

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.
This form is prescribed as FORM PTO-1465)



REQUEST FOR EX PARTE REEXAMINATION TRANSMITTAL FORM

05/09/07

1338 U.S. PTO
90008648



05/09/07

Address to:
Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: 6883/23

Date: May 7, 2007

1. This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number 5,826,259 issued October 20, 1998. This request is made by:

patent owner. third party requester.

2. The name and address of the person requesting reexamination is:

William L. Anthony, Jr.
Orrick, Herrington & Sutcliffe
1000 Marsh Road
Menlo Park, CA 94025

3. a. A check in the amount of \$ _____ is enclosed to cover the reexamination fee, 37 CFR 1.20(c)(1);

b. The Director is hereby authorized to charge the fee as set forth in 37 CFR 1.20(c)(1) to Deposit Account No. 15-0665 (submit duplicate of this form for fee processing); or

c. Payment by credit card. Form PTO-2038 is attached.

4. Any refund should be made by check or credit to Deposit Account No. _____ 37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.

5. A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4)

6. CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table

7. Nucleotide and/or Amino Acid Sequence Submission
If applicable, all of the following are necessary.

- a. Computer Readable Form (CRF)
- b. Specification Sequence Listing on:
 - i. CD-ROM (2 copies) or CD-R (2 copies); or
 - ii. paper
- c. Statements verifying identity of above copies

8. A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

9. Reexamination of claim(s) 1-18 is requested.

10. A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO-1449 or equivalent.

05/22/2007 JMCDOUGA 00000001 150665 90008648
01 FC:1812 2520.00 DA

11. An English language translation of all necessary and pertinent non-English language patents and/or printed publications is included.

[Page 1 of 2]

This collection of information is required by 37 CFR 1.510. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop *Ex Parte* Reexam, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

12. The attached detailed request includes at least the following items:
- a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1)
 - b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR 1.510(b)(2)

13. A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e)

14. a. It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).

The name and address of the party served and the date of service are:

ALLEN, DYER, DOPPELT, MILBRATH & GILCHRIST P.A.

1401 CITRUS CENTER 255 SOUTH ORANGE AVENUE

Orlando, FL 32802-3791

Date of Service: May 7, 2007; or

- b. A duplicate copy is enclosed since service on patent owner was not possible.

15. Correspondence Address: Direct all communication about the reexamination to:

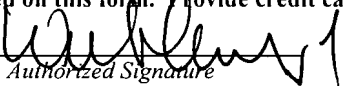
Customer Number:

OR

<input checked="" type="checkbox"/> Firm or Individual Name	Orrick Herrington & Sutcliffe, LLP William L. Anthony, Jr.				
Address (line 1)	1000 Marsh Road				
Address (line 2)					
City	Menlo Park	State	CA	Zip	94025
Country	USA				
Telephone	650 614-7400	Fax	650 614-7401		

16. The patent is currently the subject of the following concurrent proceeding(s):
- a. Copending reissue Application No. _____
 - b. Copending reexamination Control No. _____
 - c. Copending Interference No. _____
 - d. Copending litigation styled: _____

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

 5-8-07
 Authorized Signature Date

Williams L. Anthony, Jr. 24771
 Typed/Printed Name Registration No., if applicable

- For Patent Owner Requester
 For Third Party Requester



05/09/07

PATENT

IN THE UNITED STATES PATENT OFFICE

1338 U.S. PTO
90008648



05/09/07

Request For *Ex Parte* Reexamination Of:

U.S. Patent No. 5,826,259

Inventor: Karol Doktor

Assignee: Financial Systems Technology
Pty. Ltd.

Filed: May 22, 1997

Issued: October 20, 1998

For: Easily Expandable Data
Processing Systems and
Method

INDEX FOR *EX PARTE*
REEXAMINATION OF U.S. PATENT
NO. 5,826,259

Mail Stop Ex Parte Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Enclosed please find Foundry Networks, Inc.'s request for *ex parte* reexamination of U.S. Patent No. 5,826,259. Included with the request is a compact disk that contains all exhibits and references in PDF format. The request comprises the following documents:

	<u>DOCUMENT</u>	<u>NO. OF PAGES</u>
USPTO From SB/08A		2
Exhibits to Form SB/08A:		
Exhibit PA-A	U.S. Patent No. 4,506,326	28

	<u>DOCUMENT</u>	<u>NO. OF PAGES</u>
Exhibit PA-B	U.S. Patent No. 4,774,661	16
Exhibit PA-C	U.S. Patent No. 4,918,593	38
Exhibit PA-D	Toby J. Teorey, et al., <u>A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model</u> , Computing Surveys (June 1986)	25
Exhibit PA-E	Daniel R. Dolk, et. al., <u>A Relational Information Resource Dictionary System</u> , Computing Practices, Communications of the ACM (January 1987)	14
Exhibit PA-F	M.M. Zloof, <u>Query-by-Example: A Data Base Language</u> , IBM Systems Journal, No. 4 (1977)	20
Exhibit PA-G	Tsichritzis, <u>LSL: A Link and Selector Language</u> , Proceedings of the 1976 ACM SIGMOD International Conference on Management of Data, Washington, D.C. (June 2-4, 1976)	11
Exhibit PA-H	Munz, Rudolf, <u>The Well System: A Multi-User Database System Based on Binary Relationships and Graph-Pattern-Matching</u> , 3 Information Systems 99-115 (Pergamon Press 1978)	17
Exhibit PA-I	Munz, Rudolf, <u>Design of the Well System</u> , in Entity-Relationship Approach to Systems Analysis and Design. Proc. 1st International Conference on the Entity Relationship Approach	18
Exhibit PA-J	Ashok Malhotra, Yakov Tsalalikhin, Donald P. Pazel, Luanne M. Burns and Harry M. Markowitz, <u>Implementing an Entity-Relationship Language on a Relational Data Base</u> , IBM Research Report RC 12134 (#54499) (Aug. 27,	27

