

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAP America, Inc.
Petitioner,

v.

Valtrus Innovations Ltd., and Key Patent Innovations Ltd.,
Patent Owners.

IPR2025-00416
U.S. Patent No. 7,313,575

**PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 7,313,575**

TABLE OF CONTENTS

Page

I.	INTRODUCTION	25
II.	COMPLIANCE WITH IPR REQUIREMENTS.....	25
A.	Grounds for Standing	25
B.	Mandatory Notices	26
C.	Fee Authorization	28
III.	CHALLENGED CLAIMS	28
IV.	THE 575 PATENT	28
A.	Effective Filing Date	28
B.	Person of Ordinary Skill in the Art	28
C.	Overview of the 575 Patent	29
V.	CLAIM CONSTRUCTION	30
VI.	PRINCIPAL PRIOR ART.....	31
A.	Summary of Chen.....	31
VII.	CLAIMS 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, AND 35-36 ARE UNPATENTABLE	34
A.	Ground 1: Claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35- 36 Would Have Been Obvious Based on Chen.....	34
1.	Claim 1	34
	[1.0] A data services handler for execution on a computing system comprising:	34
i.	Error! Bookmark not defined.	
a.	<i>a data services handler</i>	34
b.	<i>for execution on a computing system</i>	36
ii.	[1.1] an interface executing on the computing system for communicating between a data store and applications that supply and consume data; and.....	38
iii.	[1.2] a real time information director (RTID) executing on the computing system that transforms	

	data for supply, consumption, or both by the applications under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data, the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand, the RTID further comprising:	43
a.	<i>a real time information director (RTID) executing on the computing system</i>	43
b.	<i>transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata</i>	44
c.	<i>that defines a security model and data integrity rules for application to the data</i>	48
d.	<i>the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand</i>	49
iv.	[1.3] at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and	50
v.	[1.4] at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption or both.	52
2.	Claim 2	54
i.	[2.0] The data services handler according to claim 1 further comprising:	54
ii.	[2.1] a data mapper collecting data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.	54
3.	Claim 4	56

i.	[4.0] The data services handler according to claim 1 wherein the real time information director further comprises:	56
ii.	[4.1] the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.	56
4.	Claim 6	57
i.	[6.0] The data services handler according to claim 1 wherein the real time information director further comprises:	57
ii.	[6.1] a logic that represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.	58
5.	Claim 7	60
i.	[7.0] The data services handler according to claim 1 wherein the real time information director further comprises:	60
ii.	[7.1] a logic that represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.	60
6.	Claim 10	62
i.	[10.0] The data services handler according to claim 1 wherein:	62
ii.	[10.1] the real time information director receives and publishes data documents having a content and a format defined by document metadata.	62
7.	Claim 11	63
i.	[11.0] The data services handler according to claim 1 wherein the real time information director further comprises:	63

ii.	[11.1] a logic that reads a schema of tables identified in metadata; and	63
iii.	[11.2] a logic that automatically generates and caches prepare SQL statements from a content metadata portion of document metadata according to the schema.	63
8.	Claim 12	64
i.	[12.0] The data services handler according to claim 1 further comprising:	64
ii.	[12.1] a servlet that invokes the real time information director synchronously when an inbound data document is delivered via HyperText Transport Protocol (HTTP):	65
iii.	[12.2] a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS);	66
iv.	[12.3] a servlet that invokes the real time information director synchronously when an outbound data document is requested via HyperText Transport Protocol (HTTP); and.....	67
v.	[12.4] a logic that invokes the real time information director asynchronously when an outbound data document is requested via Java Message Service (JMS).	68
9.	Claim 13	68
i.	[13.0] The data services handler according to claim 1 wherein the real time information director further comprises:	68
ii.	[13.1] a subscription handler that responds to a triggering event by publishing a document according to content and format specified by Document Metadata, and sending the document to a designated queue or topic.	68
10.	Claim 14	70

i.	[14.0] The data services handler according to claim 1 wherein the real time information director further comprises:	70
ii.	[14.1] a security manager that identifies an agent on behalf of whom information is queried or updated in a document request, dynamically assigns an internal unique identifier to consumers and agents upon registration;	70
iii.	and the real time information director grants access to individual documents based on security policies according to a relationship between an agent, a consumer affected by the agent, and permissions granted by the consumer.	72
11.	Claim 17	73
i.	[17.0] The data services handler according to claim 1 wherein the real time information director further comprises:	73
ii.	[17.1] an audit log handler that responds to changes in the data store by compiling and storing information based on metadata that determines content and detail of the compiled information.....	73
12.	Claim 18	74
i.	[18.0] The data services handler according to claim 1 wherein the real time information director further comprises:	74
ii.	[18.1] a data cleanser that receives data elements identified by Document Metadata, normalizes the data elements, queues source and destination information for detection of duplicates, and cross-references duplicates in the data store.	74
13.	Claim 19.....	76
i.	[19.0] The data services handler according to claim 1 wherein the real time information director further comprises:	76

ii.	[19.1] an aggregation engine that computes aggregates in memory from data selected from the data store.....	76
14.	Claim 20.....	77
i.	[20.0] The data services handler according to claim 1 wherein the real time information director further comprises:	77
ii.	[20.1] a real-time scoring engine.....	77
15.	Claim 21	77
i.	[21.0] The data services handler according to claim 1 wherein the real time information director further comprises:	77
ii.	[21.1] a business rules engine that defines a business object model capable of enabling access to data described by Document Metadata, performing computations on data relevant to business rules specified by the business object model, concluding a decision based on computation results, and executing the decision.	77
16.	Claim 22	79
i.	[22.0] An article of manufacture comprising:	79
ii.	[22.1] a controller-usable storage medium having a computer readable program code embodied therein for managing data services, the computer readable program code further comprising:	80
iii.	[22.2] a code causing the controller to transfer data among a data store and applications that supply and consume data;	81
iv.	[22.3] a code causing the controller to transform data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class;	82

v.	[22.4] a code causing the controller to dynamically instantiate, create, and cache for reuse at least one disassembler and at least one assembler according to demand;	83
vi.	[22.5] a code dynamically created by the controller as defined by polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and.....	83
vii.	[22.6] a code dynamically created by the controller as defined by polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table.....	84
17.	Claim 23	84
i.	[23.0] The article of manufacture according to claim 22 further comprising:	84
ii.	[23.1] a code causing the controller to transform data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.	84
18.	Claim 24	84
i.	[24.0] The article of manufacture according to claim 22 further comprising:	84
ii.	[24.1] a code causing the controller to collect data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.	85
19.	Claim 25	85
i.	[25.0] The article of manufacture according to claim 22 further comprising:	85
ii.	[25.1] a code causing the controller to represent metadata as stateless polymorphic objects that are permanent after initial construction, whereby	

	methods of stateful meta data classes delegate operation to the stateless poly morphic metadata objects.	85
20.	Claim 28.....	86
	i. [28.0] An enterprise system comprising:.....	86
	ii. [28.1] a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers; and	86
	iii. [28.2] a real time information director (RTID) executing on at least one of the interlinked servers that manages information flow between a data store and applications that supply and consume data and dynamically instantiates, creates, and caches for reuse disassemblers and assemblers according to demand,.....	87
	iv. [28.3] the RTID that creates at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables and create at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption, or both.	88
21.	Claim 31	88
	i. [31.0] The enterprise system according to claim 28 wherein:	88
	ii. [31.1] the RTID represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.	89
22.	Claim 32	89

i.	[32.0] The enterprise system according to claim 28 wherein:	89
ii.	[32.1] the RTID represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.	89
23.	Claim 33	89
i.	[33.0] A data services handler according to claim 1 further comprising:	89
ii.	[33.1] at least one generator that is dynamically created by the RTID as defined by polymorphic metadata and generates an extended Markup Language (XML) Schema that describes format of the documents, the at least one disassembler, assembler, and generator that use a single descriptive polymorphic metadata.	90
24.	Claim 35	92
i.	[35.0] The data services handler according to claim 33 wherein the real time information director further comprises:	92
ii.	[35.1] a query logic that queries the database to determine information about tables in the database that the document uses, the information including field name, field datatype, and field allowance of nulls.	92
25.	Claim 36	93
i.	[36.0] A method for executing on a computer system for managing data services comprising:	93
ii.	[36.1] transferring data among a data store and applications that supply and consume data;	93
iii.	[36.2] transforming data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according	

	to a predetermined object class, the behaviors and attributes being defined specific to object class;	94
iv.	[36.3] dynamically instantiating, creating, and caching for reuse at least one disassembler and at least one assembler according to demand;	94
v.	[36.4] disassembling inbound data documents into multiple records in preparation for insertion into metadata tables; and.....	94
vi.	[36.5] assembling outbound data documents from at least one record selected from at least one metadata-selected table.....	94
VIII.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	94
A.	The Board Should Not Deny Institution Under 35 U.S.C. § 325(d).....	94
B.	The Board Should Not Deny Institution Under 35 U.S.C. § 314(a).....	95
IX.	CONCLUSION.....	98

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Apple, Inc. v. SEVEN Networks, LLC</i> , IPR2020-00156 (June 15, 2020).....	97
<i>Samsung Elecs. Co. Ltd. v. Dynamics Inc.</i> , IPR2020-00505 (Aug. 12, 2020)	98
<i>Sand Revolution II, LLC v. Continental Intermodal Group-Trucking LLC</i> , IPR2019-01393 (June 16, 2020).....	97
<i>SAP America, Inc. v. Valtrus Innovations Limited, and Patent Platform Services, LLC</i> , 24-cv-54-GBW (DDE)	26
<i>SAP America, Inc. v. Valtrus Innovations Limited et al.</i> , 1:24-cv-00756-GBW (D. Del.).....	26
<i>Valtrus Innovations Ltd. v. SAP America, Inc. et al.</i> , 2:24-cv-00024-JRG (E.D. Tex.)	26
<i>Valtrus Innovations Ltd. v. SAP America, Inc. et al.</i> , 2:24-cv-00533 (E.D. Tex.).....	26
<i>Vivid Techs., Inc. v. Am. Sci. & Eng.'g. Inc.</i> , 200 F.3d 795 (Fed. Cir. 1999).	31
Statutes	
35 U.S.C. § 102(b)	32
35 U.S.C. § 103	25, 28
35 U.S.C. § 314(a)	95
35 U.S.C. § 325(d)	94

Other Authorities

37 C.F.R. § 42.100(b)	30
37 C.F.R. § 42.104(a).....	25

CLAIM LISTING

Claim 1	
1.0	A data services handler for execution on a computing system comprising:
1.1	an interface executing on the computing system for communicating between a data store and applications that supply and consume data; and
1.2	a real time information director (RTID) executing on the computing system that transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data, the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand, the RTID further comprising:
1.3	at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and
1.4	at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption or both.
Claim 2	
2.0	The data services handler according to claim 1 further comprising:
2.1	a data mapper collecting data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.

Claim 4	
4.0	The data services handler according to claim 1 wherein the real time information director further comprises:
4.1	the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.
Claim 6	
6.0	The data services handler according to claim 1 wherein the real time information director further comprises:
6.1	a logic that represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.
Claim 7	
7.0	The data services handler according to claim 1 wherein the real time information director further comprises:
7.1	a logic that represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.
Claim 10	
10.0	The data services handler according to claim 1 wherein:
10.1	the real time information director receives and publishes data documents having a content and a format defined by document metadata.
Claim 11	
11.0	The data services handler according to claim 1 wherein the real time information director further comprises:

11.1	a logic that reads a schema of tables identified in metadata; and
11.2	a logic that automatically generates and caches prepare SQL statements from a content metadata portion of document metadata according to the schema.
Claim 12	
12.0	The data services handler according to claim 1 further comprising:
12.1	a servlet that invokes the real time information director synchronously when an inbound data document is delivered via HyperText Transport Protocol (HTTP);
12.2	a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS);
12.3	a servlet that invokes the real time information director synchronously when an outbound data document is requested via HyperText Transport Protocol (HTTP); and
12.4	a logic that invokes the real time information director asynchronously when an outbound data document is requested via Java Message Service (JMS).
Claim 13	
13.0	The data services handler according to claim 1 wherein the real time information director further comprises:
13.1	a subscription handler that responds to a triggering event by publishing a document according to content and format specified by Document Metadata, and sending the document to a designated queue or topic.
Claim 14	
14.0	The data services handler according to claim 1 wherein the real time information director further comprises:

14.1	a security manager that identifies an agent on behalf of whom information is queried or updated in a document request, dynamically assigns an internal unique identifier to consumers and agents upon registration;
14.2	and the real time information director grants access to individual documents based on security policies according to a relationship between an agent, a consumer affected by the agent, and permissions granted by the consumer.
Claim 17	
17.0	The data services handler according to claim 1 wherein the real time information director further comprises:
17.1	an audit log handler that responds to changes in the data store by compiling and storing information based on metadata that determines content and detail of the compiled information.
Claim 18	
18.0	The data services handler according to claim 1 wherein the real time information director further comprises:
18.1	a data cleanser that receives data elements identified by Document Metadata, normalizes the data elements, queues source and destination information for detection of duplicates, and cross-references duplicates in the data store.
Claim 19	
19.0	The data services handler according to claim 1 wherein the real time information director further comprises:
19.1	an aggregation engine that computes aggregates in memory from data selected from the data store.
Claim 20	
20.0	The data services handler according to claim 1 wherein the real time information director further comprises:

20.1	a real-time scoring engine.
Claim 21	
21.0	The data services handler according to claim 1 wherein the real time information director further comprises:
21.1	a business rules engine that defines a business object model capable of enabling access to data described by Document Metadata, performing computations on data relevant to business rules specified by the business object model, concluding a decision based on computation results, and executing the decision.
Claim 22	
22.0	An article of manufacture comprising:
22.1	a controller-usable storage medium having a computer readable program code embodied therein for managing data services, the computer readable program code further comprising:
22.2	a code causing the controller to transfer data among a data store and applications that supply and consume data;
22.3	a code causing the controller to transform data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class;
22.4	a code causing the controller to dynamically instantiate, create, and cache for reuse at least one disassembler and at least one assembler according to demand;
22.5	a code dynamically created by the controller as defined by polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and

22.6	a code dynamically created by the controller as defined by polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table.
Claim 23	
23.0	The article of manufacture according to claim 22 further comprising:
23.1	a code causing the controller to transform data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data
Claim 24	
24.0	The article of manufacture according to claim 22 further comprising:
24.1	a code causing the controller to collect data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.
Claim 25	
25.0	The article of manufacture according to claim 22 further comprising:
25.1	a code causing the controller to represent metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful meta data classes delegate operation to the stateless poly morphic metadata objects.
Claim 28	
28.0	An enterprise system comprising:

28.1	a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers; and
28.2	a real time information director (RTID) executing on at least one of the interlinked servers that manages information flow between a data store and applications that supply and consume data and dynamically instantiates, creates, and caches for reuse disassemblers and assemblers according to demand,
28.3	the RTID that creates at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables and create at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption, or both.
Claim 31	
31.0	The enterprise system according to claim 28 wherein:
31.1	the RTID represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.
Claim 32	
32.0	The enterprise system according to claim 28 wherein:
32.1	the RTID represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.

