumgardner
Nelson Bumgardner Conroy PC
carder@nelbum.com
brent@nelbum.com
R2Solutions-IPR@nelbum.com