

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

**SAP America, Inc.
Petitioner**

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

**DECLARATION OF KEVIN ALMEROTH, PH.D. UNDER 37 C.F.R. § 1.68
IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

	Page
EXHIBIT LIST	9
I. INTRODUCTION	11
II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	12
III. LEVEL OF ORDINARY SKILL IN THE ART	24
IV. RELEVANT LEGAL STANDARDS	25
V. OVERVIEW OF THE 575 PATENT	27
VI. CLAIM CONSTRUCTION	30
VII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE.....	30
VIII. THE PRINCIPAL PRIOR ART	31
A. Summary of Chen	31
IX. CHEN RENDERS OBVIOUS CLAIMS 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, AND 35-36.....	34
A. Detailed Analysis of the Challenged Claims	34
1. Claim 1	34
i. [1.0] A data services handler for execution on a computing system comprising:.....	34
ii. [1.1] an interface executing on the computing system for communicating between a data store and applications that supply and consume data; and.....	42
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:	48
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	61
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.....	63
2. Claim 2	66
i. [2.0] The data services handler according to claim 1 further comprising:.....	66
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.....	66
3. Claim 4	69
i. [4.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	69
ii. [4.1] the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.	70
4. Claim 6	71
i. [6.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	71
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.	71
5. Claim 7	73
i. [7.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	73
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.	73
6. Claim 10	76
i. [10.0] The data services handler according to claim 1 wherein:	76
ii. [10.1] the real time information director receives and publishes data documents having a content and a format defined by document metadata.	76
7. Claim 11	77

i.	[11.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	77
ii.	[11.1] a logic that reads a schema of tables identified in metadata; and 77	
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.....	79
8.	Claim 12	80
i.	[12.0] The data services handler according to claim 1 further comprising:.....	80
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):.....	80
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);	82
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.....	84
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).....	85
9.	Claim 13	85
i.	[13.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	85
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.....	86
10.	Claim 14	88
i.	[14.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	88
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;	88
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.	91
11. Claim 17	92
i. [17.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	92
12. Claim 18	95
i. [18.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	95
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.....	95
13. Claim 19	98
i. [19.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	98
ii. [19.1] an aggregation engine that computes aggregates in memory from data selected from the data store.....	98
14. Claim 20	99
i. [20.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	99
ii. [20.1] a real-time scoring engine.	99
15. Claim 21	100
i. [21.0] The data services handler according to claim 1 wherein the real time information director further comprises:.....	100
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.....	100
16. Claim 22	102

i.	[22.0] An article of manufacture comprising:.....	102
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:.....	104
iii.	[22.2] a code causing the controller to transfer data among a data store and applications that supply and consume data;	105
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;	106
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;	108
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	109
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.....	110
17.	Claim 23	111
i.	[23.0] The article of manufacture according to claim 22 further comprising:.....	111
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.	111
18.	Claim 24	114
i.	[24.0] The article of manufacture according to claim 22 further comprising:.....	114
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.....	115
19.	Claim 25	118

i.	[25.0] The article of manufacture according to claim 22 further comprising:.....	118
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 polymorphic metadata objects.	119
20.	Claim 28	121
i.	[28.0] An enterprise system comprising:	121
ii.	[28.1] a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers; and 121	
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, ...	123
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.	125
21.	Claim 31	127
i.	[31.0] The enterprise system according to claim 28 wherein:.....	127
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.	127
22.	Claim 32	128
i.	[32.0] The enterprise system according to claim 28 wherein:.....	128
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.	128

23.	Claim 33	129
i.	[33.0] A data services handler according to claim 1 further comprising: 129	
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.	129
24.	Claim 35	131
i.	[35.0] The data services handler according to claim 33 wherein the real time information director further comprises:.....	131
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.....	132
25.	Claim 36	134
i.	[36.0] A method for executing on a computer system for managing data services comprising:.....	134
ii.	[36.1] transferring data among a data store and applications that supply and consume data;.....	134
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; 135	
iv.	[36.3] dynamically instantiating, creating, and caching for reuse at least one disassembler and at least one assembler according to demand; .	136
v.	[36.4] disassembling inbound data documents into multiple records in preparation for insertion into metadata tables; and.....	136
vi.	[36.5] assembling outbound data documents from at least one record selected from at least one metadata-selected table.	137
X.	CONCLUSION.....	137

EXHIBIT LIST

No.	Exhibit 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, Kevin Almeroth, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Petitioner in the matter of the *Inter Partes* Review of U.S. Patent No. 7,313,575 (“the ’575 Patent”) to Carr et al.

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, and 35-36 (“the Challenged Claims”) of the ’575 Patent are anticipated or otherwise would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this declaration, I have studied:

- the ’575 Patent, Ex.1101.
- the prosecution history of the ’575 Patent (“’575 File History”), Ex.1102.
- W. Chen, et al., “DB2 UDB Evaluation Guide for Linux and Windows,” *IBM Redbooks*, April 2003. Available: <https://www.redbooks.ibm.com/> (Ex. 1105).

- The additional references I refer to below and/or are identified in the exhibit list.

5. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of network communications and security as described below.

6. Unless otherwise noted, all emphasis in any quoted material has been added. Claim terms are italicized.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My complete qualifications and professional experience are described in my Curriculum Vitae, a copy of which can be found in Ex-1104. The following is a brief summary of my relevant qualifications and professional experience.

8. I am currently a Professor Emeritus in the Department of Computer Science at the University of California, Santa Barbara (UCSB). While at UCSB, I held faculty appointments and was a founding member of the Computer Engineering (CE) Program, Media Arts and Technology (MAT) Program, and the Technology Management Program (TMP). I also served as the Associate Director of the Center for Information Technology and Society (CITS) from 1999 to 2012. I have been a faculty member at UCSB since July 1997.

9. I hold three degrees from the Georgia Institute of Technology: (1) a Bachelor of Science degree in Information and Computer Science (with minors in Economics, Technical Communication, and American Literature) earned in June 1992; (2) a Master of Science degree in Computer Science (with specialization in Networking and Systems) earned in June 1994; and (3) a Doctor of Philosophy (Ph.D.) degree in Computer Science (Dissertation Title: Networking and System Support for the Efficient, Scalable Delivery of Services in Interactive Multimedia System, minor in Telecommunications Public Policy) earned in June 1997. During my education, I took a wide variety of courses as demonstrated by my minors. My undergraduate degree also included a number of courses more typical of a degree in electrical engineering including digital logic, signal processing, and telecommunications theory.

10. One of the major concentrations of my research over the past 30+ years has been the delivery of multimedia content and data between computing devices, including through various network architectures. In my research, I have studied large-scale content delivery systems, and the use of servers located in a variety of geographic locations to provide scalable delivery to hundreds or thousands of users simultaneously. I have also studied smaller-scale content delivery systems in which content is exchanged between individual computers and portable devices. My work has emphasized the exchange of content more efficiently across computer networks,

including the scalable delivery of content to many users, mobile computing, satellite networking, delivering content to mobile devices, and network support for data delivery in wireless networks.

11. In 1992, the initial focus of my research was on the provision of interactive functions (e.g., VCR-style functions like pause, rewind, and fast-forward) for near video-on-demand systems in cable systems; in particular, how to aggregate requests for movies at a cable head-end and then how to satisfy a multitude of requests using one audio/video stream broadcast to multiple receivers simultaneously. This research has continually evolved and resulted in the development of techniques to scalably deliver on-demand content, including audio, video, web documents, and other types of data, through the Internet and over other types of networks, including over cable systems, broadband telephone lines, and satellite links.

12. An important component of my research has been investigating the challenges of communicating multimedia content, including video, between computers and across networks including the Internet. Although the early Internet was used mostly for text-based, non-real time applications, the interest in sharing multimedia content, such as video, quickly developed. Multimedia-based applications ranged from downloading content to a device to streaming multimedia content to be instantly used. One of the challenges was that multimedia content is

typically larger than text-only content, but there are also opportunities to use different delivery techniques since multimedia content is more resilient to errors. I have worked on a variety of research problems and used a number of systems that were developed to deliver multimedia content to users. One content-delivery method I have researched is the one-to-many communication facility called “multicast,” first deployed as the Multicast Backbone, a virtual overlay network supporting one-to-many communication. Multicast is one technique that can be used on the Internet to provide streaming media support for complex applications like video-on-demand, distance learning, distributed collaboration, distributed games, and large-scale wireless communication. The delivery of media through multicast often involves using Internet infrastructure, devices and protocols, including protocols for routing and TCP/IP.

13. Starting in 1997, I worked on a project to integrate the streaming media capabilities of the Internet together with the interactivity of the web. I developed a project called the Interactive Multimedia Jukebox (IMJ). Users would visit a web page and select content to view. The content would then be scheduled on one of a number of channels, including delivery to students in Georgia Tech dorms delivered via the campus cable plant. The content of each channel was delivered using multicast communication.

14. In the IMJ, the number of channels varied depending on the capabilities of the server including the available bandwidth of its connection to the Internet. If one of the channels was idle, the requesting user would be able to watch their selection immediately. If all channels were streaming previously selected content, the user's selection would be queued on the channel with the shortest wait time. In the meantime, the user would see what content was currently playing on other channels, and because of the use of multicast, would be able to join one of the existing channels and watch the content at the point it was currently being transmitted.

15. The IMJ service combined the interactivity of the web with the streaming capabilities of the Internet to create a jukebox-like service. It supported true Video-on-Demand when capacity allowed, but scaled to any number of users based on queuing requested programs. As part of the project, we obtained permission from Turner Broadcasting to transmit cartoons and other short-subject content. We also connected the IMJ into the Georgia Tech campus cable television network so that students in their dorms could use the web to request content and then view that content on one of the campus's public access channels.

16. More recently, I have also studied issues concerning how users choose content, especially when considering the price of that content. My research has examined how dynamic content pricing can be used to control system load. By

raising prices when systems start to become overloaded (i.e., when all available resources are fully utilized) and reducing prices when system capacity is readily available, users' capacity to pay as well as their willingness can be used as factors in stabilizing the response time of a system. This capability is particularly useful in systems where content is downloaded or streamed on-demand to users.

17. As a parallel research theme, starting in 1997, I began researching issues related to wireless devices and sensors. In particular, I was interested in showing how to provide greater communication capability to “lightweight devices,” i.e., small form-factor, resource-constrained (e.g., CPU, memory, networking, and power) devices. Starting in 1998, I published several papers on my work to develop a flexible, lightweight, battery-aware network protocol stack. The lightweight protocols we envisioned were similar in nature to protocols like Bluetooth, Universal Plug and Play (UPnP) and Digital Living Network Alliance (DLNA).

18. From this initial work, I have made wireless networking—including ad hoc, mesh networks and wireless devices—one of the major themes of my research. My work in wireless networks spans the protocol stack from applications through to the encoding and exchange of data at the data link and physical layers.

19. At the application layer, even before the large-scale “app stores” were available, my research looked at building, installing, and using apps for a variety of purposes, from network monitoring to support for traditional computer-based

applications (e.g., content retrieval) to new applications enabled by ubiquitous, mobile devices. For example, my research has looked at developing applications for virtually exchanging and tracking “coupons” through “opportunistic contact” among mobile wireless devices (i.e., communication among devices moving into communication range with each other). In many of the courses I have taught there is a project component. Through these projects I have supervised numerous efforts to develop new “apps” for download and use across a variety of mobile platforms.

20. Toward the middle of the protocol stack, my research has also looked to build wireless infrastructure support to enable communication among a set of mobile devices unaided by any other kind of network infrastructure. These kinds of networks are useful either in challenged network environments (e.g., when a natural disaster has destroyed existing infrastructure) or when suitable support for network communication never existed. The deployment of such networks (or even the use of traditional network support) are critical to support services like disaster relief, catastrophic event coordination, and emergency services deployment.

21. Yet another theme is monitoring wireless networks, in particular different variants of IEEE 802.11 compliant networks, to (1) understand the operation of the various protocols used in real-world deployments, (2) use these measurements to characterize use of the networks and identify protocol limitations and weaknesses, and (3) propose and evaluate solutions to these problems. I have

successfully used monitoring techniques to study wireless data link layer protocol operation and to improve performance by enhancing the operation of such protocols. For wireless protocols, this research includes functions like network acquisition and channel bonding.

22. One theme in my wireless network research has been cross-layer solutions and innovations. As mentioned above, with greater wireless device use and network support, we envisioned new application paradigms and services, for example, when mobile devices come into contact with each other. Instead of relying on existing infrastructure to relay communication, the devices are able to discover each other and communicate directly. Other examples include discovering and using location information to enhance users' experiences. Network support and novel applications span use a variety of network architectures supporting users on foot, in vehicles, and across varying terrains and environments. Finally, we studied how communication efficiency can be supported through intelligent handoffs as well as location and movement prediction.

23. Protecting networks, including their operation and content, has been an underlying theme of my research almost since the beginning of my research career. Starting in 2000, I have been involved in several projects that specifically address security, network protection, and firewalls. After significant background work, a team on which I was a member successfully submitted a \$4.3M grant proposal to the

Army Research Office (ARO) at the Department of Defense to propose and develop a high-speed intrusion detection system. Key aspects of the system included associating streams of packets and analyzing them for viruses and other malware. Once the grant was awarded, we spent several years developing and meeting the milestones of the project. A number of my students worked on related projects and published papers on topics ranging from intrusion detection to developing advanced techniques to be incorporated into firewalls. I have also used firewalls, including their associated malware detection features, in developing techniques for the classroom to ensure that students are not distracted by online content.

24. My recent work ties some of the various threads of my past research together. I have investigated content delivery in online social networks and proposed reputation management systems in large-scale social networks and marketplaces. On the content delivery side, I have looked at issues of caching and cache placement, especially when content being shared and the cache has geographical relevance. We were able to show that effective caching strategies can greatly improve performance and reduce deployment costs. Our work on reputation systems showed that reputations have economic value, and as such, creates a motivation to manipulate reputations. In response, we developed a variety of solutions to protect the integrity of reputations in online social networks. The techniques we developed for content

delivery and reputation management were particularly relevant in peer-to-peer communication and recommendations for downloadable “apps.”

25. As an important component of my research program, I have been involved in the development of academic research into available technology in the market place. One aspect of this work is my involvement in the Internet Engineering Task Force (IETF). The IETF is a large and open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. I have been involved in various IETF groups including many content delivery-related working groups like the Audio Video Transport (AVT) group, the MBone Deployment (MBONED) group, Source Specific Multicast (SSM) group, the Inter-Domain Multicast Routing (IDMR) group, the Reliable Multicast Transport (RMT) group, the Protocol Independent Multicast (PIM) group, etc. I have also served as a member of the Multicast Directorate (MADDOGS), which oversaw the standardization of all things related to multicast in the IETF. Finally, I was the Chair of the Internet2 Multicast Working Group for seven years.

26. My involvement in the research community extends to leadership positions for several academic journals and conferences. I am the co-chair of the Steering Committee for the ACM Network and System Support for Digital Audio and Video (NOSSDAV) workshop and on the Steering Committees for the

International Conference on Network Protocols (ICNP), ACM Sigcomm Workshop on Challenged Networks (CHANTS), and IEEE Global Internet (GI) Symposium. I have served or am serving on the Editorial Boards of IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Network, ACM Computers in Entertainment, AACE Journal of Interactive Learning Research (JILR), and ACM Computer Communications Review. I have co-chaired a number of conferences and workshops including the IEEE International Conference on Network Protocols (ICNP), IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), International Conference on Communication Systems and Networks (COMSNETS), IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), the International Workshop On Wireless Network Measurement (WiNMee), ACM Sigcomm Workshop on Challenged Networks (CHANTS), the Network Group Communication (NGC) workshop, and the Global Internet Symposium, and I have served on the program committees for numerous conferences.