Claim 33	
33.0	A data services handler according to claim 1 further comprising:
33.1	at least one generator that is dynamically created by the RTID as defined by polymorphic metadata and generates an extended Markup Language (XML) Schema that describes format of the documents, the at least one disassembler, assembler, and generator that use a single descriptive polymorphic metadata.
Claim 35	
35.0	The data services handler according to claim 33 wherein the real time information director further comprises:
35.1	a query logic that queries the database to determine information about tables in the database that the document uses, the information including field name, field datatype, and field allowance of nulls.
Claim 36	
36.0	A method for executing on a computer system for managing data services comprising:
36.1	transferring data among a data store and applications that supply and consume data;
36.2	transforming data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class;
36.3	dynamically instantiating, creating, and caching for reuse at least one disassembler and at least one assembler according to demand;
36.4	disassembling inbound data documents into multiple records in preparation for insertion into metadata tables; and

36.5	assembling outbound data documents from at least one record selected from at least one metadata-selected table.
------	---

PETITIONER’S EXHIBIT LIST

No.	Description
1101	U.S. Patent No. 7,313,575
1102	U.S. Patent No. 7,313,575 File History
1103	Declaration of Kevin Almeroth, Ph.D.
1104	CV of Kevin Almeroth, Ph.D.
1105	W. Chen, et al., “DB2 UDB Evaluation Guide for Linux and Windows,” <i>IBM Redbooks</i> , April 2003. Available: https://www.redbooks.ibm.com/
1106	IEEE Standard Glossary of Computer Networking Terminology
1107	IEEE Standard Glossary of Software Engineering Terminology
1108	Declaration of Martin Farach-Colton, M.D., Ph.D., In Support of Plaintiffs’ Proposed Claim Constructions (Dec. 20, 2024)
1109	Declaration of Sylvia D. Hall-Ellis, Ph.D.
1110	Maria Sueli Almeida, et al., <i>Getting Started with Data Warehouse and Business Intelligence</i> , IBM (Aug. 1999)
1111	Microsoft Computer Dictionary (5th ed. 2002)
1112	Java Message Service (Nov. 9, 1999)
1113	Juan M. Andrade, et al., <i>The Tuxedo System</i> (1996)
1114	XML Extender Administration and Programing (Ver. 8, 2002)
1115	Director Vidal’s Memorandum Regarding Discretionary Denial Procedure, Dated June 21, 2022
1116	First Amended Docket Control Order in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1117	Motion to Dismiss for Lack of Standing (Public Version) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1118	Motion to Reconsider Ruling on Motion to Dismiss for Lack of Standing (Public Version) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)

1119	Original Complaint for Patent Infringement in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00533-JRG (E.D. Tex.)
1120	Plaintiffs' Notice Regarding <i>Valtrus II</i> in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1121	SAP's Invalidity Contentions (Cover Pleading) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)

I. INTRODUCTION

Petitioner SAP America, Inc. and SAP, SE submits this Petition for *Inter Partes* review of claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35-36 (the “Challenged Claims”) of U.S. Patent No. 7,313,575 (the “575 Patent” (Ex-1101)), assigned to Patent Owners, Valtrus Innovations Ltd. and Key Patent Innovations Ltd. Petitioner respectfully submits that the Challenged Claims of the 575 Patent are unpatentable under 35 U.S.C. § 103 in view of the prior art references discussed herein. Accordingly, it is respectfully requested that the Board institute an *inter partes* review of the 575 Patent.

II. COMPLIANCE WITH IPR REQUIREMENTS

A. Grounds for Standing

As required by 37 C.F.R. § 42.104(a), Petitioner certifies that the 575 Patent “is available for inter partes review and that the Petitioner is not barred or estopped from requesting an inter partes review challenging the patent claims on the grounds identified in the petition.” Neither Petitioner nor any real party in interest has filed a civil action challenging the validity of a claim of the 575 Patent. This petition is being filed no more than one year after the date on which Petitioner or real parties in interest or privies of Petitioner were served with a complaint alleging infringement of the 575 Patent.

B. Mandatory Notices

Real Party-in-Interest: Petitioner identifies SAP America, Inc. and SAP SE (collectively, “SAP”) as real parties-in-interest.

Related Matters: No previous IPR or post-grant proceeding has been filed challenging the 575 Patent. Valtrus has asserted the 575 Patent in litigation against SAP in *Valtrus Innovations Ltd. v. SAP America, Inc. et al.*, 2:24-cv-00024-JRG (E.D. Tex.); Patent Owners have asserted the 575 Patent in litigation against SAP in *Valtrus Innovations Ltd. v. SAP America, Inc. et al.*, 2:24-cv-00533 (E.D. Tex.); and SAP filed a declaratory judgment action of non-infringement of the 575 Patent against Patent Owners and Patent Platform Services, LLC in *SAP America, Inc. v. Valtrus Innovations Limited et al.*, 1:24-cv-00756-GBW (D. Del.) and against Valtrus and Patent Platform Services, LLC in *SAP America, Inc. v. Valtrus Innovations Limited, and Patent Platform Services, LLC*, 24-cv-54-GBW (DDE) (voluntarily dismissed).

Lead and Back-up Counsel

Lead Counsel	Back-up Counsel
Louis L. Campbell (Reg. # 59,963) Winston & Strawn LLP 255 Shoreline Drive, Suite 520 Redwood City, CA 94065 Tel: (650) 858-6500 LLcampbell@winston.com	Scott M. Border (Reg. # 77,744) Winston & Strawn LLP 1901 L Street NW Washington, DC 20036 Tel: (202) 282-5000 SBorder@winston.com
	Samuel W. Riebe (Reg. # 78,044) Winston & Strawn LLP 2121 N. Pearl Street, Suite 900

	Dallas, TX 75201 Tel: (214) 453-6500 SRiebe@winston.com
	Michael Bittner (<i>Pro hac vice</i> to be submitted) Winston & Strawn LLP 2121 N. Pearl Street, Suite 900 Dallas, TX 75201 Tel: (214) 453-6500 MBittner@winston.com
	Matt R. McCullough (<i>Pro hac vice</i> to be submitted) Winston & Strawn LLP <i>Pro hac vice</i> to be submitted Winston & Strawn LLP 255 Shoreline Drive, Suite 520 Redwood City, CA 94065 Tel: (650) 858-6500 MRMcCullough@winston.com
	William Logan (<i>Pro hac vice</i> to be submitted) <i>Pro hac vice</i> to be submitted Winston & Strawn LLP 800 Capitol Street, Suite 2400 Houston, TX 77002 Tel: (713) 651-2600 WLogan@winston.com
	Patrick Clark (<i>Pro hac vice</i> to be submitted) Winston & Strawn LLP <i>Pro hac vice</i> to be submitted 2121 N. Pearl Street, Suite 900 Dallas, TX 75201 Tel: (214) 453-6500 PClark@winston.com

Service Information: Service information is provided in the designation of counsel above. Petitioner consent to service of all documents via electronic mail at IPR-SAP-Val@winston.com.

C. Fee Authorization

The Office is authorized to charge all fees due in connection with this proceeding to Deposit Account No. 50-1814.

III. CHALLENGED CLAIMS

Ground 1: Claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35-36 are unpatentable as obvious under 35 U.S.C. § 103 based on Chen (Ex-1105).

IV. THE 575 PATENT

A. Effective Filing Date

The 575 Patent was filed June 14, 2004, and does not claim priority to any other application. Thus, Petitioner assumes for the purposes of this Petition that **June 14, 2004**, is the effective filing date.

B. Person of Ordinary Skill in the Art

A Person of Ordinary Skill in The Art (“POSITA”) in June 2004 would have had a working knowledge of the telecommunications and networking art pertinent to the 575 Patent. Ex-1103, ¶ 33. A POSITA would have had a bachelor’s degree in computer science, computer engineering, or an equivalent, and two years of relevant experience in networking and network security, and related fields. Lack of professional experience can be remedied by additional education, and vice versa. *Id.*

C. Overview of the 575 Patent

The 575 Patent describes a “data services system” wherein a “data services handler comprises an interface between a data store and applications that supply and consume data, and a real time information director (RTID) that transforms data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.”¹ Ex-1101, 1:54-60.

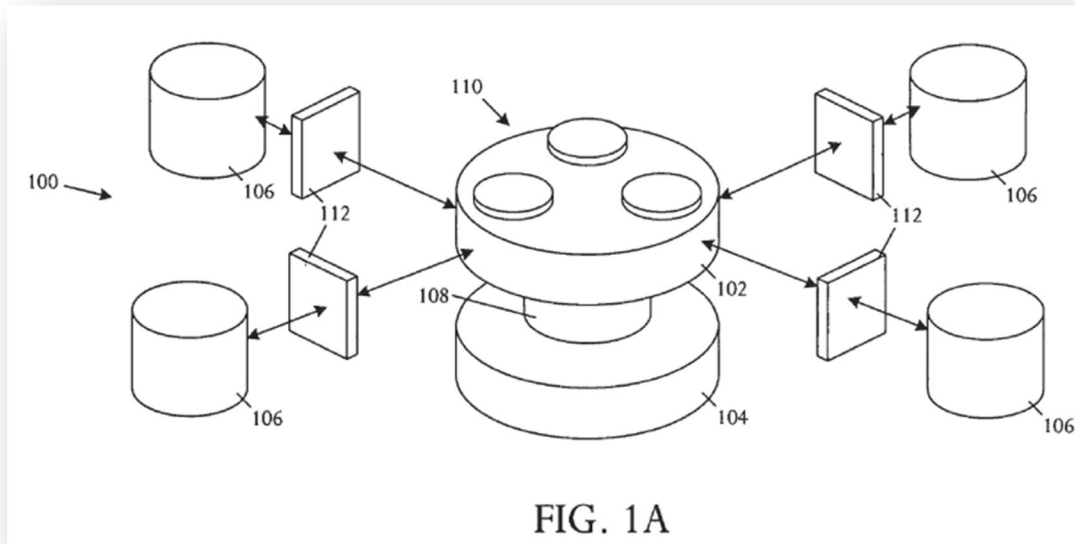
The RTID “is responsible for most or all data input and output to the real-time data store.” Ex-1101, 3:24-26. It defines document classes by “descriptive metadata” that builds both document “assemblers” and “disassemblers.” Ex-1101, 3:58-64. The RTID “supplies multiple services for clients of the data store including security, auditing, data cleansing, deduplication aggregation, real-time scoring, business rules, accumulators, user-written functions, and the like.” Ex-1101, 4:13-17.

“Descriptive metadata is typically very concise in comparison to procedural code.” Ex-1101, 4:27-28. “Polymorphic metadata can be used to enable extension of the [RTID] to facilitate and address new functionality and behaviors.” Ex-1101, 4:34-36. The RTID uses metadata for multiple purposes, including “controlling disassembly of documents for loading into a database, controlling assembly of

¹ As explained in Section V, Petitioner has applied “polymorphic metadata” as “metadata defined through a polymorphic mechanism, such as hierarchal inheritance.”

documents from records in the database, [and] generating an eXtended Markup Language (XML) schema that describes the document format.” Ex-1101, 4:18-26.

Figure 1A shows an embodiment of the 575 Patent’s invention:



Ex-1101, Fig. 1A. Figure 1A includes “data services handler 100 that comprises an interface 102 between a data store 104 and applications, shown as applications 106, which supply and consume data, and a real time information director 108.” Ex-1101, 5:7-11. RTID 108 “transforms data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.” Ex-1101, 5:11-14.

V. CLAIM CONSTRUCTION

Claims are given their “ordinary and customary meaning” as understood by a POSITA and the patent’s prosecution history. 37 C.F.R. § 42.100(b).

Patent Owner has asserted through an expert in a parallel proceeding that “*polymorphic metadata*” means “metadata defined through a polymorphic mechanism, such as hierarchal inheritance.” Ex-1108, ¶ 130. Petitioner has adopted that meaning for the limited purpose of this Petition.

Patent Owner has also asserted through an expert in a parallel proceeding that “*logic*” refers to “software.” Ex-1108, ¶ 144 (“A person of ordinary skill would have been familiar with this way of referring to software as ‘logic’ and would have interpreted the claims accordingly for all such terms.”). Petitioner has adopted that meaning for the limited purpose of this Petition.

Otherwise, because a POSITA would find the challenged claims unpatentable under any interpretation consistent with their plain and ordinary meaning in the context of the 575 Patent, the Board need not expressly construe the claim terms. *See Vivid Techs., Inc. v. Am. Sci. & Eng. ’g. Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

VI. PRINCIPAL PRIOR ART

A. Summary of Chen

The printed publication titled *DB2 UDB Evaluation Guide for Linux and Windows* and authored by W. Chen, et al. (“Chen”) (Ex-1105) was publicly accessible no later than April 2003 as a part of the *IBM Redbooks* series. Ex-1109, ¶¶ 43-49. Chen was disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the art, exercising reasonable diligence,

could have located it prior to June 14, 2004. Ex-1109, ¶¶ 43-52. (explaining Chen was published, copyrighted, and cataloged prior to June 14, 2004).

Chen is prior art to the 575 Patent at least under pre-AIA 35 U.S.C. § 102(b). Chen is also analogous art to the 575 Patent as it is directed toward the same field of endeavor—data services—and reasonably pertinent to a problem described by the 575 Patent: the need for “zero latency performance so that a business can integrate, synchronize, and route data across the enterprise, all in **real time**.” Ex-1101, Abstract (“A data services handler ...”), 1:31-33; Ex-1105, 1 (“DB2 Universal Database (UDB) is a ... relational database management system ...”).

Chen is directed to “a secure and reliable system to manage data or content for a wide variety of customer applications, such as e-Commerce, customer relationship management (CRM), enterprise resource planning (ERP), supply chain management (SCM), business intelligence (BI), Web portals, content and knowledge management.” Ex-1105, 2.

