

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TESSELL, INC.,
Petitioner,

v.

NUTANIX, INC.,
Patent Owner.

Case No. IPR2025-00322
Patent No. 11,010,336

DECLARATION OF SAMRAT BHATTACHARJEE, PH.D.

Tessell Exhibit 1003 Tessell v. Nutanix
--

TABLE OF CONTENTS

I.	PERSONAL AND PROFESSIONAL BACKGROUND	1
II.	MATERIALS REVIEWED AND CONSIDERED	5
III.	MY UNDERSTANDING OF PATENT LAW	10
	A. Anticipation	12
	B. Obviousness.....	13
IV.	THE '336 PATENT	15
	A. Technical Background and Described Embodiments	15
	B. The Challenged Claims	32
	C. Prosecution History	33
	D. Person of Ordinary Skill in the Art (“POSA”).....	34
V.	GROUND 1: Sivasubramanian and Zha Render Obvious Claims 1, 5- 8, 10-15, and 19-21	36
	A. Sivasubramanian (EX1005).....	36
	1. Overview	37
	a. Database-Provisioning UIs	49
	i. GUI	51
	ii. Command-Line	56
	b. Backup “Workflow”	58
	2. Motivation to Customize Sivasubramanian’s UI	61
	B. Zha (EX1006).....	63
	C. Sivasubramanian+Zha Combination	67
	1. UI for retention period.....	78
	2. UI for protection schedule.....	84
	D. Claim-by-Claim Analysis.....	85
	1. Claim 1	85
	a. [1PRE]: “A method comprising”	85
	b. [1A1]: “receiving, by a processor executing computer- readable instructions stored on a memory...”	85

c.	[1A2]: “a user request for provisioning a source database...”	87
i.	GUI	87
ii.	Command Line	90
d.	[1A3]: “in a hyperconverged infrastructure system”	90
e.	[1B]: “receiving, by the processor, selection of a database engine type”	95
i.	GUI	95
ii.	Command Line	97
f.	[1C]: “receiving, by the processor, selection of a Service Level Agreement (“SLA”) and a protection schedule”	98
i.	GUI	98
ii.	Command Line	99
g.	[1D]: “provisioning, by the processor, the source database based upon the database engine type; and”	99
h.	[1E]: “setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database”	100
i.	[1F]: “wherein the transactional logs record transactions occurring on the source database since a most recent snapshot”	104
j.	[1G]: “and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.”	106
2.	Claim 5: “The method of claim 1, wherein provisioning the source database comprises creating a new database as the source database.”	108
3.	Claim 6: “The method of claim 1, wherein provisioning the source database comprises registering an existing database as the source database.”	108
4.	Claim 7: “The method of claim 1, wherein the first frequency and the second frequency are the same.”	110

5. Claim 8	111
a. [8PRE]-[8A]: “The method of claim 1, further comprising: presenting, by the processor, a group of database engine types that are supported by the hyperconverged infrastructure system; and”	111
b. [8B]: “receiving, by the processor, the selection of the database engine type from the group of database engine types.”	112
6. Claim 10: “The method of claim 1, further comprising: presenting, by the processor, a plurality of levels of the SLA; and receiving, by the processor, selection of one of the plurality of levels of the SLA.”	113
7. Claim 11: “The method of claim 1, wherein the protection schedule comprises a time period for designating at least one of a daily snapshot, a weekly snapshot, a monthly snapshot, and a quarterly snapshot.”	113
8. Claim 12	116
a. [12PRE]: “A database management system comprising:”	116
b. Limitations [12A1]-[12G]	117
9. Claim 13: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive the user request, the selection of the database engine type, and the SLA via a dashboard.”	118
10. Claim 14: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to create a cloned database from the source database based on the snapshots and the transactional logs.”	119
11. Claim 15: “The database management system of claim 12, wherein the first frequency and the second frequency are different.”	120
12. Claim 19	120
a. [19PRE]: “A non-transitory computer readable media with computer-executable instructions embodied thereon that, when executed by a processor in a hyperconverged	

infrastructure system, cause the processor to perform a process comprising:"	120
a. Limitations [19A]-[19G].....	121
13. Claim 20: "The non-transitory computer readable media of claim 19, wherein provisioning the source database comprises creating a new source database or registering an existing database as the source database."	122
14. Claim 21: "The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for: presenting a group of database engine types that are supported by the hyperconverged infrastructure system; and receiving the selection of the database engine type from the group of database engine types."	122
VI. GROUND 2: Sivasubramanian, Shekar, and Zha Render Obvious	
Claims 1-23	122
A. Shekar (EX1007)	122
B. Sivasubramanian+Shekar+Zha Combination.....	126
C. Claim-by-Claim Analysis	130
1. Claims 1, 5-8, 10-15, 19-21	130
2. Claim 2	131
a. [2PRE]-[2A]: "The method of claim 1, further comprising: receiving, by the processor, selection of a first set of parameters; and"	131
b. [2B]: "creating, by the processor, a source database virtual machine based on the first set of parameters, wherein the source database resides on the source database virtual machine."	132
3. Claim 3: "The method of claim 2, wherein the first set of parameters comprises at least one of a software profile, a compute profile, a network profile, and a database parameters profile."	135
a. "software profile"	135
b. "compute profile"	137

c. “network profile”	138
d. “database parameters profile”	140
4. Claim 4: “The method of claim 1, further comprising: converting, by the processor, a size of the source database into a number of virtual disks; and associating, by the processor, the number of virtual disks with the source database.”	141
5. Claim 9: “The method of claim 1, further comprising: receiving, by the processor, selection of at least one of a sample set of parameters for creating a source database virtual machine for hosting the source database; and creating, by the processor, the source database virtual machine based upon the at least one of the sample set of parameters.”	146
6. Claim 16: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive at least one parameter for creating a source database virtual machine, and associate the source database with the source database virtual machine.”	147
7. Claim 17: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive a size of the source database, and convert the size into a number of virtual disks to be associated with the source database.”	147
8. Claim 18: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive selection of an existing source database virtual machine, and create the source database on the existing source database virtual machine.”	148
9. Claim 22: “The non-transitory computer readable media of claim 19, wherein the processor further executes the computer- executable instructions for: receiving selection of at least one of a sample set of parameters for creating a source database virtual machine; and creating the source database virtual machine based upon the at least one of the sample set of parameters.”	148
10. Claim 23: “The non-transitory computer readable media of claim 19, wherein the processor further executes the computer- executable instructions for: presenting a plurality of levels of	

the SLA; and receiving selection of one of the plurality of levels of the SLA.”	149
VII. CLAIM LISTING	149

I, Samrat Bhattacharjee, Ph.D., declare:

1. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioner Tessell, Inc., to assess claims 1-23 (the “challenged claims”) of U.S. Patent No. 11,010,336 (“the ’336 patent”). I am being compensated for my time at my standard rate of \$700.00 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of the *inter partes* review of the ’336 patent.

I. PERSONAL AND PROFESSIONAL BACKGROUND

2. My qualifications for forming the opinions set forth in this declaration are summarized below and explained in more detail in my current curriculum vitae, provided as EX1004. EX1004 also includes a list of my publications and patents.

3. I am a tenured professor in the Computer Science Department at the University of Maryland. I am also an Affiliate Professor with the Department of Electrical and Computer Engineering at the University of Maryland. I have been an appointed faculty member at the University of Maryland since 1999.

4. I received a Ph.D. in computer science from the Georgia Institute of Technology in 1999. Before that I received a B.S. degree in Mathematics and Computer Science with Highest Distinction (Summa Cum Laude) and was recognized as the Outstanding Department Major at Georgia College and State University, Milledgeville, Georgia in 1994.

5. I have been involved in the research and development of computing systems for over two decades. I perform research and I teach in the areas of computer networks, distributed systems, data storage and backup, and networking, among others. A major theme of my research has been the development of technology to improve the performance and security of data transfers and data storage on networks. My research has examined problems ranging from precursors to software-defined networking, operating systems, secure data transfer and storage, large scale Internet video distribution, the design of novel network architectures for better and secure content distribution and storage, efficiency of content distribution over wireless networks, and so on. In the course of my research, I have incorporated various types of databases and other mass storage systems into systems that I have built. I have also written numerous software programs, including software programs with user interfaces.

6. For example, starting in the late 1990s, the focus of my research was the development of network and operating system technology to design a new network architecture that allowed for more efficient data transfer. Part of my Ph.D. research was to develop new and more efficient architectures for video delivery on the Internet, and I have published papers on this architecture during my graduate studies. I continued to work on video delivery and security and storage of digital content as a faculty member, and have published various papers in these

areas. During 2007, I was a visiting researcher at AT&T Labs, and one of the projects I focused on was a video content delivery platform, which was implemented using networked massed storage. This work resulted in both publications and a granted U.S. patent (U.S. 8,752,100 B2).

7. Over the course of my career, I have conducted a number of research projects that have related to the technologies underlying databases or that have used databases as components of larger systems. For example, I have conducted research on various techniques used to improve reliability and data recovery, including replication and snapshots, for distributed datasets, including storage infrastructure for databases. In 2006, I presented a paper on possible techniques for implementing a distributed version of Google's search capability at the 2nd International IEEE Workshop on Networking Meets Databases. In 2007, I was granted a patent related to secure data transfer that used a database to store cryptographic and other information. In addition, as part of my research, I have used cloud-based databases, including those hosted on Amazon AWS. For example, in 2018, I published a paper related to a project that began in 2016 for allowing users to anonymously upload sensitive information. This system used MySQL databases implemented on the Amazon AWS service. Furthermore, I have worked on the intersection of operating systems, security, and highly performant large-scale data access systems for over 20 years, all of which is related

to and relevant to database research, and I have reviewed database literature in the course of my academic research.

8. Along with working on research problems related to the topics mentioned above, I routinely taught this material in both graduate and undergraduate courses. A number of my papers in these domains are with students at the University of Maryland, many of whom started in these areas after taking my classes.

9. For over 25 years, I have developed professional and academic experience in the field of computer systems. For example, I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, and workshops relating to computer networking, data storage, data security. Over the years I have received several fellowships, prizes, and awards for teaching excellence and technical papers, including an Alfred P. Sloan Jr. Fellowship in 2004 and the National Science Foundation CAREER award in 2001. I have also received an award from the SIGCOMM foundation for a “Test of Time” paper (that is given to an influential paper published ten years ago). My research work has been supported by multiple grants from the US National Science Foundation, and the Department of Defense. I have also started a Joint Ph.D. program with the University of Maryland and the Max Planck Society in Germany, and co-founded

the annual Cornell, Maryland, Max Planck Research School that provides research exposure to about 80 students from across the world during a week-long school.

10. I have served on numerous paper and proposal review panels and participated on program committees for ACM/IEEE groups, technical journals, and conferences in Networking, Security and Systems in the aforementioned areas of computer science. I have created research software for all of my projects, much of which is available online for others to build on.

11. I am a named inventor on four U.S. Patents. These patents are generally related to computer networking and the delivery of services over networks and video delivery systems, and some of the systems described in these patents used databases.

12. I have served as an expert witness and technical consultant in litigation and Inter Partes Review matters concerning videoconferencing, data conferencing and screen sharing, voice over IP (VoIP) telephony, multimedia networking, distributed systems, operating systems, computer networks, and datacenter networking, among others. I have testified in several trials and depositions as an expert witness. My curriculum vitae is provided as Exhibit 1004.

II. MATERIALS REVIEWED AND CONSIDERED

13. My findings, as explained below, are based on my years of education, research, experience, and background in the field of computer science, as well as

my investigation and study of relevant materials for this declaration. When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art (“POSA”), as set forth in Section IV below. In forming my opinions, I have studied and considered the materials identified in the list below, as well as any additional materials cited herein.

Exhibit	Description
1001	U.S. Patent No. 11,010,336
1002	Prosecution History of U.S. Patent No. 11,010,336
1004	Curriculum Vitae of Dr. Bhattacharjee
1005	U.S. Patent No. 8,713,060 (“Sivasubramanian”)
1006	U.S. Patent No. 8,150,808 (“Zha”)
1007	U.S. Patent App. Pub. No. 2018/0232142 (“Shekar”)
1008	U.S. Patent No. 6,542,893 (“Quernemoen”)
1009	The American Heritage Dictionary of the English Language (5th ed. 2011) (“Heritage”)
1010	U.S. Patent No. 7,873,684 (“Souder”)
1011	U.S. Patent App. Pub. No. 2018/0324279 (“Barton”)
1012	Using Oracle Database Cloud Service – Creating a Database Deployment (2017) (“Oracle”) (Wayback Machine Capture, January 18, 2017), available at https://web.archive.org/web/20170118071307/https://docs.oracle.com/en/cloud/paas/database-dbaascloud/csdbi/create-db-deployment.html
1013	Amazon Relational Database Service User Guide (2016) (“RDS”) (Wayback Machine Capture, April 12, 2016), available at https://web.archive.org/web/20160412105355/http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf
1014	U.S. Patent App. Pub. No. 2009/0319582 (“Simek”)
1015	U.S. Patent No. 7,331,000 (“Ohno”)

Exhibit	Description
1016	U.S. Patent App. Pub. No. 2016/0182328 (“Bhasin”)
1017	U.S. Patent App. Pub. No. 2016/0077925 (“Tekade”)
1018	U.S. Patent App. Pub. No. 2019/0324865 (“Weissman”)
1019	U.S. Patent App. Pub. No. 2016/0335369 (“Picard”)
1020	U.S. Patent No. 11,308,114 (“Moghe”)
1021	U.S. Patent App. Pub. No. 2014/0229698 (“Sivasubramanian-698”)
1022	“Taking a Backup Using Snapshots” (“LVM-Howto”) (Wayback Machine Capture, January 20, 2007), available at https://web.archive.org/web/20070120200004/https://tldp.org/HOWTO/LVM-HOWTO/snapshots_backup.html
1023	U.S. Patent App. Pub. No. 2004/0059878 (“Madany”)
1024	U.S. Patent App. Pub. No. 2006/0161879 (“Lubrecht”)
1025	PCT Pub. No. WO 2010/106578 (“Utsunomiya”)
1026	U.S. Patent No. 10,855,554 (“Freeman”)
1027	U.S. Patent No. 11,182,372 (“Jain”)
1028	U.S. Patent App. Pub. No. 2016/0057077 (“Gomatam”)
1029	U.S. Patent No. 11,036,696 (“Higginson”)
1030	U.S. Patent No. 9,305,068 (“Esposito”)
1031	U.S. Patent No. 7,873,610 (“Poulsen”)
1032	“Creating Instances,” Google Cloud (“Google”) (Wayback Machine Capture, September 18, 2018), available at https://www.archive.org/web/20180918103721/https://cloud.google.com/sql/docs/mysql/create-instance
1033	U.S. Patent App. Pub. No. 2003/0046353 (“Chung”)
1034	U.S. Patent App. Pub. No. 2006/0253472 (“Wasserman”)
1035	U.S. Patent App. Pub. No. 2005/0027661 (“Lober”)
1036	U.S. Patent App. Pub. No. 2018/0173452 (“Hu”)
1037	U.S. Patent No. 10,078,465 (“Wu”)
1038	U.S. Patent No. 9,830,082 (“Srinivasan”)

Exhibit	Description
1039	U.S. Patent App. Pub. No. 2016/0266913 (“Achillopoulos”)
1040	U.S. Patent App. Pub. No. 2017/0262232 (“Zhao”)
1041	U.S. Patent App. Pub. No. 2019/0116237 (“Gibson”)
1042	U.S. Patent No. 10,206,092 (“Mellquist”)
1043	U.S. Patent No. 11,243,703 (“Watkins”)
1044	U.S. Patent App. Pub. No. 2015/0207683 (“Adogla”)
1045	U.S. Patent App. Pub. No. 2014/0018947 (“Ales”)
1046	U.S. Patent App. Pub. No. 2013/0332251 (“Ionnidis”)
1047	U.S. Patent App. Pub. No. 2006/0107006 (“Green”)
1048	U.S. Patent App. Pub. No. 2008/0059894 (“Cisler”)
1049	U.S. Patent App. Pub. No. 2013/0110779 (“Taylor”)
1050	U.S. Patent App. Pub. No. 2016/0306840 (“Deshmukh”)
1051	U.S. Patent App. Pub. No. 2018/0121521 (“Bivol”)
1052	U.S. Patent No. 10,496,302 (“Gaurav”)
1053	U.S. Patent No. 10,268,551 (“Wong”)
1054	U.S. Patent App. Pub. No. 2014/0108339 (“Marsden”)
1055	U.S. Patent No. 8,335,765 (“McAlister”)
1058	Affidavit of Nathaniel E. Frank-White
1059	RESERVED
1060	U.S. Patent No. 11,120,011 (“Priebe”)
1061	U.S. Patent No. 7,634,679 (“Quintanillo”)
1062	U.S. Patent App. Pub. No. 2009/0276771 (“Nickolov”)
1063	U.S. Patent App. Pub. No. 2016/0188421 (“Karinta”)
1064	U.S. Patent App. Pub. No. 2017/0243275 (“Goens”)
1065	U.S. Patent App. Pub. No. 2016/0378361 (“Uriel”)
1066	RESERVED
1067	U.S. Patent App. Pub. No. 2017/0228227 (“Winterfeldt”)

Exhibit	Description
1068	U.S. Patent App. Pub. No. 2014/0337844 (“Jin-844”)
1069	U.S. Patent No. 9,582,297 (“Jin-297”)
1070	U.S. Patent No. 10,572,292 (“Bhandari”)
1071	U.S. Patent App. Pub. No. 2015/0188775 (“VanDerWalt”)
1072	U.S. Patent App. Pub. No. 2018/0063088 (“Hardy”)
1073	U.S. Patent No. 4,742,482 (“Inskeep”)
1074	U.S. Patent App. Pub. No. 2015/0207682 (“Nichele”)
1075	U.S. Patent App. Pub. No. 2013/0232497 (“Jalagam”)
1076	U.S. Patent App. Pub. No. 2010/0017801 (“Kundapur”)
1077	U.S. Patent App. Pub. No. 2006/0029096 (“Babbar”)
1078	UK Patent Application No. GB2451943 (“Wren-Hilton”)
1079	U.S. Patent App. Pub. No. 2013/0238331 (“Mikan”)
1080	U.S. Patent App. Pub. No. 2016/0239637 (“Miller”)
1081	U.S. Patent App. Pub. No. 2016/0066201 (“Kerpez”)
1082	U.S. Patent App. Pub. No. 2018/0060119 (“Zamir”)
1083	U.S. Patent App. Pub. No. 2015/0188789 (“Jayaprakash”)
1084	U.S. Patent App. Pub. No. 2015/0207703 (“Gallagher”)
1085	U.S. Patent App. Pub. No. 2006/0161462 (“Sharma”)
1086	RESERVED
1087	U.S. Patent App. Pub. No. 2002/0082858 (“Heddaya”)
1088	<i>Supported Platforms: MySQL Database, available at https://web.archive.org/web/20181106221826/https://www.mysql.com/support/supportedplatforms/database.html (“MySQL”)</i>
1089	U.S. Patent No. 11,061,709 (“Cao”)
1090	U.S. Patent No. 10,705,755 (“Wang”)
1091	U.S. Patent App. Pub. No. 2017/0371547 (“Fruchtman”)
1092	U.S. Patent No. 10,824,522 (“Sadavarte”)
1093	U.S. Patent No. 8,146,080 (“Carter”)

Exhibit	Description
1094	U.S. Patent App. Pub. No. 2009/0276833 (“Paul”)
1095	U.S. Patent App. Pub. No. 2017/0315877 (“Kaplingat”)
1096	U.S. Patent App. Pub. No. 2002/0019920 (“Reuter”)
1097	U.S. Patent App. Pub. No. 2016/0246503 (“Phelan”)
1098	U.S. Patent App. Pub. No. 2017/0068469 (“Shankar”)
1099	U.S. Patent App. Pub. No. 2011/0138383 (“Le”)
1100	U.S. Patent App. Pub. No. 2017/0119753 (“Gong”)
1101	A screenshot of the Expedia webpage displaying the “Preferred class” dropdown menu in its expanded state, Expedia website (Wayback Machine Capture, July 15, 2018), available at https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand
1102	Excerpts of McGraw-Hil Dictionary of Scientific and Technical Terms (6 th ed. 2003) (“McGraw-Hill”)
1103	U.S. Patent App. Pub. No. 2018/0203771 (“Heidel”)
1104	U.S. Patent App. Pub. No. 2017/0357667 (“Auch”)
1105	Excerpts of New Oxford American Dictionary (1 st ed. 2001) (“Oxford”)

III. MY UNDERSTANDING OF PATENT LAW

14. In developing my opinions, I discussed various relevant legal principles with Petitioner’s attorneys. Though I do not purport to have prior knowledge of such principles, I understood them when they were explained to me and have relied upon such legal principles, as explained to me, in the course of forming the opinions set forth in this declaration. My understanding in this respect is as follows:

15. I understand that “inter partes review” (IPR) is a proceeding before the United States Patent & Trademark Office for evaluating the patentability of an issued patent’s claims based on prior-art patents and printed publications.

16. I understand that, in an IPR proceeding, including in this proceeding, the Petitioner has the burden of proving that the challenged claims of the challenged patent are unpatentable by a preponderance of the evidence. I understand that “preponderance of the evidence” means that a fact or conclusion is more likely true than not true.

17. I understand that, in IPR proceedings, claim terms in a patent are given their ordinary and customary meaning as understood by a person of ordinary skill in the art (“POSA”) in the context of the entire patent and the prosecution history pertaining to the patent. If the specification provides a special definition for a claim term that differs from the meaning the term would otherwise possess, the specification’s special definition controls. If a claim element is expressed as a “means” for performing a specified function, I understand that it covers the corresponding structure described in the specification and equivalents of the described structure. I have applied these standards in preparing the opinions in this declaration.

18. I understand that determining whether a particular patent or printed publication constitutes prior art to a challenged patent claim can require

determining the effective filing date (also known as the priority date) to which the challenged claim is entitled. I understand that for a patent claim to be entitled to the benefit of the filing date of an earlier application to which the patent claims priority, the earlier application must have described the claimed invention in sufficient detail to convey with reasonable clarity to the POSA that the inventor had possession of the claimed invention as of the earlier application's filing date. I understand that a disclosure that merely renders the claimed invention obvious is not sufficient written description for the claim to be entitled to the benefit of the filing date of the application containing that disclosure.

19. I understand that for an invention claimed in a patent to be patentable, it must be, among other things, new (novel—i.e., not anticipated) and not obvious from the prior art. My understanding of these two legal standards is set forth below.

A. Anticipation

20. I understand that, for a patent claim to be “anticipated” by the prior art (and therefore not novel), each and every limitation of the claim must be found, expressly or inherently, in a single prior-art reference. I understand that a claim limitation is disclosed for the purpose of anticipation if a POSA would have understood the reference to disclose the limitation based on inferences that a POSA

would reasonably be expected to draw from the explicit teachings in the reference when read in light of the POSA's knowledge and experience.

21. I understand that a claim limitation is inherent in a prior art reference if that limitation is necessarily present when practicing the teachings of the reference, regardless of whether a person of ordinary skill recognized the presence of that limitation in the prior art.

B. Obviousness

22. I understand that a patent claim may be unpatentable if it would have been obvious in view of a single prior-art reference or a combination of prior-art references.

23. I understand that a patent claim is obvious if the differences between the subject matter of the claim 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 relevant field at the time the invention was made. Specifically, I understand that the obviousness question involves a consideration of:

- the scope and content of the prior art;
- the differences between the prior art and the claims at issue;
- the knowledge of a person of ordinary skill in the pertinent art; and
- if present, objective factors indicative of non-obviousness,

sometimes referred to as “secondary considerations.” To my knowledge, the

Patent Owner has not asserted any such secondary considerations with respect to the patent challenged in this proceeding.

24. I understand that in order for a claimed invention to be considered obvious, a POSA must have had a reason for combining teachings from multiple prior-art references (or for altering a single prior-art reference, in the case of obviousness in view of a single reference) in the fashion proposed.

25. I further understand that in determining whether a prior-art reference's teachings would have been combined with teachings from other prior art or with other information within the knowledge of a POSA, the following are examples of approaches and rationales that may be considered:

- combining prior-art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;
- use of a known technique to improve similar devices in the same way;
- applying a known technique to a known device ready for improvement to yield predictable results;

- applying a technique or approach that would have been “obvious to try,” i.e., choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;

- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art; some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior-art reference or to combine prior-art reference teachings to arrive at the claimed invention. I understand that this teaching, suggestion or motivation may come from a prior-art reference or from the knowledge or common sense of one of ordinary skill in the art.

26. I understand that for a single reference or a combination of references to render the claimed invention obvious, a POSA must have had a reasonable expectation of success in arriving at the claimed invention by altering or combining the teachings of the applied references.