27. Furthermore, in the courses I taught at UCSB, a significant portion of my curriculum covered aspects of the Internet and network communication including the physical and data link layers of the Open System Interconnect (OSI) protocol stack, and standardized protocols for communicating across a variety of physical media such as cable systems, telephone lines, wireless, and high-speed

Local Area Networks (LANs). The courses I have taught also cover most major topics in Internet communication, including data communication, multimedia encoding, and mobile application design. My research and courses have covered a range of physical infrastructures for delivering content over networks, including cable, Integrated Services Digital Network (ISDN), Ethernet, Asynchronous Transfer Mode (ATM), fiber, and Digital Subscriber Line (DSL). For a complete list of courses I have taught, see my curriculum vitae (CV).

28. In addition, I co-founded a technology company called Santa Barbara Labs that was working under a sub-contract from the U.S. Air Force to develop very accurate emulation systems for the military's next generation internetwork. Santa Barbara Labs' focus was in developing an emulation platform to test the performance characteristics of the network architecture in the variety of environments in which it was expected to operate, and, in particular, for network services including IPv6, multicast, Quality of Service (QoS), satellite-based communication, and security. Applications for this emulation program included communication of a variety of multimedia-based services, including video conferencing and video-on-demand.

29. In addition to having co-founded a technology company myself, I have worked for, consulted with, and collaborated with companies for nearly 30 years. These companies range from well-established companies to start-ups and include

IBM, Hitachi Telecom, Turner Broadcasting System (TBS), Bell South, Digital Fountain, RealNetworks, Intel Research, Cisco Systems, and Lockheed Martin.

30. Through my graduate education, leadership with CITS, involvement in TMP, role in the development of the Internet2 infrastructure, and consulting with ISPs, I have gained a strong understanding in the role of the Internet in our society and the challenges of deploying large-scale production networking infrastructure. CITS, since its inception, has looked at the role of the Internet in society, including how the evolution of technology have created communication opportunities and challenges, including, for example through disruptive technologies like P2P. TMP looks to focus on non-purely technical issues, including, for example, state-of-the-art business methods, strategies for successful technology commercialization, new venture creation, and best practices for fostering innovation. Through my industry collaborations and Internet2 work, I have developed significant experience in the challenges of deploying, monitoring, managing, and scaling communication infrastructure to support evolving Internet services like streaming media, conferencing, content exchange, social networking, and e-commerce.

31. I am a Member of the Association of Computing Machinery (ACM) and a Fellow of the Institute of Electrical and Electronics Engineers (IEEE).

III. LEVEL OF ORDINARY SKILL IN THE ART

32. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

33. A Person of Ordinary Skill in The Art (“POSITA”) in June 2004 would have had a working knowledge of the telecommunications and networking art that is pertinent to the ’575 Patent. A POSITA would have had a bachelor’s degree or graduate 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.

34. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the ’575 Patent.

IV. RELEVANT LEGAL STANDARDS

35. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '575 Patent, I am relying on certain basic legal principles that counsel has explained to me.

36. I understand that prior art to the '575 Patent includes patents and printed publications in the relevant art that predate the priority date of the '575 Patent. For purposes of this Declaration, I am applying June 14, 2004, as the priority date of the '575 Patent, but I take no position on that issue.

37. I have been informed by counsel that a claimed invention is invalid if it is “anticipated” by the prior art. I understand that, for a patent claim to be “anticipated” by the prior art, each and every requirement of the claim must be found, expressly or inherently, in a single prior art reference as recited in the claim. I understand that claim limitations that are not expressly described in a prior art reference may still be there if they are “inherent” to the thing or process being described in the prior art.

38. I understand that if a reference incorporates other documents by reference, the incorporating reference and the incorporated reference(s) should be treated as a single prior art reference for purposes of analyzing anticipation.

39. I understand that it is acceptable to consider evidence other than the information in a particular prior art document to determine if a feature is necessarily present in or inherently described by that reference.

40. I have been informed by counsel that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA. I have also been informed by counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

41. I have been further informed by counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

V. OVERVIEW OF THE 575 PATENT

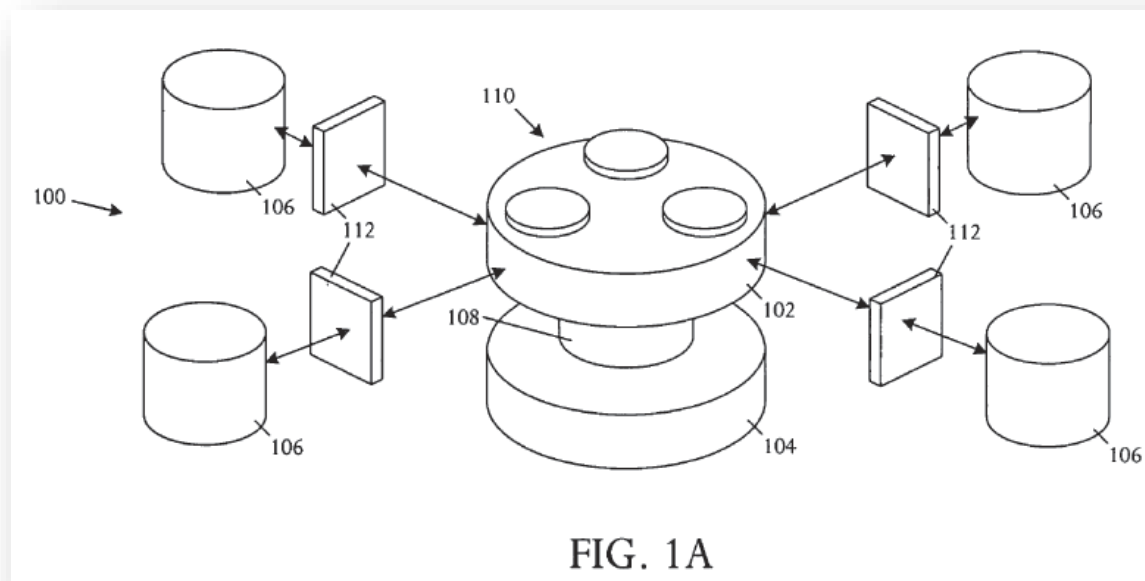
42. The 575 Patent is directed at resolving the issue of latency in an enterprise system. Ex. 1101, [1:5–50]. To achieve this, the 575 Patent discloses a “data services system” wherein a “data services handler comprises an interface between a data store and applications that supply and consume data, 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.” Ex. 1101, [1:54–60].

43. The 575 Patent notes that the RTID “is responsible for most or all data input and output to the real-time data store.” Ex. 1101, [3:24–26]. And it “prepares and caches Java Database Connectivity (JDBC) statements for each inbound or outbound document class.” Ex. 1101, [3:46–48]. Each class of documents is defined by “descriptive metadata” that is used to build both document “assemblers” and “disassemblers”. Ex. 1101, [3:58–64]. A document assembler contains “prepared statements for inserting or updating various records related to a particular class of outbound documents into the data store.” Ex. 1101, [3:51–53]. A document dissembler contains similar statements but for inbound documents.

44. The 575 Patent also describes a zero latency enterprise (ZLE) “that is capable of consolidating data that resides in multiple applications.” Ex. 1101, [4:10–12]. “ZLE supports many enterprise applications, for example including Systems, Applications, and Products (SAP) in data processing, Siebel Systems, and other

enterprise systems, that attach to the ZLE core [] via a variety of application and technology adapters.” Ex. 1101, [5:22-27]. In an embodiment of the 575 Patent, the ZLE data store caches, maintains, enriches, stores, and queues messages and information passed to and from the “appropriate applications.” Ex. 1101, [5:32-35]. The RTID of the 575 Patent “supplies multiple services for clients of the ZLE 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].

45. Figure 1A depicts a schematic block diagram of one embodiment of the invention:



Ex. 1101, Fig. 1A. Here, “data services handler 100 [] 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]. The 575 notes that “the data services handler 100 can be implemented according to a Zero Latency enterprise (ZLE) architecture or framework, a multiple-level architecture centered on a virtual hub or ZLE core 110, that caches and routes information.” Ex. 1101, [5:15–19]. “The adapters 112 convert proprietary messages to standards supported by the ZLE core 110.” Ex. 1101, [5:27–29].

VI. CLAIM CONSTRUCTION

46. It is my understanding that to properly evaluate the 575 Patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be given their “ordinary and customary meaning” as understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the unpatentability inquiry.

VII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

47. A POSITA would find the challenged claims unpatentable under any interpretation consistent with their plain and ordinary meaning. However, I have been informed that 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” and that “*logic*” refers to “software.” I therefore adopt those meanings for my analysis below. The following discussion in this Declaration provides a detailed analysis of how the asserted prior art teaches each limitation of the Challenged Claims.

48. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

49. It is my opinion that the alleged invention recited in the Challenged Claims would have been obvious in view of the teachings of the asserted prior art and the knowledge of a POSITA.

VIII. THE PRINCIPAL PRIOR ART

A. Summary of Chen

50. The printed publication titled *DB2 UDB Evaluation Guide for Linux and Windows* and authored by W. Chen, et al. (“Chen”) (Ex. 1105) was published in April 2003 as a part of the *IBM Redbooks* series. I understand that Chen is therefore prior art to the 575 Patent at least under pre-AIA 35 U.S.C. § 102(b).

51. Chen seeks to address the need for “zero latency performance so that a business can integrate, synchronize, and route data across the enterprise, all in real time” that is identified in the 575 Patent. Ex. 1101, [1:31-33]. Specifically, Chen

describes a “complete multimedia, Web-ready, relational database management system” known as the “DB2 Universal Database (UDB).” Ex. 1105 at 1.

52. Chen teaches that its system is “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 at 2. According to Chen, the system is suitable for a “wide range of customer applications, which can be deployed on a variety of hardware and software platforms.” Ex. 1105 at 1. It is “scalable and industrial strength” and can serve “both structure and unstructured data management needs of a single user to very large enterprises.” *Id.*

53. Below is an overview of the system architecture depicted with respect to Figure 1-2:

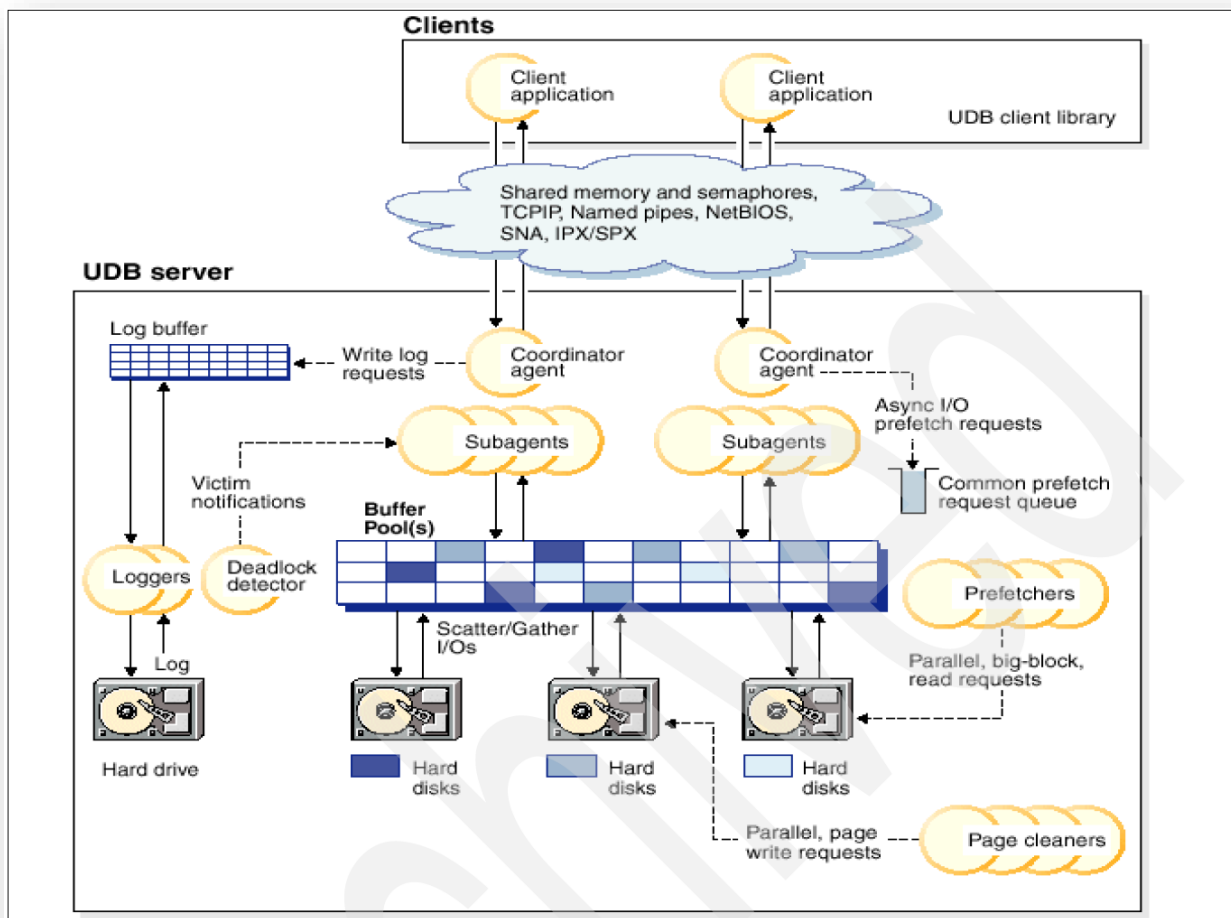


Figure 1-2 DB2 UDB architecture overview

Ex. 1105, Fig. 1-2.

54. As shown in Figure 1-2, “[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 at 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 at 318.

IX. CHEN RENDERS OBVIOUS CLAIMS 1-2, 4, 6-7, 10-14, 17-25, 28, 31-33, AND 35-36

A. Detailed Analysis of the Challenged Claims

1. Claim 1

i. [1.0] *A data services handler for execution on a computing system comprising:*

55. Chen discloses “a data services handler for execution on a computing system.”

(A) “a data services handler”

56. In the 575 Patent, it is explained that “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.” Ex. 1101, Abstract, 1:55-60. As explained below, Chen discloses a database server system (the DB2 System, including UDB Server) that handle data services, including as set forth in the limitations of claim 1 the comprise the data services handler. Chen thus discloses a *data services handler*.