Chen teaches a “complete multimedia, Web-ready, relational database management system” suitable for a “wide range of customer applications, which can be deployed on a variety of hardware and software platforms.” Ex-1105, 1-2. This system “is a scalable and industrial strength relational database management system” that can serve “both structured and unstructured data management needs of a single user to very large enterprises.” Ex-1105, 1.

Chen's Figure 1-2 shows an overview of the system architecture:

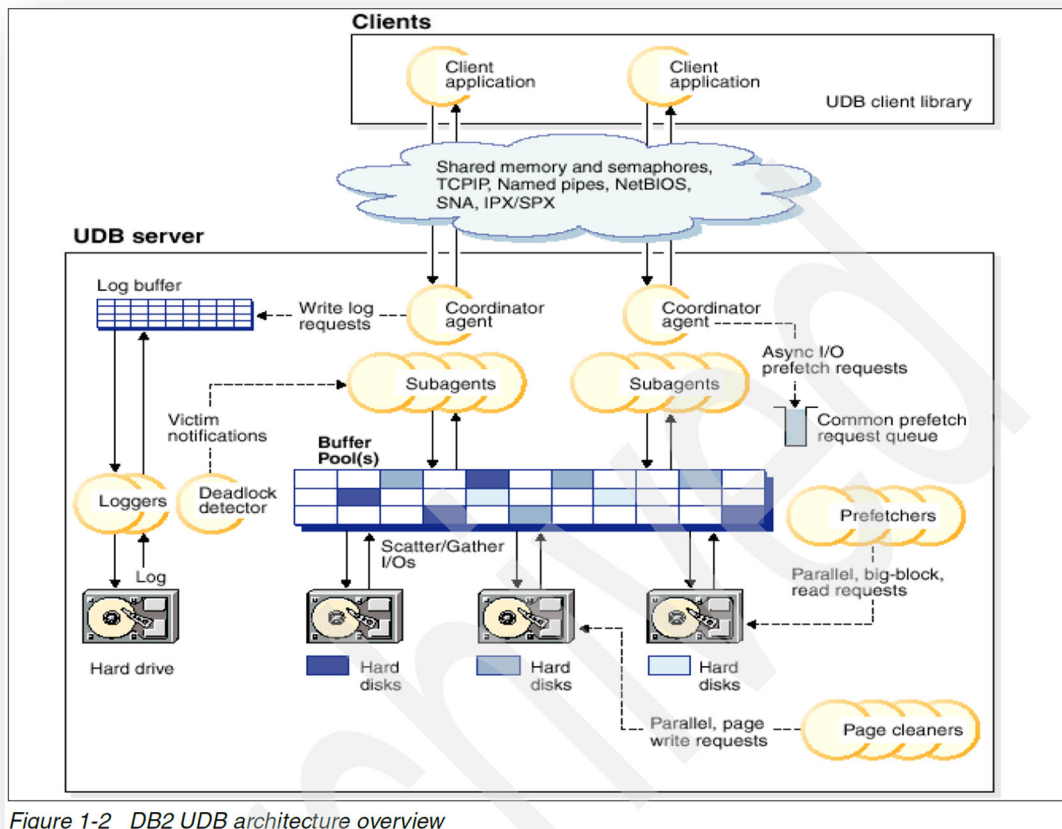


Figure 1-2 DB2 UDB architecture overview

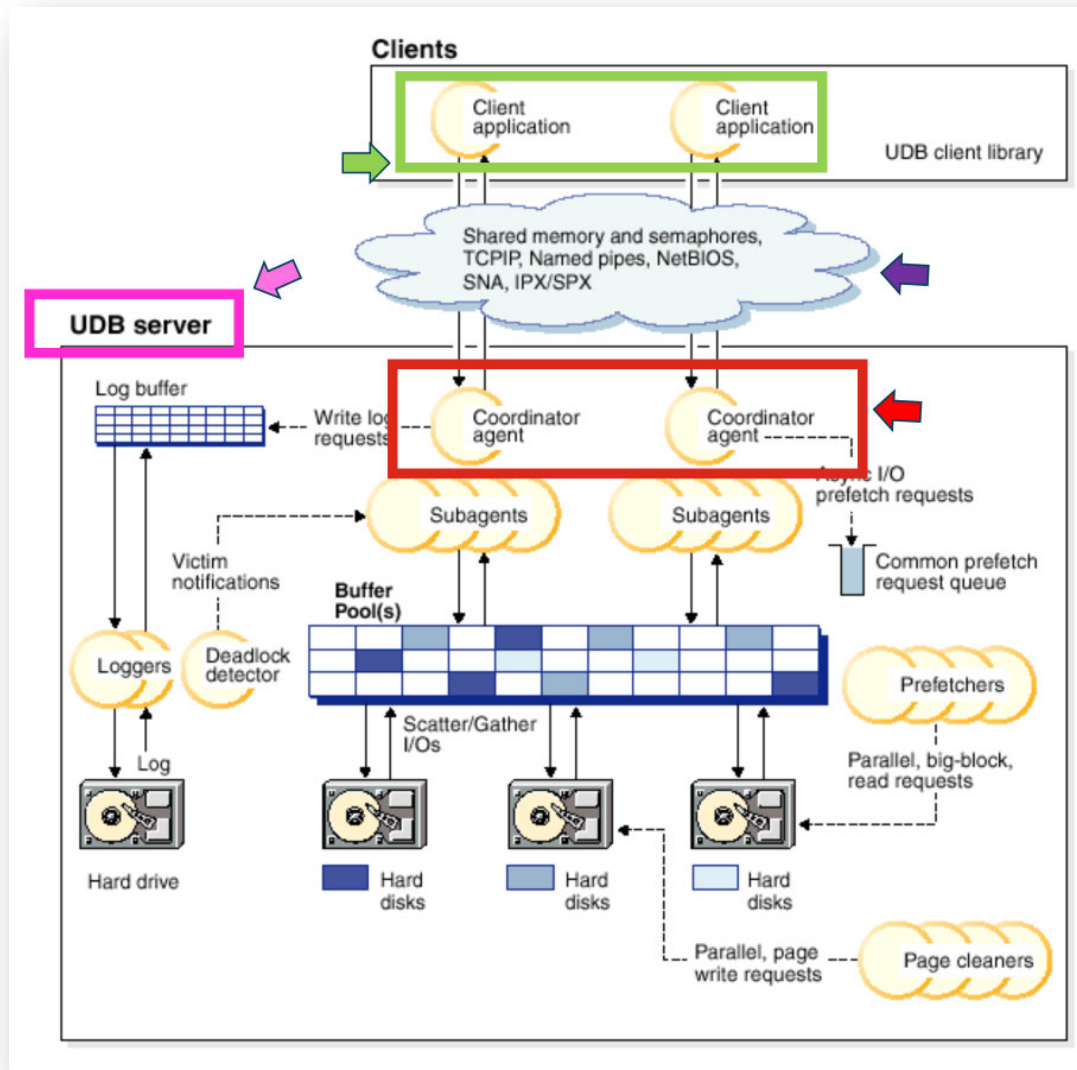
Ex-1105, Fig. 1-2. Chen discloses that “[e]ach client application links with the DB2 client library, and communicates with the DB2 Server using shared memory (local clients), or a communication protocol such as TCP/IP (remote clients).” Ex-1105, 10. Chen further discloses that “DB2 XML Extender provides the ability to store and access XML documents, and also to compose XML documents from the existing relational data or to decompose XML documents into relational data. It does so by providing a set of new data types functions, and stored procedures.” Ex-1105, 318.

VII. CLAIMS 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, AND 35-36 ARE UNPATENTABLE**A. Ground 1: Claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35-36 Would Have Been Obvious Based on Chen****1. Claim 1****i. [1.0] *A data services handler for execution on a computing system comprising:*****a. *a data services handler***

The 575 Patent describes a data services handler “compris[ing] an interface between a data store and applications that supply and consume data, and a real time information director (RTID) that transforms data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.” Ex-1101, Abstract, 1:55-60.

The DB2 System includes an interface between a data store and applications that supply and consume data. Chen describes the DB2 System as “a scalable and industrial strength **relational database management system** ... [that] is suitable for a wide range of **customer applications**.” Ex-1105, 1-2. As demonstrated below, a POSITA would have understood the DB2 System’s relational database management system to be a data store and the customer applications to be applications that supply and consume data. Ex-1103, ¶¶ 56-58.

For instance, as depicted below, the client applications (green) supply data to, and consume data from, the UDB server (pink) of the DB2 System through coordinator agents (red).



Ex-1105, 11. "Each client application links with the DB2 client library, and communicates with the DB2 Server using shared memory (local clients), or a communication protocol such as TCP/IP (remote clients)." Ex-1105, 10. The DB2

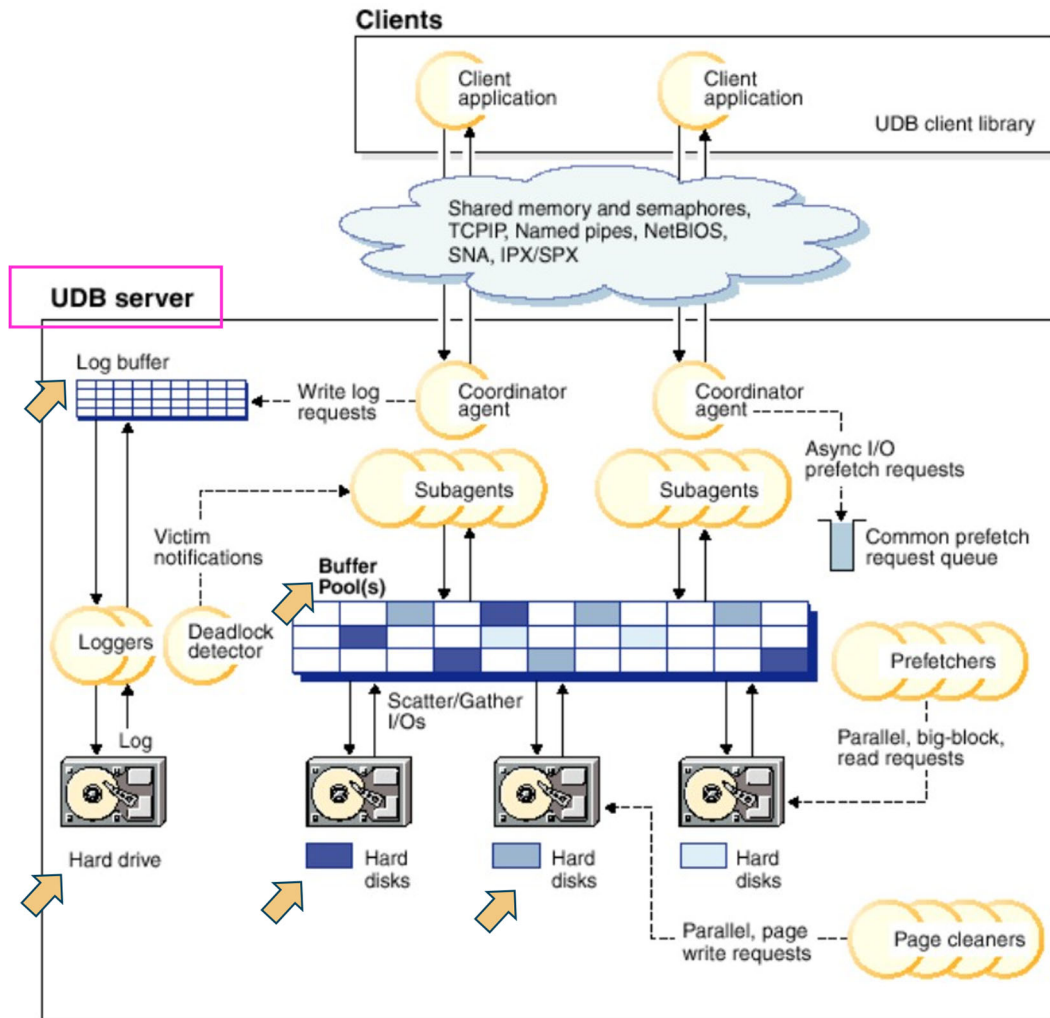
Server is a relational database that utilizes SQL translation and data flow through buffer pools for streamlining performance. Ex. 1105, 1, 10; Ex-1103, ¶ 60. Because of this, a POSITA would have understood that the system in Chen comprises an interface between the data store and customer applications and that communications between the client applications and the DB2 Server comprise supplying and consuming data. Ex-1103, ¶¶ 61-62 (citing Ex-1105, 1-2, 10).

For the reason described regarding limitation [1.2], *infra*, Chen also discloses that the DB2 System includes a real time information director (RTID) that transforms data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data. Ex-1103, ¶ 63.

Accordingly, Chen teaches or suggests “*a data services handler*.”

b. for execution on a computing system

Chen discloses that the DB2 System “can be deployed on a variety of hardware and software platforms all the way from palm devices to massive parallel multiprocessor systems.” Ex-1105, 2. This is consistent with Chen’s Fig. 1-2, which shows the DB2 System’s UDB Server (pink) deployed using conventional computer hardware, including hard drives and buffers (yellow arrows):



Ex-1105, Fig. 1-2. Chen discloses that the DB2 System executes on machines:

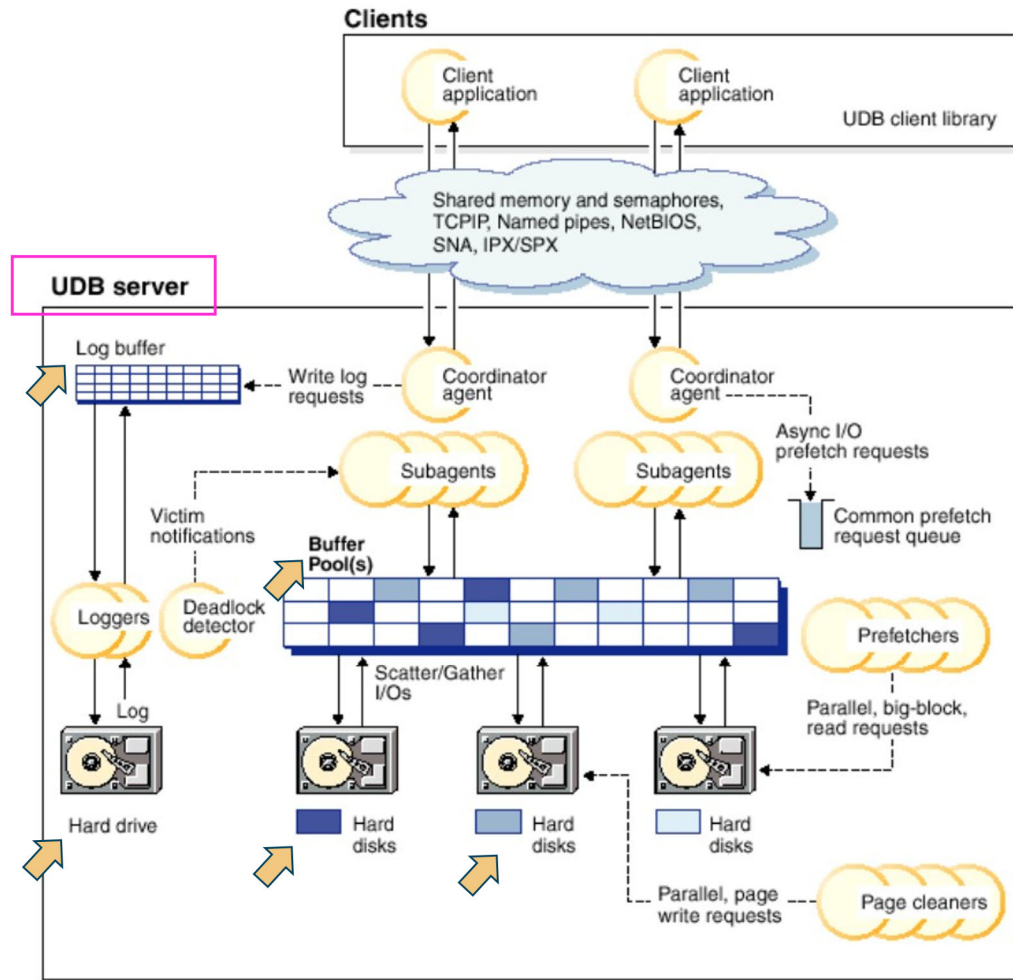
DB2 is a very scalable database product allowing you to create multi-terabyte databases through its partitioning feature by spreading a database across many machines (called a cluster). Even if a multi-terabyte database is not in your plans, partitioning a database allows for a shared nothing architecture in which each machine in the cluster performs [its] own workload independent from all other machines. This means independent processing, independent memory, independent disks.