IV. THE '336 PATENT

A. Technical Background and Described Embodiments

27. The '336 patent discloses a “database system” that receives a “user request” for “provisioning” a “source database.” *See* EX1001, Abstract, 1:24-28,

independent claims 1, 12, 19. In the Abstract, the patent refers to “[a] system and method include receiving, by a database engine of a database system associated with a virtual computing system, a user request via a dashboard for provisioning a source database with the database system[.]” Similarly, at 1:24-28, the patent says “a method is disclosed” that “database engine of a database system associated with a virtual computing system, a user request via a dashboard for provisioning a source database.” Furthermore, the patent’s independent claims 1, 12, and 19 recite “instructions” that relate to “provisioning” a “source database” or “instructions” to “provision” a “source database.” A “source database” is a database from which “snapshots” are taken to create backups; a snapshot is a point-in-time record of the database’s state. EX1001, 4:46-59 (“Protecting a database means replicating a state of the database for creating a fully functional copy of the database. Replicating the state of the database may involve creating fully functional *clones* (e.g., *back-ups*) of the database.... A cloned database may be created by periodically capturing *snapshots* of the *source* database. *A snapshot stores the state of the source database at the point in time* at which the snapshot is captured. The snapshot is thus a point in time image of the database.”), 5:20-23 (“The database protection system instance may be configured to *protect the source database by automatically capturing snapshots* of the source database.”).¹

¹ Emphasis is added throughout unless otherwise indicated.

28. Databases predated the '336 patent's priority date. Quernemoen (EX1008), which was filed in 2000 and issued in 2003, mentions that "[r]elational databases came into common use in computers over twenty years ago." Quernemoen, 1:36-37. This is consistent with my personal understanding and experience; I took a course on relational databases as an undergraduate in the early 1990s, and my understanding is that relational databases have been around since the 1970s.

29. Databases are collections of organized data. Heritage (EX1009), 462 ("database" is "[a] collection of data arranged for ease and speed of search and retrieval."). Databases can be created and managed by different types of database software, commercial examples of which include Oracle and MySQL. EX1001, 14:11-15; Sivasubramanian, 2:44-47. The patent refers to different types of database software as "database engine types." EX1001, 14:14-25. I note that in describing Figure 2, the patent mentions to a "database engine 205" that refers to a different concept, namely a component of the "database system" that manages databases of different "database engine types." EX1001, 14:14-18.

30. The '336 patent's "database system" includes a "database provisioning system" that provides "database provisioning services." EX1001, 10:44-57 ("the database system 200 includes a database engine 205... the database engine 205 is configured to implement one or more database management

services... For example, the database engine 205 is configured to provide database provisioning services to create new databases and register existing databases with the database system 200 using a database provisioning system 220.”).

31. The ’336 patent says that “[d]atabase provisioning services involve creating and/or associating databases with the database system for management and use.” EX1001, 3:32-34. As I discuss further below, the patent describes a user interface for creating a database that takes user-specified parameters and creates the database based on those parameters. Thus, as I explain further below, in the ’336 patent, “[p]rovisioning” may involve either “creating” a database with user-specified parameters or “associating” an existing database with the database system.

32. POSAs understood that “provisioning” was a term generally used in many technical contexts, including databases, to refer to configuring and providing resources to an entity that needed them. My testimony regarding a POSA’s background knowledge is corroborated by, for example, Souder (EX1010), which is assigned to Oracle. Souder explains, in a portion of its “BACKGROUND” section entitled “Provisioning for Database Grid,” that “[t]he term provisioning refers to providing and configuring the computational resources and data needed to provide a service.” Souder, 2:47-50. As another example, Barton (EX1011) says that

The term “provisioning” can be used in a variety of contexts in today’s technological environment. For example, in the telecommunication industry, the concept of network provisioning may refer to mediating a customer’s services with respect to telecommunication network elements. In enterprise computing, provisioning may refer to the enterprise-wide configuration, deployment, and management of multiple types of enterprise system resources for enterprise customers.

Barton, [0004].

33. POSAs understood that databases required resources like (1) processors and memory to run the above-mentioned database engine software, and (2) data storage to store the underlying data. My testimony is corroborated by various references. *See, e.g.,* Quernemoen, Abstract (referring to “total mass storage requirements for a relational database table including database storage requirements, application and software requirements, system table requirements, scratch and sort requirements, log file requirements, and growth requirements”); *Using Oracle Database Cloud Service – Creating a Database Deployment* (“Oracle,” EX1012), 2-4 (stating that one of the options in the “Provision New Oracle Database Cloud Service wizard” is “Compute Shape,” which is defined as “[t]he number of Oracle Compute Units (OCPUs) and amount of memory (RAM) for the compute node of the new database deployment.”); *Amazon Relational Database Service Guide* (“RDS,” EX1013), 100 (“The computation and memory capacity of a DB instance is determined by its DB instance class. You can change

the CPU and memory available to a DB instance by changing its DB instance class[.]”), 101-102 (mentioning “MySQL” and “Oracle” among “DB engines” for which various “instance classes” are available).

34. I understand that Exhibit 1012 was downloaded from the URL <https://web.archive.org/web/20170118071307/https://docs.oracle.com/en/cloud/paas/database-dbaascloud/csdbi/create-db-deployment.html>. I understand that the DNS name web.archive.org represents the website of something known as the “Wayback machine” that is maintained by an organization known as the Internet Archive. I am familiar with the Wayback machine from having used it prior to my work in this proceeding, and I understand that the Wayback Machine allows users to search archived websites maintained by the Internet Archive. Furthermore, I have reviewed Exhibit 1058, which I understand is an Affidavit of Mr. Nathaniel E. Frank-White, an employee of the Internet Archive. Based on Mr. Frank-White’s explanation of Internet Archive URLs in paragraph 5 of his declaration, and based on my own familiarity with the Wayback Machine, I understand that in an Internet Archive URL, the numerical field that appears after “web/” indicates the date and time at which the page following the number was archived. Thus, I understand the URL from which Exhibit 1012 was downloaded is the Internet Archive’s record of the website at the URL <https://docs.oracle.com/en/cloud/paas/database-dbaascloud/csdbi/create-db->

deployment.html archived on January 18, 2017. I also understand that Exhibit 1013 was downloaded from the URL <https://web.archive.org/web/20160412105355/http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf>. Based on my familiarity with the Wayback Machine as detailed here in paragraph 34, I understand the URL from which Exhibit 1013 was downloaded is the Internet Archive's record of the website at the URL <http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf> archived on April 12, 2016.

35. In addition, when describing the “process 300” illustrated in Figure 3, the patent says “[t]he process 300 may be used to implement the database provisioning service. Thus, the process 300 may be used to create a new database or register an existing database.” EX1001, 15:62-66.

36. Thus, POSAs understood that the patent's “database provisioning system” (EX1001, 10:44-57) can either create a new database and provision it with resources, or provision resources to an already-existing database.

37. The patent describes two different “example user interfaces”—one for “creating” and one for “registering” a “database.” EX1001, 22:23-25 (“Turning now to FIGS. 4A-5E, example user interfaces for creating and registering a database are shown, in accordance with some embodiments of the present disclosure.”).

38. “[T]o initiate a database provisioning service,” the user navigates to a “database page” that has a “create button” and a “register button.” EX1001, 23:43-24:3, Fig. 4B (annotated below). The exemplary database-creation interface is shown in Figs. 4C-4G. EX1001, 22:25-26.

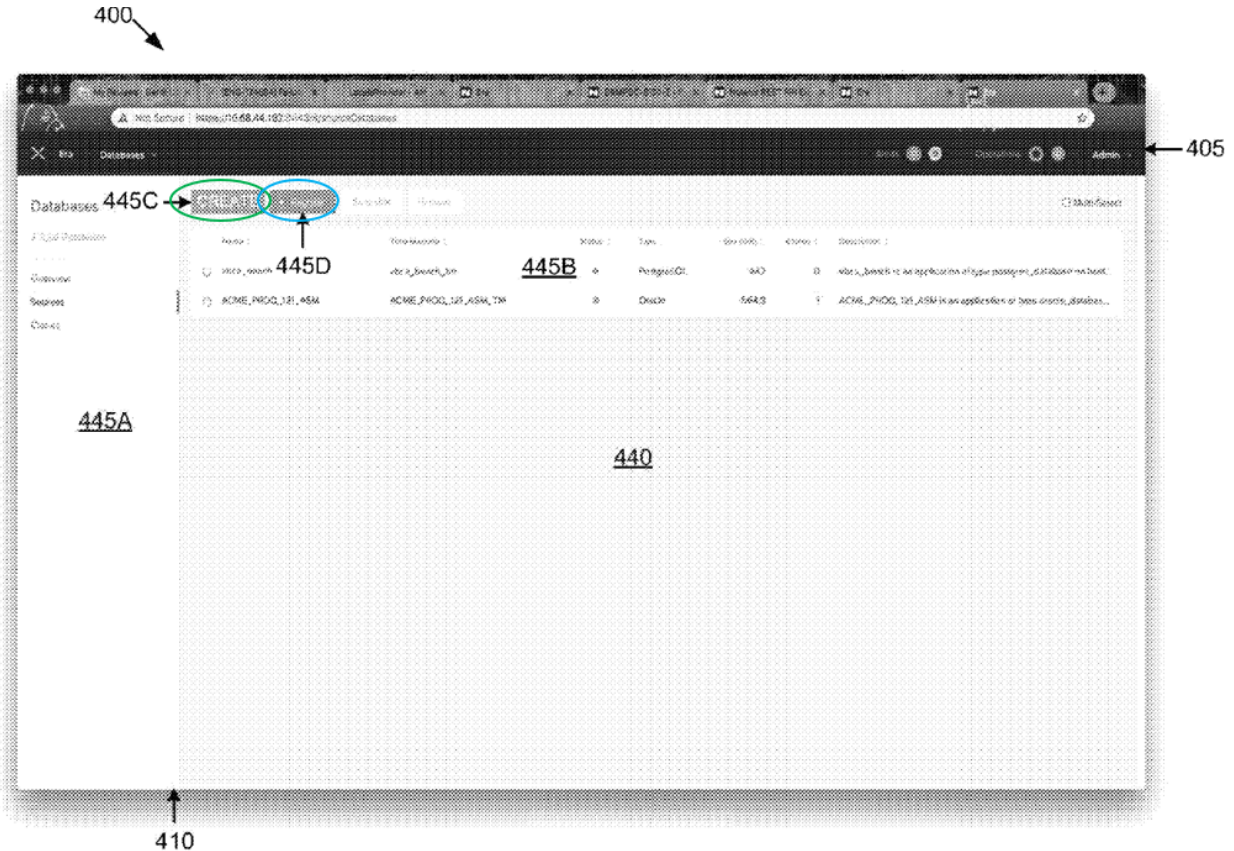


FIG. 4B

EX1001, Fig. 4B (annotated)

39. Once the user presses the “create” button, the use can enter various parameters on subsequent screens. EX1001, 24:4-10 (“Upon selecting the create button 445C, the database engine 205, and particularly the database creation system 235 of the database engine may present a user interface 450, shown in FIG.

4C. The user interface 450 identifies the various steps of creating the new database and highlights the current step.”). I show examples of such parameters in the table below, along with excerpts from the relevant figures.

selection of a “database engine type” from among types that are “supported by the database system,” such as “Oracle, PostgreSQL, etc.” (EX1001, 14:22-24, 24:15-23, FIG. 4C). As I discussed above, the patent refers to different types of database software as “database engine types.” EX1001, 14:14-25.

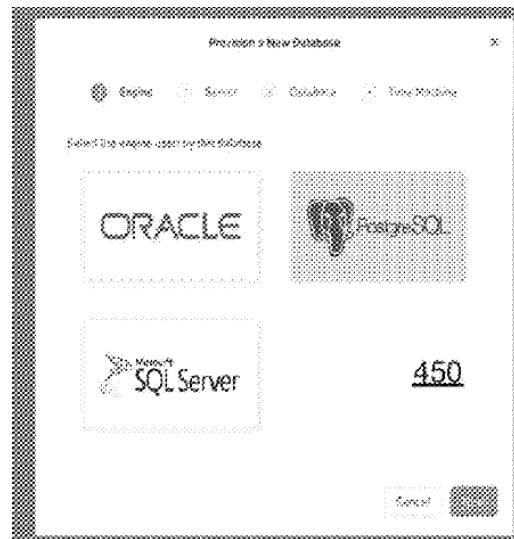


FIG. 4C (excerpt)

“name 460A,” “description 460B,” “size 460E,” (EX1001, 24:49-63, Fig. 4E);

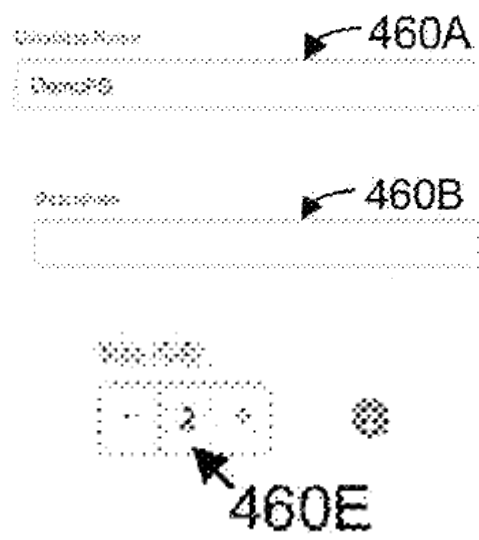


FIG. 4E (excerpts)

“protection schedule 465D” and “SLA [service level agreement] level 465C” (EX1001, 25:1-17, Fig. 4F)

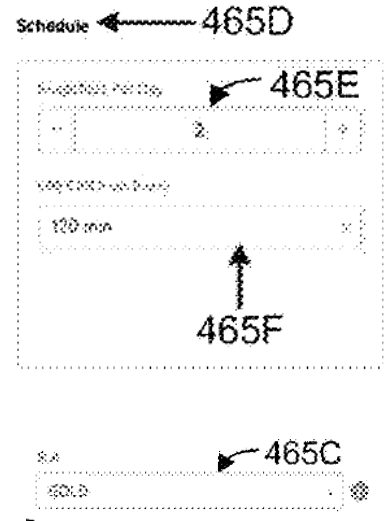


FIG. 4F (excerpts)

40. The patent states that “[u]pon providing the various parameters in the user interface 465, the user may select a ‘Next’ button to send the selections to the database creation system 235 and start the database creation. The database creation system 235 may display a status of the database creation in a user interface 470 of FIG. 4G.” EX1001, 25:17-22. POSAs understood from context that “the selections” refers not just to the parameters on Fig. 4F, but includes the other database creation parameters entered on previous screens (since Fig. 4F is the last screen in a series of screens for entering parameters). Thus, POSAs understood that, in the ’336 patent’s database creation user interface (“UI”), once the parameters are entered on the different screens, the user presses a button—the

“Next” button on Figure 4F—to send the parameters to a “database creation system” to “start the database creation.” EX1001, 25:17-22.

41. The user invokes UIs for “registering an existing database” by pressing “register” instead of “create” on the “database page.” EX1001, 25:43-54. The user can then enter various parameters on subsequent screens, including an “SLA and protection schedule.” EX1001, 25:60-67. “Upon receiving all of the parameters from the user, the database registration system... registers the database.” EX1001, 26:1-2.

42. The “protection schedule” referenced above that is specified on the screen shown in Fig. 4F (EX1001, 25:8) may define a daily schedule for capturing “snapshots” from the database. EX1001, 25:8-17 (“Within the protection schedule 465D, the user interface 465 requests the user to provide a number of snapshots 465E desired each day... and time periods 465G for identifying which snapshot to designate as the daily snapshot...”), 5:64-67 (“The protection schedule may allow the user to define the frequency of snapshots and transactional logs to be captured each day, and the time-period for capturing daily, weekly, monthly, and/or quarterly snapshots based upon the SLA.”).

43. The patent refers to a “snapshot” as a “point in time image of the database.” EX1001, 4:57-59 (“A snapshot stores the state of the source database at the point in time at which the snapshot is captured. The snapshot is thus a point in

time image of the database.”). The concept of taking a point-in-time image of a database was well-known to a POSA, and the prior art referred to such an image as a “snapshot”; snapshots could be used for backup purposes because they could be used to restore the database to a certain point in time. *See, e.g.*, Simek (EX1014), [0002] (“Database Snapshots are a feature of some database management systems (‘DBMS’) and allow a user to specify a point in time at which to preserve the contents of the database.”); Ohno (EX1015), 1:54-61 (“In order to restore the contents of the database up to a point in time when the data has consistency, a snapshot must be acquired in advance in a state in which data has consistency.... Snapshot is to maintain data to be snapshot at a certain point in time in an external storage, and it is a function provided by most storage systems.”); Zha, 11:33-44 (“The system configuration manager 315 also allows a user with appropriate roles and privileges to setup policies specifying the schedule with which the point-in-time copy manager 310 retrieves PIT copies of databases....”).

44. The “service level agreement” or “SLA” referenced above that is specified on the screen shown in Fig. 4F (EX1001, 25:8) specifies how long snapshots are kept. EX1001, 19:12-23 (“[a]n SLA is an agreement... that outlines, among other things, the protection scope of the database. The protection scope defines for how long data from the database being created or registered is retained...”), 19:45-47 (“For example, the continuous protection parameter within

the SLA defines the duration in days for which all captured snapshots and transactional logs are retained.”), 20:58-62 (“As an example, if a snapshot is to be stored for 30 days, on the 31st day, that snapshot may be deleted. Thus, each snapshot and transactional log is managed based on the SLA and protection schedule independent from other snapshots and transactional logs.”).

45. Retaining snapshots or other types of backup data for some user-defined duration was also well-known to a POSA. *See, e.g.*, Bhasin (EX1016), [0009] (“...in some embodiments the snapshots are maintained according to a configured or configurable retention time period...”); RDS, 16 (describing a configurable “Backup Retention Period” setting as “the number of days you want automatic backups of your database to be retained”), 111-112 (“The preferred backup window is the user-defined period of time during which your DB instance is backed up.”); Tekade (EX1017), [0046] (“FIG. 6 depicts a user interface for configuring a single SLA protecting a database with an associated log on different schedule... This Snapshot policy states that a database backup will run every day 605 with exception setting 606 and operating window start time 607 and end time 608. The backup will run every 24 hours specified by 609... the RPO field 613 specifies log backup runs every 30 minutes when database log backup is enabled.”), Fig. 6; Simek, [0034] (“the user interface 200 includes a control 208

that allows a user to specify a parameter defining the frequency at which the database snapshots 122 should be created.”), Fig. 2.

46. Although the exemplary interfaces discussed above are graphical, the ’336 patent notes that its “dashboard” “receive[s] user inputs” via either a “graphical user interface (‘GUI’)” or a “text-based user interface[.]” EX1001, 11:61-64, 12:28-43.

47. The ’336 patent claims that its “database system” provides a “user friendly, intuitive user interface [UI] that solicits information from and conveniently walks the user through the various steps for creating a new database.” EX1001, 4:8-11. The patent claims that its UI enables database creation “within minutes” and “hides the complexity” of database creation. EX1001, 4:8-19 (“The database system of the present disclosure provides a user friendly, intuitive user interface that solicits information from and conveniently walks the user through the various steps for creating a new database within minutes... By virtue of providing standardized configurations, the present disclosure simplifies the database creation process for the user. The user interface also hides the complexity of creating the database from the user.”). The patent claims that this is in contrast to conventional systems in which, according to the ’336 patent, the process of “provisioning” a database required “multiple days” of work by “multiple entities.” EX1001, 3:32-55; *see also* EX1001, 4:8-19, Abstract (“A system and method include receiving,

by a database engine of a database system associated with a virtual computing system,” various inputs and “also include provisioning, by the database engine,” a database “based upon” the received inputs), Figs. 4C, 4E.

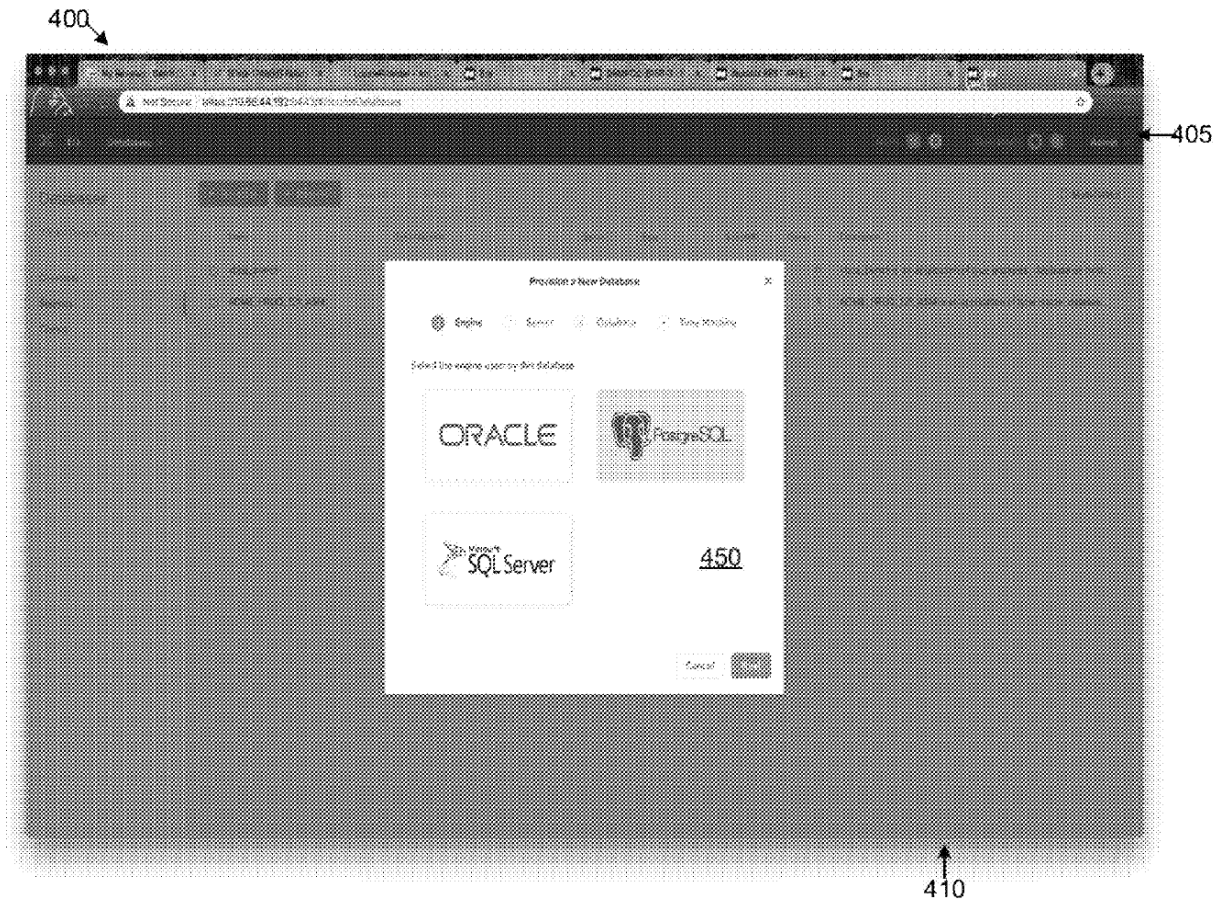


FIG. 4C

EX1001, FIG. 4C

actions to be performed in the data environment.... A workflow can be instantiated that includes tasks necessary to perform the action.”).

49. Sivasubramanian’s control plane replaces “tedious manual procedures” previously required for tasks such as “provisioning and scaling data storage.” Sivasubramanian, 1:46-56 (“Typically, performing tasks such as provisioning and scaling data storage are tedious manual procedures... There is no easy way for a customer to dynamically adjust the data storage capacity, or manage other such aspects of a data repository.”), 2:17-21 (“Systems and methods in accordance with various embodiments of the present disclosure may overcome one or more of the aforementioned and other deficiencies experienced in conventional approaches to managing aspects of data storage in an electronic environment.”).

50. In contrast to the manual procedures previously used, which “[took] a significant amount of time and effort,” “a user or customer [could] instead submit a call including information such as a type of hardware and a version of a database product” to Sivasubramanian’s control plane, which then “perform[ed] the necessary task.” Sivasubramanian, 9:49-61.

51. Amazon had also released a commercial product, the Amazon Relational Database Service, that contained a similar interface to that shown in Sivasubramanian. The Amazon Relational Database Service User Guide (“RDS,”

EX1013) from 2016 documents that functionality in a printed publication that is also prior art to the '336 patent. *See* RDS, 13-18, 22.

The screenshot shows the 'Specify DB Details' page in the AWS RDS console. It is divided into two main sections: 'Instance Specifications' and 'Settings'.

Instance Specifications:

- DB Engine:** mysql
- License Model:** general-public-license (dropdown)
- DB Engine Version:** 5.6.19a (dropdown)
- DB Instance Class:** - Select One - (dropdown)
- Multi-AZ Deployment:** - Select One - (dropdown)
- Storage Type:** - Select One - (dropdown)
- Allocated Storage*:** 5 GB

There are two informational messages (blue boxes with speech bubbles):

- One above the instance class dropdown: "Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions."
- One below the storage type dropdown: "Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. Click here for more details."