	<u>DOCUMENT</u>	<u>NO. OF PAGES</u>
	1986)	
Exhibit PA-K	Rudolph Munz, "Das WEB-Modell" (translated pages)(1976).	13
Exhibit PA-L	Gio Wiederhold, "Database Design Second Edition" (1995).	30
Exhibit PA-M	Pin-Shan Chen, <u>The entity-relationship model – A basis for the enterprise view of data</u> (1977).	8
Exhibit PA-N	Mark L. Gillenson, <u>Database Step-by- Step</u> 2nd Edition (1990). Other References	40
Exhibit PAT-A1	U.S. Patent No. 5,826,259	46
Exhibit PAT-A2	Preliminary Infringement Contentions filed by FST in <u>Financial Systems Technology, et al. v. Oracle Corporation</u> ("PICS").	50
Exhibit PAT-A3	FST's Response to the Notice of Non- Compliant Amendment, filed on Sept. 21, 2006.	47
Exhibit PAT-A4	U.S. Reissue App'n, Amendment Filed July 25, 2006, 11/152,835.	7
Exhibit PAT-A5	FST's Information Disclosure Statement (IDS) in the 90/007,707 re-examination (stamped by the USPTO on October 23, 2006).	7
Exhibit PAT-A6	Second Office Action in the Reissue/Reexamination Proceedings for the '259 Patent.	44
Exhibit PAT-A7	Livingston Enterprises, Inc., Configuration Guide for PortMaster Products (Dec. 1995)	52
Exhibit OTH-A	Full copy of the complaint filed by Patent Owner in <u>Financial Systems</u>	6

DOCUMENT

NO. OF
PAGES

Technology, et al. v. Oracle Corporation, Case No. 2:04-CV-358-TJW (E.D. Tex.) filed on October 12, 2004.

Exhibit OTH-B

The IBM Dictionary of Computing Terms 87 (8th Ed. 1987).

4

Exhibit OTH-C

Telebit Corp., Telebit NetBlazer®
Version 2.3 Release Notes (March 25, 1994)

5

Certificate of Service

1

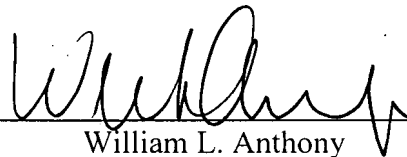
Postcard

1

CD

Respectfully submitted,

Dated: May 8, 2007



William L. Anthony
Reg. No. 24771
Attorney for Oracle Corporation

OHS West:260228539.1



05/09/07

PATENT

1338 U.S. PTO
90008648



05/09/07

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Request for Reexamination of:

U.S. Patent No. 5,826,259

Inventor: Karol Doktor

Assignee: Financial Systems Technology
(Intellectual Property) Pty Ltd
Melbourne, Australia

Filed: May 22, 1997

Issued: October 20, 1998

For: Easily Expandable Data
Processing System and Method

REQUEST FOR *EX PARTE*

REEXAMINATION OF U.S. PATENT NO.
5,826,259

ATTACHMENT TO FORM 1465

MATTER IN REEXAMINATION

ATTN: EXAMINER LUKE S. WASSUM

GAU: 2167

Mail Stop Ex Parte Reexam
Commissioner for Patents
P.O. Box 1450,
Alexandria, VA 22313-1450

TABLE OF CONTENTS

	Page
TABLE OF EXHIBITS	5
LIST OF EXHIBITS	5
A. Prior Art (PA)	5
B. Relevant Patent Materials (PAT)	6
C. Other Documents (OTH)	6
REQUIREMENTS UNDER 37 C.F.R. § 1.510	8
A. PAYMENT OF FEES; 37 C.F.R. § 1.510(A)	8
B. STATEMENT POINTING OUT EACH SUBSTANTIAL NEW QUESTION OF PATENTABILITY; 37 C.F.R. § 1.510(B)(1)	8
C. IDENTIFICATION OF CLAIMS FOR REEXAMINATION; 37 C.F.R. § 1.510(B)(2)	8
D. APPLICATION OF CITED PRIOR ART; 37 C.F.R. § 1.510(B)(2)	9
E. COPIES OF THE PRIOR ART; 37 C.F.R. § 1.510(B)(3)	9
F. COPY OF U.S. PATENT 5,826,259; 37 C.F.R. § 1.510(B)(4)	9
G. CERTIFICATION OF SERVICE ON PATENT OWNER; 37 C.F.R. § 1.510(B)(5)	9
II. CLAIMS FOR WHICH RE-EXAM IS REQUESTED	10
III. STATEMENT OF SUBSTANTIAL NEW QUESTION OF PATENTABILITY	11
A. The Prior Art	11
B. New Question of Patentability	12
IV. EXPLANATION OF THE PERTINENCE AND MANNER OF APPLYING CITED PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED BASED ON PRIOR ART	14
A. Teorey	14
B. Huber	18
C. Kumpati	22
D. Dolk	24
E. Zloof	26
F. Shaw	28
V. DISCUSSION OF FST'S RESPONSE TO NOTICE OF NON-COMPLIANT AMENDMENT	29