Ex-1105, 346-47. The DB2 System can be loaded on computing devices ranging from “[l]aptops supporting mobile users to massive parallel systems with terabytes of data and/or thousands of users.” Ex-1105, 2. A POSITA thus would have understood Chen to disclose that the DB2 System is “*for execution on a computing system*.” Ex-1103, ¶ 68; Ex-1111, 200 (defining “execute” as “loading the machine code of the program into memory and then performing the instructions”).

Accordingly, Chen teaches or suggests limitation [1.0].

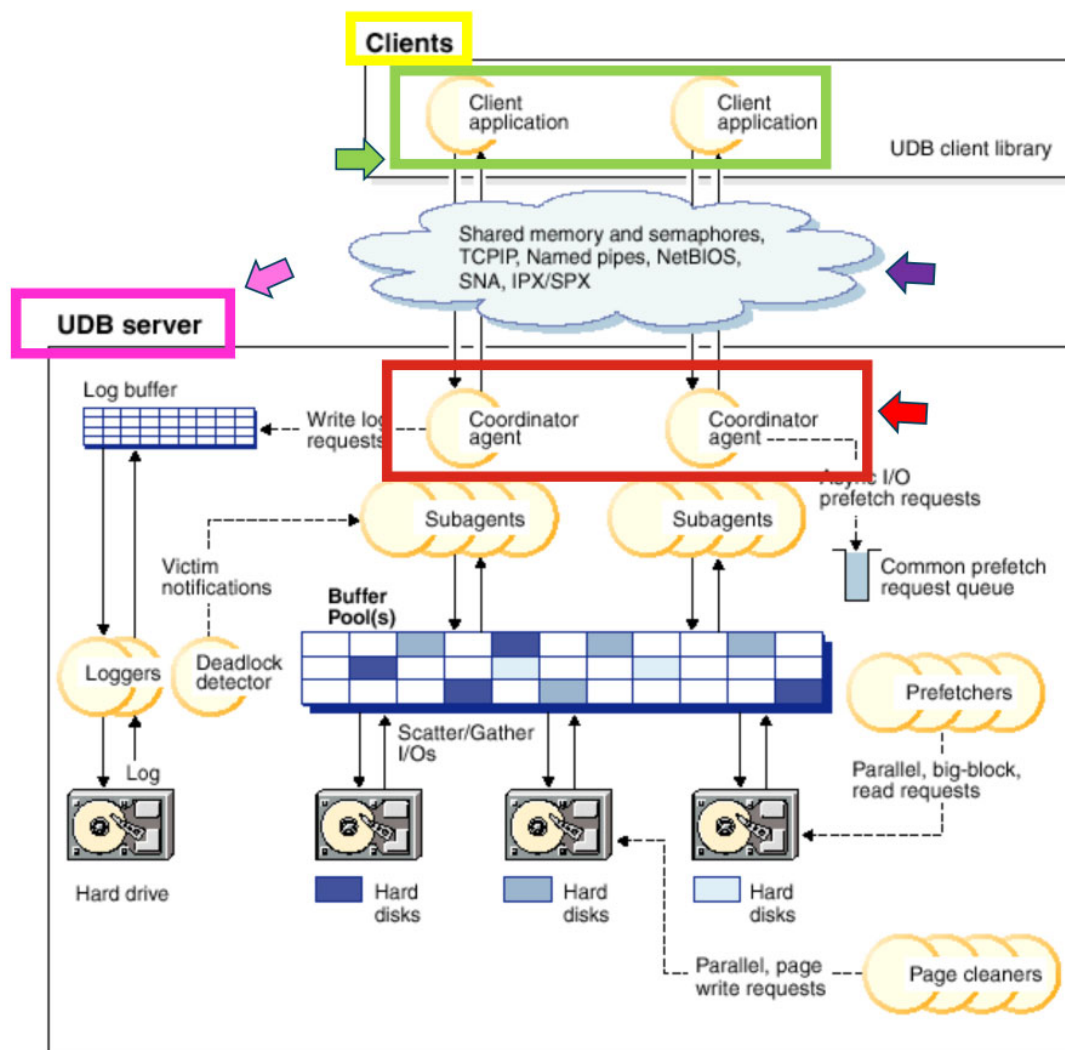
- ii. ***[1.1] an interface executing on the computing system for communicating between a data store and applications that supply and consume data; and***

As shown in Figure 1-2, Chen’s DB2 system, designed “*for execution on a computing system*” as explained for limitation [1.0], includes UDB server (pink):



Ex-1105, Fig. 1-2. UDB server in Chen is a database server—specifically a DB2 Universal Database server. Ex-1105, 1. UDB server includes persistent and volatile storage—e.g., the hard drives and buffers in Figure 1-2 (orange arrows). Because the DB2 UDB system “manage[s] data or content for a wide variety of customer applications” (Ex-1105, 1-2), a POSITA would have understood UDB server to comprise “a data store.” Ex-1103, ¶ 72 (citing Ex-1110, 42 (identifying DB2 as one of the “Data Stores Supported”)).

Chen teaches that UDB server can be operated as a local application or (more pertinently) a network server, and UDB server “can be access[ed] from many client workstations over many networks.” Ex-1105, 3, 10. As illustrated in Figure 1-2, Chen explains that these clients (outlined yellow) may contain client applications (outlined green):



Ex-1103, ¶ 73; Ex-1105, Fig. 1-2.

The data store—UDB server (pink)—includes coordinator agents (“*an interface executing on the computing system*”) (red). Ex-1105, 11. These coordinator agents act as “*an interface executing on the computing system*” by “coordinat[ing] communication” between clients. A coordinator agent is assigned a specific application and “coordinates communication” with clients, e.g., sending and receiving data over a TCP/IP network (purple arrow). Ex-1105, 10 (“Each client application ... communicates with the DB2 Server using ... a communication protocol such as TCP/IP”), 11 (“DB2 agents include coordinator agents and subagents, and are the most common type of DB2 processes which carry out the bulk of SQL processing on behalf of applications. DB2 assigns a coordinator agent with an application, and this agent coordinates the communication and processing for this application.”), 179 (illustrating how applications connect in Figure 5-13). Thus, a POSITA would have understood that DB2 server discloses “*communicating between a data store*” based on Chen’s description of how it conducts SQL processing for client applications. Ex-1103, ¶ 77.

As a more specific example of how data is supplied and consumed, Chen discloses that “provid[ing] the ability to **store and access XML documents**, and also to compose XML documents from the existing relational data or to decompose XML documents into relational data...[by] providing a set of new data types,

functions, and stored procedures.” Ex-1105, 318-19. This data is accessible from the Internet, allowing a user to build “e-business or Internet applications.”

Ex-1105, 2-3. Further, “[e]ach client application links with the DB2 client library, and communicates with the DB2 Server using shared memory (local clients), or a communication protocol such as TCP/IP (remote clients).” Ex-1105, 10. Because of their ability to supply and consume data, a POSITA would have understood that Chen’s client applications are “*applications that supply and consume data.*”

Ex-1103, ¶ 76; *see id.* ¶ 74 (citing Ex-1111, 31).

Accordingly, Chen discloses “*an interface executing on the computing system*” (the coordinator agents) “*for communicating between a data store*” (the UDB server) “*and applications that supply and consume data*” (the client applications). Ex-1103, ¶ 78; Ex. 1105, 3, 10-13.

Chen thus discloses limitation [1.1].

- iii. ***[1.2] a real time information director (RTID) executing on the computing system that transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data, the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand, the RTID further comprising:
 - a. *a real time information director (RTID) executing on the computing system****

The 575 Patent explains that a real time information director (RTID) is “an integrated set of services” that “is responsible for most or all data input and output to the real-time data store” and “receives and sends data in the form of eXtended Markup Language (XML) documents.” Ex-1101, 3:22-28.

Chen discloses a database management system for “storing ... XML documents in a centralized repository, as well as the ability to quickly search for information or to trigger automatic data interchange when a particular action occurs.” Ex-1105, 318-19. The system makes data accessible for supply and consumption by means of the interface. Ex-1105, 318-19. Chen explains that this is executed on the UDB Server by way of DB2 XML Extender, which “is responsible for most or all data input and output to the real-time data store” and “receives and sends data in the form of eXtended Markup Language (XML) documents.” Ex-1101, 3:22-29; Ex-1103, ¶ 84. Because the DB2 XML Extender sends and receives XML documents while being responsible for this data in the data store, it

discloses an “RTID” as described in the 575 Patent. Ex. 1103, ¶ 84. For the same reasons described regarding [1.1], and the additional reasons described below, the DB2 System, which includes the UDB Server and DB2 Extender, “*execute[s] on a computing system.*” Ex-1103, ¶ 85; Ex-1105, 2.

Accordingly, Chen’s UDB Server incorporating DB2 XML Extender discloses “*a real time information director (RTID) executing on the computing system.*” Ex-1103, ¶ 86; Ex-1105, 318-19.

b. transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata

Chen discloses that DB2 XML Extender on UDB Server (the “RTID”) “*transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata.*” In particular, Chen teaches that server-side activity (such as the composition and decomposition of XML documents described below) “is controlled by Engine Dispatchable Units (EDUs).” Ex-1105, 11. This includes DB2 agents and subagents on the UDB Server that carry out the communication and processing for data to and from clients (e.g., the “client applications” in Chen’s Figure 1-2). Ex-1105, 11. Thus, the UDB Server incorporating the DB2 XML Extender (“RTID”) performs the transformations described below “*for supply, consumption, or both by the applications.*” Ex-1103, ¶¶ 87, 94; Ex-1105, 11.

Specifically, Chen discloses that the UDB Server, including DB2 XML Extender, “*transforms data ... under direction of polymorphic metadata.*” Chen discloses that “[w]ith DB2 XML Extender you can either store the entire XML document in DB2 as a user-defined data type or map the XML content into columns in DB2 tables.” Ex-1105, 319. Chen further discloses transformation of data through the XML Collections method:

You can populate the columns of one or more DB2 tables starting from an XML document. The tags in the XML document and the XML document itself is not stored in DB2; **only the values of the elements and attributes you are interested in are stored in DB2. The mapping between elements and attributes in the XML document and the DB2 tables is given by the DAD file.**

Ex-1105, 319-20. Chen further explains that:

Each database includes a set of system catalog tables, which describe the logical and physical structure of the data. DB2 creates and maintains an extensive set of system catalog tables for each database. **These tables contain information about the definitions of database objects, such as user tables, views, and indexes, as well as security information about the authority that users have on these objects.**

Ex-1105, 10.

Chen's transformation of the XML document into a new data type so only the relevant values are stored based on user-defined datatypes (e.g., the mapping stored in the DAD file) is an implementation of polymorphism, as the user-defined datatypes represent classes that determine the different numbers of types of attributes relevant in the document. Ex-1103, ¶ 91 (citing Ex-1105, 319-20 ("DB2 XML Extender also provides you with a mapping scheme called Document Access Definition (DAD), which is a file used to map the XML document to relational data. The DAD is an XML formatted document which allows you to associate XML document structure to a DB2 database when using either XML Columns or XML Collections. The structure of DAD files is different when using an XML Column or an XML Collection.")).

Chen also teaches using a Document Type Definition (DTD) repository to "validate the XML documents you are storing within the DB2." Ex-1105, 320. "When a database is enabled for XML, a DTD reference table called DTD_REF is created. Each row of the table contains a DTD with additional metadata information about it." Ex-1105, 320. A POSITA would have understood this DTD repository to define classes by document type for extracting metadata into the database, which is a form of polymorphism. Ex-1103, ¶¶ 92-93.

Thus, the above disclosures in Chen regarding DAD files and DTD repositories teach polymorphic metadata as identified in the 575 Patent: in "the

illustrative embodiment,” “[p]olymorphism enables different metadata objects to have different numbers and types of attributes, determined by class,” and “[b]ehaviors can be defined that are specific to certain classes of objects, and the attributes specific to those behaviors need only be specified in those classes of objects.” Ex-1101, 13:1-12; Ex-1103, ¶¶ 92-93.

Chen’s DB2 can also “extend the capabilities of the database to meet your specific organizational requirements,” including supporting more advanced applications involving multimedia data and “new XML, spatial and high-speed searching extenders are available, along with object-relational extensions to the SQL language.” Ex-1105, 2. In doing so, Chen discloses the use of Java “to create DB2 objects, such as stored procedures and user defined functions.” Ex-1105, 186.