Settings:

- DB Instance Identifier*:** [text input]
- Master Username*:** [text input]
- Master Password*:** [password input]
- Confirm Password*:** [password input]

At the bottom, there is a legend: "* Required". To the right of the legend are three buttons: "Cancel", "Previous", and "Next Step".

RDS, 15

B. The Challenged Claims

52. Independent claims 1, 12, and 19 recite a processor receiving “selection” of a “database engine type,” “Service Level Agreement (‘SLA’),” and “protection schedule.” The processor then provisions a “source database” of the selected “database engine type” and sets up “automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,” with additional limitations further defining the “transactional

logs” and “protection schedule.” The dependent claims recite additional limitations including (1) creating a “source database virtual machine” that the “source database” resides on (*e.g.*, claim 2), (2) converting the “size” of the “source database” into a number of “virtual disks,” and (3) additional details about the user interface for accepting parameters (*e.g.*, claims 3, 13)selection of certain parameters.

53. Based on my review of the ’336 patent’s claims, I note that the patent does not claim any novel implementation of databases. For example, the patent does not claim any new database type, or any new technology for storing the underlying data in the database.

C. Prosecution History

54. I have reviewed the prosecution history of the ’336 patent (EX1002, “’336 FH”).

55. The examiner rejected the originally-filed claims over Weissman (EX1018) in view of Picard (EX1019). EX1002, 256. Applicants scheduled an examiner interview and, in the agenda (1) proposed amending claim 1 to recite what became Element [1F] (*see* the claim listing in Section VII below) and (2) argued Weissman did not disclose as-filed claim 7 (whose subject matter was later incorporated into the independent claims) because Weissman did not disclose a “first frequency” for capturing snapshots and a “second frequency” for capturing

transactional logs. EX1002, 275-276. The examiner agreed that the incorporating 7 into claim 1 overcame the prior art rejection. EX1002, 278. The applicants then amended independent claims 1, 12, and 19 to add Elements [1F-G]/[12F-G]/[19F-G], and also amended various dependent claims. EX1002, 287-291. The examiner then allowed the claims. In the statement of “[r]easons for allowance,” the examiner stated that “the prior art does not teach or fairly suggest” the subject matter of the independent claims, but did not identify or discuss any specific reference. EX1002, 324-326.

D. Person of Ordinary Skill in the Art (“POSA”)

56. I have been informed and understand that for purposes of assessing whether prior-art references disclose every element of a patent claim (thus “anticipating” the claim) and/or would have rendered the claim obvious, the patent and the prior-art references must be assessed from the perspective of a person having ordinary skill in the art (“POSA”) to which the patent is related, based on the understanding of that person at the time of the patent claim’s priority date. I have been informed and understand that a POSA is presumed to be aware of all pertinent prior art and the conventional wisdom in the art, and is a person of ordinary creativity. I have applied this standard throughout my declaration.

57. The ’336 patent involves technology in the field of user interfaces for deploying database services. *See* EX1001, Abstract (“The system and method also

include provisioning, by the database engine, the source database...”), 1:24-2:9 (describing various “aspects of the present disclosure” that each include “provisioning the source database” based on inputs), 3:45-49 (“Upon receiving the database configuration, another entity of the database provider may configure a database virtual machine for hosting the database, while yet another entity may configure the networking settings to facilitate access to the database upon creation.”). *See also* my discussion of the embodiments in Section IV.A above.

58. I have been asked to provide my opinions as to the state of the art in this field by December 2018. I use this timeframe because the face of the ’336 patent indicates an earliest claimed priority date of December 27, 2018. Whenever I offer an opinion in this declaration about the knowledge of a POSA, the manner in which a POSA would have understood the claims of the ’336 patent or its description, the manner in which a POSA would have understood the prior art, or what a POSA would have been led to do based on the prior art, I am referencing the December 2018 timeframe, even if I do not say so specifically in each case.

59. In my opinion, a person of ordinary skill (POSA) at the time of the ’336 patent’s claimed December 27, 2018 priority date would have at least a bachelor’s degree in computer science, computer engineering, or a related field, with two (2) years of academic and/or industry experience with database technologies. A higher level of education may substitute for less experience.

60. By 2018, I held a Ph.D. in computer science from the Georgia Institute of Technology, and I had more than 20 years of experience using, researching, and/or teaching about networked storage, storage security, storage virtualization, and databases. Therefore, I was a person of more than ordinary skill in the art during the relevant timeframe. However, I worked with many people who fit the characteristics of the POSA, and I am familiar with their level of skill. When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art, as set forth above.

V. GROUND 1: SIVASUBRAMANIAN AND ZHA RENDER OBVIOUS CLAIMS 1, 5-8, 10-15, AND 19-21

61. I have been asked to provide my opinion concerning whether claims 1, 5-8, 10-15, and 19-21 of the '336 patent would have been obvious to a POSA in light of Sivasubramanian and Zha, the prior-art references identified in the Petition. For the reasons explained below, it is my opinion that each of these claims would have been obvious to a POSA.

A. Sivasubramanian (EX1005)

62. As I discussed in Section IV.A above, “provisioning” in the '336 patent encompasses either (1) creating and allocating resources to a new database; or (2) registering an existing database and providing resources. EX1001, 3:32-34, 15:62-66.

63. As I explain below, in my opinion, POSAs reading Sivasubramanian understood that although Sivasubramanian uses the terminology slightly differently and illustrates the functionality with fewer graphical figures than the '336 patent, Sivasubramanian discloses UIs for the same two above-mentioned processes that fall within the scope of “provisioning” in the '336 patent.

1. Overview

64. Sivasubramanian concerns “the *creation, provisioning*, and management of data stores and instances” using a “control environment.” Sivasubramanian, Abstract. *See also* Sivasubramanian, 10:5-10 (“Using components such as those discussed above, FIG. 3 illustrates an example process 300 by which a customer can request the performance of a control-related task with respect to at least one data instance in a data environment, here the data plane, using the control plane or a similar data control service.”), Fig. 3. “[D]ata store’ refers to... any device or combination of devices capable of storing, accessing, and retrieving data”; such a device or devices may include “databases.” Sivasubramanian, 3:15-19 (“As used herein the term ‘*data store*’ refers to any device or combination of devices *capable of storing, accessing, and retrieving data, which may include* any combination and number of data servers, *databases*, data storage devices, and data storage media, in any standard, distributed, or clustered environment.”).

65. POSAs understood that Sivasubramanian’s “control environment” (also called “control plane”) can manage various “workflows” that perform tasks in a “data plane” (also called a “data environment”). See Sivasubramanian, Abstract, 2:21-24, Fig. 2. Sivasubramanian’s Abstract says:

A user can *call into* an externally-facing interface of the *control environment*, the *call being analyzed to determine actions to be performed in the data environment*. A monitoring component of the control plane also can periodically communicate with the data environment to determine any necessary actions to be performed, such as to recover from faults or events in the data environment. *A workflow* can be instantiated that *includes tasks necessary to perform the action*.

Sivasubramanian, Abstract. Sivasubramanian’s “control environment” is also called a “control plane,” and Sivasubramanian’s “data environment” is also called a “data plane.” Sivasubramanian, 2:21-24 (“In particular, various embodiments provide a separate control environment, or control plane, that can be used to control aspects of a data environment, or data plane.”). As shown below, Sivasubramanian’s Figure 2 shows the “Workflow 216,” which I discuss in more detail below, inside the “control plane.”

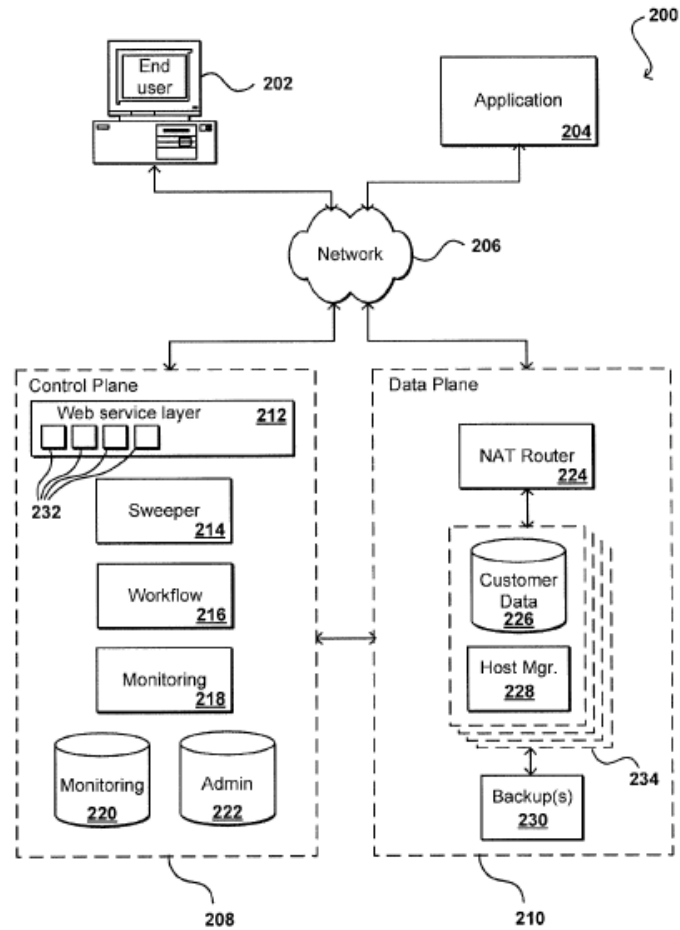


FIG. 2

Sivasubramanian, Fig. 2

66. A “workflow” is “a sequence of tasks” for “a specific job.”

Sivasubramanian, 6:25-26.

67. As I explain in ¶¶ 68-70 below, POSAs understood that in Sivasubramanian, “workflows” may be initiated in response to “requests” that

users submit to the control plane. *See Sivasubramanian*, 2:26-32, 5:4-10, 5:35-44, 6:12-23.

68. Sivasubramanian states that “a user or customer can submit a *request* to the control *plane through an externally-visible application programming interface (API), for example*, which can be analyzed to determine actions to be performed in the data plane.” Sivasubramanian, 2:26-32. Sivasubramanian later says that an “*RDS* in one embodiment *provides a separate ‘control plane’* that includes components... useful for managing aspects of the data storage. In one embodiment, a set of data management application programming interfaces (*APIs*) or other such interfaces *are provided that allow a user* or customer to *make calls into the RDS* to perform certain tasks relating to the data storage.”

Sivasubramanian, 5:4-10. “RDS” in Sivasubramanian refers to “relational database service.” Sivasubramanian, 4:59.

69. Sivasubramanian further says that the “control plane” in the “RDS” embodiment “includes a *Web services layer*... The *Web services layer* also can include a set of APIs 232 (or other such interfaces) for *receiving* Web services *calls or requests* from across the network 206, which the Web services layer can parse or otherwise analyze to determine the steps or actions needed to act on or process the call.” Sivasubramanian, 5:35-44. Finally, Sivasubramanian says

the Web services layer might place instructions or information for the *‘create database’ request* in the admin data store 222, or a similar job queue, and the sweeper can periodically check the admin data store for outstanding jobs. Various other approaches can be used as would be apparent to one of ordinary skill in the art, such as the Web services layer sending a notification to a sweeper that a job exists. The *sweeper component can pick up the ‘create database’ request*, and *using information for the request can send* a request, call, or other such *command to a workflow component 216 operable to instantiate at least one workflow for the request*.

Sivasubramanian, 6:12-23.

70. Thus, Sivasubramanian discloses that when a “user or customer... submit[s] a request to the control plane through,” for example, an API (Sivasubramanian, 2:26-32), a “Web services layer” in the control plane receives that “*request*” (Sivasubramanian, 5:35-44), and a “sweeper” component in the Web Services Layer issues a “command to a *workflow* component” that “instantiates a *workflow* for the *request*” (Sivasubramanian, 6:12-23).

71. As I will explain in Section V.A.1.a below, in Sivasubramanian, requests for initiating workflows can be submitted to the control plane using either graphical or textual interfaces.

72. One “workflow” the control plane can manage is “create database.” Sivasubramanian, 6:59-65. As explained below, POSAs understood that

Sivasubramanian’s “create database” workflow includes what the ’336 patent calls “provisioning” a database, for multiple reasons.

73. **First**, Sivasubramanian says the “create database” workflow “includes tasks such as *provisioning a data store instance*.” Sivasubramanian, 6:59-65.

Sivasubramanian uses the term “data instance” or “data store instance” to refer to a database together with the resources—such as processing power, memory, and storage—that the database relies on. In other words, the “instance” is a component including the database and the engine and hardware the database runs on, and other entities can access a database via its instance. For example, Sivasubramanian says “[a]n example ‘create *database*’ workflow... includes tasks such as provisioning a *data store instance*, allocating a volume of off-instance persistent storage, *attaching* the persistent *storage* volume to the *data store instance*.”

Sivasubramanian, 6:59-62. Additional tasks performed by the workflow include “allocating and attaching a DNS address or other port, interface, or identifier which the customer can use to *access or otherwise connect* to the *data instance*.”

Sivasubramanian, 6:62-65. In addition, Sivasubramanian says “customers can create a data store using a ‘Create*Database*’ or similar API. The user can call a Web service to specify any desired values for *an instance type (which describes the CPU and memory capacity)*, storage size, repository name, port, and other such values.” Sivasubramanian, 11:48-51. *See also* Sivasubramanian, 9:56-10:4

(“Using a control plane... a user or customer can instead submit a call including information such as a *type of hardware and a version of a database product*. The control plane or service can then perform the necessary tasks to *create*, delete, modify, expand, or otherwise modify *a data store or data storage instance*. Once provisioned, the user has native access to the *data instance(s)*... There is no restriction or modification of query models or other such functionality, as a user can continue to use applications built on MySQL, Oracle, or other *database* technology.”). I also note that Sivasubramanian refers to choosing between a MySQL or Oracle “instance” when creating a “database.” Sivasubramanian, 6:43-50 (“In this example, a workflow can be instantiated using a workflow template for creating a *database* and applying information extracted from the original request. For example, if the request is for a MySQL® Relational Database Management System (RDBMS) instance, as opposed to an Oracle® RDBMS or other such instance, then a specific task will be added to the workflow that is directed toward MySQL *instances*.”).

74. *Second*, Sivasubramanian also references using a “Create*Database*” API (“application programming interface,” *see* Sivasubramanian, 2:26-32) to “*provision*” a “*repository*,” and then using “Describe*Database*” to check if the “repository” is “provisioned.” *See* Sivasubramanian, 11:47-56. Sivasubramanian states:

...customers can create a data store using a “Create**Database**” or similar API. The user can call a Web service to specify any desired values for an instance type (which describes the CPU and memory capacity), storage size, *repository name*, port, and other such values. The customer could also utilize a “Describe**Database**” or similar API to poll on the status of *the repository* to determine the state of the repository, such as *whether the repository state is provisioned*.

Sivasubramanian, 11:47-56. POSAs understood that if a possible “status” of the “repository” after the “CreateDatabase” is used is that “the repository” is “provisioned,” this means that the “CreateDatabase” API performs provisioning of the repository.

75. POSAs understood that the “repository” that Sivasubramanian references being “provisioned” by the “CreateDatabase” API is a *database*, for multiple reasons, as I explain below.

76. First, as I noted above, Sivasubramanian at 11:47-56 refers to commands issued using APIs called “CreateDatabase” and “DescribeDatabase” when discussing the “repository” being provisioned. This context indicates to a POSA that “repository” is being used for “database.”

77. Furthermore, Sivasubramanian describes a database as a type of repository and frequently uses those terms interchangeably. At 15:37-38, Sivasubramanian refers to “a repository such as a relational database,” indicating that a database is a type of repository. *See also* 4:58-62 (“while the service is

referred to herein as a ‘relational *database* service’, it should be understood that such a service can be used with any appropriate *type of data repository* or data storage”). Additionally, Sivasubramanian repeatedly uses the terms “database” and “repository” interchangeably. *See, e.g.*, Sivasubramanian, 4:49 (“relational *database* service (‘RDS’)”), 5:15-20 (“FIG. 2 illustrates an example of an *RDS* implementation... In this example, a computing device 202 for an end user is shown to be able to make calls through a network 206 into a control plane 208 to perform a task such as to provision a *data repository* of the data plane 210.”), 1:47-56 (“Typically, performing tasks such as provisioning and Scaling data storage are tedious manual procedures, in which a customer has to provide a *database* administrator (DBA) or similar expert user with configuration information and requirements... The DBA typically then has to enable, tune, and optimize the *data repository*. There is no easy way for a customer to dynamically adjust the data storage capacity, or manage other such aspects of a data *repository*.”), 11:52-56 (“The customer could also utilize a ‘Describe*Database*’ or similar API to poll on the status of the *repository* to determine the state of the *repository*, such as whether the *repository* state is provisioned.”), 12:24-33 (“As part of a Create*Database* API, for example, customers in one embodiment can be expected to supply the username and password for a special *repository* user, such as may be called a ‘*Repository* Owner.’ A *Repository* Owner is a special type of user who owns the

repository schema objects. After creation of a *repository*, a customer can perform tasks such as to add more users to using a ‘Create**Database**User’ API, remove users using a ‘Delete**Database**User’ API, and list customers using a ‘Describe**Database**Users’ API.”), 13:18-21 (“Once the request has been submitted, the control plane can asynchronously execute the request to provision the *repository*. A ‘Describe**Databases**’ or similar API can be used to determine the status of the request.”), 12:64-13:5 (stating that “in the following example, the customer generates a request to create a new *data repository* using a command line tool” and showing an example use of “rds-create-*database*” command), 13:27-33 (“Once the *repository* has been provisioned and is available, the customer can perform various actions on the data plane.... The customer in this example runs a command (i.e., on the source MySQL *database* server using a compatible client utility)....”). In addition, as I discuss in Section V.A.1.a.i below, Sivasubramanian’s Figure 5 shows an input field called “Repository Name” above a field entitled “Database engine,” while Sivasubramanian’s exemplary command line command has a field called “dbName.”

78. As I noted in paragraphs 67-70 above (discussing, *e.g.*, Sivasubramanian, 2:26-32, 5:35-44, 6:12-23), in Sivasubramanian, workflows may be initiated in response to requests submitted via APIs. POSAs would have

understood from the name that the “Create**Database**” API causes initiation of the “create database” workflow.

79. **Third**, the ’336 patent uses “provisioning” to encompass creating a database and giving it resources, as I noted in Sections IV.A and V.A above. As I noted above, Sivasubramanian discloses the “create database” workflow “provisioning” a “repository,” *i.e.* database, and also provisioning a “data instance” representing a database and its resources. POSAs understood from the name “create database” that the workflow “**creates**” a database and then gives it resources, *e.g.* the resources in the data instance; thus, Sivasubramanian’s “create database” workflow does the same thing the ’336 patent describes as “provisioning.”

80. **Fourth**, the ’336 patent says “[d]atabase provisioning services involve creating and/or associating databases with the database system **for management and use.**” EX1001, 3:32-34. Likewise, following completion of Sivasubramanian’s “create database” workflow that includes “provisioning,” the database, and the “data instance” on which it runs (as discussed above), can be **managed** by the control plane, and also can be **used**. *See, e.g.,* Sivasubramanian, 6:59-67 (“An example ‘create database’ workflow... might include[] tasks such as provisioning a data store instance” and “allocating and attaching a DNS address or other address, port, interface, or identifier which the customer can use to access or

otherwise connect to the data instance”), 9:4-8 (similar), 2:26-44 (“A user or customer can submit a request to the control plane... which can be analyzed to determine actions to be performed in the data plane, such as actions that create... a data store or data storage instance.... Once provisioned, a user can native access to the data instance(s) in the data plane, and can simply point existing applications (such as MySQL applications) to the DNS (domain name system) address or other location information for the particular instance.”), 9:45-65 (describing advantages of provisioning with control plane and explaining that “[o]nce provisioned, the user has native access to the data instance(s)”), 12:1-36 (describing additional APIs including “‘ModifyDatabase’ API” enabling customers to “increase the storage size, change the instance type, or modify various other fields”), 13:18-57, (stating that “Once the request has been submitted, the control plane can asynchronously execute the request to provision the repository” and referring to “state” changing to “‘Created’ once the provisioning has been completed,” and then saying “Once the repository has been provisioned and is available, the customer can perform various actions on the data plane” and describing various actions such as “verify[ing] that the required tables are created” to confirm that the “repository is ready for use by one of the customer’s applications,” at which point “[t]he customer changes the connect string... for the application to point to the new instance database”), 13:58-15:31 (describing other actions such as adjusting user privileges or “adjust[ing] the

size of an instance for compute or processing needs” (14:45-46)), 19:32-64 (discussing workflow for backup).

81. Additionally, as I noted in Section IV.A, the ’336 patent also uses the term “provisioning” to refer to “registering” an existing database and providing resources to it. EX1001, 3:32-34. The patent describes an exemplary “registering” interface where a user identifies an existing database and then enters parameters for providing it resources. EX1001, 25:43-67, Fig. 5A-5E. As I will discuss in §V.A.1.a, Sivasubramanian likewise discloses UIs that let users identify an existing database and specify parameters for providing it resources.

a. Database-Provisioning UIs

82. Sivasubramanian says “F[igure] 5 illustrates an example of a display” that is an “example” of an “interface page... allow[ing] a user to *submit requests* to the *control plane*.” Sivasubramanian, 12:49-55. Sivasubramanian also describes Figure 5’s exemplary “display” and the below-referenced command-line tool as alternative ways to submit “calls to the control plane.” Sivasubramanian, 12:53-13:5 (“...*FIG. 5 illustrates an example of a display 500*, here a page rendered in a browser application, that can allow a user to make calls into the control plane... *In other embodiments*, a user can manually (or otherwise) create and submit Web service calls to the control plane. *In the following example*, the customer generates a request to create a new data repository using a *command line*

tool.”). Thus, POSAs understood Sivasubramanian to disclose using a GUI like Figure 5’s to submit requests to invoke workflows. I discuss the GUI further in Section V.A.1.a.i below.

83. Sivasubramanian discloses a “command line tool” that is an “example” way to “submit Web services calls to the control plane.” Sivasubramanian, 12:62-66. Sivasubramanian also says users can “call a Web service” to invoke the “‘CreateDatabase’ or similar API”; this API invokes the “create database” workflow (*see* my discussion in Section V.A.1 above). Sivasubramanian, 11:47-52. Thus, POSAs understood that the command-line tool could be used to submit requests to invoke workflows. I discuss the command-line further in Section V.A.1.a.ii below.

84. As I explain in Sections §§V.A.1.a.i-V.A.1.a.ii below, the GUI and command-line, respectively, can be used to invoke workflows to “provision” a database (in the ’336 patent’s meaning) by creating a new database and allocating it resources and/or allocating resources to an existing database. Throughout this Declaration, I will use the term “database-provisioning UI” to refer to a UI that performs either of the two “provisioning” processes I mentioned above.

i. GUI

85. Sivasubramanian's Figure 5 "display" lets the user "select an action to be performed 504," and select "options for the actions" including "database engine" (506) and "version information" (508). Sivasubramanian, 12:53-62.

The screenshot shows a web browser window titled 'Web Browser' with a menu bar (FILE, EDIT, VIEW, FAVORITES, TOOLS, HELP) and an address bar. The main content area is titled 'Control interface page'. It contains several form fields and a 'Task' dropdown menu. The 'Task' dropdown is circled in red and labeled '504'. The 'Provision' option is selected and circled in green. The 'Create' option is circled in blue. The 'Delete' and 'Modify' options are also visible. The 'Database engine' dropdown is labeled '506' and shows 'MySQL'. The 'Version' dropdown is labeled '508' and shows '5'. Other fields include 'Repository name' (Repository A), 'Storage capacity' (150), 'Partition scheme', 'Security scheme', and 'Access key'. There are 'Save' and 'Cancel' buttons at the bottom right.

Sivasubramanian, Fig. 5

86. In Figure 5 (annotated above), the "action to be performed 504" (Sivasubramanian, 12:59-60), labeled "Task," shows the "Provision" option selected. POSAs understood that the "Provision" "action"/"Task" provisions a *database*, and that the "Repository name" is a database name, because the interface includes an option for selecting a "*Database* engine," and because of

Sivasubramanian’s repeated use of “repository” for “database” (as I noted *supra* §V.A.1). Sivasubramanian, 12:56-62, Fig. 5. Furthermore, the figure shows that the “Repository [*i.e.*, database] name” field lets the user select from options via a drop-down menu (indicated by the down arrow), indicating that the “Provision” “action”/“Task” provisions an already-***existing*** database the user selects, rather than creating a new database. Sivasubramanian, 12:38-40 (“In this example a customer wants to ***provision a new*** data instance, ***instead of maintaining and managing an existing MySQL database***”).