TABLE OF CONTENTS
(continued)

	Page
A. FST Identification of Alleged “Benefits Achieved by the Claimed Invention” is Unavailing as the “Benefits” are Unclaimed, and Because the “Benefits” are Disclosed by Prior Art.....	30
B. The ‘259 Patent Claims are Non-Statutory	38
C. The ‘259 Patent is Invalid Under 35 USC 102/103 Over the Munz, Malhotra and Tschritzis References	39
VI. DISCUSSION OF FST’S RESPONSE TO THE EXAMINER’S SECOND OFFICE ACTION.....	43
A. Wiederhold’s “Database Design Second Edition” Discloses Definition Tables	43
B. Requester Agrees with Examiner’s Section 101 Rejections.....	45
C. The Munz Reference is Anticipatory Prior Art	48
D. The Malhotra Reference is Anticipatory Prior Art.....	52
E. The Claims Are Not Entitled to a Presumption of Validity.....	53
VII. APPLICATION OF PRIOR ART PATENTS AND PUBLICATIONS	54
A. Teorey and Huber	54
B. Teorey and Kumpati.....	80
C. Dolk, Teorey, Zloof and/or Shaw	108
D. Tschritzis, Munz, Zloof, and Shaw References.....	132
VIII. CONCLUSION	138

TABLE OF EXHIBITS

LIST OF EXHIBITS

The exhibits to the present Request are arranged in three groups: prior art ("PA"), relevant patent prosecution file history, patents, and claim dependency relationships ("PAT"), and other ("OTH").

A. Prior Art (PA)

- PA-SB/08A USPTO Form SB/08A
- PA-A U.S. Patent No. 4,506,326 to Philip S. Shaw, et al., Apparatus and Method for Synthesizing a Query for Accessing a Relational Database, issued March 19, 1985, filed Feb. 28, 1983 ("Shaw").
- PA-B U.S. Patent No. 4,774,661 to Murari Kumpati, Database Management System with Active Data Dictionary, issued Sept. 27, 1988, filed Nov. 19, 1985 ("Kumpati").
- PA-C U.S. Patent No. 4,918,593 to Val. J. Huber, Relational Database System, issued April 17, 1990, filed January 8, 1987 ("Huber").
- PA-D Toby J. Teorey, et al., A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, Computing Surveys, Vol. 18, No. 2, June 1986, pp. 197-222 ("Teorey").
- PA-E Daniel R. Dolk, et. al., A Relational Information Resource Dictionary System, Computing Practices, Communications of the ACM, Vol. 30, No. 1, January 1987 ("Dolk").
- PA-F M.M. Zloof, Query-by-Example: A Data Base Language, IBM Systems Journal, No. 4, 1977, pp. 324-343 ("Zloof").
- PA-G Tsichritzis, LSL: A Link and Selector Language, Proceedings of the 1976 ACM SIGMOD International Conference on Management of Data, Washington, D.C. June 2-4, 1976 ("Tsichritzis").
- PA-H Munz, Rudolf, The Well System: A Multi-User Database System Based on Binary Relationships and Graph-Pattern-Matching, 3 Information Systems 99-115 (Pergamon Press 1978) ("Munz I").
- PA-I Munz, Rudolf, Design of the Well System, in Entity-Relationship Approach to Systems Analysis and Design. Proc. 1st International Conference on the Entity Relationship Approach, 505-522 (1979) ("Munz II")

- PA-J Ashok Malhotra, Yakov Tsalalikhin, Donald P. Pazel, Luanne M. Burns and Harry M. Markowitz, Implementing an Entity-Relationship Language on a Relational Data Base, IBM Research Report RC 12134 (#54499) (Aug. 27, 1986) (“Malhotra”).
- PA-K Rudolph Munz, “Das WEB-Modell” (translated pages), pp. 155-156, Fig. 10.2.1, (1976) (“Munz III”), with English translation.
- PA-L Gio Wiederhold, “Database Design Second Edition”, Discloses Definition Tables, Sections 7-3-1, 7-3-7, 7-4-4, 7-4-5, and 9-7-6 and Figs. 8-5, 8-7, 8-9 (1995).
- PA-M Pin-Shan Chen, The entity-relationship model – A basis for the enterprise view of data 77 (1977).
- PA-N Mark L. Gillenson, Database Step-by-Step 141-42, 2d Ed. (1990).