57. A POSITA would have understood that Chen’s DB2 Universal Database server is one example of a data store, as it is a place to store data and was

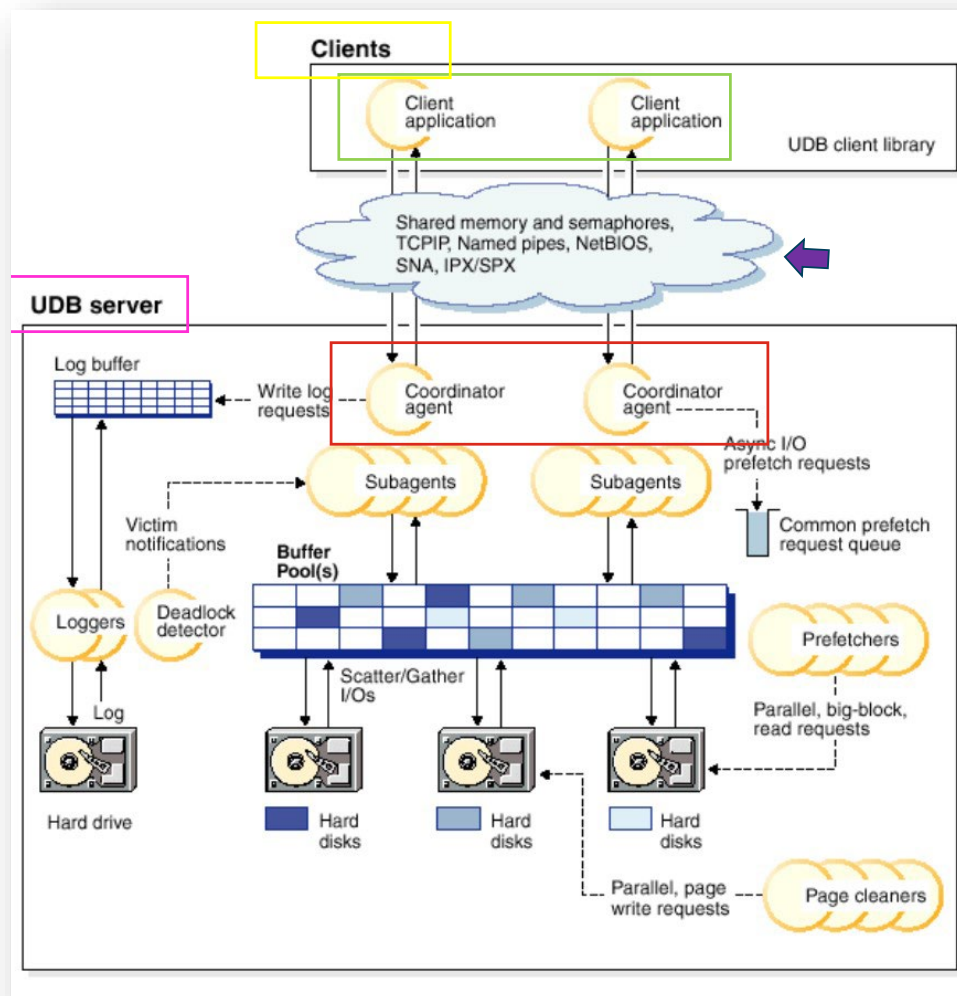
well-known as a data store in the industry. See Ex-1110, 42 (identifying DB2 as one of the “Data Stores Supported”). Chen states that the disclosed DB2 UDB system is “a high performance middleware software that provides a secure and reliable system to **manage data or content for a wide variety of customer applications.**” Ex. 1105 at 1-2. Thus, under the plain and ordinary meaning, Chen discloses a “a data store.”

58. As Chen explains, it is a relational database that “presents data as a collection of tables... The data in a table is logically related, and relationships can be defined between tables. Data can be viewed and manipulated based on mathematical principles and operations called relations.” Ex. 1105 at 7-8. Chen discloses that relational databases are part of the DB2 system. Ex. 1105 at 2, 6. Therefore, a POSITA would have further understood the “relational database management system” of Chen to comprise a “data store.”

59. Next, Chen notes that the “DB2 UDB is...flexible enough to serve both structured and unstructured data management needs of a single user to very large enterprises. It is suitable for a wide range 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 at 1-2. A POSITA would have understood that the structured and unstructured data would likely not be in a form

that is inherently useable by the client applications—there would need to be some interface that translates the data for use in the applications.

60. Moreover, Chen discloses that in addition to the relational databases the DB2 system also comprises database objects known as buffer pools. Ex. 1105 at 10. “A buffer pool is the amount of main memory allocated to cache table and index data pages as they are being read from disk, or being modified.” Ex. 1105 at 10. The buffer pool is meant to “improve system performance” by allowing data to be accessed “from memory” which is faster than accessing the data “from disk.” Ex. 1105 at 10. Figure 1-2 below illustrates how data flows from the hard disks through the buffer pool(s) to the client applications:



Ex. 1105 at Fig. 1-2 (annotated). Also disclosed in Figure 1-2 are “DB2 agents” which “are the most common type of DB2 processes which carry out the bulk of the SQL processing on behalf of the applications.” Ex. 1105 at 11. A POSITA would have known that SQL is a standard language for communicating with and manipulating relational databases. Ex-1113, 427 (defining “Structured Query Language (SQL) as “a standard language for accessing relational database management systems”). In the figure above, the client applications (green) supply

data to, and consume data from, the UDB server (pink) through coordinator agents (red).

61. In light of the disclosure of relational databases that use of SQL and the disclosure of data flow from hard disks through buffer pools that streamline system performance to client applications, it would be obvious to a POSITA that the system described in Chen must comprise an interface between the relational database management system (DB2 Server, the *data store*) and the variety of customer applications interfacing with the DB2 System. Ex. 1105 at 1-2.

62. Further, according to Chen, “[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 at 10. Thus, it would also be clear to a POSITA that said applications consist of supplying and consuming data.

63. As will be described in Section IX.A.1.iii, 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. Thus, it would be obvious to a POSITA that the DB2 System described in Chen comprises a “data services handler.”

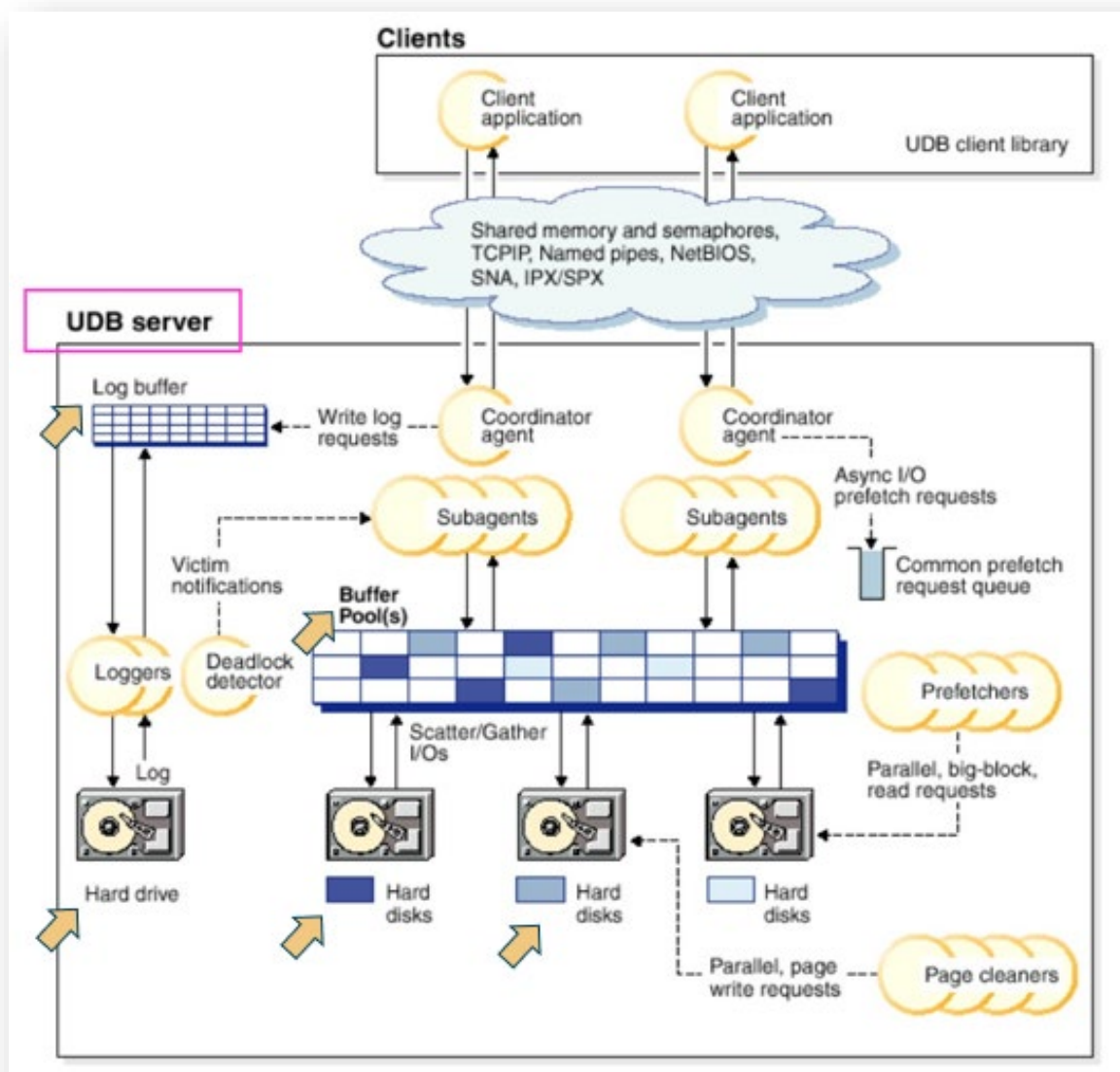
64. Accordingly, Chen discloses a “*data services handler*” (e.g., the DB2 System, including UDB Server).

(B) “for execution on a computing system”

65. Chen system discloses this limitation.

66. For example, the ’575 Patent discusses that its computing system “communicat[es] between a data store and applications” and “transforms data.” Ex. 1101 at Claim 1. The ’575 Patent further discloses that “[m]etadata can be modified on a computing system of various types,” those types including “a workstation.” Ex. 1101 at [42:40-41]. Given this context, a POSITA would have understood that “a computing system” as used in the 575 Patent would include any system where hardware and software components together input, process, store, and/or output data.

67. Chen states that its 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 at 2. A POSITA would have understood that these “variety of hardware and software platforms” include the exemplary “workstation” explicitly disclosed in the 575 Patent as well as a variety of other “*computer system[s]*.” Moreover, Figure 1-2 depicts the UDB Server of the DB2 system, including conventional computer hardware like hard drives and buffers.



Ex. 1105 at Figure 1-2 (annotated).

68. Lastly, the DB2 System (which includes the UDB Server) discloses that it is designed to execute on both single machines and clusters:

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 at 346–347. Each of these disclosures would teach a POSITA that the DB2 System is able to “run on everything from Laptops supporting mobile users to massive parallel systems with terabytes of data and/or thousands of users” and therefore “meet the demands of today’s heterogeneous computing environments.” Ex. 1105 at 1–2. According to the Microsoft Computer Dictionary, “execute” means “loading the machine code of the program into memory and then performing the instructions.” Ex. 1111, 200. This is also true for the DB2 System’s coordinator agents (discussed below as assemblers and disassemblers), which are described as “EDUs [Engine Dispatchable Units]” that “are implemented as threads in a single process on Windows-based platforms and as processes on UNIX,” (Ex. 1105 at 11).

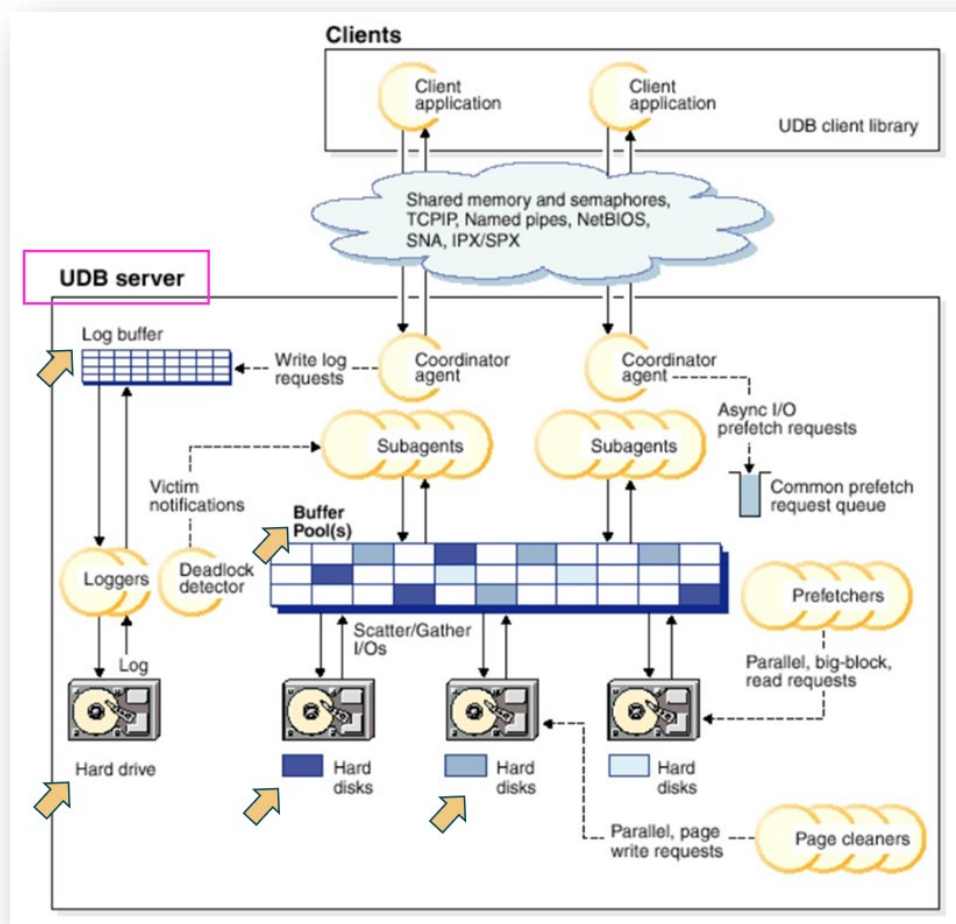
69. Accordingly, Chen discloses “*a data services handler for execution on a computing system.*”

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

70. Chen discloses “an interface executing on the computing system for communicating between a data store and applications that supply and consume data.” As explained below, in my opinion, the coordinator agents function as an *interface between the data store and applications*, because they facilitate communication and information passing between the client applications and DB2 database.

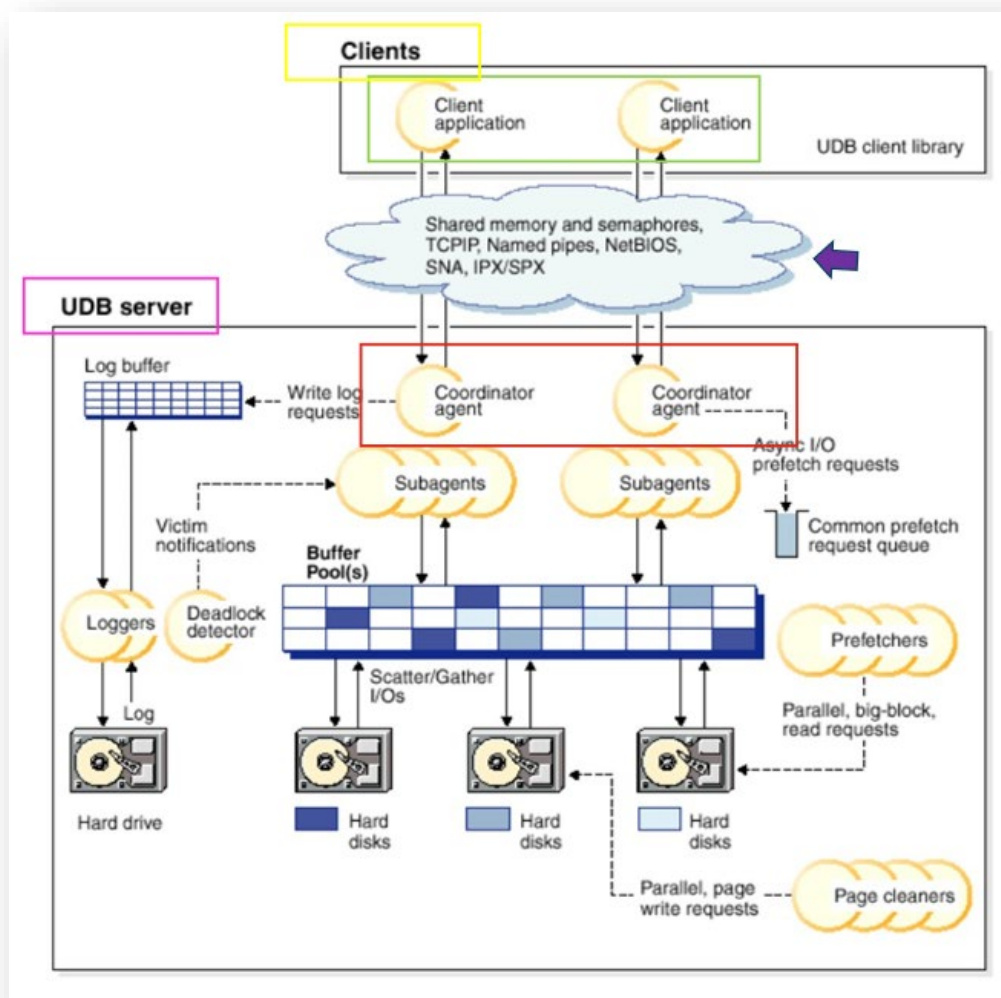
71. As explained previously, the DB2 system, including the UDB server, is designed for execution on a computing system. *See supra* Section IX.A.1.i(B).