Further, Chen’s use of Java as well as XML is polymorphic, as admitted by the 575 Patent. Ex-1101, 36:25-26 (“As an object-oriented language, Java supports polymorphism.”); 36:48-49 (“Alternatively, metadata can be represented in eXtended Markup Language (XML), which is also polymorphic.”); Ex-1105, 185-86, 318. The above disclosures describing extracting metadata based on classes defined in DAD and DTD files, and implementing meta data extraction using Java, are also consistent with Patent Owner’s position in a parallel proceeding that “polymorphic metadata” means “metadata defined through a polymorphic

mechanism, such as hierarchal inheritance. Ex-1103, ¶ 92; Ex-1109, ¶ 130; Ex. 1105, 319-20; Ex-1111, 293, 412.

c. that defines a security model and data integrity rules for application to the data

Chen discloses that the polymorphic metadata (e.g., the DAD files and the DTD repositories) “*defines a security model and data integrity rules for application to the data.*” Chen explains that the DTD repositories and DAD files can provide for “[d]ocument validation.” Ex-1105, 320. This document validation involves the DB2 XML Extender using DTD to store incoming data and DAD to map the XML document to relational data. Ex-1105, 320. A POSITA would have understood that the metadata would include authentication information and would utilize encryption based on the mapping scheme. Ex-1103, ¶ 99; Ex-1105, 319-20. Accordingly, the DTD repositories and DAD files, separately and in combination, “*defines a security model and data integrity rules for application to the data*” (e.g., extracting the data from a document or exporting the data to a document) of the data. Ex-1103, ¶ 99.

Chen also describes “a combination of external security services and internal access control information to secure the information kept in the database.” Ex-1105, 211. “[T]he user can [be] specific in table design to ensure the data integrity. The database manager enforces the constraints. DB2 contains these four types of constraints.” Ex-1105, 173. These constraints are data integrity rules. Ex-

1103, ¶ 100; Ex. 1111, 143 (defining “data integrity” as “accuracy of data and its conformity to its expected value”). One constraint is a unique constraint: “rules that forbid duplicate values in one or more columns within a table.” Ex-1105, 173. Another type is a Referential Integrity (RI) constraint: “logical rule[s] about values in one or more columns in one or more tables.” Ex-1105, 173. The DAD files and DTD repositories in Chen are thus one manner of enacting these table designs to “define a security model and data integrity rules for application to the data.” Ex-1105, 173, 211; Ex-1111, 143.

d. the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand

Chen discloses that the DB2 XML Extender (the RTID) can “*dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand.*” Chen teaches that DB2 XML Extender can “compose XML documents from the existing relational data” and “decompose XML documents into relational data.” Ex-1105, 318. Chen further teaches that server-side activity (such as the composition and decomposition of XML documents described above) “is controlled by Engine Dispatchable Units (EDUs).” Ex-1105, 11. This includes DB2 agents and subagents that carry out the communication and processing for data to and from clients. *See id.* When performing these composition and decomposition tasks, a POSITA would have understood these DB2 agents to be

functioning as the “*assemblers*” and “*disassemblers*” of the 575 Patent by using the appropriate DAD file or DTD repositories. Ex-1103, ¶¶ 103-05.

Further, Chen discloses that the EDUs (software agents comprising the assemblers and disassemblers) are “implemented as threads in a single process on Windows-based platforms and as processes on UNIX.” Ex-1105, 11. A POSITA would have understood that processes (e.g., software agents) for serving client requests over a network (such as the processes and threads described in Chen) are dynamically instantiated, created, and cached for reuse, including to serve concurrent client application requests (e.g., through forking). Ex-1103, ¶ 107 (citing Ex-1111, 221 (defining “fork”)). Chen thus discloses that the RTID (the UDB Server, including DB2 Extender) will “*dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand.*”

Chen thus discloses limitation [1.2].

- iv. ***[1.3] at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and***

As described above regarding limitations [1.2b] and [1.2d], Chen discloses “*disassemblers defined according to polymorphic metadata.*”

Chen explains these “*disassemblers*” can parse inbound data documents (XML documents) into multiple records in preparation for insertion into metadata tables (DB2 tables):

You can **populate the columns of one or more DB2 tables starting from an XML document**. The tags in the XML document and the XML document itself is not stored in DB2; only the values of the elements and attributes you are interested in are stored in DB2. The mapping between elements and attributes in the XML document and the DB2 tables is given by the DAD file.

Ex-1105, 319. The data forms multiple records before its inclusion in tables, i.e., “[y]ou can work with the data shredded (or decomposed) from the XML document as you would work with any relational data in DB2.” Ex-1105, 320, 326 (“Using the MQ XML stored procedures, you can retrieve an XML document from a message queue, decompose it into untagged data, and store the data in DB2 tables. You can also compose an XML document from DB2 data and send the document to an MQSeries message queue.”). Ex-1103, ¶ 111.

Accordingly, Chen’s disclosure of extracting the value of certain fields from an XML document received from a client application and inserting those fields into one or more DB2 tables without storing the XML document itself would be understood to mean using “*at least one disassembler*” (the DB2 agent) “*defined according to the polymorphic metadata*” (mappings in the DAD file or DTD repository) “*that disassembles inbound data documents into multiple records*” (extracting the values from the relevant elements and attributes) “*in preparation for*

insertion into metadata tables” (preparing to populate the columns of DB2 tables with the extracted elements and attributes). Ex-1103, ¶ 112; Ex. 1105, 319, 326.

Chen thus discloses this limitation.

- v. ***[1.4] at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption or both.***

As described for [1.1] and [1.2] and for the additional reasons described below, Chen discloses “*assemblers defined according to polymorphic metadata . . . that transform data for supply, consumption, or both.*”

Chen discloses software agents (DB2 agents) using DAD files or DTD repositories as *assemblers* to obtain content from a database table and transform it into a new format for supply or consumption by the client application, where the relationship between the stored input data and the data outputted to the new format is defined by the DAD file (i.e., according to polymorphic metadata). *See supra*, and VIII.A.1.iii; Ex-1105, 320 (“DB2 XML Extender also provides you with a mapping scheme called Document Access Definition (DAD), which is a file used to map the XML document to relational data. The DAD is an XML formatted document which allows you to associate XML document structure to a DB2 database when using either XML Columns or XML Collections.”).

Chen explains that the assembler can assemble outbound data documents (XML documents) from a selected table (DB2 tables):

You can create an XML document using the data in columns of one or more DB2 tables. DB2 XML Extender composes an XML document using the data stored in columns of DB2 tables. Here again, the mapping between data in the DB2 tables and the composed XML document is done using the DAD file.

Ex-1105, 320, 326 (“Using the MQ XML stored procedures, you can retrieve an XML document from a message queue, decompose it into untagged data, and store the data in DB2 tables. You can also compose an XML document from DB2 data and send the document to an MQSeries message queue.”). Thus, Chen discloses “*at least one assembler defined according to the polymorphic metadata*” (the DB2 agent applying the definitions on the DAD file or DTD repository) “*that assembles outbound data documents*” (e.g., the XML document that XML Extender is composing) “*from at least one record selected from at least one metadata-selected table*” (using the data in columns of one or more DB2 tables based on the definitions in the DAD file or DTD repository) “*whereby data is transformed for supply, consumption or both*” (the XML document is created by XML Extender for supply to, and/or consumption by, the client application). Ex-1103, ¶¶ 115-17; Ex-1105, 320, 326.

Accordingly, Chen discloses this limitation.

2. Claim 2

- i. *[2.0] The data services handler according to claim 1 further comprising:*

See [1.0a].

- ii. *[2.1] a data mapper collecting data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.*

As discussed for [1.3], Chen discloses that XML Extender includes software agents (disassemblers) that extract data from incoming files and insert that data into DB2 tables based on definitions in a DAD file and/or DTD repository. Chen explains that these incoming files may be XML documents of varying formats (i.e., each with different attribute and element tags and definitions) based on the definitions in the corresponding DAD file and/or DTD repository. Ex-1105, 319-20 (“The mapping between elements and attributes in the XML document and the DB2 tables is given by the DAD file.”).

Chen discloses that XML files have diverse formats—i.e., the format of an XML file is determined by the author based on the use of tags in the documents:

XML is a meta-markup language and is used for creating your own markup languages. Using XML, you can define the tags for your markup language. XML tags are used to

describe the contents of the document. This means that any type of data can be defined easily using XML.

Ex-1105, 318.

Chen discloses that XML documents can be stored in diverse ways:

With DB2 XML Extender you can either store the entire XML document in DB2 as a user-defined data type or map the XML content into columns in DB2 tables. DB2 XML Extender can also store the XML document as an external file in the file system with a pointer to that file stored in the DB2 database so that you can keep working with your legacy flat files but benefit from the advantages offered by DB2 at the same time.

Id., 319.

A POSITA would have understood disclosures like these in Chen to describe “a data mapper” (e.g., XML Extender software agents) “collecting data from a plurality of diverse applications” (e.g., receiving files from a plurality of client applications) “that can supply data in multiple diverse external document formats” (e.g., the client applications can send a variety of differently formatted XML files, with varying attributes and elements) “and mapping the supplied data to a predetermined unified data model” (e.g., extracting the desired attributes and elements from the XML file to corresponding entries in DB2 tables) “according to the polymorphic metadata” (e.g., determining where the elements and attributes

extracted from the XML file should be stored in the DB2 tables based on the corresponding definitions in the DAD files and/or DTD repositories). Ex-1103, ¶¶ 121-24; Ex. 1105, 318-20.

3. Claim 4

- i. ***[4.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[4.1] the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.***

As discussed for [1.3] and [1.4], Chen teaches a “*RTID using ... descriptive polymorphic metadata to build both assemblers and disassemblers.*” Ex-1105, 319-20; Ex-1103 ¶¶ 128-29. Chen further teaches using the “*same descriptive polymorphic metadata*” because it teaches using the same DAD file (the “polymorphic metadata”) to both populate DB2 tables from a provided XML file (disassembler) and to create an XML document based on information stored in the DB2 tables (assembler):

You can populate the columns of one or more DB2 tables starting from an XML document. ... **The mapping between elements and attributes in the XML document and the DB2 tables is given by the DAD file.**

You can create an XML document using the data in columns of one or more DB2 tables. ... **Here again, the**

mapping between data in the DB2 tables and the composed XML document is done using the DAD file.

Ex-1105, 319-20. The DAD file “is an XML formatted document which allows you to associate XML document structure to a DB2 database.” Ex-1105, 320. A POSITA would have understood associating an XML document structure with a DB2 database, such that XML documents having that structure can be extracted into particular columns of one or more tables, to teach or suggest using *the same descriptive polymorphic metadata* (e.g., the corresponding DAD file) ... *for a same document class* (i.e., for XML documents having the structure addressed in the corresponding DAD file). Ex-1103, ¶ 129, 319-20.

A POSITA would have understood the disclosures above to teach “*the RTID*” (UDB Server incorporating XML Extender) “*using a same descriptive polymorphic metadata*” (the DAD file) “*to build both assemblers and disassemblers for a same document class*” (to instantiate and/or configure from the DAD file the software agents that extract information from an XML document to the DB2 table and the software agents that compose corresponding XML documents from the information stored in the DB2 table). Ex-1103, ¶¶ 129-30.

4. Claim 6

- i. [6.0] *The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[6.1] a logic that represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.***

Petitioner has applied Patent Owner's position that "logic" means "software." *See supra* Section V. Chen discloses that the RTID (e.g., the UDB Server, including DB2 XML Extender) and the assemblers, and disassemblers (Chen's DB2 agents) are software. Ex-1105, 11, 318-19. A POSITA would have thus understood Chen to teach "*a logic*." Ex-1103, ¶ 133; Ex-1105, 11, 318-19.

Chen teaches that a logic (e.g., DB2 XML Extender) "*represents metadata as polymorphic objects ... according to a predetermined object class*." As discussed for limitation [4.1], Chen teaches that DBL2 XML Extender instantiates software agents ("*assemblers*" and "*disassemblers*") based on polymorphic metadata (DAD files) defined based on the structure of specific XML documents. A POSITA would have understood the DAD files to be "*object classes*" and the software agents instantiated based on the DAD files to be "*polymorphic objects*" and accordingly would have understood Chen to teach a logic that "*represents metadata as polymorphic objects* (e.g., instantiates the assemblers and disassemblers) ... *according to a predetermined object class* (based on the definitions in the DAD file for the specific XML structure that the

assemblers/disassemblers are to process).” Ex-1103, ¶¶ 134-35; Ex-1111, 100 (defining “class”), 411 (defining “polymorphism”).

Chen teaches “*enabling individual metadata objects to have distinct numbers and types of attributes, with the behaviors and attributes being defined specific to object class.*” As discussed for limitation [4.1], Chen describes using a DAD file (the “polymorphic metadata”) to both populate DB2 tables from a provided XML file (disassembler) and to create an XML document based on information stored in the DB2 tables (assembler). Ex-1105, 319-20. The DAD file “is an XML formatted document which allows you to associate XML document structure to a DB2 database.” Ex-1105, 320.

A POSITA would have understood this to teach “*enabling individual metadata objects to have distinct numbers and types of attributes* (e.g., defining in the DAD file the specific attributes that should be extracted and the corresponding columns to which they should be extracted), and *the behaviors and attributes being defined specific to object class* (e.g., the description in the DAD file defining how the attributes should be extracted and populated for a specific XML document structure).” Ex-1103, ¶ 136; Ex-1105, 319-20.

5. Claim 7

- i. ***[7.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[7.1] a logic that represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.***

As discussed for limitation [6.1], Chen teaches “*a logic that represents metadata as polymorphic objects.*”

Chen discloses that the DAD files define the mappings of elements and attributes that will be applied to an XML document structure. Ex-1105, 319-20. As explained for [1.2], the DAD files are “*polymorphic metadata.*” As explained for [1.3] and [1.4], these DAD files are used to instantiate DB2 agents (e.g., assemblers and disassemblers). As explained for [33.1], the DB2 Agent is initiated using a DAD file associated with the structure (i.e., XML Schema) of the XML document—i.e., different DAD files may be associated with different XML document structures.

A POSITA also would have understood the DB2 Agents are “*stateless ... objects*” because they are static—i.e., the mapping in the DAD files define the operation of the DB2 Agents, such that they are not modified after they are initiated (they are “*permanent after initial construction*”). Ex-1103, ¶ 141; Ex-

1101, 13:20-25 (“In an illustrative system, metadata objects are stateless. The metadata objects are not modified after initial construction, a characteristic attained, for example, in the Java language by declaring all attributes of metadata classes as final. Such attributes are set only during the construction of the metadata object.”). Chen thus teaches that the DB2 agents instantiated from DAD files are “*metadata as stateless polymorphic objects that are permanent after initial construction.*”

As explained for [1.3], Chen teaches that the RTID (UDB Server, including XML Extender) uses XML documents as an input for DB2 agents (disassemblers) that parse the XML document’s elements and attributes into the database. Because these XML documents are inputs to the DB2 agent (which is instantiated based on the DAD file), and because the XML documents are used to initiate the DB2 agents, a POSITA would have understood the input XML documents to be stateful metadata classes. Ex-1103, ¶ 142; Ex-1101, 5:1-6 (“The metadata classes are stateless and are called by the core functionality of the real time information director (RTID) whenever the real time information director is operating on relevant data. Stateful data objects are passed to the relative metadata methods as parameters.”).