B. Relevant Patent Materials (PAT)

- PAT-A1 U.S. Patent No. 5,826,259 (the ‘259 patent).
- PAT-A2 Preliminary Infringement Contentions filed by FST in Financial Systems Technology, et al. v. Oracle Corporation (“PICS”).
- PAT-A3 FST’s Response to the Notice of Non-Compliant Amendment, Filed on Sept. 21, 2006.
- PAT-A4 U.S. Reissue App’n, Amendment, Filed on July 25, 2006, 11/152,835.
- PAT-A5 FST’s Information Disclosure Statement (IDS) in the 90/007,707 re-examination, stamped by the USPTO on October 23, 2006.
- PAT-A6 Second Office Action in the Reissue/Reexamination Proceedings for the ‘259 Patent.
- PAT-A7 FST’s Response to Office Action, Filed on March 22, 2007.

C. Other Documents (OTH)

- OTH-A Full copy of the complaint filed by Patent Owner in Financial Systems Technology, et al. v. Oracle Corporation, Case No. 2:04-CV-358-TJW (E.D. Tex.) filed on October 12, 2004.
- OTH-B The IBM Dictionary of Computing Terms 87 (8th Ed. 1987).
- OTH-C Webster’s New World Dictionary of Computer Terms 107 (3d Ed. 1988).

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Request for Reexamination of:

U.S. Patent No. 5,826,259

Inventor: Karol Doktor

Assignee: Financial Systems Technology
(Intellectual Property) Pty Ltd
Melbourne, Australia

Filed: May 22, 1997

Issued: October 20, 1998

For: Easily Expandable Data
Processing System and MethodREQUEST FOR *EX PARTE*REEXAMINATION OF U.S. PATENT NO.
5,826,259

ATTACHMENT TO FORM 1465

MATTER IN REEXAMINATION

ATTN: EXAMINER LUKE S. WASSUM

GAU: 2167

Mail Stop Ex Parte Reexam
Commissioner for Patents
P.O. Box 1450,
Alexandria, VA 22313-1450

Dear Sir:

Pursuant to the provisions of 35 U.S.C. §§ 302 *et seq.* and 37 C.F.R. § 1.510, Oracle Corporation ("Oracle" or "Requester") hereby requests *ex parte* reexamination of U.S. Patent No. 5, 826,259 ("the '259 patent"). Attached as **Exhibit PAT-A1** is a copy of the '259 patent, as required under 37 C.F.R. § 1.510(b)(4). The '259 patent was issued on October 20, 1998 to Karol Doktor. On its face, the '259 patent indicates that it was assigned to Financial Systems Technology Pty Ltd. Financial Systems Technology Pty Ltd. claims it has assigned the patent to Financial Systems Technology (Intellectual Property) Pty Ltd. For convenience, both entities will be referred to as "FST" in this request. FST has stated it believes the '259 patent is enforceable and there is no terminal disclaimer, certificate of correction, or reexamination certificate.

The '259 patent is presently the subject of a merged re-issue/re-examination. Re-

issue serial number 11/152,835, reexamination serial number 90/007,707. Additionally, the '259 patent was previously the subject of litigation proceedings in the District Court for the Eastern District of Texas, styled as Financial Systems Technology, et al. v. Oracle Corporation, Case No. 2:04-CV-358-TJW. A copy of the Complaint is attached as **Exhibit OTH-A**. During these proceedings, FST prepared and served on Oracle its Preliminary Infringement Contentions ("PICs") as required under the Patent Local Rules of the Eastern District of Texas. The PICs, as admissions by the patent owner of record in a court record, may be utilized in combination with a patent or printed publication during an ex parte reexamination proceeding. United States Patent & Trademark Office, Manual of Patent Examining Procedure § 2217(II). Admissions by the patent owner as to any matter affecting patentability may be utilized to determine the scope and content of the prior art **in conjunction with patents and printed publications** in a prior art rejection, whether such admissions result from patents or printed publications or from some other source. Id. A copy of these PICs is attached as **Exhibit PAT-A2**. This litigation was dismissed without prejudice to allow FST to pursue the above-noted reissue application. FST has stated that it intends to assert the '259 patent following the reissue proceedings.

REQUIREMENTS UNDER 37 C.F.R. § 1.510

Pursuant to 37 C.F.R. § 1.510, Oracle satisfies each of the requirements for *ex parte* reexamination of the '259 patent.

A. Payment of Fees; 37 C.F.R. § 1.510(a)

Requester authorizes the Patent Office to charge Deposit Account No. 15-0665 for the fee set in 37 CFR § 120(c)(1) for reexamination. The fee for reexamination is \$8,800, and the fee for an Information Disclosure Statement is \$180.00.

B. Statement Pointing Out Each Substantial New Question of Patentability; 37 C.F.R. § 1.510(b)(1)

A statement pointing out each substantial new question of patentability based on prior patents and publications is provided in Section II.

C. Identification of Claims for Reexamination; 37 C.F.R. § 1.510(b)(2)

Requester requests reexamination of claims 1-18 of the '259 Patent, as further

discussed in Section I.

D. Application of Cited Prior Art; 37 C.F.R. § 1.510(b)(2)

A detailed explanation of the pertinency and manner of applying the cited prior art to every claim for which reexamination is requested is provided in Section III.