72. Additionally, as discussed previously, Chen discloses such a *data store* in the form of a relational database. Ex. 1105 at 1-2; Ex-1110, 42. One type of database object that the DB2 System manages is a “buffer pool” where “the amount of main memory allocated to cache table and index data pages as they are being read from disk, or being modified.” Ex. 1105 at 10. As shown below in Figure 1-2, the UDB server (outlined pink) includes persistent and volatile storage, embodied as hard drives and buffer pools, respectively (orange arrows):



Ex. 1105, Fig. 1-2 (annotated). A POSITA would therefore have understood that the UDB DB2 system comprises *a data store*, including the UDB Server.

73. Further, as previously described, the DB2 system comprises coordinator agents and databases that facilitate supply and consumption of data. *See* Section IX.A.1.i(A). For instance, Chen discloses that the DB2 UDB server can operate as a local application or network server, where it “can be access[ed] from many client workstations over many networks.” Ex. 1105 at 3. As shown in Figure 1-2, below, the client machines (yellow) can consist of client applications (green):



Ex. 1105, Fig. 1-2 (annotated).

74. As explained below, in my opinion, the coordinator agents function as an *interface between the data store and applications*, because they facilitate communication and information passing between the client applications and DB2 database. See, e.g., Ex-1111, 31.

75. For example, to access data from the *data store* (i.e., UDB Server), the client applications interface with coordinator agents: namely, the coordinator agent,

“coordinates communication” with clients and is assigned to a specific application. In one example disclosed by Chen, “[e]ach client application...communicates with the DB2 Server using...a communication protocol such as TCP/IP.” Ex. 1105 at 10. Chen also discloses that the DB2 server processes requests using “DB2 agents, [that] 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 the applications. DB2 assigns a coordinator agent with an application, and this agent coordinates the communication and processing for this application.” Ex. 1105 at 11. When selecting data, “[a]n application issues a connect database statement to connect remotely to a DB2 database through a DB2 client...[where] a coordinating agent is assigned by DB2 which then coordinates the communication and processing for this application” Ex. 1105 at 13.

76. The type of data stored in the database and how it is consumed varies. One example of this disclosed in Chen consists of the DB2 UDB “provid[ing] the ability to store and access XML documents, and also [] compos[ing] XML documents from the existing relational data or [] decompos[ing] XML documents into relational data...[by] providing a set of new data types, functions, and stored procedures.” Ex. 1105 at 318-319. And, as discussed before, “[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 at 10. Moreover, Chen explains that certain server-side agents control activity on the UDB server (*data store*):

Prefetchers: Prefetchers are present to **retrieve data from disk and move it into the buffer pool before applications need the data.** Without prefetchers applications needing to scan through large volumes of data would have to wait for data to be moved from disk into the buffer pool. Their effectiveness is increased by having multiple disks in a DB2 tablespace. **The data is striped across multiple disks and enables the prefetchers to access multiple disks in parallel to retrieve data.**

Page cleaners: Page cleaners are present to make room in the buffer pool, before agents and prefetchers **read pages from disk storage and move them into the buffer pool.** For example, if an application has updated a large amount of data in a table, many of the updated data pages in the buffer pool may not yet have been **written on to disk storage** — such pages are called dirty pages. Since prefetchers cannot place fetched data pages on to the dirty pages in the buffer pool, these dirty pages must first be flushed to disk storage and become “clean” pages, so that prefetchers can find room to place **fetched data pages from disk storage.**

Ex. 1105 at 11-12.

77. As discussed previously, in light of the disclosure of relational databases that require the use of and translation to SQL and the disclosure of data flow from hard disks through buffer pools that streamline system performance to client applications, it would be obvious to a POSITA that the system described in Chen must comprise an interface between the relational database management system and the variety of customer applications within the DB2 System. *See supra* Section IX.A.1.i(A). Further, Figure 1-2 the DB2 System, including UDB Server (the *data store*) interacts with clients (and client applications) Therefore, in light of the above discussion, a POSITA would therefore understand that, in Chen, the coordinator agents act as the interface between client applications (e.g., those submitting SQL queries) the UDB Server, using protocols like TCP/IP.

78. Accordingly, Chen discloses “*an interface*” (the coordinator agents) “*executing on the computing system*” (executing on the hardware running UDB Server) “*for communicating between a data store*” (the UDB server) “*and applications that supply and consume data*” (client applications).

- 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:***

79. Chen discloses “*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.*”

**(A) a real time information director (RTID)
executing on the computing system**

80. Chen discloses “*a real time information director (RTID) executing on the computing system.*” A POSITA would have understood the UDB Server, including DB2 XML Extender in Chen to be the “*RTID.*”

81. The 575 Patent describes a real time information director (RTID) as “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].

82. As discussed above, the UDB Server of Chen is a *data store*. See *supra* Section IX.A.1.ii.

83. Further, Chen discloses that UDB Server makes data accessible for supply and consumption, as it can “stor[e] the XML documents in a centralized repository” and has “the ability to quickly search for information or to trigger automatic data interchange when a particular action occurs.” Ex. 1105 at 318–319. The component of the DB2 server responsible for making such data accessible is the DB2 XML Extender, as described below.

84. As Chen explains, the DB2 XML Extender “provides the ability to store and access XML documents,” “compose XML documents from...existing relational data,” and “decompose XML documents into relational data.” Ex. 1105 at 318. In other words, the XML Extender is responsible for most of the XML document input and output to the UDB Server. The 575 Patent describes “[t]he real time information director is responsible for most or all data input and output to the real-time data store.” Ex. 1101 at 3:24-26. A POSITA would therefore understand that this DB2 XML Extender is an RTID as described by the 575 Patent.

85. Further, as was discussed in Section IX.A.1.i(B), the UDB Server, which includes the DB2 XML Extender (*real time information director*), *executes on a computing system*.

86. Accordingly, Chen discloses “a real time information director (RTID) executing on the computing system.”

(B) transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata

87. As explained below, Chen discloses that the UDB Serer, including DB2 XML Extender (*i.e.*, the real time information director) “*transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata.*” 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. A POSITA would have understood that, in Chen, the UDB Server incorporating the DB2 XML Extender performs the transformations described “*for supply, consumption, or both by the applications.*”

88. As I explain above regarding claim construction, I have adopted Patent Owner’s asserted meaning of “*polymorphic metadata*,” *i.e.*, “metadata defined through a polymorphic mechanism, such as hierarchal inheritance.” Ex-1108, ¶ 130..

89. The DB2 System discloses polymorphism in the sense that it has the ability to use user-defined types that are derived from, or inherit from, predefined

types. Ex. 1105 at 174 (“A user-defined type (UDT) is a data type that you derive from existing data types. UDTs enable you to extend the built-in types already available and create your own customized data types.”). *See, e.g.*, Ex-1111, 31.

90. Chen details several implementations of *polymorphic metadata*. As discussed above, the UDB server, which includes the DB2 XML Extender (the RTID) “*transforms data*.” *See supra* Section IX.A.1.iii(A). As Chen explains, “[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 at 319. “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 at 10.

91. Chen further explains that 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.**

...

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.**

Ex. 1105 at 319–320. Accordingly, Chen discloses that XML Documents are transformed into new data types, such that only the relevant values are stored based on user-defined datatypes (e.g., the mapping stored in the DAD file). The 575 Patent mentions, “metadata can be represented in eXtended Markup Language (XML), which is also polymorphic.” Ex. 1101 at 36:48-49. Chen mentions that “[t]he DAD is an XML formatted document.” Ex. 1105 at 320. XML documents are *data*. 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.”). In my opinion, a POSITA would thus have understood these disclosures to describe “*transform[ing] data*” (XML documents) ...*under direction of polymorphic metadata*” (e.g., based on the definitions in the DAD files).

92. Chen also teaches using a Document Type Definition (DTD) repository to “validate the XML documents you are storing within the DB2.” Ex. 1105 at 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 at 320. “You can insert your own DTDs into this table.” Ex. 1105 at 320. A POSITA would have understood that a given DTD repository could store different DTDs for various types of documents in one repository, where each document type has its own schema, and that all documents can be validated using different DTDs. In other words, these DTD repositories define classes by document type for extracting metadata into the database. In my opinion, a POSITA would have understood this to be another implementation of polymorphism for this metadata.

93. Finally, Chen discloses that the 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 at 2. Chen notes that “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 at 318. A POSITA would thus understand, and the 575 Patent admits, that “metadata” (e.g., XML tags) “can be represented in eXtended Markup Language (XML), which is also polymorphic.” Chen also notes

that the use of Java can “be used to create DB2 objects, such as stored procedures and user defined functions.” Ex. 1105 at 186. Like with XML, a POSITA would thus understand, and the 575 Patent admits, that “[a]s an object-oriented language, Java supports polymorphism.” Ex. 1101, [36:25-26]. In other words, in my opinion, a POSITA would have understood that Chen’s use of both XML and Java is polymorphic as taught by the 575 Patent (and common practice when realizing modern software implementations).

94. Regarding the “*supply, consumption or both*” of data “*by the applications*,” Chen discloses that, “[o]n the server side, activity is controlled by Engine Dispatchable Units (EDUs)” which “include coordinator agents, subagents,...and prefetchers.” Ex. 1105 at 11. As discussed in Section IX.A.1.ii, this includes the DB2 coordinator agents and subagents which enable communication and processing of the data to and from client applications. A POSITA would thus understand that the UDB Server, including the DB2 XML Extender (i.e., the RTID) performs the above transformations (e.g., DAD mapping scheme, DTD repositories and validation, and use of XML and Java languages for more advanced multimedia data) “*for supply, consumption, or both*” (communication) “*by the applications*.”

95. To summarize, the DB2 XML Extender “*transforms data*” by compiling it into and extracting it from XML documents based on (*under the direction of*) definitions in DAD and DTD files (*polymorphic metadata*). The

transformed data is used to store the extracted XML data in a database or to compile data from the database into XML documents (*for supply, consumption, or both*).

96. Accordingly, Chen discloses “*transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata.*”

(C) that defines a security model and data integrity rules for application to the data

97. Chen discloses transforming data under direction of polymorphic metadata, “*that defines a security model and data integrity rules for application to the data.*” As explained below, a POSITA would have understood that, in Chen, 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.

98. A POSITA would have understood that data integrity includes at least “map[ping] the incoming data to the data model” because the ’575 Patent discloses this process “cleanses and enriches data, provides data security, and insures data integrity.”

99. As discussed above, Chen explains that the combination of DTD repositories and DAD files, provide “[d]ocument validation.” Ex. 1105 at 320; *see supra* Section IX.A.1.iii(B). The DB2 XML Extender (i.e., the RTID) uses DTD to store incoming data, and it uses DAD to map the XML document to relational data.

Ex. 1105 at 320. With regards to security, it would be obvious to a POSITA that metadata can consist of authentication information and the mapping scheme can pertain to encryption. This is especially true given that Chen discloses a concern for security, as the “tables [of the DB2 system] 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 at 10. Thus, separately and in combination, DTD and DAD methods “*define[] a security model and data integrity rules.*” Ex. 1101 at Claim 1.

100. Additionally, “DB2 uses a combination of external security services and internal access control information to secure the information kept in the database... Once a user is authenticated and has access to a DB2 instance, DB2...restrict[s] the user to the set of authorities and privileges that the user is assigned to.” Ex. 1105 at 211. To ensure data integrity, “the user can [be] specific in table design, [where] [t]he database manager enforces the constraints. DB2 contains [] four types of constraints.” Ex. 1105 at 173. Those constraints include “[u]nique constraints,” “[r]eferential integrity (RI) constraints,” “check constraints,” and “informational constraints.” Ex. 1105 at 173–174. Unique constraints ensure there are no duplicate values in a column. Ex. 1105 at 173. Referential integrity constraints create a linkage between keys and columns of different tables to “ensure[] that when actions happen on the parent table they are cascaded to the other table(s) that rely on

it.” Ex. 1105 at 173. Check constraints set restrictions on data added to a table. Ex. 1105 at 174. And informational constraints allow a user to specify whether a referential or check constraint is to be enforced or not enforced in order to provide quicker results. Ex. 1105 at 174. Notably, “[u]sing ‘not enforced’ [constraints]...can also result in erroneous results unless the checking of constraints is handled by some other source.” Ex. 1105 at 174. In other words, not enforcing the constraints enumerated in Chen risks compromising security and data integrity. In light of the above, a POSITA would have recognized these various constraints defined by polymorphic metadata of the XML Extender to be ensuring security and data integrity as described by the 575 Patent.

101. Accordingly, Chen discloses transforming data under direction of polymorphic metadata (the DAD files) “*that defines a security model and data integrity rules for application to the data*” (e.g., by the DAD files enacting data integrity rules, like table designs that forbid duplicative values).

**(D) the RTID to dynamically instantiate, create, and
cache for reuse disassemblers and assemblers
according to demand**

102. Chen discloses “*the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand.*” As explained below, a POSITA would have understood DB2 agents disclosed in Chen to be functioning

as the “*assemblers*” and “*disassemblers*” of the ’575 Patent by using the appropriate DAD file or DTD repositories.

103. The 575 Patent at least explains that a “document disassembler” “contains prepared statements for inserting or updating various records related to a particular class of inbound documents into the data store.” Ex. 1101, [3:48-51]. “A ‘document assembler’ contains prepared statements for selecting various records related to a particular class of outbound documents from the data store.” Ex. 1101, [3:51-54]. Both “[a]ssemblers and disassemblers are created on-the-fly, based entirely on the descriptive metadata and on the schema of the tables read on demand from the database.” Ex. 1101, [3:59-61].

104. As discussed in Section IX.A.1.ii, the DB2 XML Extender can “compose XML documents from the existing relational data [and] decompose XML documents into relational data.” Ex. 1105 at 318. “It does so by providing a set of new data types, functions, and stored procedures.” Ex. 1105 at 318. This transformation of data is done on the server-side of the system, as mentioned in Section IX.A.1.iii(B), and “is controlled by Engine Dispatchable Units (EDUs),” where coordinator agents (called DB2 Agents) and subagents carry out the communication and processing—e.g., composition and decomposition—of data to and from the client applications. Ex. 1105 at 11; *see supra* Section IX.A.1.iii(B).