87. As shown above, Figure 5 also includes a “Create” option. POSAs understood that the “Create” option presents an interface for creating a *database*, for at least two reasons. First, Sivasubramanian describes the GUI and the “rds-create-***database***” command-line command as alternative methods for creating a database. Sivasubramanian, 12:49-13:5 (“***In some embodiments***, the user will receive software or will ***access an interface page*** through the Internet, for example, that will allow a user to submit requests to the control plane or service. ***For example, FIG. 5 illustrates an example of a display 500***, here a page rendered in a browser application, that can allow a user to make calls into the control plane... ***In other embodiments***, a user can manually (or otherwise) create and submit Web service calls to the control plane. ***In the following example***, the customer generates a request to create a new data repository using a ***command line***

tool.”). Second, Sivasubramanian describes “*database* engine” as an “option” for the “action 504,” labeled “Task 504” in Figure 5 (annotated above).

Sivasubramanian, 12:49-13:5.

88. Sivasubramanian says the Figure 5 interface “include[s] options that allow the user to enter information needed to perform a control action on the data plane.” Sivasubramanian, 12:55-57. Thus, POSAs would have understood Sivasubramanian to teach, or at minimum render obvious, that the fields for database-creation parameters presented in the “Create” interface would include fields for the parameters shown in Figure 5 for the “Provision” interface, as I explain below.

89. Sivasubramanian’s create-database workflow includes provisioning the database by allocating it resources. *Supra* §V.A.1. Thus, POSAs understood that the “Create” interface likewise included the fields on the “Provision” interface.

90. Further, as I discussed in Section V.A.1.a above and as I will discuss in greater detail below in Section V.A.1.a.ii, Sivasubramanian discloses an “rds-create-database” command-line command for creating a database.

Sivasubramanian, 13:1-5. The parameters Sivasubramanian shows “rds-create-database” accepting include parameters shown in Figure 5’s “Provision” task screen, including name, engine, and size. Sivasubramanian, 13:1-5 (“rds-create-database” command accepting “dbname,” “size,” “engine”), Fig. 5 (showing fields

for “Repository name,” Database engine,” and “Storage capacity”), 12:42-44. POSAs understood that in the parameter “dbname” shown in the “rds-create-database” command at Sivasubramanian 13:1-5, the “db” means “database,” because “db” (or “DB”) is a common abbreviation for database. *See, e.g.,* Sivasubramanian, 6:45-48 (“For example, if the request is for a MySQL® Relational Database Management System (RDBMS) instance, as opposed to an Oracle® RDBMS or other such instance.”). In addition, POSAs understood that “storage capacity” in Sivasubramanian Figure 5 is a different label for the same parameter as “size” in the “rds-create-database” command example at 13:1-5. Both the discussion of Figure 5 and the discussion of “rds-create-database” appear in Sivasubramanian in a section entitled “Customer Example.” Sivasubramanian, 12:37. At the beginning of this section, Sivasubramanian says that the “customer selects an instance to be provisioned with an initial *capacity* of 80 GB.” Sivasubramanian, 12:42-44. The number “80” is used as the value of the “size” parameter in the “rds-create-database” command example. Sivasubramanian, 13:1-5

91. Additionally, POSAs understood, or at minimum would have found obvious, that in the “Create” interface, the user could enter a new “Repository name,” *i.e.* database name as discussed above, for the new database being created,

rather than selecting the name of a previously-created database to provision as in the “Provision” interface.

92. To the extent not taught by Sivasubramanian, POSAs would have understood that having the “Create” task GUI accept inputs of parameters shown on the “Provision” task GUI would have been obvious. POSAs understood that a database “creation” GUI that also accepts parameters for provisioning a database was well-known. *See, e.g.*, RDS, 13-18 (describing interface for “Creating a MySQL DB Instance” that accepts parameters, including “engine,” “allocated storage,” and “master username” and then referring to the resulting “provision[ed] database”). Furthermore, Sivasubramanian itself teaches creating a database and then provisioning it, as I explained in Section V.A.1 above. Furthermore, POSAs understood that having the “Create” task GUI also accept input of parameters needed for provisioning would have the benefit of increased user convenience by letting the “Create” UI accept parameters needed to provision a created database (as Sivasubramanian itself teaches, as discussed above). Additionally, POSAs would have understood that accepting a new database name was logical in a UI for creating a new database. POSAs would have reasonably expected success implementing the “Create” UI in this manner because it was within a POSA’s skill, as I explain in Section V.A.2 below.

93. As I discussed in Section V.A.1.a above, Sivasubramanian’s Figure 5 GUI is used to submit requests to the control plane to initiate workflows. Thus, POSAs understood, or at least found obvious, that using the “Create” option in the Figure 5 GUI invokes the “create database” workflow, which *provisions* a new database (under the ’336 patent’s meaning) as I discussed above in Section V.A.1. Additionally, POSAs would have understood Sivasubramanian to disclose, or at least would have found obvious, that the “Provision” task likewise invokes a workflow for provisioning an existing database, since Sivasubramanian discloses users making requests/calls to the control plane to perform provisioning. Sivasubramanian, 5:16-20 (“In this example, a computing device 202 for an end user is shown to be able to make calls through a network 206 into a control plane 208 to perform a task Such as to provision a data repository of the data plane 210.”), 7:24-28 (“As mentioned, various embodiments can take advantage of a workflow service that can receive requests or calls for a current state of a process or task, such as the provisioning of a repository, and can return the current state of the process.”).

ii. Command-Line

94. The command-line tool presents a text-based interface (command line) for the user to enter text in a specified command format to input parameters for a task such as database creation. Sivasubramanian, 12:64-13:5. The command-

line format accepts input parameters as a list of pairs of parameter names and values, as shown in the example “rds-create-database” command:

```
rds-create-database --identifier customerprod --dbname customer --  
size 80 [-]-class small --engine mysql5.1 --master master_username --  
password master_password -port 4030
```

Sivasubramanian, 12:64-13:5.

95. Sivasubramanian teaches that the “rds-create-*database*” command creates a database, as indicated by the command’s name and by Sivasubramanian’s description of it as an example command for “generat[ing] a request to create a new data *repository*,” a term which in Sivasubramanian encompasses a “database,” as I discussed in Section V.A.1 above. Sivasubramanian, 12:64-13:5.

96. The command-line tool is used to submit requests to the control plane to initiate a workflow, as I discussed in Section V.A.1.a. Thus, Sivasubramanian teaches a POSA, or at least renders obvious, that the “rds-*create-database*” command invokes the “create database” workflow, which *provisions* a new database (under the ’336 patent’s meaning) as I discussed *supra* §V.A.1.

97. Furthermore, POSAs would have understood the “rds-create-database” command to be just one representative example discussed for illustration purposes, and would have been motivated to similarly implement a command-line version of the above-discussed “Provision” task in the Figure 5 GUI that provisions an existing database, as this would beneficially provide parallel

functionality with the GUI; POSAs would have expected success because implementing such a command, which may have been called, *e.g.*, “rds-provision-database,” was within a POSA’s skill as I discuss in Section V.A.2 below.

b. Backup “Workflow”

98. Sivasubramanian discloses a “recovery and backup” “workflow[.]” Sivasubramanian, 6:31-42 (“A workflow component (or system of components) in one embodiment is operable to manage and/or perform the hosting and executing of workflows for tasks Such as: repository creation, modification, and deletion; recovery and backup;”).

99. Sivasubramanian says the control plane can use “software frameworks” for “database backup and recovery.” Sivasubramanian, 16:21-27 (“As mentioned, the control plane layer can take advantage, or ‘sit on top,’ of various basic software frameworks for performing tasks such as: implementing workflows, ... and performing various database backup and recovery procedures”). Sivasubramanian also says a “framework” can handle “backing up both the data files and any associated log files.” Sivasubramanian, 19:25-28.

100. Sivasubramanian describes one exemplary “approach for backing up data” as follows:

An approach for backing up data in accordance with one embodiment *suspends data operations until s[n]apshots are taken of the appropriate data volumes*, and the log files a[re] similarly copied

into an appropriate location. For example, an Admin tier can wait for the backup window before initiating a backup procedure. Once inside the backup window, the Admin tier can create a workflow that will create a workflow instance for repository backups. ***In one example, the workflow invokes a “su[s]pendDatabaseForBackup”*** or similar API for the host manager

Sivasubramanian, 19:32-41. Sivasubramanian printed text says “the log files a similarly copied” at 19:34-35, but POSAs understood from context that this is a type and should say “log files are similarly copied.” Thus, POSAs understood that in Sivasubramanian’s exemplary “approach for backing up data,” a “su[s]pendDatabaseForBackup” command is used to “suspend[] data operations” while “s[n]apshots are taken of the appropriate data volumes.” Note that Sivasubramanian’s printed text at 19:40 says “suspendDatabaseForBackup,” but POSAs understood that this is a typo, since the same paragraph earlier refers to “suspend[ing] data operations.” Thus, I will refer in this declaration to the “su[s]pendDatabaseForBackup” API when discussing Sivasubramanian, 19:32-41. Similarly, Sivasubramanian’s printed text at 19:33 says “shapshots,” but POSAs likewise understood that this is a typo, since “shapshot” makes no sense in this context; thus, I refer in this declaration to “s[n]apshots” when discussing Sivasubramanian, 19:32-41.

101. Additionally, in Sivasubramanian’s “approach for backing up data,” “log files” are “copied” while the database is suspended. Sivasubramanian, 19:32-41. The log files are used for “replay[ing]... *transactions* in case the data files have to be *restored*”; thus, POSAs understood that Sivasubramanian’s “log files” are what other art and the ’336 patent refer to as “transaction logs” or “transactional logs.” For example, Zha says “database blocks retrieved by a point in time copy manager 310 combined with the *transaction logs* retrieved by the transaction log manager 320 *can be used to reconstruct a copy of a database* in the production system 110 corresponding to times in the past in between the times as which point-in-time copies are made.” Zha, 10:6-11. Similarly, RDS says “Amazon RDS uses these periodic data backups in conjunction with your *transaction logs to enable you to restore* your DB instance to any second during your retention period.” RDS, 112. The ’336 patent uses the terms “transaction log” and “transactional log” interchangeably to describe logs that can be used to “recover[]” a database. EX1001, 36:49-54 (“Advantageously, the snapshot and *transactional log capturing* efficiently only copies changes in the memory blocks. Furthermore, all *transactions are recorded using transaction logs* such that when a clone is created, *the source database may be recovered based on* both the snapshots and *the transaction logs to any desired state.*”). See also Priebe (EX1060), 1:14-22 (filed April 27, 2018 and referring in “BACKGROUND” to “a

transaction log stored... so that the transaction log may be used to recover the database by *replaying* the transactions recorded in the database system”); 1:61-63 (referring in “BACKGROUND” to “the transaction log (or just log)”; Quintanillo, (EX1061), 4:17-30 (filed in 2005 and issued in 2009 and referring to “*replaying transaction logs*” in order to “allow[] the database snapshot to be updated with any actions that have occurred since the snapshot was formed”).

2. Motivation to Customize Sivasubramanian’s UI

102. Although Sivasubramanian’s Figure 5 shows a particular example set of configuration parameters that can be input, in my opinion, POSAs understood that the UI shown could beneficially accept additional inputs not expressly shown, for the reasons I explain below.

103. Sivasubramanian says Figure 5 is an “*example* of a display” in an “interface” including “options that allow the user to enter *information needed* to perform a control action on the data plane,” and notes that “*other* such values” aside from those shown can be specified. Sivasubramanian, 12:53-64.

104. Sivasubramanian also teaches that the user interface to the control plane is flexible and enables the user to enter whatever options are appropriate to accomplish a desired workflow. Sivasubramanian, 11:49-52 (“The user can call a Web service to specify *any* desired values for an instance type..., storage size,

repository name, port, *and other such values.*)², 2:23-26 (“The functionality of a control plane can be provided as a set of Web services, enabling the control plane to act as a virtual database administrator (DBA).”), 11:41-47 (“...users of the control plane can perform various tasks relating to data repositories and data instances using a set of APIs or other such interfaces. While the selection and names of the example APIs are used for purposes of explanation, it should be apparent that *other selections, combinations, names, and other aspects can vary* between the various embodiments.”), 6:7-12 (“control plane” includes a “‘sweeper’ component” that “determine[s] any tasks to be executed in response to an outstanding request”), 6:45-52 (“For example, if the request is for a MySQL® Relational Database Management System (RDBMS) instance... then a specific task will be added to the workflow that is directed toward MySQL instances. The workflow component also can select specific tasks related to the amount of storage requested, any specific hardware requirements, or other such tasks.”).

105. Furthermore, Sivasubramanian’s discussion of the “rds-create-database” database-creation command shows different options for parameters than those shown in Figure 5, which in my opinion would have indicated to a POSA that other parameters could similarly be added to Figure 5’s GUI.

² Emphases added unless otherwise indicated.

Sivasubramanian, 13:1-5 (“rds-create-database” command including parameters for “identifier,” “class,” “password”).

106. In my opinion, such UI implementations were well within a POSA’s skill. Sivasubramanian describes Figure 5 as an example page rendered in a “browser application.” Sivasubramanian, 12:52-55. Sivasubramanian also discloses implementing its functionality on “[w]eb server[s]” running “applications” implemented in “any programming language.” Sivasubramanian, 23:4-16. POSAs had the skill to use “programming language[s]” to implement an application (such as a web application) to display a version of Figure 5’s exemplary UI including additional options.

107. For the same reasons I discussed above with respect to the GUI, in my opinion POSAs would have been motivated to implement Sivasubramanian’s command-line tool to accept additional inputs not shown in the example at 13:1-5. It would have been within a POSA’s skill to use a “programming language” to implement such a command line, and to implement additional commands such as the “rds-provision-database” command that I discussed in Section V.A.1.a.ii above. Sivasubramanian, 23:4-16.

B. Zha (EX1006)

108. Zha teaches techniques for creating a “virtual database” that is used for purposes including “backup.” Zha, Abstract (“A virtual database (VDB) is

created by creating a set of files in the data storage system. ... Workflows based on VDBs allow various usage scenarios based on databases to be implemented efficiently, for example, testing and development, backup and recovery.”).

109. The “virtual database” is a copy of an existing database from a particular point in time. Zha, 6:49-59 (“To create a virtual database, the database storage system 100 creates files that represent the information corresponding to the production database system 110 at a given point in time. ... Hence, a virtual copy of the production database is created for the virtual database system 130 for a given point in time in a storage efficient manner.”). Zha’s “backup operation creates a copy of the data available in a storage system such that the backup copy... can be used to reconstruct information of the original... in case the original data is lost.” Zha, 24:24-27.

110. Zha’s “[m]ethods of creating a virtual database (VDB) read different point-in-time copies of a source database,” which Zha also refers to as a “production database.” Zha, 1:66-2:6.

111. Zha also teaches retrieving “transaction logs” from the source/production database; these logs can be combined with point-in-time copies to “reconstruct a copy of a database... corresponding to times in the past in between the times [at] which point-in-time copies are made.” Zha, 9:63-10:11; *See also* Zha, 15:58-62 (“Obtaining incremental changes to the transaction logs at a

high frequency provides the ability to recreate a copy of a database from the production database system 110 at a time point in between the times that a point-in-time copy is made by the point-in-time copy manager 310.”), 18:1-3 (“a VDB can be created for any time point in between PIT copies by appropriately applying the transaction logs to a previous PIT copy.”).

112. Zha’s technique periodically copies data from a source/production database to a “Database Storage System 100” that stores data copied from source databases throughout an organization. Zha, 6:11-48, Fig. 1. For a given source database, the copied data that is stored in the central “Database Storage System” can be used to create a “virtual database[]” that represents a copy of the source database as it existed at a certain point in time. Zha, 6:49-58.

113. Zha’s above-mentioned “database storage system” (where database copies are stored) can be configured by a “database administrator” using a “system configuration manager.” Zha, 11:26-28. This “manager” lets a user “setup... the *schedule*” specifying the “frequency and times during the day” for “retriev[ing] *PIT [point-in-time] copies of databases*” and “retriev[ing] updates to online *transaction logs*”:

The system configuration manager 315 also allows a user with appropriate roles and privileges to setup *policies* specifying the *schedule* with which the point in-time copy manager 310 *retrieves PIT [point-in-time] copies of databases* in the production database systems

110 *as well as the frequency and the times at which the transaction log manager 320 retrieves updates to online transaction logs* from the production database systems 110. In an embodiment, a schedule can specify *the frequency and times during the day for the PIT* and log retrieval actions or it could be an a periodic [sic: aperiodic] schedule specifying the calendar days when the same action should take place.

Zha, 11:33-44.

114. In my opinion, POSAs would have understood that “point-in-time copies” as discussed in Zha are “snapshots” of the source database at the time the copy was made, for multiple reasons. As I noted in Section IV.A above, POSAs understood that the term “snapshot” referred to a record of the database’s state at a point in time. *See, e.g.,* Simek, [0002] (“Database Snapshots are a feature of some database management systems (‘DBMS’) and allow a user to specify a point in time at which to preserve the contents of the database.”); Ohno, 1:54-61 (“In order to restore the contents of the database up to a point in time when the data has consistency, a snapshot must be acquired in advance in a state in which data has consistency.... Snapshot is to maintain data to be snapshot at a certain point in time in an external storage, and it is a function provided by most storage systems.”).

115. Furthermore, Zha states that “transaction logs can be applied to a VDB [virtual database] created based on a PIT copy so as to create a *snapshot* of the source database for a time point that occurs *after* the PIT copy was made.”

Zha, 17:63-66. POSAs understood that if applying transaction logs created a “snapshot” of the database at a time *after* the PIT copy was made, then the PIT copy itself represents a “snapshot” *at* the time it was made, since Zha also explains that the transaction logs “can be used to reconstruct a *copy* of a database... corresponding to times in the past in *between the times as which point-in-time copies* are made.” Zha, 10:6-11.

116. Zha also discloses “policies defined by a database administrator” for operations including “purging of information... including point-in-time copies of databases and transaction log information.” Zha, 11:45-54. Such “policies” may include a “schedule for purging the point-in-time copy after a retention period” (Zha, claim 11, 35:18-20), and may also “a schedule for purging the transaction logs after a retention period” (Zha, claim 13, 35:24-26).

C. Sivasubramanian+Zha Combination

117. In my opinion, POSAs seeking to implement Sivasubramanian’s “backup” workflow that I discussed *supra* §V.A.1.b would have considered references like Zha regarding certain implementation details. As I explain below, although Sivasubramanian and Zha use slightly different terminology, both discuss database backup using snapshots and transaction logs, with Zha’s teachings supplying further details. POSAs would have found it obvious to implement

Sivasubramanian using Zha’s teachings in multiple independent ways, as I discuss below.

118. Sivasubramanian’s discussion of “snapshots” appears in a paragraph describing “[an] approach for backing up data”; this paragraph immediately follows a paragraph that introduces Sivasubramanian’s discussion of “*repository* and data backup” and that also notes that “*approaches* can differ from various *database engines*.” Sivasubramanian, 19:20-35. Thus, POSAs understood from the language and structure of Sivasubramanian that Sivasubramanian discloses *database* snapshots. This understanding from Sivasubramanian’s text is reinforced by an earlier passage in the text. When discussing database creation commands, Sivasubramanian says:

The customer may also want to take a *snapshot* of the *production instance* so that the test instance is fully populated and comparable with the production dat[a]. The customer decides that, for the particular needs of the testing procedure, the customer can utilize a *SMALL instance* for purposes of processing capacity, and can provision the same storage capacity as is used for production. The customer thus *can submit a request to clone the database* using a command line tool such as by submitting:

```
create-database --identifier customertest --dbname tcustomer --size
150 --class small --engine mysql5.1--master master_username --
password master_password--port 4030
```

Sivasubramanian, 15:1-15. POSAs would have understood this as a discussion of “tak[ing] a snapshot of a production instance” so that a “test instance” created with a “create-database” command is capable of then being “fully populated and comparable with the production dat[a].” I note that Sivasubramanian’s text says “production date” at 15:3, but POSAs would have understood that this is a typo and should have said “production data,” since “production date” would not make sense in context.

119. Additionally, Sivasubramanian’s exemplary “approach” uses a “su[s]pend**Database**ForBackup” command to “suspend[] data operations” while “s[n]apsshots are taken of the appropriate data volumes,” as I discussed in Section V.A.1.b above. Sivasubramanian, 19:32-41. Sivasubramanian also discloses that a database’s data is stored on “volume[s]” allocated to a database’s storage “instance.” Sivasubramanian, 6:59-63 (““create database’ workflow... might include... provisioning a data store instance... [and] attaching [a] persistent storage volume to the data store instance”), 8:22-25, 15:56-60.

120. Moreover, POSAs understood that taking snapshots of underlying data volumes was a typical and customary way to take a database snapshot. My testimony regarding a POSA’s background knowledge is corroborated by various prior art references. For example, “LVM-Howto” (EX1022) is a blog post from 2007 entitled “Taking a Backup Using Snapshots” that discusses how to back up a

database using snapshots. LVM-Howto, 1. The technique described involves creating a “snapshot logical volume [LVM]” that can be used to copy the “changes to the underlying database volume.” LVM-Howto, 1. “LVM-Howto” is a blog post from 2007 entitled “Taking a Backup Using Snapshots” that discusses how to back up a database using snapshots. LVM-Howto, 1. The technique described involves creating a “snapshot logical volume [LVM]” that can be used to copy the “changes to the underlying database volume.” LVM-Howto, 1. I understand that Exhibit 1022 was printed from the URL

https://web.archive.org/web/20070120200004/https://tldp.org/HOWTO/LVM-HOWTO/snapshots_backup.html. Based on my familiarity with the Wayback Machine as detailed in paragraph 34 above, I understand the URL from which Exhibit 1013 was printed is the Internet Archive’s record of the website at the URL https://tldp.org/HOWTO/LVM-HOWTO/snapshots_backup.html archived on January 20, 2007. Another example is Ohno (EX1015). Ohno explains that “to restore the contents of the database up to a point in time when the data has consistency, a snapshot must be acquired in advance in a state in which data has consistency. For this purpose, it is necessary to staticize the database and acquire a snapshot thereof in a state in which data has consistency.” Ohno, 1:54-59. Ohno explains that some database management systems (“DBMS,” *see* Ohno, 1:28-29) “provide a function to staticize a database virtually.” Ohno, 2:3-6. In such

systems, “[a] typical approach is to staticize a virtual database using a virtual database staticizing function of a DBMS and to create a snapshot of a file or logical volume which is to be snapshot at the point in time when the database is staticized.” Ohno, 2:18-21. In this approach, “[a] snapshot can be created by using an LVM (Logical Volume Manager) of an, software provided by various vendors, or a snapshot creating function provided by an external storage.” Ohno, 2:26-29. In another example, McAlister³ (EX1055), whose inventors are two of the inventors of Sivasubramanian, describes taking snapshots of underlying database volumes as a way “to create snapshots of” a database “instance.”

McAlister, 17:23-40, Abstract (referring to a “replicated database instance”). As McAlister explains, “[t]o create snapshots of a non-replicated [database] instance, the workflow effectively suspends I/O to the database volumes, takes block storage-level snapshots of the volumes attached to the instance, and un-suspends the volume.” McAlister, 17:29-32. Additionally, McAlister notes that “[Amazon] RDS can enable customers to create snapshots of their instances and create new instances from these snapshots.” McAlister, 17:25-27.

³ I understand that EX1055 shares the same first-named inventor with Sivasubramanian (EX1005). To avoid confusion, I refer to EX1055 by the second-named inventor (McAlister).

121. Thus, POSAs understood Sivasubramanian to disclose taking *database* snapshots by taking snapshots of the underlying data volumes.

122. POSAs understood that “tak[ing]” “snapshots” of a database in Sivasubramanian (Sivasubramanian, 15:1-5, 19:20-64) is analogous to what Zha refers to as “retriev[ing] PIT [point-in-time] copies” (Zha, 11:33-44, 17:63-66), and likewise understood that “backing up” the “log files” in Sivasubramanian (Sivasubramanian, 19:65-20:11) is analogous to what Zha refers to as “retriev[ing] updates to online transaction logs” (Zha, 11:33-44); both references’ processes result in the snapshot/log being stored in some storage location used for storing snapshots/logs. *Compare* Sivasubramanian, 19:32-35 (“An approach for backing up data in accordance with one embodiment suspends data operations until *snapshots are taken* of the appropriate data volumes, and the *log files a similarly copied* into an *appropriate location*.”) *with* Zha, 2:19-24 (“The database blocks retrieved from the source database and stored on the storage system correspond to different point-in-time copies of the source database...”), 9:67-10:3 (“The data obtained by the transaction log manager... is stored in the storage system data store[.]”).

123. In my opinion, POSAs understood that Sivasubramanian does not restrict implementation details such as daily frequency of retrieving snapshots and transaction logs, what time of day these events occur, or retention duration of

snapshots and logs. Sivasubramanian, 19:20-20:11. However, POSAs understood it was typical to collect snapshots, logs, and other database-backup information at user-configurable frequencies and times and retain such information for user-configurable durations. My opinion regarding this aspect of a POSA's background knowledge is corroborated, for example, by Bhasin (EX1016), [0009] ("...in some embodiments the snapshots are maintained according to a configured or configurable retention time period"); RDS, 16 (describing a configurable "Backup Retention Period" setting as "the number of days you want automatic backups of your database to be retained."), 111-112 ("Automated Backup" section saying, "The preferred backup window is the user-defined period of time during which your DB instance is backed up."); Tekade (EX1017), [0046] ("FIG. 6 depicts a user interface for configuring a single SLA protecting a database with an associated log on different schedule... This Snapshot policy states that a database backup will run every day 605 with exception setting 606 and operating window start time 607 and end time 608. The backup will run every 24 hours specified by 609... the RPO field 613 specifies log backup runs every 30 minutes when database log backup is enabled"), Fig. 6; Simek, [0034] ("the user interface 200 includes a control 208 that allows a user to specify a parameter defining the frequency at which the database snapshots 122 should be created."), Fig. 2.