E. Copies of the Prior Art; 37 C.F.R. § 1.510(b)(3)

Patent Office Form 1449 states the patents and printed publications upon which this Request is based. A complete copy of each listed patent and printed publication is included herewith as further outlined in Section II.

F. Copy of U.S. Patent 5,826,259; 37 C.F.R. § 1.510(b)(4)

As noted above, attached as **Exhibit PAT-A1** is a copy of the '259 patent, as required under 37 C.F.R. § 1.510(b)(4). There is no Certificate of Correction, Terminal Disclaimer, or Certificate of Reexamination.

G. Certification of Service on Patent Owner; 37 C.F.R. § 1.510(b)(5)

The undersigned certifies that a complete and entire copy of the Request for *Ex Parte* Reexamination and all supporting documents have been provided to the Patent Owner by serving the attorneys of record at the Patent Office for the '259 Patent and for the pending reissue/reexamination proceedings:

Kwok, Edward
MacPherson, Kwok, Chen & Heid LLP
2033 GATEWAY PLACE
Suite 400
San Jose CA 95110
(on file for the '259 Patent)

Allen, Dyer, Doppelt, Milbrath & Gilchrist, P.A.
1401 Citrus Center
255 South Orange Avenue
P.O. Box 3791
Orlando, FL 32802-3791
(on file for the reissue/reexamination proceedings)

The undersigned further certifies that it served an additional copy on the Patent Owners' current litigation counsel of record:

Sam Baxter, Esq.
McKool Smith P.C.
505 E. Travis, Suite 105, P.O. Box O

Marshall, TX 75760

II. CLAIMS FOR WHICH RE-EXAM IS REQUESTED

Reexamination is requested of claims 1-18 of the '259 patent in view of the disclosure in Toby J. Teorey, et al., A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, Computing Surveys, Vol. 18, No. 2, June 1986, pp. 197-222, attached as **Exhibit PA-D**.

Reexamination is also requested of claims 1-18 of the '259 patent in view of the disclosure in US Patent No. 4,918,593 to Val. J. Huber, Relational Database System, issued April 17, 1990, filed January 8, 1987, attached as **Exhibit PA-C**

Reexamination is also requested of claims 1-18 of the '259 patent in view of the disclosure in US Patent No. 4,774,661 to Murari Kumpati, Database Management System with Active Data Dictionary, issued Sept. 27, 1988, filed Nov. 19, 1985, attached as **Exhibit PA-B**.

Reexamination is also requested of claims 1-18 of the '259 patent in view of the disclosure in Daniel R. Dolk, et. al., A Relational Information Resource Dictionary System, Computing Practices, Communications of the ACM, Vol. 30, No. 1, January 1987, attached as **Exhibit PA-E**.

Reexamination is also requested of claims 1-18 of the '259 patent in view of the disclosure in M. M. Zloof, Query-by-Example: a data base language, IBM Systems Journal, No. 4, 1977, pp. 324-343, attached as **Exhibit PA-F**.

Reexamination is also requested of claims 1-18 of the '259 patent in view of the disclosure in US Patent No. 4,506,326 to Philip S. Shaw, et al., Apparatus and Method for Synthesizing a Query for Accessing a Relational Database, issued March 19, 1985, filed Feb. 28, 1983, attached as **Exhibit PA-A**.

All of the claims cited above are anticipated under 35 U.S.C. § 102 and/or rendered obvious under 35 U.S.C. § 103 in view of the five prior art publications noted above.

III. STATEMENT OF SUBSTANTIAL NEW QUESTION OF PATENTABILITY

A. The Prior Art

FST's Response filed Sept. 21, 2006, has presented certain new interpretations that it attributes to certain claims¹ in the '259 patent. Due to these new interpretations, Requester Oracle has identified additional prior art references which anticipate or render obvious the claims of the '259 patent. Of the additional prior art documents cited above, Teorey, Kumpati, Dolk, Zloof, and Shaw were not of record in the file of the '259 patent.

In addition to the foregoing, FST's amendment of claims 4, 9, 12, 17, and 18 in its first reexamination request have raised a substantial new question of patentability with respect to those claims. U.S. Reissue App'n, Amendment Filed July 25, 2006, 11/152,835, attached as **Exhibit PAT-A4**. Under the Manual of Patent Examining Procedure, the second or subsequent request for reexamination may raise a substantial new question of patentability with respect to any new or amended claim which has been proposed under 37 CFR 1.530(d) in the first (or prior) pending reexamination proceeding. United States Patent & Trademark Office, Manual of Patent Examining Procedure § 2240, II (8th ed. 2001).

Teorey, as part of the 90/007,707 re-examination, was cited as reference BA in an Information Disclosure Statement (IDS) (stamped by the USPTO on October 23, 2006²), attached as **Exhibit PAT-A5**, and initialed by the Examiner on December 18, 2006. This reference was cited more or less in the middle of 63 citations in the IDS. As this reference was effectively buried in dozens of other references, reconsideration of this reference is warranted. Also, in light of the positions taken by FST as to claim breadth and the amended claims, a substantial question of patentability exists as to the Teorey reference.