105. As discussed further below, a POSITA would have recognized that in Chen, the coordinating agent performing the composition of XML documents is an “*assembler*” and the coordinating agent performing the decomposition of XML documents is a “*dissembler*” as defined by the 575 Patent. *See infra* Sections IX.A.1.iv-v. In short, it would be obvious to a POSITA that the coordinating agent performing the composition of XML documents in Chen is an “*assembler*” as used by the ’575 Patent. For example, when a coordinating agent performs composition of XML documents, it uses “existing relational data” such as “the data stored in columns of DB2 tables” for selecting various records related to a particular class of XML documents. Ex. 1105 at 318-320. In this example, “mapping between data in the DB2 tables and the composed XML document is done using the DAD file,” which defines the particular class of documents and various records that need to be selected. Ex. 1105 at 320. In this way, the composing coordinating agent is an “*assembler*” as used in the 575 Patent’s claims. Similarly, a “*disassembler*” is the coordinating agent performing the decomposition of XML documents. For example, when a coordinating agent performs decomposition of XML documents, it “decompose[s] XML documents” (disassembles inbound documents) “into relational data” (into various records) that can be stored in DB2 tables located on the UDB server (into the data store). Ex. 1105 at 318–319. In this way, the decomposing coordinating agent is a “*disassembler*” as the term is used by the ’575 Patent.

106. Because the data stored on the hard disks is in DB2 tables, a POSITA would have further recognized that the composition and decomposition (i.e., processing) done by the coordinator agents must be done according to the respective DAD file and/or DTD repositories—according to definitions and restricts set by those files and/or repositories.

107. Chen also discloses that the EDUs, which include the coordinator agents (DB2 Aents) that are assemblers and disassemblers, are “implemented as threads in a single process on Windows-based platforms and as processes on UNIX.” Ex. 1105 at 11. A POSITA would have recognized from experience and education there are a number of benefits, particularly for a relational database software, to dynamic instantiation, creation, and cache for reuses of client requests, including scalability, load balancing, fault tolerance, and performance optimization. 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). *See* Ex-1111, 221 (“fork” means “to initiate a child process in a multitasking system after a parent process has been started”).

108. Chen discloses “*the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand.*”

109. Accordingly, Chen discloses “*a real time information director (RTID)*” (UDB Server, including XML Extender) “*executing on the computing system*” (the hardware that UDB Servers runs on) “*that transforms data for supply, consumption, or both by the applications*” (that composes or decomposes XML documents based on requests from the client applications) *under direction of polymorphic metadata*” (composition and decomposition of XML documents) “*that defines a security model and data integrity rules for application to the data,*” (based on the DAD files and DTD repositories) “*the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand*” (initiate DB2 coordinating agents as the assemblers and disassemblers for each client request).

- 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***

110. For reasons described above, (*see supra* Sections IX.A.1.ii-iii) and further reasons below, Chen discloses “*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.*”

111. As discussed in Section IX.A.1.iii(D), the coordinator agents (DB2 Agents) are disassemblers. More specifically, the coordinator agents of the DB2 system disassemble XML documents into tables and structures defined by a DAD

file or DTD repository (*polymorphic metadata*). Also, as described in Section IX.A.1.iii, the *polymorphic metadata*, DAD files and DTD repositories, direct the disassembly of the incoming XML documents for insertion into 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 at 319. Once the data is decomposed, the user can work with it as they “would work with any relational data in DB2.” Ex. 1105 at 320.

112. A POSITA would have understood that populating the DB2 tables (*insertion into metadata tables*) starting from an XML document (*inbound data document*), with only the values of elements and attributes of interest and without storing the XML document itself, would use a *disassembler*—here, a decomposition coordinator agent. The decomposition and mapping of the XML document and DB2 tables is given by the DAD file or DTD repository (*according to polymorphic metadata*) and the relevant values (*multiple records*) are extracted for insertion in tables (metadata tables) that DAD file has identified for storing the extracted information on the DB2 Server (*in preparation for insertion into metadata tables*).

113. Accordingly, Chen discloses, “*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.*”

- 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.***

114. For reasons described above, (*see supra* Sections IX.A.1.ii-iii) and further reasons below, Chen discloses “*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.*”

115. More specifically, Chen describes coordinator agents using DAD files or DTD repositories, comprised of polymorphic metadata, to direct the assembly of relational data from the DB2 database and compose an XML document for supply or consumption by the client application. *See supra* Section IX.A.1.iii. As described, “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.” Ex. 1105 at 320. Chen further describes that the *assembler* can compose outbound XML documents from a selected DB2 table:

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 at 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.”). Chen mentions that “[t]he WebSphere MQ range of products...enable application programs to communicate with each other using messages and queues.” Ex. 1105 at 325. A POSITA would have understood that the XML document composed from DB2 data in an MQSeries message queue to be an *outbound data document*.

116. Chen discloses *assembl[ing] outbound data documents from at least one record selected from at least one metadata-selected table*. As discussed above (see *supra* ¶116), the *assembling* of XML documents can be done for an MQSeries message queue (*outbound data documents*). Also discussed above (see *supra* Section IX.A.1.iii(B)), the DAD file, composed of *metadata* describes the mapping between

the XML document (*outbound data document*) and DB2 *table[s]*. Since there may be more DB2 tables for other documents, a POSITA would have understood that only relevant values (*at least one record*) *selected from* relevant tables, where relevant tables are determined by the DAD file metadata (*at least one metadata-selected table*).

117. It would have been obvious to a POSITA that the coordinator agents are “assemblers” as defined by the 575 Patent. Here, a POSITA would have recognized composing the outbound XML documents from relational data in the DB2 tables (*assembles outbound data documents from at least one record selected from at least one metadata-selected table*) as “processing” the data. Further, it would have been obvious to a POSITA that assembly of such data is directed through the use of DAD files and DTD repositories (*according to the polymorphic metadata*), which, as described in Section IX.A.1.iii, *transform[]* such data *for supply, consumption, or both*.

118. Accordingly, Chen discloses “*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.*”

2. Claim 2

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

119. For the reasons explained above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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.*

120. Chen discloses “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.”

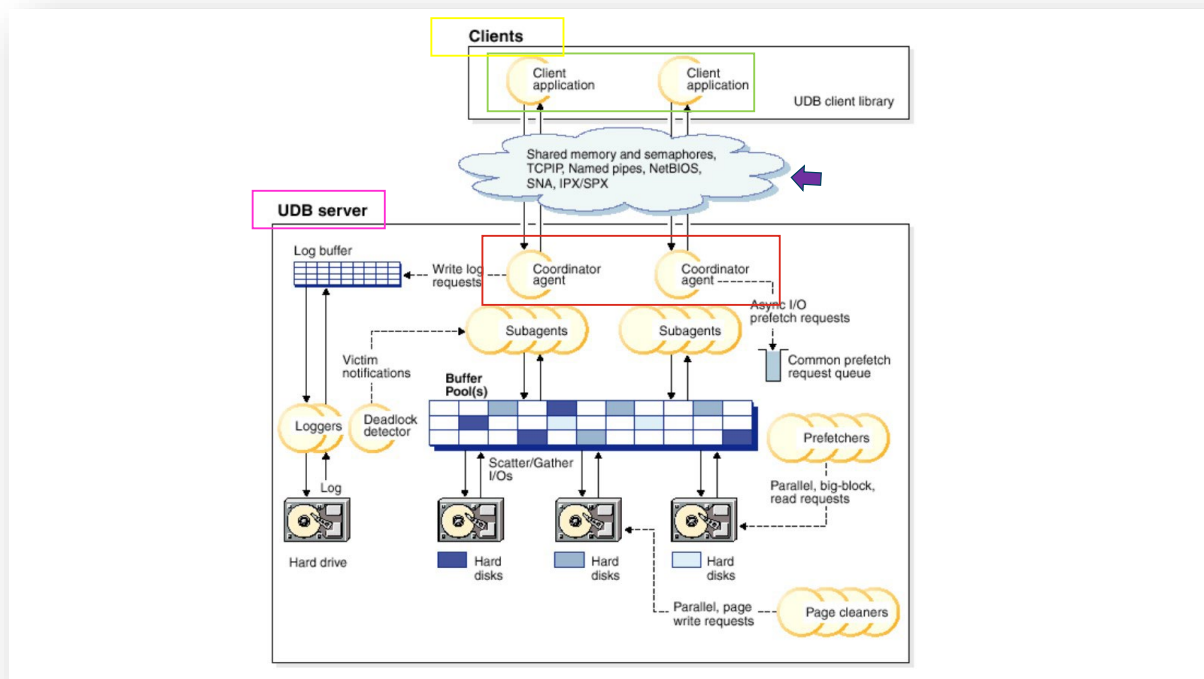
121. A POSITA would have understood that the DB2 XML Extender coordinator agents disclosed in Chen are “a data mapper” as described in Claim 2 of the ’575 Patent.

122. First, as discussed above (*see supra* Sections IX.A.1.ii-iii) the DB2 system, comprising of the DB2 XML Extender, includes coordinator agents that disassemble incoming documents and decompose that data into DB2 tables based on mappings and schemes from a DAD file and/or DTD repository, respectively. Also as previously discussed in Sections IX.A.1.ii-iii, the incoming documents can be

XML documents of various formats or schema, each having different attribute and element tags and definitions based on a given DAD file and/or DTD repository:

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 at 319-320. Figure 1-2, depicts a number of client applications that communicate with the agents, confirming that data can be collected from a plurality of diverse applications:



Ex. 1105, Fig.1-2 (annotated).

123. Chen further explains that the format of the XML documents can vary and are determined by the author by use of tags:

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....XML is a structured data format, which allows it to store complex data, whether it is originally textual, binary, or object-oriented.

Ex. 1105 at 318.

124. In sum, the DB2 XML Extender coordinator agents assemble and disassemble documents and relational databases (*collecting data...and mapping the supplied data*). Each document can be in a different format (*from a plurality of diverse applications*), whereby the relevant attributes and elements are extracted and inserted into corresponding entries in DB2 tables (*that can supply data in multiple diverse external document formats*). And mapping is dictated by the DAD files and/or DTD repositories (*to a predetermined unified data model according to the polymorphic metadata*). Thus, it would be obvious to a POSITA that the DB2 XML Extender coordinator agents disclosed in Chen are “a data mapper” as described in Claim 2 of the 575 Patent.

125. Accordingly, Chen discloses “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.”

3. Claim 4

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

126. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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

127. Chen discloses “*the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.*”

128. For reasons described above (*see supra* Section IX.A.1.ii, IX.A.1.iii, and IX.A.1.v) Chen discloses limitation 4.1.

129. More specifically, as was described in Section IX.A.1.iii, a POSITA would have understood that the UDB Server implementation of the DB2 XML Extender discloses an RTID. Chen discloses that this DB2 XML Extender (RTID) uses the same DAD file, which consists of *polymorphic metadata*, to direct coordinating agents to assemble and disassemble XML documents from and to 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 at 319-320.

130. Accordingly, Chen discloses “*the RTID using a same descriptive polymorphic metadata to build both assemblers and disassemblers for a same document class.*”

4. Claim 6

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

131. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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.***

132. Chen discloses “*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.*”

133. 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. As I explained above, I have adopted Patent Owner’s definition of “*logic*,” i.e., “*software*.” Because the RTID is a software, a POSITA would have thus understood Chen to teach *a logic*.

134. A POSITA would have known that Chen's DB2 XML Extender “represents metadata as polymorphic objects . . . according to a predetermined object class.” As I discussed above regarding limitation [4.1] (see Section IX.A.3.ii), Chen teaches that 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.

135. A POSITA would have understood a “class” in object-oriented programming to be a generalized category of objects. See, e.g., Ex-1111, 100. Likewise in object-oriented programming, a POSITA would have understood “polymorphism” to include the ability to redefine a routine in a derived class. See, e.g., *id.* 411. Consequently, 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).

136. A POSITA would have understood that using the DAD file (“*polymorphic metadata*”) in Chen to both (1) populate DB2 tables from a provided XML file, and (2) to create an XML document based on information stored in those

DB2 tables is *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).

137. Accordingly, Chen discloses “*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.*”

5. Claim 7

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

138. For the reasons I explain above (see *supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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.**

139. Chen discloses “*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.”

140. For the reasons mentioned above (*see supra* Sections IX.A.1.ii, IX.A.1.iii, IX.A.1.iv, IX.A.1.v, and IX.A.5.i), Chen discloses limitation 7.1.

141. More specifically, Chen discloses that the DAD files define the mappings of elements and attributes that will be applied to an XML document structure. Ex-1105, 319-320. As Chen explains, the DB2 tables and communications with the client applications delegate operation to the DAD files:

A DADX [(Document Access Definition Extension (DADX)] document specifies a Web Service using a set of operations that are defined by SQL statements or XML Extender Document Access Definition (DAD) documents. Web services, or functions invoked over the Internet, specified in a DADX file are called DADX or DB2 Web services.

Ex. 1105 at 322. As mentioned in Section IX.A.1.iii, the DAD files are polymorphic objects representing metadata. As explained in Sections IX.A.1.iii-iv, these DAD files are used to instantiate DB2 agents (e.g., assemblers and disassemblers). As explained for Section IX.A.23.ii, 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 would have recognized the DAD files to be stateless, in that they do not have internal state: their attributes do not change over time, method calls and interactions are independent of previous calls or interactions, and behavior is consistent for identical inputs—they are not modified after they are initiated. A POSITA would also have recognized that since these DAD files are reused for defining the representation for performing operations on the data, they must be “permanent” after initial creation. Thus, the DB2 agents instantiated from DAD files are “*metadata as stateless polymorphic objects that are permanent after initial construction.*”

142. Further, as explained for claim limitation [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.

143. Accordingly, Chen discloses “*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.*”

6. Claim 10

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

144. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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

145. Chen discloses “*the real time information director receives and publishes data documents having a content and a format defined by document metadata.*”

146. As I discussed for limitation [1.4] above, Chen teaches assemblers that create data documents (e.g., XML documents) from stored information. Ex-1105, 319-320.

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

148. Chen also discloses a Document Type Definition (DTD) repository, which “provides an XML DTD repository to store these DTDs. When a database is enabled for XML, a DTD reference table called DTD_REF is created. Each row in this table contains a DTD with additional metadata information about it.” Ex. 1105 at 320.

149. Thus, DB2 XML Extender (*RTID*) *receives* XML documents (*data documents*) for further processing. Moreover, Chen describes “map[ing] the XML document to relational data,” that “a database is enabled for XML,” and that “DB2 UDB V8 has introduced new built-in functions to allow for relational data to be queried and published as to XML” (*publishes data documents*). Ex. 1105 at 320. Additionally, “a DTD reference table called DTD_REF is created” for the XML document, and “[e]ach row in this table contains a DTD with additional metadata information about it,” (*data documents having a content and a format defined by document metadata*). Ex. 1105 at 320.

150. Accordingly, Chen discloses “*the real time information director receives and publishes data documents having a content and a format defined by document metadata.*”

7. Claim 11

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

151. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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

152. For reasons mentioned above (*see supra* Section IX.A.1.iii and IX.A.1.v), Chen discloses “*a logic that reads a schema of tables identified in*

metadata.” Because the DAD file in Chen identifies fields in the database, POSITA would have understood this to teach that DB2 agent *reads a schema of tables identified in metadata*. Ex-1111, 465 (“schema defines aspects of the database, such as attributes (fields)”)

153. More specifically, when discussing “*schema*,” the ’575 Patent discusses “a generator logic 310 that generates an eXtended Markup Language (XML) Schema that describes format of the documents.” Ex. 1101 at 9:39-41. The DAD file of the DB2 system in Chen does just that, as it is “used to map the XML document to relational data” and “is an XML formatted document which allows you to associate XML document structure to a DB2 database.” Ex. 1105 at 320. The DAD file contains “[t]he mapping between elements and attributes in the XML document and the DB2 tables.” Ex. 1105 at 320. There are DB2 rules, such as unique constraints “that forbid duplicate values in one or more columns within a table.” Ex. 1105 at 173. Additionally, as mentioned in IX.A.4.ii, there are pre-defined data types and user-defined data types derived from existing data types. A POSITA would therefore have recognized that the DB2 system, comprising the DB2 XML Extender, uses schemas of tables, such as mapping between elements and attributes, data types and cardinality.