124. Furthermore, POSAs understood that taking and storing snapshots and transaction logs required computing resources and storage space and potentially impacted the database's performance; thus, POSAs understood that allowing users of Sivasubramanian's control plane to specify snapshot and log frequencies, times, and retention periods beneficially enabled users to tune those parameters to suit their needs. My testimony regarding a POSA's background knowledge is corroborated by, *e.g.*: Madany (EX1023), [0023] (explaining how available storage limits constrain snapshot storage and noting that various schedules and retention periods can be used to mitigate constraints); RDS, 5 ("Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database."); Sivasubramanian, 19:31-41 (exemplary backup scheme requires suspension of database); Ohno, 1:54-61 (prior art snapshotting commonly required pausing database).

125. Therefore, POSAs seeking to implement Sivasubramanian's exemplary "approach for backing up data" (Sivasubramanian, 19:20-20:11) would have been motivated to use an administrator-defined "schedule" including a "frequency" and "times during the day" for taking snapshots and backing up transactional/transaction logs, and to store them for administrator-defined "retention periods," as Zha discloses. *See* Zha, 11:33-54 (discussing administrator specifying a "schedule" that can "schedule can specify the frequency and times

during the day for the PIT and log retrieval actions” and “policies” for “purging of information from the database storage system 100 including point -in-time copies of databases and transaction log information”), 35:18-20 (reciting “predetermined policy specifies a schedule for purging the point-in-time copy after a retention period.”), 35:24-26 (reciting “predetermined policy specifies a schedule for purging the transaction logs after a retention period”).

126. In the resulting “Sivasubramanian+Zha” combination, Sivasubramanian’s database-provisioning UIs (including UIs for creating new databases and allocating them resources, and UIs for provisioning existing databases) that I discussed *supra* §V.A.1.a accept user-defined snapshot and transaction-log collection frequencies, times during the day, and/or retention periods, as detailed below. Additionally, in the combination, the provisioning process would setup the database to have snapshots and logs (1) collected at the respective user-defined frequencies and times and (2) stored for the respective user-defined periods. *See* Sivasubramanian, 19:32-39 (discussing collection of snapshots and logs).

127. Alternatively, POSAs would have found it obvious to employ Zha’s technique of creating VDBs (alternatively or additionally to creating Sivasubramanian’s snapshots) to beneficially achieve the “efficient[] use [of] storage and computing resources” that Zha’s technique provides. Zha, 1:63-65.

POSAs would have been motivated to customize Sivasubramanian’s database-provisioning UIs to let the user set snapshot and log schedules and retention periods for the same above-discussed reasons, and the resulting system would incorporate the UI elements that I detail *infra* §§V.C.1-V.C.2 into Sivasubramanian’s database-provisioning UIs. The resulting system would also setup the provisioned database to have “point-in-time copies,” *i.e.* snapshots, and “transaction logs” “retriev[ed]” from it, as taught in Zha, based on user-entered parameters as I discussed above. Zha, 11:33-44; *supra* §V.B.

128. In my opinion, POSAs would reasonably have expected success with either alternative above-discussed implementation of Sivasubramanian+Zha, for the reasons I explain below.

129. As I detail *infra* §§V.C.1-V.C.2, POSAs had the skill to implement Sivasubramanian’s UIs to accept the inputs Zha teaches.

130. POSAs also had the skill to have the provisioning process setup the database to have snapshots and logs taken/retrieved and retained based on user-entered parameters. Sivasubramanian itself discloses the control plane executing tasks based on user-entered parameters. *See, e.g.*, Sivasubramanian, 11:41-52 (“users of the control plane can perform various tasks relating to data repositories and data instances using a set of APIs or other such interfaces.... As discussed in one of the examples above, customers can create a data store using a

‘CreateDatabase’ or similar API. The user can call a Web service to specify any desired values for an instance type (which describes the CPU and memory capacity), storage size, repository name, port, and other such value”), 12:64-13:5 (discussing example of user using “rds-create-database” command-line command and specifying parameter values). Furthermore, POSAs understood Sivasubramanian already discloses functionality to collect snapshots and logs. Sivasubramanian, 19:32-39 (explaining that in the exemplary “approach for backing up data” “suspends data operations until s[n]apshots are taken of the appropriate data volumes, and the log files a similarly copied into an appropriate location” and noting that “the Admin tier can create a workflow that will create a workflow instance for repository backups.”)

131. Moreover, Sivasubramanian says the “approach” it describes for backup is “*/a/n*” approach, meaning that Sivasubramanian’s control plane could accommodate other approaches like Zha’s. Sivasubramanian, 19:32.

132. In my opinion, POSAs had the skill to implement Sivasubramanian’s control plane to use Zha’s “virtual database” technique. Zha’s teachings can be implemented in software, and Sivasubramanian likewise discloses implementing the control plane using software “framework[s]” that can perform tasks including “backup and recovery.” Zha, 32:49-52; Sivasubramanian, 16:21-28.

1. UI for retention period

133. POSAs understood that it was well-known to specify a “service level” setting retention periods for database-protection-related data such as snapshots and transaction logs; such levels were often specified in a “service level agreement” (“SLA”). My testimony regarding a POSA’s background knowledge is corroborated by, *e.g.*: Tekade, [0046], Fig. 6 (depicting a “user interface for configuring a single SLA protecting a database” including specifying a “snapshot policy” dictating that a “backup image is *retained* for [a specified] number of day[s]”); Lubrecht (EX1024), [0120] (describing an SLA as “including restore time, backup window, backup success rate, rotation, and *retention policies*”); Utsunomiya (EX1025), [0004] (“Therefore, it is important how to classify the files based on the service levels. For example, in the case of archiving, a *retention period can be considered as a service level.*”); Freeman (EX1026), 11:39-43 (“*A Service Level Agreement* is a detailed specification that *captures* the detailed business *requirements* related to the creation, *retention* and *deletion* of secondary copies of the application data.”); Jain (EX1027), 19:32-35 (“For example, the retention window may be a time period defined by a service level agreement which defines the time period in which the database partition may be restorable to a particular point-in-time.”).

134. It was also well-known background knowledge to define multiple SLA “levels” named, *e.g.*, “gold,” “silver,” etc. *See, e.g.*, Gomatam (EX1028), [0046] (disclosing “cloud infrastructure system” providing “database services”), [0056]-[0058] (disclosing allowing customer to order “services” and also order “one or more service levels for the ordered services” by specifying “*a Service Level Agreement (SLA)*” from “different customer levels includ[ing] *a basic level, a silver level, and a gold level*”); Higginson (EX1029), 9:4-7 (“Next, a plurality of levels can be created, each having its own template. Merely for exemplary purposes, the levels described herein will be labeled Gold, Silver, Bronze, and so forth, to specify different database levels.”); Esposito (EX1030), 3:62-4:6 (“The user selects backup and recovery (BRS) profiles and service level agreements (SLAs) during step 340.”).

135. In Sivasubramanian+Zha, POSAs would have been motivated to have control-plane users specify Zha’s configurable snapshot/log retention periods (*supra* §V.B) by specifying an SLA level, for multiple reasons. POSAs understood that it was beneficial to give users familiar UIs, and POSAs also understood that users were familiar with SLAs to specify retention periods, as I discussed above. Moreover, POSAs also understood that users were generally familiar with specifying time intervals in UIs, since such UIs were also very well-known in the art. *See, e.g.*, Tekade, Fig. 6 (showing fields for specifying various time intervals

including a “retain for” field (item 610)); RDS, 15-16 (describing a “configure advanced settings page,” which POSAs understood is a UI, with a field for “backup retention period” that specifies “the number of days you want automatic backups of your database to be retained”); Simek, Fig. 2 (showing “Duration” field 212 that specifies “how long to keep snapshots”)

136. Furthermore, POSAs understood that letting users choose from predefined sets of parameter values was well-known and increased user convenience. For documentary evidence corroborating my testimony regarding a POSA’s background knowledge, *see, e.g.*, Higginson, 9:49-10:5 (describing user interface that presents predefined database templates and saying such techniques “allow[] a customer to quickly select a type of database based on its use”); Poulsen (EX1031), 1:16-29 (stating in “Background” that “[t]he use of profiles is a convenient way to save time in performing various types of tasks” including setting parameters for software and devices)

137. Thus, in either above-discussed Sivasubramanian+Zha implementation, POSAs would be motivated to let the user specify transaction log and snapshot retention periods in the database-provisioning UIs (including UIs for creating new databases and allocating them resources, and UIs for provisioning existing databases) that I discussed *supra* V.A.1.a via the well-known technique of SLA level selection, to beneficially increase user familiarity and convenience.

138. Furthermore, POSAs understood that accepting input of the SLA in connection with provisioning new or existing databases beneficially allowed a database to be protected as soon as it is provisioned, since parameters for snapshot and log capturing and retention would already be specified.

139. Moreover, as I explain below, it was well-known in database-creation and provisioning UIs both to (1) accept inputs related to retention of snapshots or other backup-related data; and (2) accept SLA-level selection.

140. Including fields related to database protection in UIs for database provisioning was well-known. For example, RDS discloses that the database creation options include a daily “backup window” for taking “snapshots.” *See* RDS, 16-17, 187. Oracle (EX1012) discloses that options for provisioning a new database in Oracle Database Cloud include a “Backup Destination.” Oracle, 7. And Google (EX1032) discloses that the options for creating a MySQL instance in Google Cloud SQL included “Automatic Backups” and “Binary Logging.” Google, 2-3. I understand that Exhibit 1032 was downloaded from the URL <https://www.archive.org/web/20180918103721/https://cloud.google.com/sql/docs/mysql/create-instance>. Based on my familiarity with the Wayback Machine as detailed in paragraph 34 above, I understand the URL from which Exhibit 1032 was downloaded is the Internet Archive’s record of the website at the URL

<https://cloud.google.com/sql/docs/mysql/create-instance> archived on September 18, 2018.

141. Accepting SLA level selections in UIs for database provisioning was also well-known. *See, e.g.*, Gomatam, [0046], [0056]-[0058] (SLA specified as part of an “order” for services including “database services”); Higginson, 9:49-10:12, Fig. 4 (showing various “levels” in “database provisioning wizard”).

142. In Sivasubramanian’s GUI, POSAs would have been motivated to implement SLA selection via a selectable interface like a drop-down menu, since such interfaces were well-known, including in Sivasubramanian.

Sivasubramanian, 15:59-60, Fig. 5 (*e.g.*, “Database engine”); Tekade, Fig. 6; RDS, 15 (figure showing fields such as “DB Instance Class” where user is prompted to “Select One”). And POSAs would have been motivated to implement the “rds-create-database” command and the obvious “rds-provision-database” command (*see supra* §V.A.1.a.ii) to accept a parameter whose value represents a level, like the “engine” parameter that specifies an engine. Sivasubramanian, 13:1-5. POSAs would have reasonably expected success with such UI implementations, because SLAs were well-known (*see above*), and because customizing Sivasubramanian’s UIs was within a POSA’s skill (*supra* §V.A.2).

143. Furthermore, POSAs would have found it obvious to have the SLA specify either one retention period for both snapshots and logs or separate periods

for each. POSAs understood there were two options—uniform or non-uniform—and either was an obvious design choice. POSAs would have additionally recognized that uniform periods beneficially simplify snapshot/log management. POSAs would have reasonably expected success because it was simple to apply one duration to all snapshots/logs, and because specifying one uniform retention period for data related to database backups was well-known in the art. *See, e.g.*, Tekade, [0046] (“A Snapshot policy of dbSnap of 601 as an example, it has a name field 603, with a priority level 604. This Snapshot policy states that a database backup will run every day 605 with exception setting 606 and operating window start time 607 and end time 608. The backup will run every 24 hours specified by 609, and the backup image is retained for number of day specified by 610... For example, using SLA template in FIG. 6, both database and log backup can be retained for 7 days.”); Bhasin, [0009] (“In some embodiments, the generated snapshots are kept indefinitely, but in some embodiments the snapshots are maintained according to a configured or configurable retention time period” and giving examples of “one day,” “week,” *i.e.* 7 days, “month,” *i.e.* 30 days, “year,” *i.e.* 365 days); RDS, 16 (“Backup Retention Period” parameter is “the number of days you want automatic backups of your database to be retained”), 111-112 (“Backups created during the backup window are retained for a user-configurable number of days (the backup retention period)”). A POSA would have reasonably

expected success with either approach, as both UI implementations were within a POSA's skill as I discussed in Section V.A.2 above.

2. UI for protection schedule

144. As I explain below, POSAs would have been motivated to customize Sivasubramanian's Figure 5 GUI to accept an input defining Zha's above-discussed "schedule" including a "frequency" and "times during the day" for taking snapshots and backing up transactional/transaction logs. Zha, 11:33-54; *supra* §V.B.

145. Fields related to database protection in database-provisioning UIs were well-known. RDS, 16-17, 187 ("backup window" for taking "snapshots"); Oracle, 7 ("Backup Destination"); Google, 2-3.

146. Additionally, POSAs understood that accepting a protection schedule via a database-creation and/or provisioning UI beneficially allowed the database to be protected as soon as it is created and provisioned, since the schedule would already be specified.

147. POSAs would have reasonably expected success with such UI implementations, because including fields related to database protection in database-provisioning UIs was well-known, as I discussed above, and because customizing Sivasubramanian's UIs was within a POSA's skill (*supra* §V.A.2).

148. POSAs would have found it obvious to add parameters for specifying the protection schedule to Sivasubramanian’s command-line database-provisioning commands (*supra* §V.A.1.a.ii) for the same reasons.

D. Claim-by-Claim Analysis

1. Claim 1

a. [1PRE]: “A method comprising”

149. Sivasubramanian discloses a “computer-implemented *method* of controlling at least a configuration aspect of a data environment using a control environment.” Sivasubramanian, claims 1, 4, 15; *see also* 2:17-18. As explained below, Sivasubramanian’s *method*, in view of Zha’s teachings as described *supra* V.C, *comprises* the remaining steps recited in claim 1.

b. [1A1]: “receiving, by a processor executing computer-readable instructions stored on a memory...”

150. Sivasubramanian’s invention can be implemented with a “computing device” (22:33-37) including “*computer readable media*” that can include “*storage* media,” examples of which include various “*memory*” types (23:61-24:8); these media are capable of “*storing*...computer-readable information” (23:42-48) including “*computer readable instructions*” (23:61-24:8). Sivasubramanian also says the control plane includes “computer-executable software.” Sivasubramanian, 5:35-39 (“The control plane in this embodiment includes a web services layer 212, or tier, which can include at least one web

server, for example, along with computer-executable software, application servers, or other such components.”). POSAs understood that “computer-executable software” is made up of *instructions*. Sivasubramanian also includes claims reciting a “control environment” operating “under control of one or more computer systems configured with executable *instructions*.” See Sivasubramanian, 24:20-24 (claim 1 reciting “a computer-implemented method of controlling at least a configuration aspect of a relational database using a control environment,” where the method steps are performed “under control of one or more computer systems configured with executable *instructions*”), 25:4-8 (claim 4, similar), 26:23-26 (claim 15, similar). Thus, Sivasubramanian discloses implementing its “control plane” via *computer-readable instructions stored on a memory*.

151. Furthermore, Sivasubramanian teaches that *a processor executes* the aforementioned *instructions* implementing the “control plane.” As discussed above, the *instructions* are Sivasubramanian’s above-discussed “computer-readable instructions” for the control plane; these instructions are intended to be executed by a processor. Sivasubramanian, 27:18-35 (claim 20 reciting “system for controlling a data environment using a control environment” comprising “processor” and “instructions that, when executed by” the “processor” “cause the system” to perform actions), 5:35-39 (control plane includes “computer-executable software”).

152. As explained below, all steps recited in [1A2]-[1E] are performed in the course of provisioning a database using the control plane's database-provisioning UIs; thus, since the *processor* executes the control-plane instructions as I discussed above, *the processor* performs the claim's recited tasks, including *receiving* the required *user request* ([1A2]).

c. [1A2]: “a user request for provisioning a source database...”

i. GUI

153. As I explained above generally in Section V.A, and more specifically in Sections V.A.1 (overview of Sivasubramanian) and V.A.1.a (discussing Sivasubramanian's database-provisioning UIs), Sivasubramanian discloses starting the process for provisioning a database in response to user request.

Sivasubramanian's Figure 5 “illustrates an example of a display... to make calls into the control plane” when the user “*wants to provision a new data instance*” (*i.e., a database, supra* §V.A.1). Sivasubramanian, 12:38-40, 12:52-55.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address

Control interface page

Repository name: Repository A

Task: ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine: MySQL Version: 5

Storage capacity: 150 Units: GB

Partition scheme:

Security scheme: Version: 0

Access key: Version: 0

Save Cancel

Sivasubramanian, Fig. 5 (annotated)

154. As I explained *supra* §V.A.1.a.i, control-plane users could either: (1) select the “Provision” option to *provision* an existing *database*; or (2) select the “Create” option to *provision* a newly-created database, as discussed *supra* §V.A.1.

155. The ’336 patent describes a “*source database*” as a database from which snapshots are taken to create backups. *See, e.g.*, EX1001, 4:46-57, 5:20-23. As I discuss below for Element [1E], both the “Provision” and “Create” tasks in the GUI set up the database being provisioned to have snapshots captured from it to create backups, according to either Sivasubramanian+Zha implementation I discussed *supra* §V.C; thus, in Sivasubramanian+Zha, the database that is

provisioned is a *source database*. See Zha, 1:66-2:6 (referring to backed-up database as “*source database*”).

156. Sivasubramanian describes the Figure 5 UI as an “interface page” that a “*user*... will access... through the Internet.” Sivasubramanian, 12:48-52.

POSAs understood that the *user* “access[es]” the page through the Internet by making a “*request*” that the control plane “*receives*.” Sivasubramanian, 5:35-41 (“[t]he control plane... includes a Web services layer” that “*receiv[es]* Web services calls or *requests* from across the network”). This “user” request for “access” to the Figure 5 provisioning UI is a *user request for provisioning a database*, since this request invokes the database provisioning UI (shown in Sivasubramanian Fig. 5, above). EX1001, 13:2-5 (“[i]n response to the *user request* for a database *provisioning* service, the database engine... may activate the database *provisioning* system[.]”).

157. Moreover, Sivasubramanian says Figure 5 is example “interface” that lets a “*user*... submit *requests* to the control plane,” which the control plane “*receiv[es]*”; thus, the request submitted using the Create or Provision task is also a *user request* as claimed. Sivasubramanian, 12:49-55 (“The *customer*, if not already signed up or subscribed to the control service, can sign up for the service. In some embodiments, the user [*i.e.*, customer] will... access an interface page... for example, that will allow a user to submit *requests to the control plane* or

service. For example, FIG. 5 illustrates an example of a display 500, here a page rendered in a browser application, that can allow a user to make calls into the control plane.”), 10:15-17 (“A *request*, such as a Web services call, is *received* through a *customer*-facing control plane interface component 302.”); *see also* claims 1, 4, 20, 25 (all reciting a “request” or “call” that is “receiv[ed]”).

ii. Command Line

158. Sivasubramanian’s “*rds-create-database*” command *provisions* a new database, as I discussed in Section V.A.1.a.ii above. Sivasubramanian, 12:53-13:5. Additionally, a command such as the “rds-provision-database” command that would have been obvious to implement (as I discussed in Section V.A.1.a.ii above) *provisions* an existing database. As I discussed above for the GUI, the database is a *source database*. Sivasubramanian says the command-line is an alternative example to the GUI of an “interface” that lets a “*user*... submit *requests* to the control plane”; thus, the “rds-create-database” and “rds-provision-database” commands are each a *user request* as recited that the control plane *receives*. Sivasubramanian, 12:49-67.

d. [1A3]: “in a hyperconverged infrastructure system”

159. The term “hyperconverged” appears only in the ’336 patent’s title, and is nowhere in the specification.

160. Sivasubramanian’s databases are “create[d]” and “*provision[ed]*” in the “data plane.” Sivasubramanian, 7:3-9 (“workflow component can manage execution” of “tasks” such as “create database” in the “data plane”), 2:40-44 (“Once provisioned, a user can... access... the data instance(s) in the “data plane....”), 5:15-20, 12:55-66, 13:28-29.

161. Sivasubramanian states that “*in one embodiment*,” the “data plane” is a “cloud” system. Sivasubramanian, 9:9-13. However, Sivasubramanian does not *require* that the data plane be a cloud. *See, e.g.*, Sivasubramanian, 24:19-28:56 (Sivasubramanian’s claims not requiring cloud).

162. Rather, POSAs understood that Sivasubramanian is directed towards automating workflows (as I discussed for example in Section V.A.1 above) but is flexible as to the environment where workflows are performed. Sivasubramanian, 4:48-57 (explaining that “aspects of the various embodiments can be used with any appropriate services available or offered over a network in an electronic environment”), 3:10-25, 9:35-61, 15:45-50 (referring to “customers migrate their data stores to the cloud (or elsewhere) via the control plane”), 23:17-24.

163. One well-known alternative to a cloud-based system was a “hyperconverged infrastructure” (HCI) system. EX1036, [0003] (“Hyper Converged Infrastructure (HCI) combines computing applications and storage

applications into a single infrastructure.”); EX1037, 1:13-33; Weissman, [0005] (cited by examiner, EX1002, 256).

164. The ’336 patent does not use the term “hyperconverged,” as I noted above. However, one prior art description is as follows:

Converged infrastructures are collections of compute, storage, network, and software resources that may be acquired, serviced, and supported as single assemblies. For example, a corporate IT (Information Technology) manager can purchase a single converged infrastructure system rather than having to purchase, assemble, and deploy separate computing servers, storage arrays, networking switches, associated software, and so on.

Hyper-converged infrastructures, or “HCI’s,” are converged infrastructures that provide additional integration at the sub-assembly level. For example, a hyper-converged system may include servers that perform multiple roles, such as any combination of compute, storage, and networking.

Srinivasan (EX1038), 1:15-27.

165. In Sivasubramanian+Zha, POSAs would have found it obvious to implement Sivasubramanian’s control plane to provision databases located in a “data plane” implemented as a *hyperconverged infrastructure system*, for the reasons I explain below.

166. As I noted above, HCI systems were known. Furthermore, it was known to deploy databases in hyperconverged systems. Achillopoulos (EX1039), [0242] (discussing “hyperconverged database type management system”); Zhao (EX1040), [0089] (“....solution of the present disclosure can be implemented as integration of open-source software library and flash memory or further integrated with flash memory and application server as a hyper converged system... for instance, integrating application node of NoSQL database and a non-volatile storage device....”); Srinivasan 6:46-53, Fig. 2 (disclosing a “HCI (Hyperconverged Infrastructure) system” including “physical computing servers” running various “host application instances” including “backend database programs”).

167. POSAs would have been motivated to implement the data plane as an HCI system as an alternative to a cloud to achieve the known benefits of HCI, such as resource sharing, flexibility, and ease of management. *See, e.g.,* Gibson (EX1041), [0002] (“hyperconverged systems can provide the flexibility of a public cloud infrastructure, while allowing customers to maintain control of the hardware.”); Mellquist (EX1042), 1:5-10 (stating in “BACKGROUND” that “Hyper converged and converged infrastructure includes servers, storage, networking, and integrated software and services. These systems are designed to address the cost and complexity of data center operations and maintenance by

pulling the IT components together into a single resource pool so they are easier to manage and faster to deploy.”); Watkins (EX1043), 1:6-20 (“Computer systems and clusters of computer systems utilized by enterprises may employ hardware virtualization.... Common advantages of hardware virtualization include greater overall hardware-resource utilization, improved system scalability, and simplifying/centralizing administrative management of the system. Data centers are one type of system that can use virtualization to realize these advantages. Some data center systems virtualize storage, computing, and networking resources. Such systems are sometimes referred to as hyperconverged systems. In such systems, a cluster of nodes or servers may share resources among various VMs running on the nodes.”).

168. POSAs would reasonably have expected success in such an implementation because Sivasubramanian’s “control plane” is not limited to one type of data plane, as I discussed above, and because HCI systems and their use for databases were well-known, as I also discussed above.