Huber was of record in the file of the '259 patent, but qualifies for consideration in this re-examination proceeding. Huber was cited by the Examiner on a PTO-892 form during prosecution of the great-grandparent application for this patent, but was otherwise

¹ Oracle notes that while FST may have advanced certain claim interpretations, Oracle does not necessarily adopt them, and is therefore is not bound by those interpretations.

² The same Information Disclosure Statement, included in Examiner's second office action, has a receipt stamp of July 25, 2006.

not referenced by the Examiner in any further proceedings. Huber was cited by FST on PTO-1449 forms in each subsequent filing, but Applicant failed to make references to Huber in these filings as well. MPEP § 2242.II.A permits consideration of art previously before the Examiner, where such art is presented in a new light or in a different way as compared with its use in the earlier concluded examination(s). Since Huber was not used in any manner in the prior Examinations, its use here constitutes a presentation in a new or different way. Furthermore, Huber is being used in this request in combination with art not previously before the Examiner.

Additionally, Requester Oracle presents an update to the required “detailed explanation and pertinency and manner of applying the cited prior art to every claim for which reexamination is requested” pursuant to 37 CFR 1.510 and MPEP § 2214, for the Munz, Malhotra, and Tschritzis references that are already cited and are already of record:

Tschritzis, LSL: A Link and Selector Language, Proceedings of the 1976 ACM SIGMOD International Conference on Management of Data, Washington, D.C. June 2-4, 1976, attached as **Exhibit PA-G**;

Munz, Rudolf, The Well System: A Multi-User Database System Based on Binary Relationships and Graph-Pattern-Matching, 3 Information Systems 99-115 (Pergamon Press 1978), attached as **Exhibit PA-H**;

Munz, Rudolf, Design of the Well System, in Entity-Relationship Approach to Systems Analysis and Design. Proc. 1st International Conference on the Entity Relationship Approach, 505-522 (1979), attached as **Exhibit PA-I**; and

Ashok Malhotra, Yakov Tsalalikhin, Donald P. Pazel, Luanne M. Burns and Harry M. Markowitz, Implementing an Entity-Relationship Language on a Relational Data Base, IBM Research Report RC 12134 (#54499) (Aug. 27, 1986), attached as **Exhibit PA-J**.

B. New Question of Patentability

The prior art documents discussed herein, including the additional prior art documents and the presently pending prior art documents, are closer to the subject matter of the ‘259 patent than any prior art which was cited during the prosecution of the ‘259 patent, as demonstrated in detail below. These prior art documents provide teachings not

provided during prosecution of the '259 patent.

FST now identifies four features, as listed below in Section IV, that it believes are benefits of the supposed "inventions" recited in the claims: (1) interposing metadata between the table catalog and the query; (2) using two-part keys; (3) using an inquiry table; or (4) using multi-tailed relation types. As will be discussed in detail below, all of these features are found in the additional prior art cited above (Teorey, Huber, Kumpati, Dolk, Zloof, and Shaw) and in the prior art cited in the presently pending re-issue/re-examination proceedings (Munz, Malhotra, Tschritzis). Accordingly, all of the claims of the '259 patent are either anticipated or obvious in light of the cited prior art.

Claims 1-18 specify systems and methods for retrieving data from a relational database. Presuming these distinctions are embodied in the language of the claims, a substantial new question of patentability in this reexamination is whether (1) interposing metadata between the table catalog and the query; (2) using two-part keys; (3) using an inquiry table; or (4) using multi-tailed relation types, is anticipated and/or obvious in view of the prior art cited herein.

In any event, the Teorey, Huber, Kumpati, Dolk, Zloof, and Shaw publications anticipate and/or render obvious, either alone or in combination with each other or with the prior art of record in this patent, claims 1-18 of the '259 patent. All of the references cited herein raise a substantial new issue of patentability because they anticipate or render obvious all of the claims for which reexamination is sought and, except for Huber, they were not previously of record or cited by the Examiner or the Applicants. As discussed above, Huber is being presented in a new or different way than it was used in the prior examination.

The prior art cited herein and presently of record in this merged re-issue/re-examination is more relevant to patentability than the prior art previously considered by the Examiner. For example, as discussed above, the Examiner determined that the prior art of record in the prosecution of the '259 patent did not teach using a relation instance table or an entity definition table to contain records to be retrieved while processing queries. As described below, each of the references disclosed herein, either alone or in combination, contains "relation instance tables" and an "entity definition table" and thus satisfies the deficiencies identified by the examiner as not found in the prior art. Each

reference either alone or in combination additionally contains a relation definition table, entity instance tables, and the other limitations recited in the claims of the '259 patent, making each reference more relevant to patentability than the prior art of record in the prosecution of the '259 patent. Each reference, either alone or in combination, also contains each of the four purported "benefits" cited by FST in their Response. As a consequence, these references create a substantial new question of patentability, are more relevant than the prior art of record, and should cause cancellation of claims 1-18.

IV. EXPLANATION OF THE PERTINENCE AND MANNER OF APPLYING CITED PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED BASED ON PRIOR ART.