154. Accordingly, Chen discloses “*a logic that reads a schema of tables identified in metadata.*”

- 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.***

155. Chen discloses “*a logic that automatically generates and caches prepare SQL statements from a content metadata portion of document metadata according to the schema.*” A POSITA would have understood Chen 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-1105, 156, 181, 298.

156. For all the same reasons described above regarding claim limitation 11.1 (*see supra* Section IX.A.7.ii), 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)..

157. Chen further discloses that XML Extender may be used with SQL. 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.”).

158. Accordingly, Chen discloses “*a logic that automatically generates and caches prepare SQL statements from a content metadata portion of document metadata according to the schema.*”

8. Claim 12

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

159. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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):*

160. Chen discloses “*a servlet that invokes the real time information director synchronously when an inbound data document is delivered via HyperText Transport Protocol (HTTP).*”

161. Chen discloses that “DB2 UDB is integrated with Web technology so that data can be easily accessed from the Internet or from your intranet to provide a more secure environment.” Ex. 1105 at 2. This includes communication of data by HTTP. Ex. 1105 at 146. Chen also teaches that “the focal point of [the] application environment is an Apache Web Server which can interact with a user’s web browser and provide DB2 connectivity via the PHP scripting,” meaning the system of Chen integrates with the Apache web server using PHP to script servlets for interacting with DB2. Ex. 1105 at 112; *see also* Ex. 1111 at 475 (“Servlets perform lightweight web services...”).

162. As explained above for limitation [1.2] (*see supra* Section IX.A.1.iii), the DB2 UDB server includes a DB2 XML Extender (*real time information director*), which “provides the ability to store and access XML documents, and . . . decompose XML documents into relational data.” Ex. 1105 at 318. The “*servlets*” (e.g., a PHP script) Since the DB2 UDB server is a data store for storing documents, and “integrated with Web technology so that data can be easily accessed from the Internet” capable of data communication via HTTP, it would have been obvious to a POSITA that the DB2 XML Extender would be *invoked* for decomposing the documents as documents are delivered via HTTP at the server. Additionally, a POSITA would have understood “synchronous communication” to mean “[c]omputer-to-computer communication in which transmissions are synchronized

by timing between the sending and receiving machines,” which applies to HTTP communications in this context between the requestor must wait for a response. *See* Ex. 1111 at 506 (defining “synchronous communication”); *see also* Ex. 1101 at 3:31-34 (“The real time information director is accessed synchronously via HyperText Transport Protocol (HTTP)”). Thus, within the context of the 575 patent, a POSITA would have known that HTTP is an example of synchronous access.

163. Accordingly, Chen discloses “*a servlet that invokes the real time information director synchronously when an inbound data document is delivered via HyperText Transport Protocol (HTTP).*”

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);

164. Chen discloses “*a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS).*”

165. For all the same reasons described above regarding claim limitation 12.1 (*see supra* Section IX.A.8.ii), Chen discloses this limitation. Chen further discloses that delivery of data may be accomplished by Java. For example, with respect to “connecting to a remote DB2 server,” Chen discloses that “capability offered by Java applets and applications . . . allows them to connect directly to DB2 without DB2 client code.” Ex. 1105 at 176. Chen teaches a Java Database

Connectivity application (JDBC) which “can be especially useful for accessing DB2 databases across the Internet.” Ex. 1105 at 182. And Chen discloses that “Java’s portability enables you to deliver DB2 access to clients on multiple platforms, requiring only a Java-enabled Web browser, or a Java runtime environment.” Ex. 1105 at 182. A POSITA would have understood that one standard Java method for providing data to applications was Java Message Service (JMS). 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 thus understood that exchange of data with Java applets to teach or suggesting using *Java Message Service*, which provides asynchronous communication. 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).”).

166. Accordingly, Chen discloses “*a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS).*”

- 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***

167. Chen discloses “*a servlet that invokes the real time information director synchronously when an outbound data document is requested via HyperText Transport Protocol (HTTP).*”

168. For all the same reasons described above regarding claim limitation 12.1 (*see supra* Section IX.A.8.ii), Chen teaches or suggests *a servlet that invokes the real time information director synchronously ... via HyperText Transport Protocol (HTTP).*

169. Further, as I discussed above regarding limitation [10.1], Chen teaches publishing XML document based on information stored in DB2 tables. There, Chen taught the use of a servlet to pass data synchronously between a client and the DB2 database over the internet (*see also* [12.1]). A POSITA would therefore have understood Chen to teach 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).

- 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).***

170. Chen discloses “a logic that invokes the real time information director asynchronously when an outbound data document is requested via Java Message Service (JMS).”

171. For all the same reasons described above regarding claim limitation 12.1 and 12.2 (*see supra* Section IX.A.8.ii and IX.A.8.iii), Chen discloses this limitation.

172. Given the further teachings Chen I discuss above 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).

9. Claim 13

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

173. As I explained above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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.***

174. For all the same reasons described above regarding claim limitation 10.1 (*see supra* Sections IX.A.6.ii), Chen discloses “*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.*”

175. Chen also discloses DB2 replication, which includes a Replication Center that “is used to map a source table or view to a target table.” Ex. 1105 at 313. As Chen explains:

You define a subscription set which is a group of one or more target tables (called subscription members) that will be processed as a unit by Apply. The changes from the CD tables are applied for each table separately, in the same order they occurred at the source...[where] the schedule for replication...can be either time-based or event based.

Ex. 1105 at 313.

176. A POSITA would have known that subscription handlers—namely, a publish/subscribe or post-and-subscribe paradigm—ensure that all subscribers are notified when a specific event occurs, particularly in event-driven or time-driven

systems. In the case of DB2, a POSITA would have recognized that, at either a set time or given event, the system notifies subscribers by applying changes from the CD tables. Therefore, a POSITA would have recognized that, in either paradigm, there must be a subscription handler to manage applying changes to the tables.

177. Further, Chen discloses implementing a messaging queue for transmitting XML documents. Ex-1105, 326. A POSITA would have understood this to teach or suggest constructing the XML document and then sending it to a message queue. *See* 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”).

178. As I explained regarding limitation [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).

179. Accordingly, Chen discloses “*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.*”

10. Claim 14

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

180. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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;***

181. Chen discloses “*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*” because Chen contains a database manager containing configurations for restricting client access (“*security manager*”), and also requires that “a client’s user ID and password are sent to the server for authentication” (“*identif[ying] an agent on behalf of whom information is queried or updated*”), retrieves XML documents from a database (“*performing a document request*”), user tables with at least one column of unique values that may generate a primary key (“*dynamically assign[ing] an internal unique identifier to consumers and agents upon registration*”).

182. Because Chen's "database manager" contains configurations for restricting client access, a POSITA would have understood it to be *a security manager*.

183. With respect to *the real time information director grant[ing] access to individual documents based on security policies*, Chen discloses that "DB2 uses a combination of external security services and internal access control information to secure the information kept in the database." Ex. 1105 at 211. Chen also teaches that DB2 "provide[s] security by restricting the user to the set of authorities and privileges that the user is assigned to." Ex. 1105 at 211. As Chen explains:

To access a database server, a user must be authenticated either on the client or the server. 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 at 213. Chen further discloses that:

Once 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. There are three types of authorities which are assigned to groups by the instance owner (on Linux) or an Administrative user

(on Windows). These authorities are available so that a user can maintain database objects at a high level, such as the database or tables.

Ex. 1105 at 213.

184. 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. A POSITA would have understood that the client could represent anyone, including a *user*, or an *agent* of a user. 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-1105, 211-12.

185. 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, ¶ XXX.

186. Chen teaches or suggests maintaining “user tables” (Ex-1103, 10) and that each table should contain at least one column of unique values that function as “primary keys.” Ex-1005, 173, 176 (describing “enforcing the uniqueness of a record by using a primary key”). A POSITA would have understood that a primary key would have been generated automatically when a new entry was made in the user table. 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).

- 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.***

187. Chen teaches “*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.*”

188. Chen’s security system is customizable, “us[ing] a combination of external security services and internal access control information to secure the information kept in the database.” Ex. 1105 at 211-212. For example, “DB2 continues to provide security by restricting the user to the set of authorities and privileges that the user is assigned.” Ex. 1105 at 211.

189. Chen further 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 at 213. In the Chen system, there are three types of authorities available “so that a user can maintain database objects

at high level, such as the database or tables.” Ex. 1105 at 213. Each of these is an example of “*security policies*.”

190. As another example, Chen explains the use of “authorization groups.” Ex-1103, 197. A POSITA would have understood that members of an “authorization group” would be able to act on each other’s behalf regarding group resources. This could include agents and a member of the group could be anyone (e.g., a consumer or an agent of a consumer). A POSITA would have thus 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).*”

191. 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:***

192. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- i. ***[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.***

193. Chen discloses “*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.*”

194. First, Chen’s Database Manager has a set of configurations, which includes auditing. Ex. 1105 at 197–98. A POSITA would have understood an “audit” to be an examination of activities to determine how efficiently the system is performing, especially in terms of ensuring the integrity of data. *See, e.g.*, Ex-1111, 42.

195. Chen also mentions that “[a]ny concurrent transactions that update the [DB2] table are logged [and] [o]nce the logged table changes have been forward-fitted and the new index (the shadow copy) is read, the new index is made available.” Ex. 1105 at 345. A POSITA would have recognized that forward-fitting changes often require auditing to different extents for a number of reasons: accountability and traceability, particularly for changes that impact critical systems or data (“DB2 is a database that can be trusted with your most critical applications.” Ex. 1105 at 2); compliance and regulatory requirements (DB2 can “control access to sensitive data.” Ex. 1105 at 175); risk management to minimize system failures and data breaches,

impact assessment, version control and change management; and security and access control (“Data [is distributed] to other locations...[to] maintain[] the security and performance of [] production applications.” Ex. 1105 at 310). A POSITA would have recognized that the optimal way to implement these features, including accountability, traceability, compliance and security and access control to name a few, and handle changes to the *data store* would be through an audit log handler.

196. A POSITA would further have recognized that concurrent transactions to the table are *changes in the data store*. As was discussed in Sections IX.A.1.iii(B) and IX.A.4.ii, DAD files and/or DTD repository files (*polymorphic metadata*) are used to *compil[e] and stor[e] information* in the *data store* (DB2 tables). A POSITA would have understood that the extracted metadata from XML documents (i.e., the DAD files), as discussed in Sections IX.A.1.iv-v, and the transaction log tracking database changes suggest that the compiling and storing of information is “*based on the metadata*” and “*determine[] the content and and detail of the compiled information.*” Ex. 1105 at 319 (“[O]nly the values of the elements and attributes you are interested in are stored in DB2.”).

197. Accordingly, Chen discloses “*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.*”

12. Claim 18

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

198. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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.***

199. For reasons described above (*see supra* Section IX.A.1.iii), Chen discloses “*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.*”

200. In database management, a POSITA would have understood “normalize” to be applying a body of techniques to a relational database in order to minimize the inclusion of duplicate information. *See, e.g.,* Ex-1111, 368 (“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.”). Chen discloses *a data cleanser that...normalizes the data elements, queues source and destination information for detection of duplicates, and cross-references duplicates in the data store.* Additionally, within the

DB2 system are page cleaners, which are responsible for “mak[ing] room in the buffer pool.” Ex. 1105 at 12. “If an application has updated a large amount of data in a table, many of the updated data pages in the buffer pool may not yet have been written on to disk storage—such pages are called dirty pages.” Ex. 1105 at 12. A POSITA would have known that dirty pages are identified and tracked by metadata, typically using flags like dirty bits, timestamps, and page IDs. A POSITA would also have recognized that as part of the process of writing data from the buffer to disk, the database manager would enforce unique constraints that, “forbid duplicate values in one or more columns within a table.” Ex. 1105 at 173. To impose this constraint, the system must *cross-reference[] duplicates in the data store*. Further, a POSITA would have known that the data stored in the buffer can be stored in a form different from the normalized form. Since buffer pools are responsible for temporarily holding data (whereas tables represent the actual structure and organization of data stored on disk), as part of the process of writing the data to disk, the data must be a normalized (*normalizes the data elements*). Also, a POSITA would also have known that *data cleanser[s]* often utilize queues to optimize data cleaning, and in order to enforce unique constraints while writing data from the buffer to disk, mentioned above, have understood that the *queue* would contain buffer (*source*) and disk (*destination*) information (*queues source and destination information*).

201. Chen discloses *receiv[ing] data elements identified by Document Metadata*. Chen outlines how XML documents are stored *in the data store*:

DB2 XML Extender provides two options you can use for integrating XML documents into DB2: XML Columns...XML Collections: This method allows you to...decompose XML documents into DB2 data. With the XML Collections method...[y]ou can populate the columns of one or more DB2 tables starting from an XML document...[O]nly 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 at 319-320. Similar to how, “a person may be identified using a blue cross number, a social security number, an account number, and a driver’s license number,” “the values” (*receive[d] data elements*) “of the elements and attributes you are interested in are stored in DB2. The mapping between elements and attributes...is given by the DAD file” (*identified by Document Metadata, see supra* Section IX.A.1.iii(B)) Ex. 1101, 21.

202. Since data ingestion is done prior to data recordation, it would have been obvious to a POSITA that the *data cleanser* mentioned above (*see supra* ¶188) *receives data identified by Document Metadata* (*see supra* ¶189).

203. Accordingly, Chen discloses “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.”

13. Claim 19

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

204. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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

205. Chen discloses “*an aggregation engine that computes aggregates in memory from data selected from the data store.*”

206. 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 at 316. As Chen explains, “[i]n general, a process moves source data into the warehouse”—or the *data store* of Chen. Ex. 1105 at 22. Chen also details an *aggregation engine* where “data is aggregated and summarized for warehouse use.” Ex. 1105 at 22. And this data aggregation process “can produce a single flat table or a set of summary tables.” Ex. 1105 at 22. A POSITA would have understood that the data selected to be aggregated would be from the storage memory of the warehouse.

207. Accordingly, Chen discloses “*an aggregation engine that computes aggregates in memory from data selected from the data store.*”

14. Claim 20

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

208. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

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

209. Chen discloses “*a real-time scoring engine.*”

210. Chen discloses DB2 Intelligent Miner™ Scoring, which “provides scoring technology as a DB2 extender.” Ex. 1105 at 362. Chen further explains that “[i]t extends database capabilities and enables users to deploy data mining in real-time applications with a simple SQL call.” Ex. 1105 at 362. And “DB2 Intelligent Miner Scoring enables your applications to better serve business and consumer users alike - by providing more informed recommendations, personalizing customer and supplier treatment, and building more efficiencies into the online experience to be more responsive to the specific situation at hand.” Ex. 1105 at 362. Thus, a POSITA would have recognized the DB Intelligent Miner to be a “*real-time scoring engine.*”

211. Accordingly, Chen discloses “*a real-time scoring engine.*”

15. Claim 21

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

212. For the reasons I explain above (*see supra* Section IX.A.1), Chen discloses the method of Claim 1.

- 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.***

213. Chen discloses “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.”

214. Chen discloses “DB2 Universal Database’s support for...business rules.” Ex. 1105 at 17. Chen explains that “DB2 UDB has particular strengths in supporting business intelligence applications such as data warehousing and on-line analytical processing (OLAP).” Ex. 1105 at 3. It further explains that “DB2 provides parallel database technology and query optimization resulting in the ability to help customers find competitive advantage, offer better customer service, or

reduce costs by mining their data for the knowledge required to make better decisions.” Ex. 1105 at 3. Additionally, one of the features in editing the DB2 tables, is “built-in...business rules.” Ex. 1105 at 372. As described in Sections IX.A.1.iii(B) and IX.A.4.ii the data stored in the DB2 tables is described by metadata in the DAD files and DTD repository files. Moreover, the system can “create an information catalog of the data in the warehouse...that contains business metadata.” Ex. 1105 at 315.