169. In the resulting implementation of Sivasubramanian+Zha, since the data plane is a hyperconverged infrastructure system, the *user request* (see *supra* §V.D.1.c (Element [1A2])) is *for provisioning a source database in a hyperconverged infrastructure system*.

e. [1B]: “receiving, by the processor, selection of a database engine type”

i. GUI

170. Sivasubramanian’s Fig. 5 (annotated below) shows *selection* of a “**Database engine**” (element 506) for the “Provision” task. POSAs understood that the “database engine” is a database engine *type* such as MySQL, since the “MySQL” type is shown. EX1001, 14:14-25; Sivasubramanian, 2:44-47. POSAs also understood that Figure 5 is showing a *selection* of the engine type from a drop-down menu (as indicated by the down arrow). POSAs understood that such drop-down menus are used to present options to *select* from. For example, by the patent’s 2018 priority date, POSAs would have been familiar with sites such as travel reservation sites that let a user select an option using a drop-down menu. *See, e.g.*, EX1101 (internet archive capture of Expedia website dated **July 15, 2018** showing user can select “Preferred class” with drop-down menu). Exhibit 1101 is a screenshot created at my direction of a URL that I identified, <https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand>. Based on my familiarity with the Wayback Machine as detailed in paragraph 34 above, I understand the URL from which Exhibit 1101 was screenshotted is the Internet Archive’s record of the website at the URL <https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand> archived on July 15, 2018.

Web Browser

FILE EDIT VIEW FAVORITES TOOLS HELP

Address GO

Control interface page

Repository name

Task ☒ Provision ☐ Create ☐ Delete ☐ Modify

Database engine

Storage capacity Units

Partition scheme

Security scheme Version:

Access key Version:

FIG. 5

171. POSAs understood that the Figure 5 GUI would similarly display a field for selecting a “database engine” when the “Create” task is selected, as I discussed *supra* §V.A.1.a.i.

172. Additionally, POSAs understood the Figure 5 GUI would *receive [a] selection of a database engine type* via that field. Sivasubramanian refers to users entering inputs to the control plane, including via Figure 5. Sivasubramanian,

5:59-62 (referencing “[w]eb service frontend features such as... validating user input...”), 12:55-57 (Fig. 5 UI lets user “enter information”). Additionally, Sivasubramanian repeatedly discusses the control plane considering the particular database engine specified when performing a “workflow,” and also refers to “engine” as an option for the “action” in Figure 5; POSAs therefore understood that the control plane *receives* the engine type selection in order to be consider the engine. *See* Sivasubramanian, 6:39-41 (“The workflow component also can manage differences between workflow steps used for different database engines, such as MySQL”), 10:37-40 (“different tasks can be selected for the workflow based upon factors such as the type of action requested and the type of database engine being used”), 12:58-61 (“the interface page can include options to specify a repository for the action 502, select an action to be performed 504, and specify options for the action 506, such as a database engine to be use[d]”), 19:25-31 (backup approaches can “differ from various engines”).

ii. Command Line

173. Sivasubramanian’s “rds-create-database” command-line command, which provisions a newly-created database (*supra* §V.A.1.a.ii), accepts the parameter “engine,” which POSAs understood is a *database engine type*, since the value shown is “MySQL.” Sivasubramanian, 12:64-13:5. POSAs would have found it obvious to implement an “rds-provision-database” command such as the

one I discussed in Section V.A.1.a.ii above to also accept the “engine” parameter to match the “Provision” task in the GUI, and POSAs had the skill to do so (*supra* §V.A.2). POSAs further understood the “engine” parameter is a *selection* because the user must select an engine type to enter the corresponding value for the “engine” parameter. POSAs also understood that the control plane *receives* this selection because it receives the command-line commands, as I explained *supra* §V.D.1.c.ii for Element [1A2].

f. [1C]: “receiving, by the processor, selection of a Service Level Agreement (“SLA”) and a protection schedule”

i. GUI

174. The ’336 patent states that an “*SLA is an agreement...* that outlines, among other things, the protection scope of the database. The protection scope defines for how long data from the database being created or registered is retained.” EX1001, 19:12-23.

175. Sivasubramanian+Zha’s graphical database-provisioning UI includes a UI element enabling selection of an *SLA* that specifies snapshot and transaction-log retention periods, and a UI element that accepts a user-specified “schedule” as taught by Zha. Zha, 11:33-44; *Supra* §§V.C.1-V.C.2. As I explained *infra* §V.D.1.j (for Element[1G]), this “schedule” *comprises a first and second frequency*

as the claim recites. Thus, in Sivasubramanian+Zha, Sivasubramanian’s GUI *receives selection of a Service Level Agreement (SLA) and a protection schedule.*

ii. Command Line

176. As I discussed *supra* §§V.C.1-V.C.2, in Sivasubramanian+Zha, Sivasubramanian’s command-line database-provisioning UI commands includes parameters for specifying a *selection* of an SLA and a “schedule,” as taught by Zha. Thus, in Sivasubramanian+Zha, Sivasubramanian’s command-line database-provisioning UI commands *receives selection of a Service Level Agreement (SLA) and a protection schedule.*

g. [1D]: “provisioning, by the processor, the source database based upon the database engine type; and”

177. In Sivasubramanian+Zha’s GUI, the *database engine type* selection is received via the “Provision” and “Create” tasks in the Figure 5 GUI. *See my discussion of Element [1B] above. Both of these tasks provision the source database, as I explained for Element [1A2] above. Furthermore, in Sivasubramanian+Zha’s command-line interface, the database engine type selection is received as a parameter of the “rds-create-database” and “rds-provision-database” commands, as I explained for Element [1B]. These commands also provision the source database, as I explained for Element [1A2] above.*

178. POSAs understood that the database is *provisioned based upon* the selected engine type, because Sivasubramanian discusses considering the selected engine type (such as “MySQL”) when performing workflows or tasks. *See, e.g.*, Sivasubramanian, 6:39-50, 6:59-7:3, 12:58-62 (referring to the “Database engine” field labeled element 506 in Figure 5 as a “database engine *to be use[d]*” for the “action *to be performed*”), Fig. 5.

h. [1E]: “setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database”

179. As explained above, the processor in Sivasubramanian-Zha receives a selection of an *SLA* and *protection schedule* via either the GUI or command-line interface for provisioning the *source database*. *See* my discussion of Element [1C] above. The *protection schedule* defines the frequencies at which the *snapshots* and *transactional logs*, respectively, are *captured*. *See* my discussion of Element [1G] below. And the *SLA* defines the durations for which the snapshots and logs are retained, as I explained in Section V.C.1 and in my discussion of Element [1C].

180. In Sivasubramanian+Zha, snapshots and transactional logs are captured at user-defined frequencies and retained for user-defined retention periods, as taught in Zha. *Supra* §V.C. Because the retention periods and frequencies are defined in the *SLA* and *protection schedule*, respectively, as I

discussed above, POSAs understood that the *capturing of snapshots and transactional logs is based upon the SLA and protection schedule*.

181. Furthermore, POSAs understood that the capturing is *automatic*. Zha teaches that the capturing of the snapshots and transaction logs each occur at a user-defined “frequency.” Zha, 11:33-44. POSAs would have understood that defining a “frequency” at which an event occurs causes that event to occur *automatically, i.e.*, without user involvement. Oxford (EX1105), 109-110 (“automatic” means “given or imposed as a necessary and inevitable result of a fixed rule or particular set of circumstances”).

182. At minimum, POSAs would have found automatic capturing of snapshots and transactional logs obvious. Sivasubramanian discloses that the control plane can “*automatically* perform actions in the data plane.” Sivasubramanian, 2:3-5.

183. Automatic capturing of information from a database was well-known. *E.g.*, RDS, 520, 647 (“automated DB snapshot”); Simek, Abstract (disclosing “automated creation... of database snapshots”). Furthermore, based on my familiarity with a POSA’s level of skill (including education and experience) as discussed in Section IV.D above and my own knowledge and experience, it is my opinion that POSAs understood that automation beneficially reduced the need for

user intervention, and would have been motivated to implement automatic snapshot capture in Sivasubramanian to achieve this benefit.

184. Additionally, POSAs understood that the purpose of the aforementioned *automatic capturing is for protecting the source database*. The '336 patent says “[p]rotecting a database means replicating a state of the database for creating a fully functional copy of the database”; this “[r]eplicating... may involve **creating** fully functional clones (e.g., **back-ups**),” which may in turn “be created by periodically capturing snapshots of the source database.” EX1001, 4:46-58. POSAs understood that in Sivasubramanian+Zha, the snapshot and log data is captured and retained for the purpose of “replicating a state of the database for creating a fully functional copy of the database” (EX1001, 4:46-58)—*i.e.*, for the purpose of *protecting the source database*, as explained in greater detail below. *See, e.g.*, Sivasubramanian, 15:1-15 (discussing “clon[ing]” database with “snapshots”), 19:20-20:1 (discussing “approach for backing up data” in database using snapshots and logs); Zha, 6:49-58, 17:63-66, & 25:24-34 (discussing process of creating virtual database with snapshots and logs to replicate source database); *see also* Sections V.B and V.C above.

185. **First**, Sivasubramanian’s own “approach for backing up data” (19:32), used in the first Sivasubramanian+Zha variant, performs “backup” on a “database” by capturing database “snapshots.” Sivasubramanian, 19:20-64.

Moreover, Sivasubramanian discusses “tak[ing] a snapshot of a production instance” so that a user can “clone” a database and ensure it is “fully populated” with “production [data].” Sivasubramanian, 15:1-15. Sivasubramanian’s “approach” also backs up “log files,” *i.e.* transaction logs (*supra* §V.A.1.b), that can “replay[] various transactions in case the data files have to be restored.” Sivasubramanian, 19:65-20:1. POSAs understood the “data files” that are “restored” are “data files” for a database, since Sivasubramanian earlier says “backing up both the data files and any associated log files” is done to backup a “repository,” *i.e.*, a “database.” Sivasubramanian, 19:27-31. Thus, POSAs would have understood that in the first Sivasubramanian+Zha variant, both the snapshots and logs would be retained *to protect the database* by replicating its state, per Sivasubramanian’s exemplary “approach for backup up data” (19:32).

186. ***Second***, Zha’s “virtual database” technique, used in the second Sivasubramanian+Zha variant (*supra* §V.C), “create[s] a virtual database [VDB],” by “creat[ing] files that represent the ***information corresponding to [a] production database system... at a given point in time.***” Zha, 6:49-58. The VDB can be used for Zha’s “backup” process, which “can be used to reconstruct information of the original storage system in case the original data is lost.” Zha, 25:24-34. Zha reconstructs the database by applying transaction logs to VDBs created using snapshots, as discussed *supra* §V.B (citing Zha, 17:63-66). POSAs therefore

understood that Zha’s technique “replicat[es] a state of the database for creating a fully functional copy of the database” and thus “*protects*” the database, as the ’336 patent describes, using the snapshots and transaction logs retained for the user-defined retention period. EX1001, 4:55-67.

187. Finally, POSAs understood that the *setting up* of the automatic capturing is done *by the processor*. The *processor* executes the instructions that implement Sivasubramanian’s control plane, as I discussed above for Element [1A1]. And Sivasubramanian explains that the “workflow component” in the “control plane” also controls the “recovery and backup” “workflow,” and also explains that snapshots and transactional logs captured as part of the process of “backing up data.” Sivasubramanian, 6:31-42, 19:32-64.

i. [1F]: “wherein the transactional logs record transactions occurring on the source database since a most recent snapshot”

188. In both Sivasubramanian+Zha variants that I discussed in Section V.C, the transaction logs log transactions occurring *on the source database* because the capturing of logs (and snapshots) is setup on *the source database* to protect the source database when the source database is provisioned using the Sivasubramanian’s database-provisioning UIs. *See, e.g.*, Section V.C above (discussing Sivasubramanian+Zha combination) and Section V.D.1.h for element [1E]. Sivasubramanian’s “logs,” which are used to “replay” “transactions,” are

transactional logs as claimed. Sivasubramanian, 19:65-20:1; *supra* Element [1E], §V.C. POSAs understood that the logs are useful for “replay[ing]” transactions because they *record* transactions so that transactions can be reproduced later, which was a well-known function of transactional/transaction logs. McGraw-Hill (EX1102), 1762-1763 (verb “record” means “[t]o preserve for later reproduction or reference”); Priebe (EX1060), 1:19-1:54 (referring to logs “record[ing]” transactions”); Heidel (EX1103), [0002] (“Database systems are susceptible to failure... As a result, transaction logs that record various actions performed by such database systems have been adopted.”); Auch (EX1104), [0001] (“Every transaction (where ‘transaction’ refers to a set of operations) against the data stored in [the] database causes a log record to be written into a transaction log.”). Moreover, POSAs understood, or at least would have found obvious, that the logs record transactions *since a most recent snapshot*, for multiple reasons. First, Sivasubramanian refers generically to the logs replaying “transactions,” without mention any temporal limit, indicating the transactions would continue to be logged for replaying even after a snapshot is taken. Sivasubramanian, 19:65-20:1. Furthermore, in Sivasubramanian’s exemplary approach for backing up data, “both the data files and any associated log files” are backed up, indicating to a POSA that the logs contain data for transactions after the snapshots (otherwise the logs would

be superfluous to Sivasubramanian’s backup operation). Sivasubramanian, 19:20-64.

189. In addition, in Sivasubramanian+Zha, Zha also teaches transactional logs *recording transactions since a most recent snapshot*. As discussed *supra* §V.B, Zha uses the term “point-in-time [PIT] copy” to refer to a “snapshot.” Zha, 17:63-66. Zha discloses that transaction logs “can be used to reconstruct a copy of a database... corresponding to times in the past in between the times” at which [PIT] copies, *i.e.* snapshots, of the database were taken. Zha, 10:6-11. POSAs understood that in order to create a copy of the database corresponding to a time “between” “PIT copies,” *i.e.* snapshots (as explained *supra* §V.B), the transaction logs must continue to *record* transactions after a snapshot occurs; therefore, the transaction logs record transactions *since a most recent snapshot*.

- j. [1G]: “and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.”

190. In Sivasubramanian+Zha, Sivasubramanian’s database-provisioning UIs accept a user-specified “*schedule*” as taught by Zha. *Supra* [1C], §V.C.2; Zha, 11:33-44. This “schedule” includes (1) a “*frequency*” for “retriev[ing] PIT copies,” *i.e.*, taking snapshots, (2) and a “*frequency*” for “retriev[ing] updates to online transaction logs.” Zha, 11:33-44.

191. POSAs would have understood that taking/retrieving snapshots and transactional logs involved *capturing* the snapshots. EX1001, 5:4-8 (“because *capturing* a snapshot requires quiescing... the source database... it is desirable to *take* only a minimum number of snapshots”); Heritage (EX1009), 277 (“capture” means “take captive” or “preserve in lasting form”). Therefore, Zha’s “frequency” for “retriev[ing] PIT copies” (Zha, 11:33-44) is a *first frequency at which the snapshots are captured*.

192. Likewise, POSAs understood that “retriev[ing] updates” to a log *captures* the log as it exists at retrieval time; thus, the “frequency” for “retriev[ing] updates” is a *frequency at which transactional logs are captured*. EX1001, 36:49-59 (’336 patent using “transaction log” and “*transactional log*” interchangeably); Zha, 15:21-24 (describing log update process); Heritage, 277 (“capture” means “take captive” or “preserve in lasting form”). Therefore, Zha’s “frequency” for “retriev[ing] updates” (Zha, 11:33-44) is a *second frequency for capturing one or more transaction logs, i.e., transactional logs*.

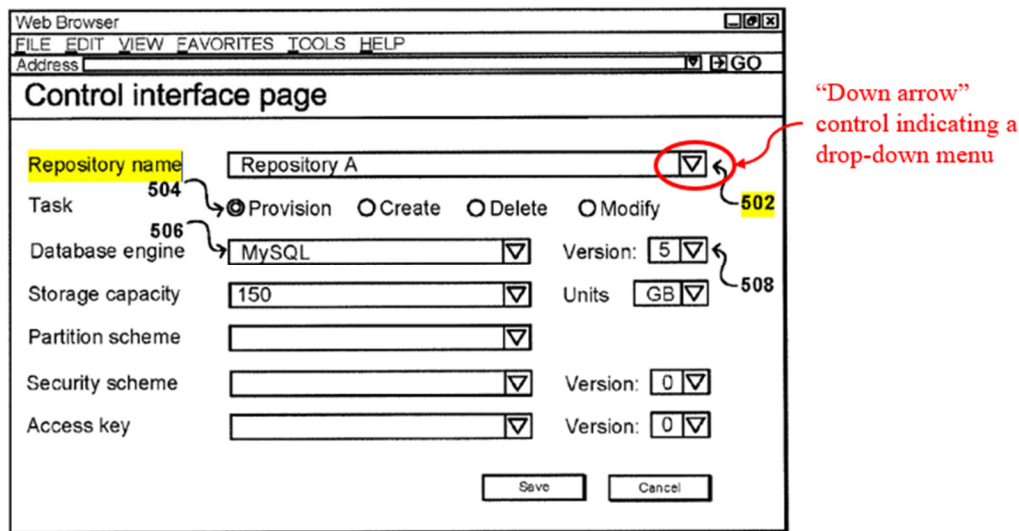
193. Therefore, the user-specified “schedule” is a *protection schedule comprising a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured*. POSAs understood that the snapshots and transactional logs are captured *for protecting the source database* for the same reasons I discussed for Element [1E].

2. Claim 5: “The method of claim 1, wherein provisioning the source database comprises creating a new database as the source database.”

194. Sivasubramanian *provisions the source database* as part of the database-creation task launched from either Figure 5’s “Create” GUI or the “rds-create-database” command. *See* my discussion of Element [1D]. In either scenario, *the provisioning comprises creating a new database as the source database.*

3. Claim 6: “The method of claim 1, wherein provisioning the source database comprises registering an existing database as the source database.”

195. Sivasubramanian’s “Provision” task in Figure 5 lets the user *provision* a *source database* that is an *existing database* identified using the “Repository name” drop-down menu. Sivasubramanian, Fig. 5; *see* my discussions above in Section V.A.1.a.i and for Element [1A2]. POSAs would likewise have found it obvious, and would reasonably expect success, to implement an “rds-provision-database” task (*supra* §V.A.1.a.ii) to *provision* an *existing* database identified with the “name” command-line parameter. Sivasubramanian, 13:1-5; *supra* §V.A.2.



Sivasubramanian, Detail of Fig. 5 (annotated)

196. In Sivasubramanian+Zha, Sivasubramanian’s database-provisioning UIs accept an SLA selection and protection schedule based on Zha’s teachings. *Supra* §§V.C, [1C]. POSAs would have understood that setting a protection schedule and SLA for an existing database *registers the existing database as the source database*, because this action sets up the existing database as the source database for capturing snapshots. *Supra* [1C], [1E]; EX1001, 4:29-34, 25:43-67, FIGS. 5A-5E (describing “registering” an existing database by providing connection information for the database and then specifying an SLA and protection schedule for it).

197. The ’336 patent says “existing databases” that are registered “*may* have been created outside of the database system,” indicating this is not *required*. If claim 6 is found to require it, however, POSAs would have found it obvious to

implement Sivasubramanian’s database-provisioning UIs to allow a control-plane user to *register* a previously-unknown *existing database*. POSAs understood this would beneficially enable performing Sivasubramanian’s workflows, including the “recovery and backup” workflow, on existing databases, replacing “tedious manual procedures” for those databases. Sivasubramanian, 1:46-52, 6:31-42. POSAs would have reasonably expected success with such a UI implementation. Sivasubramanian discloses that (1) the control plane accepts input of parameters used for database access, such as DNS address and port, and (2) the control plane can “provision and manage” *pre-existing* databases in a data plane controlled by a “provider” separate from the control plane’s “provider.” Sivasubramanian, 6:59-67, 9:37-44, 11:40-52. Implementing Sivasubramanian’s UIs to accept these parameters was within a POSA’s skill. *Supra* §V.A.2.

4. Claim 7: “The method of claim 1, wherein the first frequency and the second frequency are the same.”

198. In Sivasubramanian+Zha, the user defines both (1) a “*frequency*” for taking snapshots and (2) a “*frequency*” for backing up transaction/transactional logs. Zha, 11:33-44; *supra* §V.C. These frequencies satisfy the claimed *first* and *second* frequency, respectively. *Supra* [1G]. POSAs understood that these frequencies *are the same* if the user chooses the same values for them. POSAs would have found it obvious for a user to set the same frequency because this was

one of only two options, *i.e.*, same or not, and because setting the same frequency beneficially reduced the need for a user to pick two different frequencies to select.

5. Claim 8

- a. [8PRE]-[8A]: “The method of claim 1, further comprising: presenting, by the processor, a group of database engine types that are supported by the hyperconverged infrastructure system; and”

199. Sivasubramanian’s Figure 5 (annotated below) shows the “***Database engine***” type (element 506) being selected from a drop-down menu. *Supra* [1B]. POSAs understood that such drop-down menus are used to *present a group* of options to select from. For example, by the patent’s 2018 priority date, POSAs would have been familiar with sites such as travel reservation sites that let a user pick an option using a drop-down menu. *See, e.g.*, EX1101 (internet archive capture of Expedia website dated **July 15, 2018** showing user can select “Preferred class” with drop-down menu). Exhibit 1101 is a screenshot created at my direction of a URL that I identified,

<https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand>. Based on my familiarity with the Wayback Machine as detailed in paragraph 34 above, I understand the URL from which Exhibit 1101 was screenshotted is the Internet Archive’s record of the website at the URL <https://web.archive.org/web/20180715133951%20https://www.expedia.com/#expand> archived on July 15, 2018.

200. Sivasubramanian’s control plane “*support[s]* several different database engines,” *i.e.*, database engine types. Sivasubramanian, 9:61-64; *supra* [1B]. In Sivasubramanian+Zha, POSAs would have found it obvious that the database engine types the control plane “support[s]” would be engine types supported by the *hyperconverged infrastructure system*, since the control plane provisions databases to run on the hyperconverged infrastructure system. *Supra* §V.D.1.d (Element [1A3]). POSAs would have found it obvious to *present* this *group* for selection in the Figure 5 GUI because presenting the supported group beneficially allows the option of provisioning of databases running on any of the supported engine types. Because the *processor* executes the control plane instructions (*supra* §V.D.1.b (Element [1A1])), the *presenting* is done *by the processor*.

b. [8B]: “receiving, by the processor, the selection of the database engine type from the group of database engine types.”

201. Figure 5 shows *receiving, by the processor, the selection of the “MySQL” database engine type* from the “Database engine” drop down menu. *Supra* [1B]. This selection is *from the group of database engine types* in the menu. *Supra* [8PRE]-[8A].

6. **Claim 10: “The method of claim 1, further comprising: presenting, by the processor, a plurality of levels of the SLA; and receiving, by the processor, selection of one of the plurality of levels of the SLA.”**

202. In Sivasubramanian+Zha, SLAs are differentiated from each other by *levels* (*supra* §V.C.1) and the *processor receives the selection* of an SLA via Sivasubramanian’s Figure 5 GUI. *Supra* [1C]. POSAs would have found it obvious to implement the control plane to *present* the various levels, *i.e., a plurality of levels*, to the user via a UI element for selection, *e.g., a drop-down menu* as disclosed in Sivasubramanian’s Figure 5 (*e.g. element 506*), and then *receive* the selection of the *level*, since this would beneficially result in SLA selection for the database. Receiving such selection was within a POSA’s skill. *Supra* §V.A.2. Because the *processor* executes the control plane instructions the *presenting* and *receiving* are done *by the processor*. *Supra* [1A1].

7. **Claim 11: “The method of claim 1, wherein the protection schedule comprises a time period for designating at least one of a daily snapshot, a weekly snapshot, a monthly snapshot, and a quarterly snapshot.”**

203. The ’336 patent uses “*time[-]period*” in the context of snapshots to refer to a time of day at which a snapshot is captured. EX1001, 5:64-67 (“The protection schedule may allow the user to define ... the *time-period* for capturing daily, weekly, monthly, and/or quarterly snapshots”), 20:18-24 (referring to

“another *time period*” as an alternative to “the protection schedule defin[ing] that the snapshot captured at 11:00 AM is the daily snapshot”).

204. The patent describes the “daily” snapshot as the snapshot taken at a “designated” daily time; this snapshot is stored for a longer duration than other snapshots taken that day (e.g., 90 days v. 30). EX1001, 19:61-67, 21:4-12 (“GOLD” SLA example). “Weekly,” “monthly,” and “quarterly” snapshots can be similarly designated. EX1001, 20:1-48.

205. In Sivasubramanian+Zha, Sivasubramanian’s UI accepts a “schedule” specifying “times during the day” for the “point in time” copies, i.e., *snapshots*, as Zha teaches. Zha, 11:33-44; *supra* §V.C. This “schedule” is *the protection schedule* recited in the claim. *See* Element [1C] discussion above. Thus, the *protection schedule comprises a time period for designating at least one of a daily snapshot*. This satisfies claim 11 if due to the claim language “at least one of... and,” the claim can be met by *designating* any listed item. I note that the ’336 patent’s specification says “in those instances where a convention analogous to ‘at least one of A, B, and C, etc.’ is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (e.g., ‘a system having at least one of A, B, and C’ would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.” EX1001, 39:31-40.