Claims 1-18 of the '259 patent are considered to be fully anticipated under 35 U.S.C. § 102 or obvious under 35 U.S.C. § 103 by the prior art references to Teorey, Huber, Kumpati, Dolk, Zloof, and Shaw. These references are summarized below, with an explanation and detailed charts showing how each prior art reference, alone or in combination, meets all of the recited features of claims 1-18 of the '259 patent.

Claims 1-18 of the '259 patent are also considered to be fully anticipated under 35 U.S.C. 102 by the prior art references to Tsichritzis, Munz and Malhotra. The Tsichritzis, Munz and Malhotra references are discussed in detail in Requester's First Request for Reexamination. As part of Requester's presentation of the required "detailed explanation and pertinency and manner of applying the cited prior art to every claim for which re-examination is requested" in the reexamination (37 CFR 1.510; MPEP § 2214), Requester addresses certain mischaracterizations of the '259 patent and the cited prior art made by FST, and explains how these prior art references meet all of the recited features of the claims 1-18 of the '259 patent, even in light of FST's new arguments.

A. Teorey

Teorey, in combination with (1) Huber and (2) Zloof and/or Shaw teaches all of the claims set forth in the '259 patent³. In addition to the forgoing combination, Teorey,

³ Teorey teaches all elements of all of the claims set forth in the '259 patent. At a minimum, the definition table limitations in claims 1, 7, 8, 10, 15, 16, 17, and 18 are rendered obvious by Teorey, in combination with Huber, to the extent that Teorey does not anticipate these limitations. Further, the inquiry table limitations in claims 5, 6, and 14 are rendered obvious by Teorey, in combination with Zloof and Shaw, to the extent that Teorey does not anticipate these limitations.

in combination with (1) Kumpati and (2) Zloof and/or Shaw also teaches all of the claims set forth in the '259 patent⁴. The Teorey article discloses a conceptual schema and methodology for creating and extending a large relational databases, using the well-established Entity-Relationship ("E-R") approach to database design. Under the entity-relationship approach, information is presented in terms of entities, their attributes, and the associations between entity occurrences, also called relationships. Entity sets are the principal objects about which information is collected, and denote persons, places, things, or events of informational interest. Relationships represent the real-world associations among entities. The concepts and terms entities, entities sets, relationships, and relationship sets are all disclosed throughout the Teorey article. Teorey also teaches the use of a data dictionary to map the contents of the database. Relation and entity definitions also appear throughout the article, as do table instances and records of entities and relationships.

As disclosed in greater detail in the claim charts that follow, Teorey teaches the connection between entities and relations: a relation defines the relationship between two or more entities. Further, these entity and relation types may be defined within and accessed in a relational database through a data dictionary, which contains both entity and relation definitions. Records of these entity and relation types may be stored in instance tables, wherein each table may contain a plurality of entity records or a plurality of relation records. Also, a plurality of relation or entity instance tables may comprise an entity-relation database. Further, keys within a relation and entities may uniquely identify the relation or entity.

Figure 13 and Table 1 depict various entity relation types, including SKILL-USED, ASSIGNED-TO, and BELONGS-TO. They also depict entity types SKILL, DEPARTMENT and DIVISION.

⁴ Teorey teaches all elements of all of the claims set forth in the '259 patent. At a minimum, the definition table limitations in claims 1, 7, 8, 10, 15, 16, 17, and 18 are rendered obvious by Kumpati, in combination with Huber, to the extent that Teorey does not anticipate these limitations. Further, the inquiry table limitations in claims 5, 6, and 14 are rendered obvious by Teorey, in combination with Zloof and Shaw, to the extent that Teorey does not anticipate these limitations.

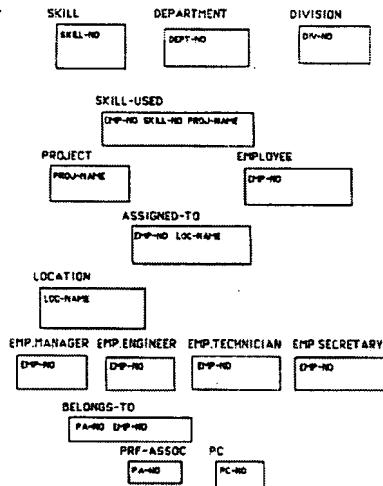


Figure 13. Company personnel and project database candidate relations.

Fig. 13 of the Teorey article depicts a number of relation and entity types.

Table 1. Transformation of Entities and Relationships to Relations (Example)

Step 2.1. Entities to relations

1. DIVISION(*DIV-NO*, ..., HEAD-EMP-NO)
2. DEPARTMENT(*DEPT-NO*, DEPT-NAME, ROOM-NO, PHONE-NO, ..., DIV-NO, MANAG-EMP-NO)
3. EMPLOYEE(*EMP-NO*, EMP-NAME, JOB-TITLE, ..., DEPT-NO, SPOUSE-EMP-NO, PC-NO)
4. SKILL(*SKILL-NO*, ...)
5. PROJECT(*PROJ-NAME*, ...)
6. LOCATION(*LOC-NAME*, ...)
7. EMP.MANAGER(*EMP-NO*, ...)
8. EMP.ENGINEER(*EMP-NO*, ...)
9. EMP.TECHNICIAN(*EMP-NO*, ...)
10. EMP.SECRETARY(*EMP-NO*, ...)
11. PC(*PC-NO*, ...)
12. PRF.ASSOC(*PA-NO*, ...)