215. Further, a POSITA would have recognized that these rules are *specified by [a] business object model*. As Chen explains, “in supporting business intelligence applications such as data warehousing,” DB2 has a data warehouse center that “[p]erform[s] calculations to summarize the data [and] [w]rit[e] the summarized data to a target in a separate database.” Ex. 1105 at 314. Chen also states that, “[d]ata warehousing allows you to create stores of informational data, which is...extracted from operational data and then transformed for decision-making.” Ex. 1105 at 314. A POSITA would have known that, in the business intelligence context and when the goal is decision-making, summaries of data can—and often do—include conclusions based on computation results.

216. Finally, Chen discloses that the DB2 Data Warehouse Center “supports...application sources like WebSphere MQ queues” (Ex. 1105 at 314), which are an integral part of the “MQSeries Workflow,

[which]...enables...execution...of complete business processes that span systems, applications and people.” Ex. 1105 at 325. And with “DB2 Connect...[it] [i]ntegrates new Web-based applications with existing core business applications.” Ex. 1105 at 374. Additionally, key features to DB2 include “automation and self-management enhancements combined with [a] low price [that] provides remarkable value.” Ex. 1105 at 1. In light of the above discussed integration of automation, MQSeries Workflow and DB2 Connect, conclusions leading to a decision, it would have been obvious to a POSITA to execute the decision as well.

217. Accordingly, Chen discloses “*a business rules engine (i.e., warehouse agent) 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.*”

16. Claim 22

i. [22.0] An article of manufacture comprising:

218. Chen discloses “*an article of manufacture.*”

219. For reasons described above (*see supra* Section IX.A.1.i), Chen discloses limitation 22.0. “DB2, IBM’s premier relational database software...spans a wide variety of Linux, UNIX and Windows platforms and the IBM iSeries™...and zSeries™...server lines. DB2 UDB...can be deployed on a variety of hardware and

software platforms.” Ex. 1105 at 1-2. This “UDB architecture” comprises servers with hard drives:

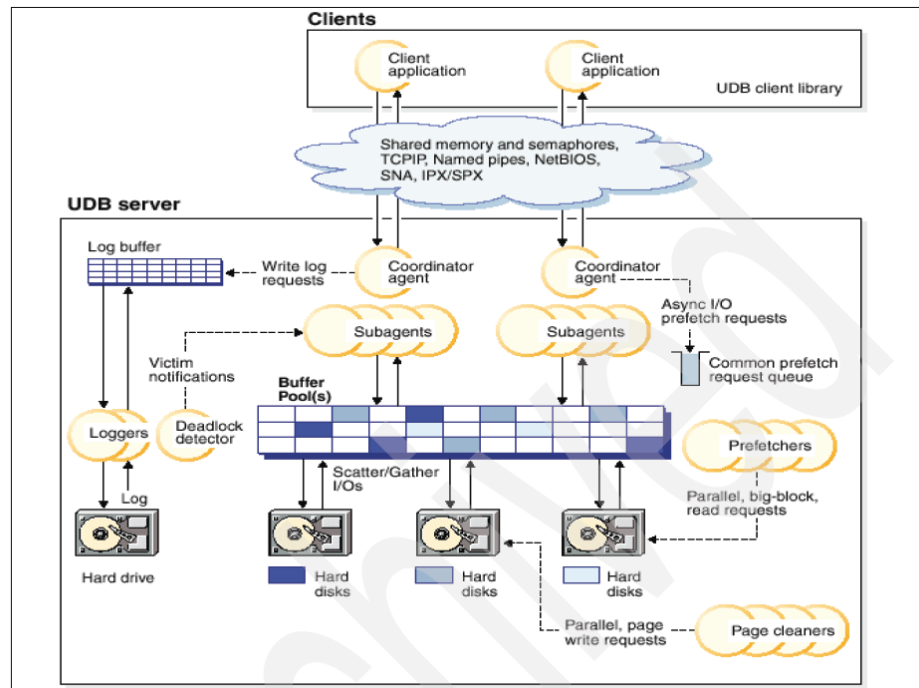


Figure 1-2 DB2 UDB architecture overview

Ex. 1105, Fig.1-2. A POSITA would have known that any software distributed on a tangible medium can be considered an article of manufacture, as it is deployable—and often involves installation—on the respective physical hardware through means such as a CD, DVD, USB drive or downloadable file.

220. Accordingly, Chen discloses “*an article of manufacture.*”

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:***

221. Chen discloses “*a controller-usable storage medium having a computer readable program code embodied therein for managing data services.*”

222. The UDB DB2 system “is a high performance middleware software that provides a secure and reliable system to manage data or content for a wide variety of customer applications.” Ex. 1105 at 2. Given that it is a relational database software that is middleware software (*computer readable program code*), a POSITA would have known that the middleware sits between an application and the database and is often integrated into the database server and stored on a hard disk or other types of storage devices (*controller-usable storage medium*) similar to most software applications. See Ex-1111, 132 (defining the central processing unit (“CPU”) as the control unit).

223. Chen discloses *a computer readable program code...for managing data services.*” According to Merriam-Webster dictionary, manage is defined as “to handle or direct with a degree of skill.” As described above (*see supra* Section IX.A.1.i(A)), the DB2 system (*computer readable program code*) comprises “a data services handler” (*for managing data services*).

224. Accordingly, Chen discloses “a controller-usable storage medium having a computer readable program code embodied therein for managing data services.”

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

225. Chen discloses “a code causing the controller to transfer data among a data store and applications that supply and consume data.”

226. For reasons described above (*see supra* Section IX.A.1.i, IX.A.1.ii and IX.A.16.ii), Chen discloses limitation 22.2. As described above (*see supra* Section IX.A.1.ii), the interface for communicating between (to *transfer data among*) a data store and applications that supply and consume data is the coordinator agents of Chen. A POSITA would have known that in Software Design, components can function as both an interface and *controller*. A POSITA would also have known that a *controller* can process SQL queries and manage data access, “DB2 agents, [that] include coordinator agents and subagents...carry out the bulk of SQL processing on behalf of the applications. DB2 assigns a coordinator agent with an application, and this agent coordinates the communication and processing for this application.” Ex. 1105 at 11. In light of the above discussion, it would have been clear to a POSITA that the *controller[s]* (“coordinator agents and sub agents”) *transfer data among*

[the] data store and [said] applications that supply and consume data caused by the DB2 UDB software (see supra Section IX.A.16.ii).

227. Accordingly, Chen discloses “*a code causing the controller to transfer data among a data store and applications that supply and consume data.*”

- 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;***

228. Chen discloses “*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.*”

229. For reasons described above (see supra Section IX.A.6.ii and IX.A.16.iii), Chen discloses limitation 22.3. Furthermore, as mentioned in Section IX.A.1.iii, the DB2 system uses DAD files and DTD repository files to convert between XML documents and DB2 tables for transfer between applications and the data store. A “coordinating agent...coordinates the communication and processing for [an] application [and] [t]he database agent performs the necessary operations on the data pages in the buffer and returns the result to the user application.” Ex. 1105

at 13. A POSITA would have recognized “the necessary operations on the data pages in the buffer” to include *transform[ing] [the] data for transfer* using the DAD files or DTD repository files (*representation of metadata as polymorphic objects, see supra* Section IX.A.1.iii(B)) and, *enabl[ing] 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 [the] object class*. Chen mentions that it is possible to “store [an] entire XML document in DB2 as a user-defined data type.” Ex. 1105 at 319. A “user-defined type...[is] a data type that you derive from existing data types. UDTs enable you to extend the built-in types already available.” Ex. 1105 at 174. A POSITA would have understood that the user-defined type (XML document) comprises *individual metadata objects [that] have distinct numbers and types of attributes according to a predetermined object class* (“built-in types”). *[T]he behaviors and attributes [are] defined specific to [the] object class*, “[t]he classic example is to create a UDT of pounds and a UDT of dollars...you cannot compare the values of a pounds and dollars without writing a [user defined function]...[m]uch the same way you cannot directly compare two built-in numerics without using a function.” Ex. 1105 at 174. As mentioned in Section IX.A.16.iii, this is implemented in software code, where “[t]he application code runs in the client process and the server code runs in separate processes.” Ex. 1105 at 10.

230. Accordingly, Chen discloses “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.”

- 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;***

231. Chen discloses “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.”

232. For all the same reasons described above regarding claim limitation 6.1 (see *supra* Section IX.A.4.ii), Chen discloses limitation. As mentioned in Section IX.A.16.iii, DB2 is implemented in software code, where “[t]he application code runs in the client process and the server code runs in separate processes.” Ex. 1105 at 10. As discussed above, the DB2 agents are *assembler[s]* and *disassembler[s]* (see *supra* Section IX.A.1.iii(D)). Additionally, DB2 agents are “assign[ed]...[to] an application,” meaning to handle new applications (*according to demand*) in real-time through the life of the applications, requires *dynamically instantiate[ing]*, *create[ing]*, and *cach[ing]* for reuse DB2 agents (*at least one disassembler and at*

least one assembler). Ex. 1105 at 11. It would have been obvious to a POSITA that software code is *causing the controller to dynamically instantiate, create, and cache for reuse at least one disassembler and at least one assembler*.

233. Accordingly, Chen discloses “*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.*”

- 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***

234. Chen discloses “*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.*”

235. For all the same reasons described above regarding claim limitations 1.3 (*see supra* Section IX.A.1.iv), Chen discloses this limitation.

236. As discussed above (*see supra* Section IX.A.16.ii), DB2 is implemented as computer readable program *code*. Additionally (*see supra* Section IX.A.1.i), the DB2 agents are components of the DB2 system, meaning they are implemented as *code*. As previously discussed (*see supra* Section IX.A.1.iv), the DB2 agents *disassemble[] inbound data documents into multiple records in preparation for insertion into metadata tables as defined by polymorphic metadata*.

Finally, as described earlier (*see supra* Section 22.4), the DB2 agents (*code*) are instantiated as threads or processes to serve client application requests, and are there for to be “*dynamically created by the controller*,” i.e. the UDB Server.

237. Accordingly, Chen discloses “*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.*”

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.*

238. Chen discloses “*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.*”

239. For all the same reasons described above regarding claim limitations 1.4 (*see supra* Section IX.A.1.v), Chen discloses this limitation.

240. As discussed above (*see supra* Section IX.A.16.ii), DB2 is implemented as computer readable program *code*. Additionally (*see supra* Section IX.A.1.i), the DB2 agents are components of the DB2 system, meaning they are implemented as *code*. As previously discussed (*see supra* Section IX.A.1.v), the DB2 agents *assemble[] outbound data documents from at least one record selected from at least one metadata-selected table as defined by polymorphic metadata.*

Finally, as described earlier (*see supra* Section 22.4), the DB2 agents (*code*) are *dynamically created by the controller*.

241. Accordingly, Chen discloses “*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.*”

17. Claim 23

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

242. For the reasons I explain above (*see supra* Section IX.A.16), Chen discloses the method of Claim 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.*

243. Chen discloses “*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.*”

244. For all the same reasons described above regarding claim limitation 1.2 (*see supra* Section IX.A.1.iii), Chen discloses this limitation. As described above (*see supra* Section IX.A.16.ii), the DB2 system is implemented by computer

readable program *code*. Additionally, the DB2 XML Extender is part of the DB2 system (*see supra* Section IX.A.1.i), meaning it is implemented as *code*.

245. It was previously discussed (*see supra* Section IX.A.1.iii(B)), that the DB2 XML Extender (*code*) *transforms data under direction of polymorphic metadata*. Recall that 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 at 319. Additionally, these *transform[ations]* from XML content into DB2 tables is done 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 at 319-320. Within the XML Collections method, *transform[ation of] data* is done *under direction of* DAD files (*polymorphic metadata*):

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 and XML Collection.

Ex, 1105 at 320. As described above (*see supra* Section IX.A.1.iii(B)), DAD files are considered *polymorphic metadata*.

246. Chen also discloses *transform[ing] data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data*. Chen states:

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 at 10. As discussed above (*see supra* Section IX.A.1.iii(C)), this explicit disclosure regarding security along with a POSITA’s knowledge and experience would imply that said *polymorphic metadata* (DAD and DTD files) *define[] a security model...for application to the data*. Regarding *data integrity rules*, as previously mentioned (*see supra* Section 1.2.c), Chen validates documents by utilizing DAD and DTD files (*polymorphic metadata*) and the database manager enforces constraints that include “[u]nique constraints,” “[r]eferential integrity (RI) constraints,” “check constraints,” and “informational constraints.” Ex. 1105 at 173–174.

247. Accordingly, Chen discloses “*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.*”

18. Claim 24

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

248. For the reasons I explain above (*see supra* Section IX.A.16), Chen discloses the method of Claim 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.***

249. Chen discloses “*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.*”

250. For the reasons described above (*see supra* Section IX.A.1.iii and IX.A.2.ii), Chen discloses limitation 24.1. As mentioned above (*see supra* Section IX.A.1.i and IX.A.16.ii), the DB2 agents, being parts of the DB2 system, are implemented as code (*a code causing the controller*). Chen discloses “DB2 agents...are the most common type of DB2 processes which carry out the bulk of SQL processing on behalf of applications.” Ex. 1105 at 11. A POSITA would have understood that in order to “carry out...SQL processing on behalf of applications,” it would first be necessary to *collect data from a plurality of diverse (client) applications*.

251. Chen discloses the client *applications...supply data in multiple diverse external document formats*. The 575 Patent does not provide an explicit definition of *external document formats*, but it can be determined from various disclosures. For instance, “the quantity of metadata used for a native format document is a small

fraction of that used for an externally defined format, such as an IDoc.” Ex. 1101 at 16:62-64. The 575 Patent also states “[d]ocument metadata is subdivided into two parts...The second part defines the format of the XML document. For SAP IDocs or for any other externally standardized document type, both parts of the document metadata are explicitly defined.” Ex. 1101 at 16:30-35. Here it is clear that different document types can be considered *diverse external document formats*. Chen mentions differing document types:

The Document Type Definition (DTD) files are used to validate the XML documents you are storing within DB2. The DB2 XML Extender provides an XML DTD repository to store these DTDs. When a database is enabled for XML, a DTD reference table called DTD_REF is created. Each row in this table contains a DTD with additional metadata information about it. You can insert your own DTDs into this table...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.

Ex. 1105 at 320. Clearly, Document Type Definition implies that there are varying document types (*diverse external document formats*). Additionally, according to Merriam-Webster Dictionary, format is “a method of organizing data (as for storage).” A POSITA would have understood that a mapping scheme between a document and relational data could be considered a method of organizing data as for storage. Thus, the use of DAD files also implies *diverse external document formats*.

252. Chen also discloses *mapping the supplied data to a predetermined unified data model according to polymorphic metadata*. A POSITA would have known that a *unified data model* consolidates data from various sources (*a plurality of diverse applications*) into a single, cohesive dataset. A POSITA would also have known that the DB2 tables where data from the client applications (*plurality of diverse applications*) is stored, is a single, cohesive data set (*unified data model*). As mentioned above (*see supra* ¶239), “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,” stored in the DB2 tables (*mapping the supplied data to a predetermined unified data model*). Ex. 1105 at 320. Additionally,

as discussed above (*see supra* Section IX.A.1.iii(B)), the DAD and DTD repository files are *polymorphic metadata*.

253. It would have been obvious to a POSITA that the software code *caus[es] the controller* (coordinator agents) *to collect data from a plurality of diverse applications* (client applications) *that can supply data in multiple diverse external document formats* (varying XML document structures) *and mapping the supplied data to a predetermined unified data model* (DB2 tables) *according to the polymorphic metadata* (DAD and/or DTD repository files).