206. If the claim requires that **one** snapshot collected at **one** designated daily time be treated as “**the**” daily snapshot, this would have been obvious. Designating specific snapshots as daily (or weekly/monthly/quarterly) snapshots was well-known. Green (EX1047), [0163]-[0165] (describing technique for “layering of the snapshots” including “daily,” “weekly,” “monthly,” “quarterly,” and “yearly” snapshots); Madany (EX1023), [0023]; Cisler (EX1048), [0060]; Taylor (EX1049), [0164]; Deshmukh (EX1050), [0053] (“Snapshots may then be retained such that a set of hourly backups exists for the most recent day, a set of daily snapshots are retained for one week, a set of weekly snapshots are retained for a month, and a set of monthly snapshots may be retained as long as specified by the user.”); Bivol (EX1051), [0025] (“In some implementations, these specific time intervals may be daily, weekly, monthly, quarterly, or yearly snapshots.”).

207. POSAs would have found it obvious to implement Sivasubramanian+Zha’s schedule UI (*supra* §V.C.2) to additionally accept input of the daily snapshot time. This was a known UI option that would beneficially let a user choose the best time for a daily snapshot based on their own circumstances; for example, the user could choose a time at which database usage is expected to be low, or choose a time at which all activity in the database for the day is expected to have been completed such that the daily snapshot has the most complete picture of that day’s activity. RDS, 16-17, 187 (daily “backup window”

during which “snapshot” is taken); Gaurav, 11:14-20 (“Element 123a is selected and indicates that snapshots are taken on a daily and/or weekly basis. In particular elements 124 and 125 collectively indicate that snapshots are taken each day at 12:00 AM. Element 126 denotes a retention policy in that snapshots created are retained for 7 days and then deleted. Element 126 notes additional snapshot criteria that affects when snapshots are deleted.”), Fig. 2. POSAs would have expected success because the UI implementation was within a POSA’s skill. *Supra* §V.A.2.

208. To the extent the claim requires *all* the recited items, this would have been obvious; it was known to specify time different periods for capturing snapshots for different snapshot frequencies, and this would have provided the same benefits discussed in the previous paragraph for daily snapshot time designation and would have been within a POSA’s skill. *See, e.g.,* Green, [0165]; Gaurav, 10:20-22, 11:14-20, Fig. 2; *supra* §V.A.2.

8. Claim 12

a. [12PRE]: “A database management system comprising:”

209. Sivasubramanian describes “*/s/*ystems and methods” that “provide a relational *database* service (‘RDS’)” that lets users “*configure relational databases,*” a task Sivasubramanian refers to as “*database management.*”

Sivasubramanian, 4:48-53. This *database management system*, as described in Sivasubramanian+Zha, includes the remaining components required in claim 12.

b. Limitations [12A1]-[12G]⁴

210. The remaining limitations of claim 12 are obvious over Sivasubramanian+Zha for at least the reasons discussed for the prior elements indicated below:

Element	Prior Element
[12A1]	[1A1]
[12A2]	[1A1]
[12A3]	[1A2]-[1A3]
[12B]	[1B]
[12C]	[1C]
[12D]	[1D]
[12E]	[1E]
[12F]	[1F]
[12G]	[1G]

⁴ The claim limitation reference numbers are defined in the Claim Listing in §VII.

9. **Claim 13: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive the user request, the selection of the database engine type, and the SLA via a dashboard.”**

211. Sivasubramanian+Zha receives the user request, the selection of the database engine type, and the SLA via the Figure 5 GUI. Supra [1A]-[1C]:

The screenshot shows a web browser window titled "Web Browser" with a menu bar (FILE, EDIT, VIEW, FAVORITES, TOOLS, HELP) and an address bar. The main content area is titled "Control interface page". It contains the following form elements:

- Repository name:** A text input field containing "Repository A".
- Task:** A group of radio buttons with labels "Provision", "Create", "Delete", and "Modify". The "Provision" button is selected.
- Database engine:** A dropdown menu showing "MySQL".
- Version:** A dropdown menu showing "5".
- Storage capacity:** A text input field containing "150".
- Units:** A dropdown menu showing "GB".
- Partition scheme:** A dropdown menu.
- Security scheme:** A dropdown menu.
- Version:** A dropdown menu showing "0".
- Access key:** A dropdown menu.
- Version:** A dropdown menu showing "0".

At the bottom right, there are "Save" and "Cancel" buttons. Reference numerals are placed as follows: 504 points to the "Provision" radio button, 506 points to the "MySQL" dropdown, 502 points to the "Repository A" text field, and 508 points to the "5" version dropdown.

Sivasubramanian, Detail of Fig. 5

212. POSAs would have understood that a user interface that enables a user to provision and manage databases, such as Sivasubramanian’s, was commonly referred to as a “*dashboard*.” *see, e.g.*, Nickolov, [0123]-[0144], [0502], [0504]-[0507]; Karinta, [0297]-[0307], Fig. 8A. Accordingly, Sivasubramanian+Zha meets claim 13.

10. Claim 14: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to create a cloned database from the source database based on the snapshots and the transactional logs.”

213. Sivasubramanian+Zha teaches capturing snapshots and transactional logs of the source database according to the protection schedule. *Supra* [1C], [1E], §V.C. Sivasubramanian also describes using snapshots and transactional logs from a “production instance” (*i.e.*, a *source database*) to create a “test instance” of the database that “is fully populated” with the data from the production instance. Sivasubramanian, 15:1-3. Sivasubramanian refers to this process as creating a “**clone**” of the production instance of the database (*i.e.*, a *cloned database from the source database based on the snapshots and the transactional logs*). Sivasubramanian, 15:7-9.

214. Additionally, Zha describes using “**database blocks** retrieved by a point in time copy [*i.e.*, snapshot] manager” “combined with the **transaction logs** retrieved by the transaction log manager... to reconstruct” (*i.e.*, *create*) “**a copy of a database**” (*i.e.*, a *cloned database from the source database based on the snapshot and the transactional logs*). Zha, 10:6-12.

11. Claim 15: “The database management system of claim 12, wherein the first frequency and the second frequency are different.”

215. In Sivasubramanian+Zha, the user defines both (1) a “*frequency*” for taking snapshots and (2) a “*frequency*” for backing up transaction/transactional logs. Zha, 11:33-44; *supra* §V.C. These frequencies satisfy the claimed *first* and *second* frequency, respectively. *Supra* [1G]. Since the user sets these frequencies, POSAs understood that these frequencies *are different* if the user chooses different frequencies. POSAs would have found it obvious for a user to set different frequencies because this was one of only two options, *i.e.*, same or not. Furthermore, it was known for transactional log backup frequencies to be higher than snapshot frequencies (*e.g.*, Tekade, [0041], [0045]-[0046], Fig. 6), and POSAs understood this arrangement beneficially allowed the database’s state to be restored to multiple points in time “between the times at which point-in-time copies [*i.e.*, snapshots, *see supra* §V.B] are made,” as Zha discloses. Zha, 10:6-12.

12. Claim 19

- a. [19PRE]: “A non-transitory computer readable media with computer-executable instructions embodied thereon that, when executed by a processor in a hyperconverged infrastructure system, cause the processor to perform a process comprising:”**

216. Sivasubramanian states that its invention can be implemented with a “computing device” including “computer readable media” storing “information such as computer readable instructions,” where the “media” includes “ROM” and

“disk.” Sivasubramanian, 22:33-37, 23:39-48, 23:61-24:8; *see also* 24:20-24, 25:4-8, 26:23-26. Thus, Sivasubramanian discloses implementing its “control plane” (described above) via a *non-transitory computer-readable media with computer-executable instructions embodied thereon*.

217. In Sivasubramanian, the aforementioned instructions are executed by a *processor* in a *hyperconverged infrastructure system* for the reasons I explained for Elements [1A1] and [1A3] above.

a. Limitations [19A]-[19G]

218. The remaining limitations of claim 19 are obvious over Sivasubramanian+Zha for at least the reasons discussed for the prior elements indicated in the table below:

Element	Prior Element
[19A]	[1A1]-[1A3]
[19B]	[1B]
[19C]	[1C]
[19D]	[1D]
[19E]	[1E]
[19F]	[1F]
[19G]	[1G]

13. Claim 20: “The non-transitory computer readable media of claim 19, wherein provisioning the source database comprises creating a new source database or registering an existing database as the source database.”

219. *See* my discussion of claims 5 and 6.

14. Claim 21: “The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for: presenting a group of database engine types that are supported by the hyperconverged infrastructure system; and receiving the selection of the database engine type from the group of database engine types.”

220. *See* my discussion of claim 8.

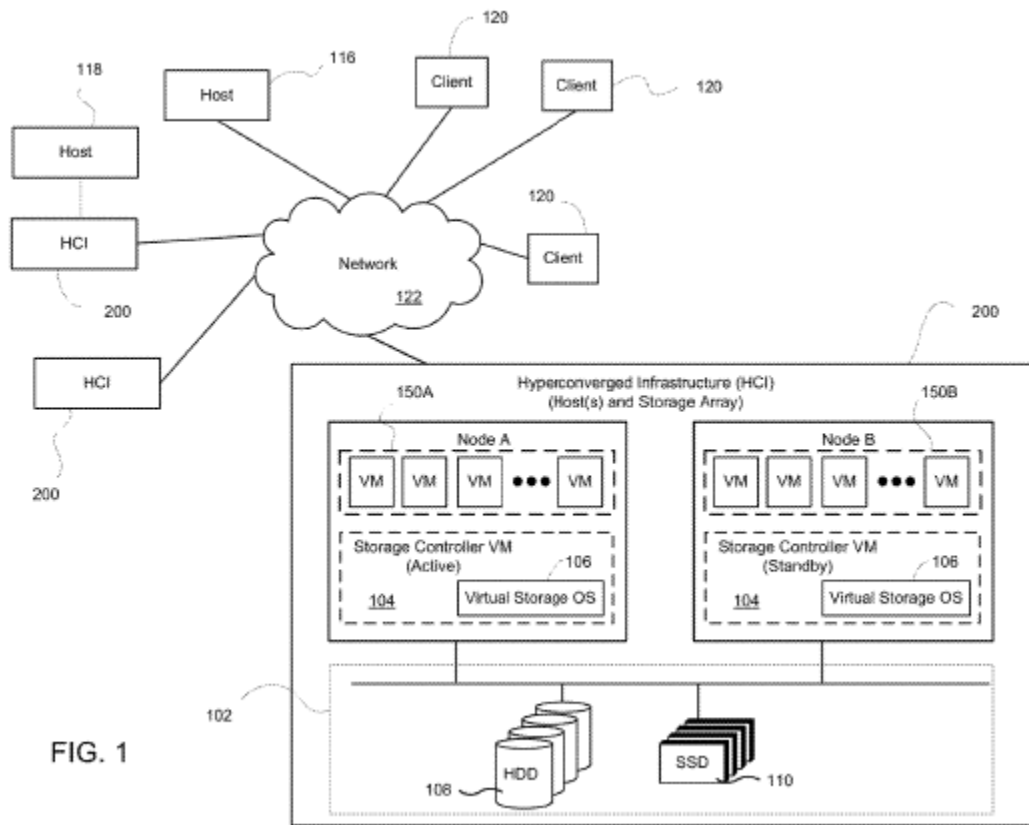
VI. GROUND 2: SIVASUBRAMANIAN, SHEKAR, AND ZHA RENDER OBVIOUS CLAIMS 1-23

221. I have been asked to provide my opinion concerning whether claims 1, 12, and 19 of the '336 patent would have been obvious to a POSA in light of Sivasubramanian, Zha, and Shekar, the prior-art references identified in the Petition. For the reasons explained below, it is my opinion that each of these claims would have been obvious to a POSA.

A. Shekar (EX1007)

222. Shekar discloses a “[h]yperconverged infrastructure” (“HCI”) providing “storage” and “compute” capabilities. Shekar, Title (“Hyperconverged infrastructure supporting storage and compute capabilities”), [0014] (“FIG. 2A illustrates a hyperconverged infrastructure providing storage and computing resources...”). Shekar’s goal is to provide a “storage device” with “high

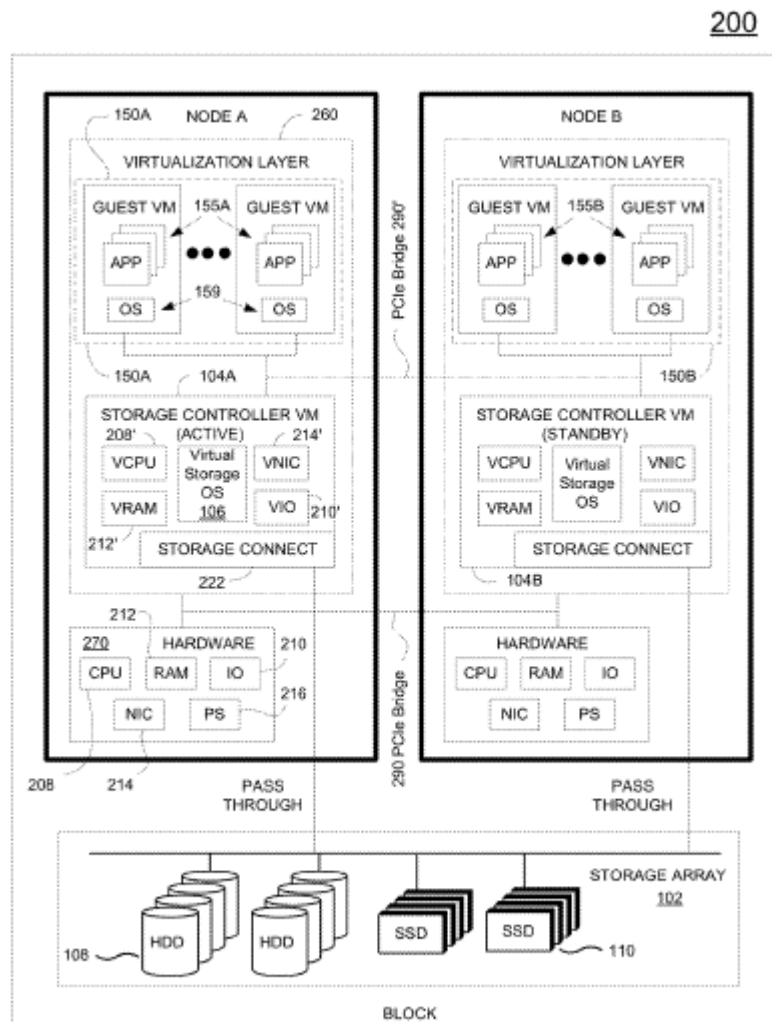
performance.” Shekar, [0005] (“What is needed is a storage device capable of processing IOs with high performance.”). Shekar discloses a “data storage system” that contains “one or more hyperconverged infrastructures (HCI)... [that] provide both compute (e.g., host applications) and storage.” Shekar, [0032], Fig. 1.



Shekar, Fig. 1

223. Shekar’s HCI includes two “nodes” that include a “virtual storage connect module” connecting the node to a “storage array.” Shekar, [0037] (“FIG. 2A illustrates a hyperconverged infrastructure (HCI) 200 including two nodes (e.g., node A and node B)...”), [0043] (“a *virtual storage connect module 222* is

configured for sending and receiving data to and from the HDDs 108 and SSDs 110. In one embodiment, the virtual storage connect module is able to directly access the physical storage array 102.”), Fig. 2A (showing element 222 inside both Node A and Node B).



Shekar, Fig. 2A

224. The storage array “provide[s] storage services to one or more host applications executing on the corresponding HCI 200.” Shekhar, [0035].

225. The applications run on “guest virtual machines [VMs]” executing on the hardware of each node. Shekar, [0008] (“The system includes a first node including first hardware and a first virtualization layer. The first hardware includes a first central processing unit (CPU). The first virtualization layer supports a first plurality of guest virtual machines utilizing the first hardware while running a plurality of first applications.”), [0038] (“...node A includes hardware 270 and a virtualization layer 260 that creates and manages the operating systems 159 of one or more guest VMs 150A, wherein the guest VMs 150A are each executing one or more applications 155A. Similarly, node B includes hardware 270 and a virtualization layer 260 that creates and manages the operating systems 159 of one or more guest VMs 150B, wherein the guest VMs 150B are each executing one or more applications 155B.”), Fig. 2A.

226. Each node also includes a “storage VM” (or “storage controller VM”). Shekar, [0033], Fig. 2A. The storage VMs on the two nodes “cooperatively manage access to the storage array” by using an “active/standby” scheme where the standby controller can take over access to the array if the active controller fails. Shekar, [0033] (“The storage VMs operating on node A and node B cooperatively manage access to the storage array 102...”), [0040] (“the virtual storage controllers

104A and 104B operating on node A and node B cooperatively manage access to the storage array 102, such as implementing an active/standby configuration”), [0041]-[0042] (describing operation of active/standby configuration).

B. Sivasubramanian+Shekar+Zha Combination

227. Sivasubramanian’s control plane is flexible regarding data-plane type. *Supra* Ground 1, [1A3]. In view of Shekar, POSAs would have been motivated to provide the option of a “data plane” implemented using one or more of Shekar’s “HCIs,” for multiple reasons, as I explain below.

228. First, as I explained for Element [1A3] in Ground 1, POSAs understood that hyperconverged infrastructure was a known option for implementing a “data plane” with known benefits.

229. Second, POSAs would have been specifically motivated to use Shekar’s “HCI[s]” (which I discussed above in §VI.A) to beneficially achieve the “high performance” of Shekar’s HCIs in a “data plane.” Shekar, [0005].

230. Third, as I noted above in §VI.A, Shekar discloses that each HCI includes a “storage array” that provides “storage services” to “host applications” running in the HCI. Shekar, [0035]. POSA understood that “database[s]” such as those provisioned by Sivasubramanian (Sivasubramanian, 6:59-67; *supra* §V.A.1) are applications that use storage services, and thus that would benefit from incorporation of Shekar’s HCI. Sivasubramanian, 6:43-58 (stating that “a

workflow can be instantiated using a workflow template for creating a database” and referring to example “request... for a MySQL® Relational Database Management System (RDBMS) instance, as opposed to an Oracle® RDBMS or other such instance,” and stating that the workflow can “select specific tasks related to the amount of storage requested, any specific hardware requirements, or other such tasks”).

231. The resulting Sivasubramanian+Shekar system includes a “control plane” as taught by Sivasubramanian (*supra* §V.A.1) that executes workflows in a data plane implemented using one or more of Shekar’s HCIs (*supra* §VI.A). In this system, databases, which are “applications” as noted above, run on “virtual machines” (“VMs”) running on HCIs, as taught in Shekar. Shekar, [0008]-[0010], [0032], [0035], [0038], [0040]; *supra* §VI.A. Additionally, as detailed *infra* claims 2 and 3, in the resulting system, Sivasubramanian’s graphical and command-line database-provisioning UIs are implemented to accept additional parameters for configuring the VMs.

232. Furthermore, POSAs would have found it obvious to incorporate Zha’s teachings regarding user-defined snapshot and transaction-log collection frequencies, times during the day, and/or retention periods, and also incorporate UI elements to gather such information, into the Sivasubramanian+Shekar system for the same reasons as the Sivasubramanian+Zha combination. *Supra* §V.C.

233. The resulting Sivasubramanian+Shekar+Zha combination is identical to the Sivasubramanian+Zha combination of Ground 1 plus two aspects: (1) the data plane is implemented using Shekar's HCIs, with databases running on VMs on the HCIs; and (2) additional UI elements are added for configuring VMs, as discussed above and detailed in Claims 2-3 below.

234. POSAs would reasonably have expected success in the above-described implementation for multiple reasons, as I explain below.

235. Using hyperconverged systems for databases was known in the art. *Supra* §V.D.1.d (Element [1D]). Running databases on virtual machines ("VMs") was also well-known in the art. Uriel, [0047]; Winterfeldt (EX1067), [0021]; Jin-844, [0007]; Jin-297 (EX1069), 3:16-19.

236. Additionally, POSAs understood that a VM emulates a physical computer system; thus, POSAs would have reasonably expected that running databases on VMs on Shekar's HCIs would not change the functions of the databases in any aspect relevant to Sivasubramanian's control plane. My testimony regarding a POSA's background knowledge is corroborated by, *e.g.*, Bhandari (EX1070), 1:10-17 (filed in 2017 and explaining in "BACKGROUND" that "A virtual machine (VM) is an emulation of a computer system... Virtual machines resemble physical computer architectures and provide functionality of a physical computer.").

237. Moreover, POSAs understood that Shekar’s “HCI” includes components that are functionally equivalent to components that Sivasubramanian describes as being used to run databases in the data plane. Shekar’s HCI includes a “storage array” providing “storage services” to “host applications” running in the HCI. Shekar, [0035]; *supra* §VI.A. Similarly, Sivasubramanian discloses that a “data instance” for a database (*see supra* §V.A.1), is “allocate[ed]” and “attach[ed]... to” “off-instance persistent storage.” Sivasubramanian, 6:59-65. As another example, each of Shekar’s “virtual host systems” (also called “VMs”) on which “applications” run includes a “VCPU” (virtual CPU) and “virtual memory (RAM).” Shekar, [0046]. POSAs understood that “VCPU” stands for “virtual CPU,” since a “V” or “v” at the beginning of an abbreviation was a common way to indicate “virtual.” *See, e.g.*, Shekar, [0030] (referring to “virtual machines (VMs)”). Similarly, Sivasubramanian discloses that “instances” for databases may have an “instance type” that “describes the CPU and memory capacity.” Sivasubramanian, 11:49-52.

238. Furthermore, Shekar explains that the nodes in an HCI include “one or more network interface cards (NICs)... for exchanging data packages through a network.” Shekar, [0039]. Thus, POSAs understood that Sivasubramanian’s control plane would be able to communicate with the databases running on VMs in the HCIs to execute workflows.

239. Finally, implementing Sivasubramanian’s UI to accept additional parameters was within a POSA’s skill. *Supra* §V.A.2.

C. Claim-by-Claim Analysis

1. Claims 1, 5-8, 10-15, 19-21

240. As explained *supra* Ground 1, [1A3], Sivasubramanian discloses that databases are provisioned in the “data plane.” Sivasubramanian, 7:3-9, 2:40-44, 5:15-20, 12:55-66, 13:28-29. As explained *supra* §VI.B, Sivasubramanian+Shekar+Zha is identical to Sivasubramanian+Zha except that (1) the “data plane” may be implemented using Shekar’s HCIs, with the databases running on VMs running on the HCIs; and (2) the Sivasubramanian+Shekar+Zha UIs accept additional parameters.

241. Shekar describes its HCI as an “integrated *system*.” Shekar, [0037]. Thus, in Sivasubramanian+Shekar+Zha, the source database is provisioned on a VM *in a hyperconverged infrastructure system*. Sivasubramanian+Shekar+Zha meets all remaining limitations of claim 1, and also meets claims 5-8, 10-15, and 19-21, for the same reasons given *supra* §V for Sivasubramanian+Zha in Ground 1.

2. Claim 2

- a. [2PRE]-[2A]: **“The method of claim 1, further comprising: receiving, by the processor, selection of a first set of parameters; and”**

242. Shekar’s VMs have properties including “operating system” and amount of “memory (RAM).” Shekar [0046], Fig. 2B. Shekar’s VMs can also be “configured.” Shekar, [0038]. Furthermore, it was known to customize the properties of VMs, including VMs on which databases ran. Bhandari, 1:11-15 (VMs “can be customized” with “computing resources” including “RAM,” and “operating system”); Winterfeldt, [0029]-[0035] (describing “[VM] templates” for VMs running databases); VanDerWalt (EX1071), [0158]-[0159] (describing UIs for configuring VM memory, storage, and CPUs); Hardy (EX1072), [0066] (“an executable virtual machine 136 can be created from virtual machine parameters within the virtual machine profile 125”); Jin-297, 1:19-25, 7:49-8:2.

243. Thus, in Sivasubramanian+Shekar+Zha, where a database runs on a VM in Shekar’s HCI (*supra* §VI.B), POSAs would have found it obvious to implement Sivasubramanian’s graphical and command-line database-provisioning UIs (*supra* §V.A.1.a) to *receive a selection of a set of values* for VM properties, including OS and RAM, to customize the VM. POSAs would have been motivated to implement this functionality to beneficially give control-plane users flexibility

to customize VMs based on the database's needs, and such an implementation would have been within a POSA's skill. *Supra* §V.A.2.

244. The aforementioned values for VM properties like OS and memory are *parameters*. *E.g.*, EX1001, 16:51-54, 17:13-48; Bhandari, 8:19-24.

Furthermore, that set of parameters is a *first set* distinct from other parameters like database engine type, SLA, or protection schedule. *Supra* Ground 1, [1B]-[1C].

245. Because the *processor* executes the instructions for the control plane, the *receiving* is done *by the processor*. *Supra* Ground 1, [1A1].

- b. **[2B]: “creating, by the processor, a source database virtual machine based on the first set of parameters, wherein the source database resides on the source database virtual machine.”**

246. As explained *supra* [2A], the *first set of parameters* are parameters for a virtual machine. Thus, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha to *create a virtual machine based on the first set of parameters*.