Chen thus discloses “*a logic (UDB Server, including XML Extender) that represents metadata as stateless polymorphic objects that are permanent after*

initial construction (that instantiates DB2 agents as assemblers based on the DAD files that are polymorphic metadata), *whereby methods of stateful metadata classes* (e.g., the input XML document) *delegate operation to the stateless polymorphic metadata objects* (indicate to the DB2 agent via the XML schema which DAD file should be used to assemble or disassemble the XML document).”

6. Claim 10

- i. ***[10.0] The data services handler according to claim 1 wherein:***

See [1.0a].

- ii. ***[10.1] the real time information director receives and publishes data documents having a content and a format defined by document metadata.***

As described for [1.2], Chen discloses “*the real time information director* (UDB Server, including DB2 XML Extender) *receives ... data documents having a content and a format* (XML documents) *defined by document metadata* (e.g., the values and attributes transformed through the XML Collections method).”

As discussed for [1.4], Chen teaches assemblers that create “*data documents*” (e.g., XML documents) from stored information. Ex-1105, 319-20.

Chen further teaches that “[t]he XML document can also be retrieved and used by your XML application.” *Id.* A POSITA would understand this to teach that XML Extender “*publishes*” the XML document when it is retrieved. Ex-1103, ¶ 147.

7. Claim 11

- i. ***[11.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[11.1] a logic that reads a schema of tables identified in metadata; and***

As explained for [1.3], Chen teaches “a logic” (DB2 agent) that reads a DAD file (polymorphic metadata) that specifies into which database fields an XML document’s attributes should be inserted. Ex. 1103, ¶ 152; Ex-1105, 319. Because the DAD file identifies fields in the database, POSITA would have understood this to teach that DB2 agent “reads a schema of tables identified in metadata.” Ex-1103, ¶ 153; Ex-1111, 465 (“schema defines aspects of the database, such as attributes (fields)”).

- iii. ***[11.2] a logic that automatically generates and caches prepare SQL statements from a content metadata portion of document metadata according to the schema.***

As explained for [1.3], Chen teaches “a logic (DB2 agent) that inserts data from an XML document into fields of a database based on a *content metadata portion of document metadata* (e.g., the attributes of the XML document) *according to the schema* (e.g., based on the definition of the database tables and fields in the DAD file).”

Chen further discloses using SQL with XML Extender. Ex. 1105, 318 (“With DB2 XML Extender, you can use SQL as the main access method to your XML documents, which makes your task much easier.”). Chen discloses creating SQL statements dynamically. Ex-1105, 181 (“Dynamic SQL statements are those that your application builds and executes at run time.”). Chen discloses caching dynamic SQL statements. Ex-1105, 156 (“For dynamic SQL, db2expln examines the sections in the SQL cache.”), 298 (“DB2 will now have a specific amount of pages that will cache SQL statements which will speed up execution if the same query is required more than once.”).

A POSITA would have understood these disclosures to teach or suggest “*a logic (e.g., DB2 agent) that automatically generates (e.g., builds a dynamic SQL statement) and caches prepare[d] SQL statements (e.g., caches the SQL statements to speed up execution) from a content metadata portion of document metadata (e.g. based on the attributes of the XML document to be inserted into the database) according to the schema (e.g., based on the table and fields in the database identified in the DAD file).*” Ex-1103, ¶¶ 155-57; Ex-1105, 156, 181, 298.

8. Claim 12

- i. ***[12.0] The data services handler according to claim 1 further comprising:***

See [1.0a].

- ii. ***[12.1] a servlet that invokes the real time information director synchronously when an inbound data document is delivered via HyperText Transport Protocol (HTTP):***

Chen discloses web enablement: “DB2 UDB is integrated with Web technology so that data can be easily accessed from the Internet” Ex-1105, 2. This includes connecting by HTTP. Ex-1105, 146. Chen teaches integration with the Apache web server using PHP to script servlets for interacting with DB2. Ex-1105, 112 (“the focal point of our application environment is an Apache Web Server which can interact with a user’s web browser and provide DB2 connectivity via the PHP scripting”), 150; Ex-1111, 475 (“Servlets perform lightweight web services...”). Within the context of the 575 patent, HTTP is an example of synchronous access. Ex-1103, ¶ 162 (citing Ex-1111, 506 (defining “synchronous communication”); Ex-1101, 3:31-34 (“The real time information director is accessed synchronously via HyperText Transport Protocol (HTTP)”). Chen further discloses that XML documents can be stored in and parsed by XML Extender. Ex-1105, 319-20.

Given the above disclosures, a POSITA would have understood Chen to teach or suggest “*a servlet (e.g., a PHP script) that invokes the real time information director synchronously (e.g., connects to XML Extender over the internet) when an inbound data document is delivered via HyperText Transport*

Protocol (HTTP) (e.g., when a document is transmitted via the Apache web server).” Ex-1103, ¶¶ 162-63; Ex-1105, 2, 112, 146, 319-20.

iii. ***[12.2] a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS);***

Chen discloses the use of Java to connect to DB2. Ex-1105, 150 (“This is the capability offered by Java applets and applications, which allows them to connect directly to DB2 without DB2 client code.”). For Java connectivity, Chen discloses that “DB2’s Java support includes JDBC, a vendor-neutral dynamic SQL interface that provides data access to your application through standardized Java methods.” Ex-1105, 182. A POSITA would have understood that one standard Java method for providing data to applications was Java Message Service (JMS). Ex-1103, ¶ 165; Ex-1112, 14 (“JMS provides a common way for Java programs to create, send, receive and read an enterprise messaging system’s messages.”). A POSITA would have also understood that JMS provides for asynchronous communication. Ex-1103, ¶ 165; Ex-1112, 64 (describing “Asynchronous Delivery” for JMS); Ex-1101, 3:31-34 (“The real time information director is accessed ... asynchronously via Java Message Service (JMS).”). Chen further discloses that XML documents can be stored in and parsed by DB2 via XML Extender. Ex-1105, 319-20.

Given the above disclosures, a POSITA would have understood Chen to teach or suggest “*a logic (e.g., a Java application) that invokes the real time*

information director asynchronously (the connects to XML Extender for asynchronous communication) when an inbound data document is delivered via Java Message Service (JMS) (when an XML document has been uploaded to the java application using JMS)." Ex-1105, ¶¶ 165-66.

Accordingly, Chen teaches or suggests this limitation.

- iv. ***[12.3] a servlet that invokes the real time information director synchronously when an outbound data document is requested via HyperText Transport Protocol (HTTP); and***

As described above regarding [12.1], Chen teaches or suggests "*a servlet that invokes the real time information director synchronously ... via HyperText Transport Protocol (HTTP).*"

Further, as described for [10.1], Chen teaches publishing XML documents based on information stored in DB2 tables. Given Chen's teachings of using a servlet to pass data synchronously between a client and the DB2 database over the internet (see [12.1]), a POSITA would have understood Chen to teach or suggest providing for a client to request such an XML document be published via the servlet—i.e., publishing the created XML document ("*an outbound data document*") when it "*is requested*" (by the "*servlet*"). Ex-1103, ¶ 169.

- v. ***[12.4] a logic that invokes the real time information director asynchronously when an outbound data document is requested via Java Message Service (JMS).***

As discussed for [12.2], Chen teaches “a logic that invokes the real time information director asynchronously ... via Java Message Service (JMS). As discussed for [12.3], Chen teaches *an outbound data document*.”

Given Chen’s further teachings of using a Java application communicating via JMS to interface with DB2 ([12.1]), a POSITA would have understood Chen to teach or suggest that the Java application would provide for requesting an XML document be created and published via the Java application using JMS—i.e., publishing the created XML document (“*an outbound data document*”) when it “*is requested*” (by Java application using JMS). Ex-1103, ¶ 172.

9. Claim 13

- i. ***[13.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[13.1] a subscription handler that responds to a triggering event by publishing a document according to content and format specified by Document Metadata, and sending the document to a designated queue or topic.***

Chen discloses “messaging models to distribute XML data and documents,” including a publish/subscribe communication paradigm. Ex-1105, 261. Chen explains that for publish/subscribe, “[o]ne or more publishers send a message to a

publication service which distributes the message to interested subscribers.” *Id.*

Because Chen’s publication service distributes messages to interested subscribers, a POSITA would have understood it to be “*a subscription handler.*” Ex-1103, ¶¶ 175-76.

A POSITA would have understood that a publish/subscribe (or post-and-subscribe) communication paradigm involves transmitting messages to a subscriber when some event has occurred. Ex-1105, ¶ 176; Ex-1113, xxxix, 40. Accordingly, a POSITA would have understood Chen to teach or suggest that “*a subscription handler*” “*responds to a triggering event by publishing a document*” (the XML document). Ex-1103, ¶ 176.

Further, Chen discloses implementing a messaging queue for transmitting XML documents. Ex-1105, 326. A POSITA would understand this to teach or suggest constructing the XML document and then sending it to a message queue. Ex-1103, ¶ 177; Ex-1114, 287 (describing “[c]onstruct[ing] an XML document from data that is stored in the XML collection tables specified in the DAD file, and send[ing] the XML document to an MQ message queue”). A POSITA thus would have understood Chen to disclose “*sending ... to a designated queue or topic.*” Ex-1103, ¶¶ 177-79; Ex-1105, 287.

As explained for [1.4], which describes assemblers that generate XML documents from database entries using DAD files (polymorphic metadata), Chen

teaches that the XML document is generated “*according to content and format specified by Document Metadata*” (the DAD file). Ex-1103, ¶ 178.

Chen thus teaches or suggests this limitation.

10. Claim 14

- i. ***[14.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[14.1] a security manager that identifies an agent on behalf of whom information is queried or updated in a document request, dynamically assigns an internal unique identifier to consumers and agents upon registration;***

Chen discloses that “[t]o access a database server, a user must be authenticated either on the client or the server.” Ex-1105, 211. “The database manager configuration file contains configuration parameters to define how clients are authenticated. Once a user is authenticated and has access to a DB2 instance, DB2 continues to provide security by restricting the user to the set of authorities and privileges that the user is assigned to.” Ex-1105, 211. Because Chen’s “database manager” contains configurations for restricting client access, a POSITA would have understood it to be “*a security manager*.” Ex. 1103, ¶ 182.

Chen also discloses that “a client’s user ID and password are sent to the server for authentication.” Ex-1105, 212. Authentication must be complete before the client can access the database. *Id.*, 211 (“To access a database server, a user

must be authenticated either on the client or the server.”). Based on this a POSITA would have understood that Chen discloses identifying the client user who is attempting to access the database before the client can undertake tasks on the database—e.g., querying or updating information. Ex-1103, ¶ 184. A POSITA would have understood that the client could represent anyone, including a “*user*” or an “*agent*” of a user. Ex-1103, ¶ 184. Accordingly, a POSITA would have understood Chen to teach or suggest “*identif[ying] an agent on behalf of whom information is queried or updated.*” Ex-1103, ¶ 184; Ex-1105, 211-12.

Chen discloses the ability to retrieve XML documents from the database. Ex-1105, 318. A POSITA would have understood requesting the retrieval of the XML document to mean “*performing a document request.*” Ex-1103, ¶ 185.

Chen teaches or suggests maintaining “user tables” (Ex-1105, 10) and that each table should contain at least one column of unique values that function as “primary keys.” Ex-1105, 173, 176 (describing “enforcing the uniqueness of a record by using a primary key”). A POSITA would have understood that a primary key would be generated automatically when a new entry was made in the user table. Ex-1103, ¶ 186. A POSITA would accordingly have understood Chen to teach or suggest “*dynamically assign[ing] an internal unique identifier (e.g., automatically assigning a unique primary key in the user database) to consumers and agents (users) upon registration (when entered into the user table).*” *Id.*

- iii. ***[14.2] and the real time information director grants access to individual documents based on security policies according to a relationship between an agent, a consumer affected by the agent, and permissions granted by the consumer.***

Chen discloses a security system that “uses a combination of external security services and internal access control information to secure the information kept in the database.” Ex-1105, 211. Chen explains that “DB2 continues to provide security by restricting the user to the set of authorities and privileges that the user is assigned.” Ex-1105, 211. Chen discloses that “[o]nce the user is authenticated and attached to the DB2 instance, the user will be able to use DB2 objects based on the authorities and privileges that are given.” Ex-1105, 213.

Chen further discloses the use of “authorization groups.” Ex-1105, 197. Because a POSITA would have understood that members of an authorization group would be able to act on each other’s behalf (i.e., as agents) regarding group resources, and because a POSITA would understand that a member of the group could be anyone (e.g., a consumer or an agent of a consumer), a POSITA would have understood Chen to teach or suggest “*grant[ing] access to individual documents based on security policies* (e.g., granting access to retrieve and XML document based on whether a the requested has permission) *according to a relationship between an agent a consumer affected by the agent* (determining whether the requested has permission based on whether the requester (e.g., the

agent) is in the appropriate group), *and permissions granted by the consumer* (e.g., whether the consumer has permitted the requested XML document to be retrieved by the group).” Ex-1103, ¶ 190.

Accordingly, Chen discloses this claim limitation.

11. Claim 17

- i. ***[17.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[17.1] an audit log handler that responds to changes in the data store by compiling and storing information based on metadata that determines content and detail of the compiled information.***

Chen teaches that UDB Server maintains transaction logs that track changes in a database. Ex-1105, 336. Chen discloses that this transaction log is used to “reconcile discrepancies.” *Id.* Because the transaction log compiles and stores information that UDB Server uses to identify discrepancies, a POSITA would have understood Chen to disclose “*an audit log handler (UBS Server) that responds to changes in the data store (reacts to changes to the database) by compiling and storing information (transcribing changes into the transaction log).*” Ex-1103, ¶ 195; Ex-1111, 42 (defining “audit” as “an examination of ... activities, ... to determine how efficiently the overall system is performing, especially in terms of ensuring the integrity ... of data”).