254. Accordingly, Chen discloses “*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.*”

19. Claim 25

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

255. For the reasons I explain above (*see supra* Section IX.A.16), Chen discloses the method of Claim 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 polymorphic metadata objects.***

256. Chen discloses “a code causing the controller to 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.”

257. For the reasons described above (*see supra* Section IX.A.1.iii, IX.A.1.v, and IX.A.5.ii), Chen discloses limitation 25.1. As described above (*see supra* Section IX.A.18.ii), the DB2 agents *represent metadata* using DAD files. Also described above (*see supra* Section IX.A.5.ii), the DAD files (*metadata*) is *represent[ed]* as stateless polymorphic objects that are permanent after initial construction.

258. Chen also discloses *methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects*. As discussed previously, *see supra* Section IX.A.5.ii, a POSITA would have understood that DB2 tables and client applications are considered *stateful metadata classes*. Furthermore, as described above (*see supra* Section IX.A.5.ii), the *stateful metadata classes delegate operation to the stateless polymorphic metadata objects* (DAD files):

A DADX [(Document Access Definition Extension (DADX)] document specifies a Web Service using a set of operations that are defined by SQL statements or XML Extender Document Access Definition (DAD) documents. Web services, or functions invoked over the Internet, specified in a DADX file are called DADX or DB2 Web services.

Ex. 1105 at 322. Chen mentions that “[t]he Web services object runtime framework (WORF) provides an environment to easily create simple XML based Web services that access DB2 data.” Ex. 1105 at 322. A POSITA would have understood that the Web services could be used by client applications (*stateful metadata classes*). Additionally, since these Web services are defined by the DAD files (*stateless polymorphic metadata objects*), the client applications (*stateful metadata classes*) are *delegat[ing] operation* to the DAD files (*stateless polymorphic metadata objects*). Because an XML Extender is software, a POSITA would have understood the XML Extender to be “*a code causing the controller to*” perform the above.

259. Accordingly, Chen discloses “*a code causing the controller to 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.*”

20. Claim 28

i. [28.0] *An enterprise system comprising:*

260. Chen discloses “[a]n enterprise system.”

261. Chen describes the DB2 UDB as an *enterprise system*. Chen notes that the “DB2 UDB is...flexible enough to serve both structured and unstructured data management needs of a single user to very large enterprises.” Ex. 1105 at 1. Chen further teaches that “[i]t is suitable for a wide range 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 at 1-2. In addition, DB2 UDB product is offered as an “DB2 Universal Database Enterprise Server Edition.” Ex. 1105 at 359. Finally, Chen mentions “DB2 UDB provides a single database that can be used across an enterprise.” Ex. 1105 at 3.

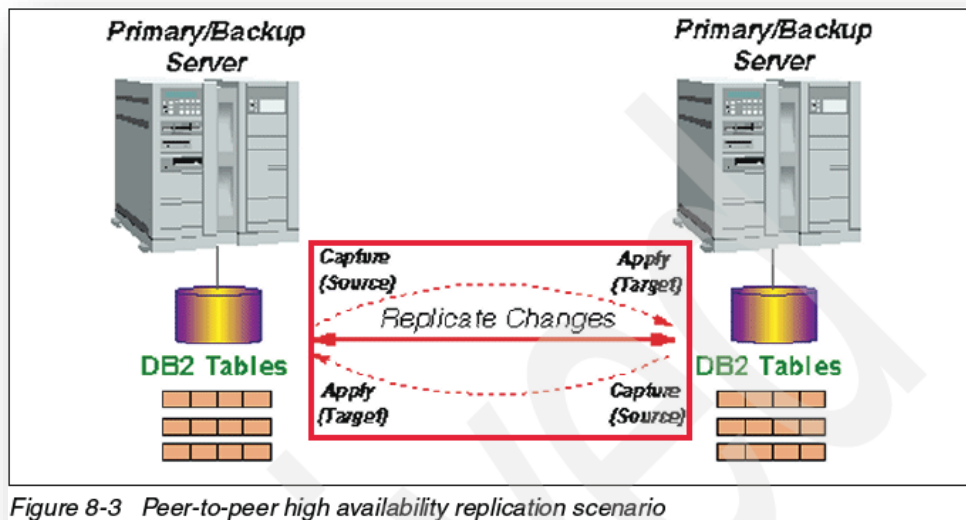
262. Accordingly, Chen discloses “[a]n enterprise system.”

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

263. Chen discloses “a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers.”

264. Chen discloses that its DB2 UDB system uses a *plurality of interlinked computer servers*. For one, Chen teaches using multiple primary servers for “peer-

to-peer replication.” Ex. 1105 at 333. For another, Chen states that “two primary servers acting as peers may be implemented, each serving as a backup for the other when a database failure.” Ex. 1105 at 333. Another example is disclosed in Figure 8-3, which illustrates such server configuration:



Ex. 1105 at 333 (annotated). Ex. 1105 at 3.

265. As pointed out in the figure, the two servers are *interlinked* and exchanges data between each other. Chen discloses that “[p]eer-to-peer replication is bidirectional,” and “[e]ach peer server must be able to handle local changes as well as the replicated changes from all other peers.” Ex. 1105 at 333. Chen further notes that “[a]pplications can run on any peer system, so query workload can be balanced across the peer configuration.” Ex. 1105 at 333. 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.*”

266. Moreover, for all the same reasons described above regarding claim limitation 1.1 (*see supra* Section IX.A.1.ii), Chen further discloses *networks enabling multiple users to access applications and data on the servers*, as the DB2 UDB can support access by “thousands of users.” Ex-1105, 2-3. Also as described for [1.1], Chen teaches that users can access multiple Client Applications that are consuming/supplying data from /to the UDB Server.

267. Accordingly, Chen discloses “*a plurality of interlinked computer servers and networks enabling multiple users to access applications and data on the servers.*”

- 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,***

268. Chen discloses “*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.*”

269. For all the same reasons described above regarding claim limitations [1.1]–[1.3] (*see supra* Section IX.A.1.i–IX.A.1.iv), Chen discloses this limitation. For example, DB2 “can be deployed on a variety of hardware and software platforms all the way from palm devices to massive parallel multiprocessor systems,” meaning it *execut[es] on at least one of the interlinked servers*. Ex. 1105 at 2. Chen also discloses that “DB2 UDB is a scalable and industrial strength relational database management system...[that] is a high performance middleware software” (*manag[er of] information flow between a data store and applications that supply and consume data*). Ex. 1105 at 1-2.

270. As discussed above (*see supra* Section IX.A.1.iii(A)), Chen discloses a *real time information director (RTID)* (DB2 XML Extender) *executing on a computing system (the UDB server)*. In view of previous discussion (*see supra* Section IX.A.20.iii and ¶ 257), said computing system is *at least one of the interlinked servers*. Also mentioned above (*see supra* Section IX.A.1.iii(A)), the DB2 XML Extender (*real time information director*) is responsible for most of the XML document input and output to the UDB Server (*manag[ing] information flow between a data store and applications that supply and consume data*). Additionally, mentioned above (*see supra* Section IX.A.1.iii(D)), the *real time information director...dynamically instantiates, creates, and caches for reuse disassemblers and assemblers according to demand*.

271. Accordingly, Chen discloses “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.”

- 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.***

272. Chen discloses “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.”

273. For all the same reasons described above regarding claim limitations 1.2–1.4 (see *supra* Section IX.A.1.iii–IX.A.1.v), Chen discloses this limitation.

274. As discussed above (*see supra* Sections IX.A.1.iii and IX.A.1.iv), Chen discloses *the RTID...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*. Additionally, as previously mentioned (*see supra* Sections IX.A.1.iii and IX.A.1.v), Chen discloses *the RTID...create[s] 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*.

275. Accordingly, Chen discloses “*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*.”

276. Accordingly, Chen discloses Claim 28.

21. Claim 31

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

277. For the reasons I explain above (see *supra* Section IX.A.20), Chen discloses the method of Claim 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.*

278. Chen discloses “*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.*”

279. For all the same reasons described above regarding claim limitation 6.1 (see *supra* Section IX.A.4.ii), Chen teaches this limitation. As discussed above (see *supra* Section IX.A.4.ii), the DB2 XML Extender (RTID) represents meta data as polymorphic objects (DAD and DTD repository files. Also described above (see *supra* Section IX.A.4.ii), individual metadata objects...have distinct numbers and types of attributes according to a predetermined object class. Finally, as discussed above (see *supra* Section IX.A.16.iv), the behaviors and attributes [of] predetermined object class being defined specific to object class. Clearly, the RTID (DB2 XML Extender) represents metadata as polymorphic objects (DAD files and

DTD repository files) *to have distinct numbers and types of attributes according to a predetermined object class* (user-defined types and existing data types), *the behaviors and attributes being defined specific to object class.*

280. Accordingly, Chen discloses “*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.*”

22. Claim 32

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

281. For the reasons I explain above (*see supra* Section IX.A.20), Chen discloses the method of Claim 32.

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.*

282. Chen discloses “*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.*”

283. For all the same reasons described above regarding claim limitations 1.2, 1.4, 7.1, and 25.1 (*see supra* Section IX.A.1.iii, IX.A.1.v, IX.A.5.ii, and IX.A.19.ii), Chen discloses this limitation. From the above discussion, *the RTID*

(DB2 XML Extender) *represents metadata as stateless polymorphic objects (DAD file and/or DTD repository file) that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects (DADX files define a set of operations from the DAD file).*

284. Accordingly, Chen discloses “*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.*”

23. Claim 33

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

285. For the reasons I explain above (see *supra* Section IX.A.20), Chen discloses the method of Claim 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.*

286. Chen discloses “*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.”

287. For all the same reasons described above regarding claim limitations [1.2], [1.4], and [10.1] (*see supra* Section IX.A.1.iii, IX.A.1.v, and IX.A.6.ii), Chen teaches this limitation. A POSITA would have known that an XML Schema defines the structure *that describes the format of the document*[], specifying attributes and elements that can appear in a document, the number and order of child elements, and the data types for elements and attributes. A POSITA would have understood that some of the information in the XML Schema is in the DAD and DTD repository files (*defined by polymorphic metadata*). A POSITA would have understood that, as new schemas are used with new application communications (*dynamically created*), a *generator* would be used to do so. Further, the DB2 database object is a schema paired, and *the RTID* (DB2 XML Extender) utilizes *polymorphic metadata* (DAD files and/or DTD repository files) to map XML documents to relational data—a POSITA would have recognized the schema containing the DAD and/or DTD repository files is done via a generator. Such generator generates the XML Schema that defines the format of the XML document based on the DAD file. 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.”). Additionally, as the *RTID* manages communications between application (see *supra* Section IX.A.1.iii(B)), the *generator...is dynamically created by [the] RTID*. Furthermore, since the same DAD files are used for creating the schema and disassembling and assembling documents (i.e., a DAD file with consistent mappings), *the at least one disassembler, assembler, and generator...use a single descriptive polymorphic metadata*.

288. Accordingly, Chen discloses “*at least one generator that is dynamically created by 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.*”

24. Claim 35

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

289. For the reasons I explain above (see *supra* Section IX.A.23), Chen discloses the method of Claim 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.***

290. Chen discloses “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.”

291. For all the same reasons described above regarding claim limitations 11.1 (see *supra* Section IX.A.7.ii) and the additional reasons described below, Chen discloses “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.”

292. To the extent that one may argue that the sections of Chen cited within Section IX.A.7.ii do not fully disclose these claim limitations, Chen further discloses a “db2look” tool—a software tool that extracts data from the database (“query logic that queries the database”)—to “capture the table definitions” for a particular database (“to determine information about tables.”) Ex. 1105 at 262-263; see also Ex. 1111 at 433 (defining “query” as “extract data from a database and present it”). Chen also teaches that DAD files define “[t]he mapping between elements and attributes in the XML document and the DB2 tables.” Ex-1105, 319-320. 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. 1105 at 319-320. Additionally, a POSITA would have understood that in the below annotated CREATE TABLE statement, “DEPTNUMB” is a field name, “SMALLINT” is a data type, and “NOT NULL” allows nulls in the field (“*the information including field name, field datatype, and field allowance of nulls*”):

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

Ex-1105, 263; Ex-1114, 72 (describing “CREATE TABLE” parameters). In order for the PHP routine, from the above disclosure, to read the information, it would be necessary for the DB2 system to read such information as well. Chen also discloses this is done through *query logic that queries the database*, “input of any query and...execution is achieved by clicking on the query button. The query results are returned and written dynamically into an HTML table...The program looks for the first field in the record as the primary key.” Ex. 1105 at 144.

293. DB2 also includes the ability to look for (and subsequently compress) null values (*field allowance of nulls*). Ex. 1105 at 214; *see also* Ex. 1105 at 5 (“Null

and default compression can significantly reduce the disk footprint of a large DB2 system.”).

294. Finally, given that “DB2 [is]...relational database software,” (Ex. 1105 at 1) and DB2 agents communicate with applications through SQL (Ex. 1105 at 11), it would have been obvious to a POSITA that the above-described interactions with tables are queries and comprise *query logic*.

295. Accordingly, Chen discloses “*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.*”

25. Claim 36

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

296. Chen discloses “[a] method for executing on a computer system for managing data services.”

297. For all the same reasons described above regarding claim limitation 1.0 (see *supra* Section IX.A.1.i), Chen teaches this limitation.

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

298. Chen discloses “*transferring data among a data store and applications that supply and consume data.*”

299. For all the same reasons described above regarding claim limitation 1.0, 1.1, 22.2 (*see supra* Section IX.A.1.i, IX.A.1.ii and IX.A.16.iii), Chen teaches this limitation. Chen discloses the interface for communicating between (to *transfer data among*) a data store and applications that supply and consume data is the coordinator agents.

300. Accordingly, Chen discloses “*transferring data among a data store and applications that supply and consume data.*”

- 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;***

301. Chen discloses “*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.*”

302. For all the same reasons described above regarding claim limitation 6.1 and 22.3 (*see supra* Section IX.A.4.ii and IX.A.16.iv), Chen discloses this limitation. Chen discloses “perform[ing] necessary operations on the data pages in the buffer and return[ing] the result to the user application” (*transforming data for transfer*) Ex. 1105 at 13. As stated above (*see supra* Section IX.A.1.iv, IX.A.1.v, IX.A.17.ii,

and IX.A.18.ii), DB2 relational data is converted to XML documents for transfer using DAD and DTD repository files (*using a representation of metadata as polymorphic objects*). Also previously discussed (*see supra* Section IX.A.4.ii, IX.A.16.iv and IX.A.21.ii), this *enabl[es] 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.*

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

303. Chen discloses “*dynamically instantiating, creating, and caching for reuse at least one disassembler and at least one assembler according to demand.*”

304. For all the same reasons described above regarding claim limitations 1.1 and 1.2 (*see supra* Section IX.A.1.ii and IX.A.1.iii), Chen teaches this limitation.

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

305. Chen discloses “*disassembling inbound data documents into multiple records in preparation for insertion into metadata tables.*”

306. For all the same reasons described above regarding claim limitations 1.1 and 1.2 (*see supra* Section IX.A.1.ii and IX.A.1.iii), Chen teaches this limitation.

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

307. Chen discloses “*assembling outbound data documents from at least one record selected from at least one metadata-selected table.*”

308. For all the same reasons described above regarding claim limitations 1.1 and 1.2 (*see supra* Section IX.A.1.ii and IX.A.1.iii), Chen teaches this limitation.

X. CONCLUSION

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1101 of Title 18 of the United States Code.

January 15, 2025
Date



Kevin C. Almeroth, Ph.D.