247. As explained *supra* [2A], the parameters are received via the database provisioning UIs. These UIs are used to provision the *source database*. *Supra* Ground 1, [1D]. In Sivasubramanian+Shekar+Zha, a database runs as an “application” on a VM running on an HCI, as taught in Shekar. Shekar, [0008]; *supra* §VI.B.

248. In Shekar, “applications” such as databases “resid[e] on” the VMs on Shekar’s HCIs. *See* Shekar, [0038], [0035], [0046], Fig. 2B (showing “applications 155” inside “VM”). Shekar refers to applications as “residing on” HCIs. Shekar, [0035]. Because Shekar also teaches that the VMs on the HCIs are what “execute” the applications, POSAs would have understood that “applications” such as databases “resid[e] on” the VMs on Shekar’s HCIs. Shekar, [0038]; *see also* Shekar, [0046], Fig. 2B (showing “applications 155” inside “VM”).

249. Thus, in Sivasubramanian+Shekar+Zha, the *source database resides on the created virtual machine*.

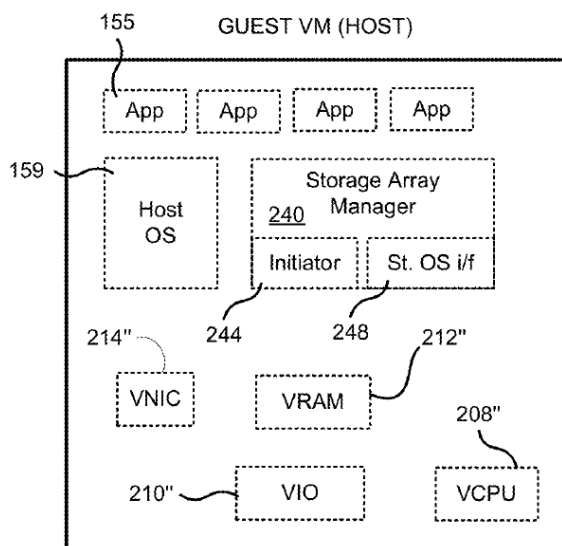


FIG. 2B

Shekar, Fig. 2B

250. The '336 patent describes a “database virtual machine” as a VM that “host[s]” a database, and describes a “*source database virtual machine*” as a virtual machine “associated with” the “storage” used by a source database. *See, e.g.,* EX1001, 3:32-49, 14:59-61.

251. Similarly, Shekar refers to VMs as “*host[s]*” for applications, and each VM includes a “storage array manager” that allows “applications” to access the “storage array.” *See, e.g.,* Shekar, [0046]-[0047], Fig. 2B. In Sivasubramanian+Shekar+Zha, the databases running on Shekar’s VMs use

Shekar's storage array for storage. *Supra* §VI.B. Thus, the aforementioned virtual machine on which the source database resides is a *source database virtual machine*.

252. Because the *processor* executes the instructions for the control plane, the *creating* is done *by the processor*. *Supra* Ground 1, [1A1].

3. Claim 3: “The method of claim 2, wherein the first set of parameters comprises at least one of a software profile, a compute profile, a network profile, and a database parameters profile.”

a. “software profile”

253. The '336 patent says a “*software profile* defines the software and operating system [OS] parameters for the database engine type that is selected” during the database “provisioning” process. EX1001, 17:13-19, 15:52-57. As explained *supra* [2A], the *first set of parameters* comprises the OS for the VM that the database of the selected engine type runs on. Additionally, Sivasubramanian discloses specifying “version information” for the *selected database engine type* in both the Figure 5 GUI and the command-line. Sivasubramanian, 12:57-13:5, Fig. 5; *supra* Ground 1, [1B]. Thus, the *first set of parameters* further comprises the database version, and the operating system (OS) and database version together comprise a *software profile*.

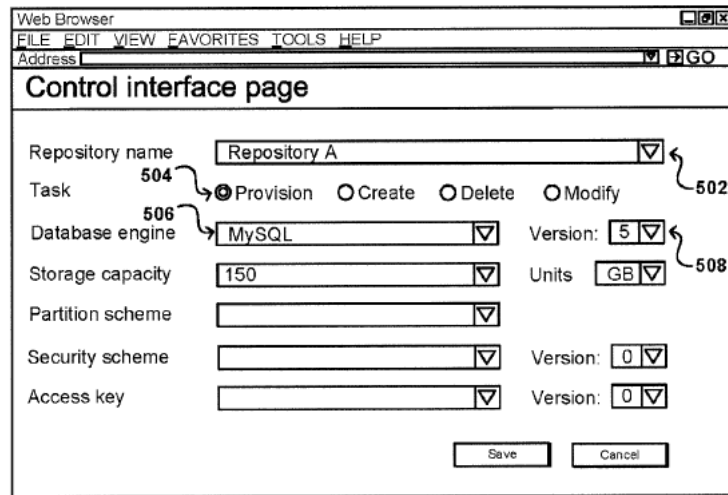


FIG. 5

Sivasubramanian, Fig. 5

254. If the claim requires the “software profile” be a predefined set of values, this would have been obvious. Defining sets of parameters values was well-known; such sets were often called “profiles” or “templates.” *See, e.g.*, Inskeep, Abstract, 1:31-34; RDS, 412; Winterfeldt, [0029]; Nickolov (EX1062), [0734], Fig. 9; Hardy, [0066]; Nichele, [0002]; Jalagam, [0028]; Kundapur, [0022]; Babbar, [0050].

255. It was likewise well-known that allowing selection of a predefined “profile”/“template” beneficially increased user convenience by reducing the need for users to choose and separately input parameter values. *see, e.g.*, Wren-Hilton,

1-2; Mikan, [0003]; Poulsen, 1:16-29; Miller, [0225]-[0226] & Fig. 16A; Kerpez, [0105].

256. POSAs understood that certain versions of applications like database engines were compatible with, or optimized for, specific OSs or versions thereof. *See, e.g.,* Zamir, [0027]; Jayaprakash, [0013]; Gallagher, [0012]; Sharma, [0031]; Heddaya, [0015]; MySQL, 1. Thus, in Sivasubramanian+Shekar+Zha, where the user may customize the source database's VM (*see supra* [2PRE]-[2A]), POSAs would have found it obvious to implement Sivasubramanian's database creation and provisioning UIs so the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) includes a predefined operating system and supported database engine version, *i.e., a software profile*; this would beneficially increase user convenience and was within a POSA's skill. *Supra* §V.A.2.

b. “compute profile”

257. A “compute profile defines the size/configuration of [a] source database VM,” *e.g.,* “the number of vCPUs [virtual CPUs]” or the “memory capacity.” EX1001, 17:42-46.

258. The *first set of parameters* comprises the amount of memory for the VM. *Supra* [2A]. Additionally, each VM in Shekar has a “VCPU.” Shekar, [0046]. In Sivasubramanian+Shekar+Zha, where the user may customize the source database's VM, POSAs would have found it obvious to implement

Sivasubramanian’s database creation and provisioning UIs so the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) includes a number of VCPUs and an amount of memory. Both were well-known configuration options for databases, and such an implementation was within a POSA’s skill. *Supra* §§VI.B; *see, e.g.*, RDS, 14 (user can select “DB Instance Class” with certain amount of “memory”), 104-105 (different “instance classes” have different amounts of “Memory (GB)” and numbers of “vCPU[s],” and explaining that “vCPU” stands for “vCPU – A virtual CPU, or virtual central processing unit”); Adogla, [0032] (“Compute instance configurations may also include... [s]ize of compute instances, such as a particular number of virtual CPU cores, memory, cache, storage, as well as any other performance characteristic.”).

259. If the claim requires a predefined set of these parameter values, this was a known UI feature that would have been obvious for the convenience reasons discussed above for software profiles.

c. “network profile”

260. Sivasubramanian’s “example ‘create database’ workflow” includes “allocating and attaching a DNS address or other address, port, interface, or identifier,” and “a user is provided with the DNS address and a port address... to access the [database] instance.” Sivasubramanian, 6:59-67, 11:40-52.

Additionally, Sivasubramanian's exemplary invocation of "rds-create-database" shows a "port" parameter being used. Sivasubramanian, 13:1-5.

261. Thus, POSAs would have been motivated to implement Sivasubramanian's database-provisioning UIs in Sivasubramanian+Shekar+Zha so that the *first set of parameters* received for configuring the source database's VM (*supra* [2PRE]-[2A]) includes the DNS address and port; this would beneficially enable access to the provisioned database by an entity "provide[d] with" that "DNS address and port." Sivasubramanian, 6:59-67.

262. A "***network profile***" in the '336 patent "identifies the network location of the database being created... to facilitate access to the database after creation." EX1001, 17:31-33.

263. POSAs understood that in Sivasubramanian, the "DNS address" identifies a database's network "location." Sivasubramanian, 2:40-44 (referencing "the ***DNS (domain name system) address or other location information***" for a database instance); *see also* Chung, [0003] ("DNS facilitates Internet communication by associating domain names with the numeric (IP, 'Internet Protocol') network ***addresses***."). POSAs also understood that the "port" identifies the network location of a database, because an external entity "use[s]" a "port address" to "access" the database. Sivasubramanian, 6:65-67. Thus, the DNS address and port together constitute a *network profile*.

264. If the claim requires a *network profile* be a predefined set of database-access-related parameter values, this was a known UI feature that would have been obvious for the convenience reasons discussed *supra* §VI.C.3.a for software profiles. *See, e.g.*, RDS, 412 (discussing “select connection profile” dialog box).

d. “database parameters profile”

265. “The database parameter profile defines the custom parameters that are applied to the database being created or registered.” EX1001, 17:51-53. The patent’s specification does not state which parameters qualify as “custom” parameters.

266. Sivasubramanian discloses that the database creation and provisioning UIs accept parameters including “Partition scheme,” “Security scheme,” “Access key,” and “password.” Sivasubramanian, 13:1-5, Fig. 5. Thus, POSAs would have been motivated to implement Sivasubramanian’s database creation and provisioning UIs in Sivasubramanian+Shekar+Zha so that the *first set of parameters* received for configuring the VM (*supra* [2PRE]-[2A]) include such parameters, which constitute a *database parameters profile*.

267. If claim 3 is interpreted to require that the database parameters profile be a single predefined set of values, this was a known UI feature that would have been obvious for the convenience reasons discussed above for software profiles. *see, e.g.*, RDS, 459-460 (discussing “DB Parameter Groups”).

4. **Claim 4: “The method of claim 1, further comprising: converting, by the processor, a size of the source database into a number of virtual disks; and associating, by the processor, the number of virtual disks with the source database.”**

268. The '336 patent refers to the user specifying a “database size to determine how much storage is needed for storing” the database.” EX1001, 18:46-52, 4:18-22. The exemplary UI also includes a database “size” field. EX1001, 24:51-63, Fig. 4E. The patent describes an exemplary “cluster” in a “virtual computing system” that includes a “storage pool” containing “storage components” “configure[able] as virtual disks for access by...database VMs.” EX1001, 6:5-7, 8:18-22, Fig. 1. The “storage pool” may include “network-attached” or “direct-attached” storage. EX1001, 6:36-41.

269. POSAs understood that a “virtual disk” is a unit of storage allocated to a virtual machine that provides a virtual representation of physical storage. Cao (EX1089), 1:15-16; Wang (EX1090), 1:13-14; Fruchtman (EX1091), [0002]. The '336 patent states that a database may use either one virtual disk or multiple virtual disks as storage for its data. EX1001, 11:19-26 (“a database storage device (e.g., a virtual disk from the storage pool)”), 14:61-63 (“the source databases stored within the source database storage may be stored within the virtual disks associated with the source database VMs”), 35:20-22 (“The *virtual disk(s)* on which the source database is stored may be composed of or divided into one or more memory

blocks.’’). Based on my review, the specification does not identify anything inventive about the choice of one virtual disk v. multiple, and as I explain below, using multiple virtual disks was known.

270. Sivasubramanian discloses the user of the control plane specifying a “**storage** capacity” or “**storage** size” for a database being created or provisioned. Sivasubramanian, 11:49-52 (“The user can call a Web service to specify any desired values for an instance type (which describes the CPU and memory capacity), storage size, repository name, port, and other such values.”), 12:62-13:5 (“a user can manually (or otherwise) create and submit Web service calls to the control plane. In the following example, the customer generates a request to create a new data repository using a command line tool,” and showing “rds-create-database” command called with “size” parameter), Figure 5.

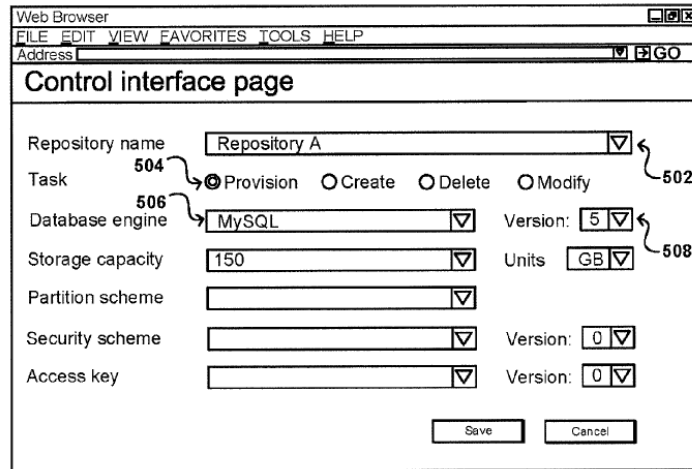


FIG. 5

271. Thus, POSAs understood that in Sivasubramanian+Shekar+Zha, the *source database*, which is provisioned as a result of using Sivasubramanian’s database-provisioning UIs (*supra* Ground 1, [1D]), has a *size*.

272. In Sivasubramanian+Shekar+Zha, the *source database* is provisioned on a virtual machine running on Shekar’s HCI. *Supra* §VI.C.1. In Shekar’s HCI, the VMs use a “storage array” for data storage. Shekar, [0032]. The array may be “local” to the HCI or accessible over a “network.” Shekar, [0034]-[0035]. Storage array access is provided by “virtual storage controllers” on the HCI. Shekar, [0040].

273. POSAs understood that it was well-known and customary to use virtual disks to provide storage for virtual machines. Cao, 1:15-16; Wang, 1:13-

14; Fruchtman, [0002]; Sadavarte (EX1092), 1:23-26; Carter (EX1093), 7:36-39; Jin-297, 3:16-26. It was also well-known for databases to use virtual disks to store data. Paul (EX1094), [0003]; Kaplingat (EX1095), [0052]; Goens (EX1064), [0019]; Reuter (EX1096), [0063]. POSAs understood that using virtual disks on shared storage such as Shekar's storage array provided benefits such as "streamlin[ing] storage management and configuration" and "enabl[ing] virtualization management techniques." Phelan (EX1097), [0002]; Shankar (EX1098), [0001]. Thus, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha so that the *source database* running on a VM on one of Shekar's HCIs used *virtual disks* to store data; such an implementation would have been within a POSA's skill, as using virtual disks as storage with both databases and VMs was well-known in the art, as discussed above.

274. POSAs also understood that it was customary for virtual disks to have a certain size, such as "10 GB." Le (EX1099), [0002]; *see also* Wang, 1:15-17; Gong (EX1100), [0003]. Sivasubramanian discloses examples of users selecting database sizes for the source database of 80 or 150 GB. Sivasubramanian, 12:38-13:5, Fig. 5. Thus, based on simple mathematics, POSAs would have found it obvious to *convert* the *size* of the source database into *a number of virtual disks* to ensure that enough virtual disks are allocated to store the database; *e.g.*, a minimum of eight 10-GB virtual disks would be needed for an 80-GB database.

POSAs would reasonably have expected success implementing such a conversion since it required simple mathematical operations. Since the *processor* in the control plane performs the provisioning, the processor also performs the *converting*. *Supra* Ground 1, [1A1], [1D].

275. Finally, POSAs would have understood that the *processor* would configure the source database so that it can use the virtual disks as storage, since the point of specifying storage capacity when provisioning a database is so that the database can use that storage. Sivasubramanian, 6:59-65, claim 10. Since this allows the database and virtual disks to collectively achieve the functionality of accessing data, this constitutes *associating* the *source database* with the *number of virtual disks*. EX1001, 38:44-51 (“In a conceptual sense, any arrangement of components to achieve the same functionality is effectively “associated” such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality can be seen as “associated with” each other such that the desired functionality is achieved, irrespective of architectures or intermedial components.”).

5. **Claim 9: “The method of claim 1, further comprising: receiving, by the processor, selection of at least one of a sample set of parameters for creating a source database virtual machine for hosting the source database; and creating, by the processor, the source database virtual machine based upon the at least one of the sample set of parameters.”**

276. As discussed above for claim 3, the '336 patent refers to different types of “profiles” as each defining multiple parameters, *i.e., sets of parameters*. EX1001, 17:13-19, 15:52-57, 17:42-46, 17:31-33, 17:51-53. The patent refers to “sample” profiles as predefined profiles the user can choose from, in contrast to custom profiles. EX1001, 17:19-30.

277. In Sivasubramanian+Shekar+Zha, the *processor receives parameters* in the form of various profiles and *creates the source database virtual machine based [up]on those parameters*. *Supra* claims 2-3. POSAs would have found it obvious to implement the control plane in Sivasubramanian+Shekar+Zha to provide at least one *sample profile, i.e., sample set of parameters*, to the user, and then *receive the selection* of that sample, because providing predefined profiles would increase user convenience and was within a POSA's skill. *Supra* §V.A.2. For example, POSAs would have been motivated to provide predefined sets of operating systems and corresponding supported database software versions to eliminate the need for the user to memorize which software version runs on which operating system. *Supra* claim 3.

6. **Claim 16: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive at least one parameter for creating a source database virtual machine, and associate the source database with the source database virtual machine.”**

278. As explained above, in Sivasubramanian+Shekar+Zha, the processor *executes the computer-readable instructions to receive a set of parameters, i.e. at least one parameter, for creating a source database virtual machine and configures the database to run on that virtual machine, i.e., associates the source database with the source database virtual machine. Supra, claim 2.*

7. **Claim 17: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive a size of the source database, and convert the size into a number of virtual disks to be associated with the source database.”**

279. As discussed above, in Sivasubramanian+Shekar+Zha, the UI receives an input of a *size of the source database. Supra claim 4. Because the processor executes the computer-readable instructions for the control plane (supra Ground 1, [1A1]), the processor executes the instructions to receive the size. Additionally, as discussed supra claim 4, the processor converts the size into a number of virtual disks to be associated with the source database.*

8. **Claim 18: “The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive selection of an existing source database virtual machine, and create the source database on the existing source database virtual machine.”**

280. As explained above, in Sivasubramanian+Shekar+Zha, the *source database is created on a source database virtual machine* that is a VM on Shekar’s HCI. *Supra* §VI.C.2.b ([2B]).

281. As an alternative to creating a new VM as discussed *supra* claim 2, POSAs would have found it obvious to implement Sivasubramanian+Shekar+Zha so that Sivasubramanian’s UI allows the user to selecting an existing *source database virtual machine*. POSAs understood that allowing UIs that allowed a user to select an existing VM for deployment of a database or another application were known in the art. *See, e.g.*, VanDerWalt, [0135]-[0138], Figs. 18-19. POSAs also understood that allowing the user to pick an existing VM would beneficially give the user the option of avoiding the added steps of creating a new VM. Such a UI was within a POSA’s skill, as I discussed *supra* §V.A.2.

9. **Claim 22: “The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for: receiving selection of at least one of a sample set of parameters for creating a source database virtual machine; and creating the source database virtual machine based upon the at least one of the sample set of parameters.”**

282. *See* my discussion of claim 9.

10. **Claim 23:** “The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for: presenting a plurality of levels of the SLA; and receiving selection of one of the plurality of levels of the SLA.”

283. See my discussion of claim 10.

VII. CLAIM LISTING

The following claim listing assigns element labels (e.g., [1A1], [1A2], etc.)

to certain claims for clarity.

Claim 1
[1PRE] A method comprising:
[1A1] receiving, by a processor executing computer-readable instructions stored on a memory,
[1A2] a user request for provisioning a source database
[1A3] in a hyperconverged infrastructure system;
[1B] receiving, by the processor, selection of a database engine type;
[1C] receiving, by the processor, selection of a Service Level Agreement (“SLA”) and a protection schedule;
[1D] provisioning, by the processor, the source database based upon the database engine type; and
[1E] setting up, by the processor, automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[1F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[1G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 2
[2PRE] The method of claim 1, further comprising:

[2A] receiving, by the processor, selection of a first set of parameters; and
[2B] creating, by the processor, a source database virtual machine based on the first set of parameters, wherein the source database resides on the source database virtual machine.
Claim 3
The method of claim 2, wherein the first set of parameters comprises at least one of a software profile, a compute profile, a network profile, and a database parameters profile.
Claim 4
[4PRE] The method of claim 1, further comprising:
[4A] converting, by the processor, a size of the source database into a number of virtual disks; and
[4B] associating, by the processor, the number of virtual disks with the source database.
Claim 5
The method of claim 1, wherein provisioning the source database comprises creating a new database as the source database.
Claim 6
The method of claim 1, wherein provisioning the source database comprises registering an existing database as the source database.
Claim 7
The method of claim 1, wherein the first frequency and the second frequency are the same.
Claim 8
[8PRE] The method of claim 1, further comprising:
[8A] presenting, by the processor, a group of database engine types that are supported by the hyperconverged infrastructure system; and
[8B] receiving, by the processor, the selection of the database engine type from the group of database engine types.
Claim 9
[9PRE] The method of claim 1, further comprising:

[9A] receiving, by the processor, selection of at least one of a sample set of parameters for creating a source database virtual machine for hosting the source database; and
[9B] creating, by the processor, the source database virtual machine based upon the at least one of the sample set of parameters.
Claim 10
[10PRE] The method of claim 1, further comprising:
[10A] presenting, by the processor, a plurality of levels of the SLA; and
[10B] receiving, by the processor, selection of one of the plurality of levels of the SLA.
Claim 11
The method of claim 1, wherein the protection schedule comprises a time period for designating at least one of a daily snapshot, a weekly snapshot, a monthly snapshot, and a quarterly snapshot.
Claim 12
[12PRE] A database management system comprising:
[12A1] a memory comprising computer-readable instructions stored thereon; and
[12A1] a processor that executes the computer-readable instructions to:
[12A3] receive a user request for provisioning a source database in a hyperconverged infrastructure system;
[12B] receive selection of a database engine type;
[12C] receive selection of a Service Level Agreement (“SLA”) and a protection schedule;
[12D] provision the source database based upon the database engine type; and
[12E] set up automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[12F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[12G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 13

The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive the user request, the selection of the database engine type, and the SLA via a dashboard.
Claim 14
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to create a cloned database from the source database based on the snapshots and the transactional logs.
Claim 15
The database management system of claim 12, wherein the first frequency and the second frequency are different.
Claim 16
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive at least one parameter for creating a source database virtual machine, and associate the source database with the source database virtual machine.
Claim 17
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive a size of the source database, and convert the size into a number of virtual disks to be associated with the source database.
Claim 18
The database management system of claim 12, wherein the processor further executes the computer-readable instructions to receive selection of an existing source database virtual machine, and create the source database on the existing source database virtual machine.
Claim 19
[19PRE] A non-transitory computer readable media with computer-executable instructions embodied thereon that, when executed by a processor in a hyperconverged infrastructure system, cause the processor to perform a process comprising:
[19A] receiving a user request for provisioning a source database in the hyperconverged infrastructure system;
[19B] receiving selection of a database engine type;

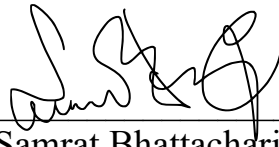
[19C] receiving selection of a Service Level Agreement (“SLA”) and a protection schedule;
[19D] provisioning the source database based upon the database engine type; and
[19E] setting up automatic capturing of snapshots and transactional logs based upon the SLA and the protection schedule for protecting the source database,
[19F] wherein the transactional logs record transactions occurring on the source database since a most recent snapshot,
[19G] and wherein the protection schedule comprises a first frequency at which the snapshots are captured and a second frequency at which the transactional logs are captured for protecting the source database.
Claim 20
The non-transitory computer readable media of claim 19, wherein provisioning the source database comprises creating a new source database or registering an existing database as the source database.
Claim 21
[21PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[21A] presenting a group of database engine types that are supported by the hyperconverged infrastructure system; and
[21B] receiving the selection of the database engine type from the group of database engine types.
Claim 22
[22PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[22A] receiving selection of at least one of a sample set of parameters for creating a source database virtual machine; and
[22B] creating the source database virtual machine based upon the at least one of the sample set of parameters.
Claim 23
[23PRE] The non-transitory computer readable media of claim 19, wherein the processor further executes the computer-executable instructions for:
[23A] presenting a plurality of levels of the SLA; and

[23B] receiving selection of one of the plurality of levels of the SLA.

I declare that all statements made herein of my own knowledge are true, that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

I declare under penalty of perjury that the foregoing is true and correct.

Dated: December 18, 2024

A handwritten signature in black ink, appearing to read 'Samrat Bhattacharjee', is written over a horizontal line.

Samrat Bhattacharjee, Ph.D.