As the transaction log tracks database changes, and Chen teaches extracting metadata from XML documents ([1.3] and [1.4]), a POSITA would have understood Chen to teach or suggest compiling and storing information “*based on the metadata*” in the transaction log that was parsed from the XML documents. Ex-1103, ¶¶ 195-96. As the changes are being transcribed in the transaction log (including the metadata), a POSITA would have understood that the extracted metadata “*determines content and detail of the compiled information.*” Ex-1103, ¶ 196.

Accordingly, Chen teaches this limitation.

12. Claim 18

- i. ***[18.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[18.1] a data cleanser that receives data elements identified by Document Metadata, normalizes the data elements, queues source and destination information for detection of duplicates, and cross-references duplicates in the data store.***

As discussed for [1.3] and [1.4], Chen discloses UDB agents that extract elements and attributes from XML documents based on the definitions in DAD files (polymorphic metadata). A POSITA would have understood such UDB agents to “*receive[] data elements* (e.g., attributes and elements from the XML document) *identified by Document Metadata* (DAD files).”

Chen teaches applying “[u]nique constraints,” which “are rules that forbid duplicate values in one or more columns within a table.” Ex-1105, 173. “The database manager enforces the constraints.” *Id.* A POSITA would have understood that this suggests applying normalization, a process that minimizes the inclusion of duplicative information and improves security and performance of the database. Ex-1103, ¶ 200; Ex-1111, 368 (defining “normalization”) (“In database management, to apply a body of techniques to a relational database in order to minimize the inclusion of duplicate information. Normalization greatly simplifies query and update management, including security and integrity considerations, although it does so at the expense of creating a larger number of tables.”).

As a result, a POSITA would have found it obvious based on Chen and the POSITA’s education and experience to “*normalize[] the data elements*” (e.g., apply normalization techniques to the XML document elements and attributes), “*queue[] source and destination information for detection of duplicates*” (check for duplicate entries in the database), and “*cross-reference[] duplicates in the data store*” (create relationships in the database identifying the existing entry instead of creating a duplicate in the database). Ex-1103, ¶ 200.

Chen thus renders this limitation obvious.

13. Claim 19

- i. ***[19.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[19.1] an aggregation engine that computes aggregates in memory from data selected from the data store.***

Chen discloses a process that “contains a series of steps that perform a transformation and movement of data for a specific warehouse use.” Ex-1105, 316. Two steps of this process include “mov[ing] source data into the warehouse” and then “aggregate[ing]” and “summariz[ing]” the data “for warehouse use.” Ex-1105, 316-17. A POSITA would have understood that this “process” is an “*aggregation engine*” and that its computations take place “in memory” because it is a software process specifically designed to aggregate data. Ex-1103 ¶ 206; Ex-1105, 316-17. Accordingly, a POSITA would have understood Chen to teach or suggest “*an aggregation engine (the software process) that computes aggregates in memory (e.g., that summaries and aggregates the data) from data selected from the data store (e.g., from the data selected to be moved to the warehouse).*” Ex-1103, ¶ 206.

Accordingly, Chen teaches or suggests the limitation.

14. Claim 20

- i. ***[20.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[20.1] a real-time scoring engine.***

Chen discloses DB2 Intelligent Miner™ Scoring, which “provides scoring technology as a DB2 extender.” Ex-1105, 362. Chen further explains this extender “enables users to deploy data mining in **real-time applications.**” Ex-1105, 22.

From these disclosures, a POSITA would have understood DB Intelligent Miner to be a “*real-time scoring engine.*” Ex-1103, ¶ 210; Ex-1105, 362.

15. Claim 21

- i. ***[21.0] The data services handler according to claim 1 wherein the real time information director further comprises:***

See [1.0], [1.2].

- ii. ***[21.1] a business rules engine that defines a business object model capable of enabling access to data described by Document Metadata, performing computations on data relevant to business rules specified by the business object model, concluding a decision based on computation results, and executing the decision.***

Chen discloses that “DB2 UDB has particular strengths in supporting business intelligence applications,” including “support for . . . business rules.” Ex-1105, 3, 17. Chen discloses “data warehousing capabilities.” *Id.*, 314. “Data

Warehousing allows you to create stores of informational data, which is data that is extracted from the operational data and then transformed for decision-making.” *Id.*

Chen discloses that data warehouse operations include “[e]xtracting data from the operational database,” “[c]leaning the data,” “[p]erforming calculations to summarize the data,” and “[w]riting the summarized data to a target in a separate database.” *Id.* Chen further discloses that “[w]arehouse agents manage the flow of data between the data sources and the target warehouses” and that warehouse agents are software executing on common operating systems, like Linux and Windows. Ex-1105, 316.

Because Chen discloses “[e]xtracting data from the operational database” (Ex-1105, 136) and further discloses (*see* [1.2]) inserting information into databases by parsing attributes from XML documents based on DAD files (polymorphic metadata), a POSITA would have understood Chen to teach or suggest “*enabling access to data* (e.g., retrieving data from the operational database) *described by Document Metadata* (where the data was loaded into the database based on a DAD file).” Ex-1103, ¶¶ 215-17; Ex-1105, 136.

Because Chen discloses “[p]erforming calculations to summarize the data” and that the data warehouse maintains data “for decision making” (Ex-1105, 313), a POSITA would have understood Chen to teach or suggest “*performing computations* (e.g., calculating) *on data relevant to business rules* (e.g., on data

that needs to be summarized for decision making) and *concluding a decision* (e.g., undertaking decision making) *based on computation results* (e.g., based on the calculations that were performed and/or their summaries) and *executing the decision* (performing the decision making).” Ex-1103, ¶ 215; Ex-1105, 314.

Because Chen teaches or suggests that the warehouse agents performs the above steps, a POSITA would have understood a warehouse agent to be “*a business rules engine*” and that it “*defines a business object model capable of*” performing those steps. Ex-1103, ¶ 217.

Accordingly, Chen teaches or suggests this limitation. Ex-1105.

16. Claim 22

i. [22.0] *An article of manufacture comprising:*

Chen discloses a “UDB architecture” comprising servers with hard drives:

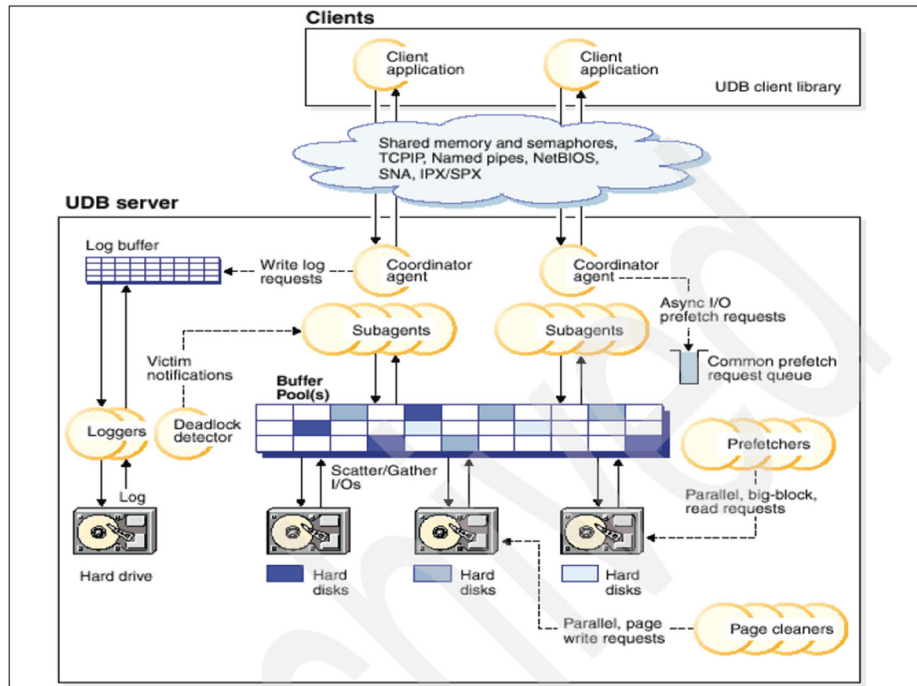


Figure 1-2 DB2 UDB architecture overview

Ex-1105, Fig. 1-2. That architecture is “an article of manufacture.” Ex-1103, ¶ 219.

- ii. ***[22.1] a controller-usable storage medium having a computer readable program code embodied therein for managing data services, the computer readable program code further comprising:***

Chen discloses that the UDB server, and by extension the UDB DB2 System, comprises hardware components, including disks and buffers. Ex-1105, Figure 1-2. A POSITA would have understood this to disclose “a storage medium.” Ex-1103, ¶ 222. Chen discloses that the UDB DB2 System “can be deployed on a variety of hardware and software platforms all the way from palm devices to massive parallel multiprocessor systems.” Ex-1105, 2. A POSITA would have understood this to teach that the UDB Server comprises “a controller-usable

storage medium.” Ex-1103, ¶ 222; Ex-1111, 132 (defining “CPU” as “control unit”).

Chen discloses that UDB Server comprises “*computer readable program code ... for managing data services.*” As discussed for [1.0], Chen’s DB2 System includes an interface comprising “a data services handler.” A POSITA would have recognized that interface as “*computer readable program code ... for managing data services.*” Ex-1103, ¶¶ 223-24.

iii. [22.2] a code causing the controller to transfer data among a data store and applications that supply and consume data;

As described for [1.1], Chen discloses the DB2 System, which includes UDB Server (“*a data store*”), Client Applications (“*applications that supply and consume data*”), and further discloses that UDB Server comprises DB2 agents that coordinate communications between (“*transfer data among*”) UDB Server and Client Applications. Ex-1103, ¶ 226. Given this, and because DB2 agents are software that executes on a processor (“*controller*”) of the DB2 System (see [22.1]), a POSITA would have understood Chen to teach “*a code causing the controller to transfer data (e.g., DB2 agents) among a data store (UDB Server) and applications that supply and consume data (Client Applications).*” *Id.*, ¶ 227-28.

- iv. ***[22.3] a code causing the controller to transform data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class;***

As discussed for [1.2], Chen teaches that UDB Server, including DB2 XML Extender, extracts attributes from XML documents based on DAD files that define relationships between the structure of specific XML documents and the tables and entries in the DB2 Database where the attributes extracted should be stored. A POSITA would have understood this to disclose “*a code causing the controller to (UDB Server, including DB2 XML Extender) transform data (extract the attributes from the XML document) for transfer (to store the attributes in the database).*” Ex-1103, ¶ 229.

As discussed for [6.1], Chen teaches the “*representation of metadata as polymorphic objects enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.*” Thus, Chen discloses this limitation.

- v. ***[22.4] a code causing the controller to dynamically instantiate, create, and cache for reuse at least one disassembler and at least one assembler according to demand;***

As described for [1.2], Chen teaches UDB Server, including DB2 XML Extender, to “*dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand.*” Because the UDB Server, including DB2 XML Extender, comprises software executing on a processor of the DB2 Server, a POSITA would understand Chen to disclose “*a code causing the controller to*” perform those actions. Ex-1103, ¶ 232.

- vi. ***[22.5] a code dynamically created by the controller as defined by polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and***

As discussed for [1.3], Chen discloses “*a code (a DB2 agent) defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables.*” Further, because Chen teaches that the DB2 agents are instantiated as threads or processes to serve client application requests (Ex-1105, 11), a POSITA would have understood the DB2 agents to be “*dynamically created*” by the UDB Server, including as needed to handle concurrent client requests. Ex-1103, ¶ 236; Ex-1111, 221 (defining “fork”).

- vii. ***[22.6] a code dynamically created by the controller as defined by polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table.***

As discussed for [1.4], Chen teaches at least one assembler (DB2 Agent) “defined [by] polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table.” As described for [22.5], Chen teaches the DB2 Agent is “code” that is “dynamically created by the controller.”

17. Claim 23

- i. ***[23.0] The article of manufacture according to claim 22 further comprising:***

See [22].

- ii. ***[23.1] a code causing the controller to transform data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.***

As described for [1.2], Chen teaches a DB2 agent “to transform data ... under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data.” As discussed for [22.2], Chen discloses the DB2 agent is “a code causing the controller” to perform do so.

18. Claim 24

- i. ***[24.0] The article of manufacture according to claim 22 further comprising:***

See [22].

- ii. ***[24.1] a code causing the controller to collect data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.***

As discussed for [2.1], Chen teaches DB2 agents that “collect[] data from a plurality of diverse applications that can supply data in multiple diverse external document formats and mapping the supplied data to a predetermined unified data model according to the polymorphic metadata.” As discussed for [22.2], Chen discloses the DB2 agents are “code causing the controller” to do so.

19. Claim 25

- i. ***[25.0] The article of manufacture according to claim 22 further comprising:***

See [22].

- ii. ***[25.1] a code causing the controller to represent metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful meta data classes delegate operation to the stateless poly morphic metadata objects.***

As discussed for [7.1], Chen teaches that XML Extender will “represent metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.” Because an XML Extender is software, it would also represent “a code causing the controller to do so.” Ex-1103, ¶ 258.

20. Claim 28**i. [28.0] *An enterprise system comprising:***

Chen describes a “DB2 UDB architecture” comprising servers that is “flexible enough to serve both structured and unstructured data management needs of a single user to very large enterprises (“*[a]n enterprise system*”).” Ex-1105, 1; Ex-1103, ¶ 261.

ii. [28.1] *a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers; and*

Chen discloses that “tables and indexes may be split across multiple database partitions,” including “**across multiple physical servers.**” Ex-1105, 7, 361. Chen discloses that “DB2 UDB can be accessed from many client workstations **over many networks.**” Ex-1105, 3. Dr. Almeroth explains that because DB2 UDB’s tables may span across several physical servers and DB2 UDB may be accessed over many networks, the servers and networks must be interlinked to facilitate communication, and thus Chen teaches or suggests “*a plurality of interlinked computer servers and networks.*” Ex-1103, ¶¶ 263-264.

Chen further explains that DB2 UDB can support access by “thousands of users.” Ex-1105, 2-3. Chen therefore discloses its system “*enabling multiple users to access ... the servers.*” Ex. 1103, ¶ 266.

Additionally, as described for [1.1], Chen teaches that users can access multiple client applications that are consuming/supplying data from/to the UDB

Server. Because Chen teaches a plurality of Client Applications consuming and supplying data, along with teaching that thousands of users can access the UDB Server, Chen teaches “*enabling multiple users to access applications and data on the servers.*” Ex. 1103, ¶ 266. Thus, Chen discloses this limitation.

- iii. ***[28.2] a real time information director (RTID) executing on at least one of the interlinked servers that manages information flow between a data store and applications that supply and consume data and dynamically instantiates, creates, and caches for reuse disassemblers and assemblers according to demand,***

As discussed for [1.2], Chen discloses “*a real time information director (RTID) (UDB Server, including XML Extender) executing on ... a server (the hardware comprising UDB Server).*” As described for [28.1], Chen discloses that UDB Server may comprise a plurality of “*interlinked servers.*” A POSITA would thus have understood Chen to teach “*a real time information director (RTID) executing on at least one of the interlinked servers.*” Ex-1103, ¶¶ 269-70.

As described for [1.2], Chen discloses that UDB Server includes communicate agents that facilitate communications between a data store (UDB Server) and applications that supply and consume data (Client Application). A POSITA would have understood this to teach that the UDB Server “*manages information flow between a data store and applications that supply and consume data.*” Ex-1103, ¶ 269.

As described for [1.2], Chen teaches that the RTID “*dynamically instantiates, creates, and caches for reuse disassemblers and assemblers according to demand.*” Ex-1103, ¶ 270. Thus, Chen discloses this limitation.

- iv. ***[28.3] the RTID that creates at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables and create at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption, or both.***

As discussed for [1.3], Chen teaches “*the RTID (UDB Server, including XML Extender) creates at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables.*”

As discussed for [1.4], Chen teaches “*the RTID (UDB Server, including XML Extender) creates at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption or both.*” Thus, Chen discloses this limitation.

21. Claim 31

- i. ***[31.0] The enterprise system according to claim 28 wherein:***

See [28].

- ii. ***[31.1] the RTID represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.***

As described for [6.1], Chen teaches “*the RTID (XML Extender) represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.*”

22. Claim 32

- i. ***[32.0] The enterprise system according to claim 28 wherein:***

See [28].

- ii. ***[32.1] the RTID represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.***

As discussed for [7.1], Chen discloses “*the RTID (UDB Server, including XML Extender) represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.*”

23. Claim 33

- i. ***[33.0] A data services handler according to claim 1 further comprising:***

See [1].

- ii. ***[33.1] at least one generator that is dynamically created by the RTID as defined by polymorphic metadata and generates an extended Markup Language (XML) Schema that describes format of the documents, the at least one disassembler, assembler, and generator that use a single descriptive polymorphic metadata.***

As discussed for [1.2], Chen discloses software agents (DB2 agents) using DAD files or DTD repositories as to map content between XML documents and databases. As also discussed for [1.2], DAD files are “*polymorphic metadata*” and DB2 agents are “*dynamically created*” as processes or threads. A POSITA would have understood an XML document to comprise “*an extended Markup Language (XML) Schema that describes format of the documents.*” Ex-1103, ¶ 287; Ex-1111, 579 (defining “XML Schema” as “[a] specification providing a common base for data description and validation in XML environments. ... It is also a standard for the description and encoding of data.”), Ex-1105, 318 (“XML stands for eXtensible Markup Language.”).

Accordingly, a POSITA would have understood Chen to teach or suggest a DB2 agent as “*a generator*” to generate the XML Schema that defines the format of the XML document based on the DAD file. Ex-1103, ¶ 287; Ex-1105, 318-20, Ex-1111, 579.

As discussed for [1.3] and [1.4], Chen also teaches each of assemblers (which assemble the content of XML documents from a database based on mappings in the DAD files) and disassemblers (which disassembles an XML

document by parsing it and storing its attributes in database tables based on mappings in the DAD files). A POSITA would thus have understood Chen to teach or suggest using the same DAD file (i.e., a DAD file with consistent mappings) for each of the assembler, disassembler, and generator that process XML documents that share the same structure (i.e., the same “class” of XML documents). Ex-1103, ¶ 287; Ex-1105, 320.

Chen thus teaches or suggests “*at least one generator (a DB2 agent that generates the XML Schema) that is dynamically created by the RTID (instantiated as a thread or process by UBS Server, including XML Extender) as defined by polymorphic metadata (based on the definitions in the DAD file) and generates an extended Markup Language (XML) Schema that describes format of the documents, the at least one disassembler (a DB2 Agent that parses an XML document to store its attributes in a database), assembler (a DB2 Agent that retrieves assets from the database to compile in a new XML document), and generator that use a single descriptive polymorphic metadata (i.e., each of the disassembler, assembler, and generator use the same DAD file for consistent mappings across XML files that share the same structure—i.e., are the same class).*”

24. Claim 35

- i. ***[35.0] The data services handler according to claim 33 wherein the real time information director further comprises:***

See [33].

- ii. ***[35.1] a query logic that queries the database to determine information about tables in the database that the document uses, the information including field name, field datatype, and field allowance of nulls.***

Chen discloses a “db2look” tool to “capture the table definitions” for a particular database. Ex-1105, 292-93. As shown in the below excerpted output from db2look, it returns “*information about tables*”:

```
CREATE TABLE "UDBRS02"."ORG" (  
  "DEPTNUMB" SMALLINT NOT NULL ,  
  "DEPTNAME" VARCHAR(14) ,  
  "MANAGER" SMALLINT ,  
  "DIVISION" VARCHAR(10) ,  
  "LOCATION" VARCHAR(13) )  
IN "USERSPACE1" ;
```

Ex-1105, 263. A POSITA would have understood that in the above CREATE TABLE statement excerpted from the output of db2look, “DEPTNUMB” (red) is a “*field name*” (the column is named “DEPTNUMB”), “SMALLINT” (blue) is a “*field datatype*” (the datatype for the field is small integer), and “NOT NULL”

(green) indicates that nulls should not be allowed for that field (“*field allowance of nulls*”). Ex-1103, 292; Ex-1114, 72 (describing “CREATE TABLE” parameters).

Because db2look is a software tool that extracts data from the database, Dr. Almeroth states it is “*a query logic that queries the database.*” Ex-1103, ¶ 292; Ex-1111, 433 (defining “query” as “extract data from a database and present it”).

Chen teaches that DAD files define “[t]he mapping between elements and attributes in the XML document and the DB2 tables.” Ex-1105, 319-20. A POSITA would have understood this to teach DB2 tables to which the XML document will have its elements and attributes stored are “*tables in the database that the document uses.*” Ex-1103, ¶ 294; Ex-1105, 319-20.

25. Claim 36

i. ***[36.0] A method for executing on a computer system for managing data services comprising:***

As demonstrated for [1.1] and [1.2], Chen describes “*executing*” (e.g., running the software components UDB Server, including XML Extender) “*on a computer system*” (e.g., the hardware components of UDB Server) “*for managing data services*” (e.g., for transferring data among the UDB Sere and Clients).

ii. ***[36.1] transferring data among a data store and applications that supply and consume data;***

See [22.2].

- iii. *[36.2] transforming data for transfer using a representation of metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class;*

See [22.3].

- iv. *[36.3] dynamically instantiating, creating, and caching for reuse at least one disassembler and at least one assembler according to demand;*

See [1.2].

- v. *[36.4] disassembling inbound data documents into multiple records in preparation for insertion into metadata tables; and*

See [1.3].

- vi. *[36.5] assembling outbound data documents from at least one record selected from at least one metadata-selected table.*

See [1.4].

VIII. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)

Denial under § 325(d) is not warranted because the challenges presented in this Petition are neither cumulative nor redundant to the prosecution of the 575 Patent, as Chen was neither cited nor considered by the Examiner during prosecution of the 575 Patent. The references in this Petition were neither relied upon during prosecution nor are cumulative of art relied upon during prosecution. Further, the

575 Patent has not been previously challenged in an IPR. In addition, the Examiner also did not have the benefit of Dr. Almeroth's detailed expert testimony and the other evidence of record.

Thus, the Board should not exercise discretion to deny institution.

B. The Board Should Not Deny Institution Under 35 U.S.C. § 314(a)

The merits of Petitioner's arguments are compelling and the evidence in support is substantial. That "alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*." Ex-1115,4-5. But in any event, the six *Fintiv* factors do not justify denying institution.

Factor 1 (stay) is **neutral**. SAP has not yet moved to stay pending this IPR.

Factor 2 (proximity to trial) is **neutral or weighs against denial**. In the originally filed litigation by Patent Owner Valtrus (*Valtrus I*), the trial date is not currently set (though jury selection is set for September 22, 2025. Ex-1116. However, the Board should not rely on the current schedule in *Valtrus I* because Patent Owner Valtrus filed suit alone (without the other named Patent Owner, Key Patent Innovations) and thus did not have Article III Standing. Ex-1117; Ex-1118. Petitioner filed a Motion to Dismiss for Lack of Standing in *Valtrus I*, reconsideration of which remains pending and casts serious doubt on whether *Valtrus I* will actually proceed along the current schedule. Indeed, in response to Petitioner's Motion to Dismiss for Lack of Standing, Patent Owners essentially

concede that Valtrus alone lacked standing because Patent Owners together filed a second, duplicative lawsuit (*Valtrus II*) in an effort to rectify the Article III standing deficiencies in *Valtrus I*. Ex-1119; Ex-1120. This conduct confirms that Patent Owners themselves doubt that the trial schedule in *Valtrus I* will hold, further supporting that the Board should not rely on that schedule at all. Additionally, the other pending litigations involving the challenged patent (*Valtrus II* and a declaratory judgment action filed by Petitioner) do not have any trial date currently set, meaning there is no reliable trial date for the Board to consider in relation to this Petition.

Even if the Board assumes the *Valtrus I* trial date applies, this factor is neutral because there will be other claims not litigated at trial that are covered by this Petition. Therefore, a final written decision will issue first at least on the claims not litigated at trial but covered by this Petition.

Factor 3 (investment in parallel proceedings) **weighs against denial**. The related litigations are in its early stages of discovery, motions to transfer and dismiss have been filed, and the investment by the parties has been relatively minimal. All litigations involving the challenged patent are at an early stage, with major deadlines several months away. Ex-1116. Patent Owners may argue that the parties have invested substantial resources on procedural motions, but the resources expended have been directed toward resolving procedural missteps perpetuated by Patent Owner Valtrus's choice to file alone (without Article III standing) and Patent

Owners’ later attempts to fix that with procedurally improper duplicative litigation. These procedural disputes (caused by Patent Owners’ actions) should not be held against Petitioner under this factor.

Factor 4 (overlap) **weighs against denial** because Petitioner hereby stipulates that, if the IPR is instituted, it will not pursue the IPR grounds in the district court. Indeed, Chen is not at issue in the District Court Litigation. Ex-1121. Thus, “[i]nstituting trial here serves overall system efficiency and integrity goals by not duplicating efforts and by resolving materially different patentability issues.” *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10, 19 (June 15, 2020); *see also Sand Revolution II, LLC v. Continental Intermodal Group-Trucking LLC*, IPR2019-01393, Paper 24, 12 (June 16, 2020). In addition, Petitioner challenges more claims than Patent Owners will ultimately assert at trial. Patent Owner Valtrus agreed to narrow to 30 total asserted claims by April 30, 2025, and a maximum of six asserted claims per asserted patent. Ex-1116, 4. Thus, Patent Owner will be limited to asserting up to six of the thirty-four asserted claims of the 575 Patent, merely a subset of the claims challenged in this Petition. *See, e.g., Apple*, No. IPR2020-00156, Paper 10, 17, 21-22.

Factor 6 (other circumstances) **weigh against denial**. Here, the merits of the Petition are particularly compelling—for example, REFERENCE is a strikingly similar disclosure to that of the 575 Patent, is directed at the same problem and

proposes the same solutions as the 575 Patent and discloses nearly identical architecture, including several nearly identical figures. The evidence presented, if unrebutted in trial, would plainly lead to a conclusion that one or more claims are unpatentable by a preponderance of the evidence.

While Factor 5 (parties) may weigh slightly in favor of denial because the parties are the same as in district court, under the “holistic view” of the remaining factors and whether integrity of the system and efficiency is best served, institution is favored. *Samsung Elecs. Co. Ltd. v. Dynamics Inc.*, IPR2020-00505, Paper 11, 15 (Aug. 12, 2020).

IX. CONCLUSION

Claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35-36 of the 575 Patent are unpatentable for the reasons demonstrated above.

Dated: January 15, 2025

Respectfully Submitted,

/s/Louis L. Campbell
Louis L. Campbell (Lead Counsel)
llcampbell@winston.com
USPTO Reg. No. 59,963
Winston & Strawn LLP
255 Shoreline Dr., Ste. 520
Redwood City, CA 94065
T: (650) 858-6500

Lead Counsel for Petitioner

CERTIFICATE OF COMPLIANCE

This Petition complies with the word count limits set forth in 37 C.F.R. § 42.24(a)(1)(i), because this Petition contains 13,963 words, excluding the parts of the Petition exempted by 37 C.F.R. § 42.24(a)(1), determined using the word count provided by Microsoft Word, which was used to prepare this Petition.

Dated: January 15, 2025

Respectfully Submitted,

/s/Louis L. Campbell

Louis L. Campbell (Lead Counsel)

llcampbell@winston.com

USPTO Reg. No. 59,963

Winston & Strawn LLP

255 Shoreline Dr., Ste. 520

Redwood City, CA 94065

T: (650) 858-6500

Lead Counsel for Petitioner

CERTIFICATE OF SERVICE

Under 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that on January 15, 2025, I caused to be served a true and correct copy of the foregoing “**PETITION *INTER PARTES* REVIEW OF U.S. PATENT NO. 7,313,575**” and accompanying exhibits by Federal Express Overnight Mail on the following:

Botos Churchill IP Law LLP
430 Mountain Avenue
Suite 401
New Providence, NJ 07974

With a courtesy copy of this Petition and supporting material was also served on litigation counsel for Patent Owners via email:

Matt Berkowitz
mberkowitz@reichmanjorgensen.com

Patrick R. Colsher
pcolsher@reichmanjorgensen.com

Aaron Morris
amorris@reichmanjorgensen.com

Navid Cyrus Bayar
nbayar@reichmanjorgensen.com

Reichman Jorgensen Lehman & Feldberg LLP
100 Marine Parkway
Suite 300
Redwood Shores, CA 94065

Khue Van Hoang
khoang@reichmanjorgensen.com
Reichman Jorgensen Lehman & Feldberg LLP
400 Madison Avenue
Suite 14d
New York, NY 1001

Philip Ekle
peklem@reichmanjorgensen.com
Savannah Harman Carnes

scarnes@reichmanjorgensen.com

Reichman Jorgensen Lehman & Feldberg LLP
1909 K Street, NW
Suite 800
Washington, DC 20006

/s/Louis L. Campbell

Louis L. Campbell (Lead Counsel)

llcampbell@winston.com

USPTO Reg. No. 59,963

Winston & Strawn LLP

255 Shoreline Dr., Ste. 520

Redwood City, CA 94065

T: (650) 858-6500

Lead Counsel for Petitioner