Step 2.2. Binary or unary relationships to relations

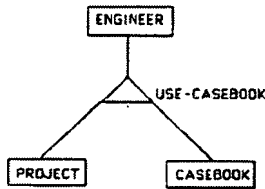
13. BELONGS-TO(*PA-NO*, *EMP-NO*)

Step 2.3. Ternary (or any n-ary) relationships to relations

14. SKILL-USED(*EMP-NO*, *SKILL-NO*, *PROJ-NAME*)
15. ASSIGNED-TO(*EMP-NO*, *LOC-NAME*, *PROJ-NAME*)

Table 1 of the Teorey article depicts a number of relationship types, such as BELONGS-TO, SKILL-USED, and ASSIGNED-TO and entity types such as DIVISION, DEPARTMENT, and EMPLOYEE. The table also depicts how an entity or relation may be defined by various attributes, one or more of which may be keys.

An engineer will use one casebook for a given project. Different engineers use different casebooks for the same project. No engineer will use the same casebook for different projects, but different engineers can use the same casebook for different projects.



Relations:
 ENGINEER(EMP-NO,)
 PROJECT(PROJ-NAME,)
 CASEBOOK(BOOK-NO,)
 USE-CASEBOOK(EMP-NO, PROJ-NAME, BOOK-NO)

FDs : EMP-NO, PROJ-NAME ---> BOOK-NO
 BOOK-NO, PROJ-NAME ---> EMP-NO
 EMP-NO, BOOK-NO ---> PROJ-NAME

USE-CASEBOOK

EMP-NO	PROJ-NAME	BOOK-NO
3	ALPHA	1001
3	BETA	1008
4	DELTA	1004
4	GAMMA	1005
8	BETA	1007
9	ALPHA	1009
9	EPSILON	1001

(a)

Fig. 10(a) (directly above) depicts a ternary relation USE-CASEBOOK of three entities, ENGINEER, PROJECT and CASEBOOK. The ternary relationship reflects the fact that under the given E-R model, an ENGINEER can use a particular CASEBOOK, depending on the PROJECT. It also includes an instance of a USE-CASEBOOK relationship table containing individual USE-CASEBOOK records and depicts how the entities and relationships may be defined, i.e. ENGINEER(EMP-NO...), PROJECT(PROJ-NAME...), CASEBOOK(BOOK-NO...), and USE-CASEBOOK(EMP-NO, PROJ-NAME, BOOK-NO...).

SKILL-USED	EMP-NO	SKILL-NO	PROJ-NAME
	38	27	GAMMA
	38	51	GAMMA
	38	27	DELTA
	38	3	DELTA

(a)

SKILL-AVAILABLE	EMP-NO	SKILL-NO	PROJ-NAME
	14	22	ALPHA
	14	22	BETA
	14	35	ALPHA
	14	35	BETA

EMP-SKILL	EMP-NO	SKILL-NO	EMP-PROJ	EMP-NO	PROJ-NAME
	14	22		14	ALPHA
	14	35		14	BETA

(b)

Fig. 4 of the Teorey article depicts relationship instances tables, each of which contain a plurality of relation instance records. Each of the relationships depicts, SKILL-USED, SKILL-AVAILABLE, and EMP-SKILL, contain multiple rows, where each row indicates an instance of a particular relationship. Thus the SKILL-AVAILABLE instance table discloses what sort of skill (SKILL-NO) a given employee (EMP-NO) has, as a function of the project (PROJ-NAME).

B. Huber

Huber, in combination with (1) Teorey and (2) Zloof and/or Shaw teaches all of the claims set forth in the '259 patent⁵. Huber describes how a database may rely on a data dictionary to manage the underlying data within the database, particularly through the use of definitions. Specifically, the Huber patent discloses a way to maintain a dependence between a user-defined field in one base table of a relational data base system and the state of a row in another relational data base system, when one base table references the other. Two base tables make up a referenced-referencing pair, and one row has a primary key, which is used as a foreign key in the set of rows of another base

⁵ Teorey teaches all elements of all of the claims set forth in the '259 patent. At a minimum, the definition table limitations in claims 1, 7, 8, 10, 15, 16, 17, and 18 are rendered obvious by Teorey, in combination with Huber, to the extent that Teorey does not anticipate these limitations. Further, the inquiry table limitations in claims 5, 6, and 14 are rendered obvious by Teorey, in combination with Zloof and Shaw, to the extent that Teorey does not anticipate these limitations.

table. A change in the row of one base table may imply changes in the other base table. Where such a dependence or relationship exists, the system must ensure the appropriate changes in both base tables.

For the purposes of this reexamination application, the key to the Huber system lies in the fact that it maintains the dependence between the user-defined fields by the use of a data dictionary and reestablishing means. Huber provides in greater detail, one implementation of a data dictionary to define data items, including entities, and to identify the locations, or files, of data instances, where records are located.

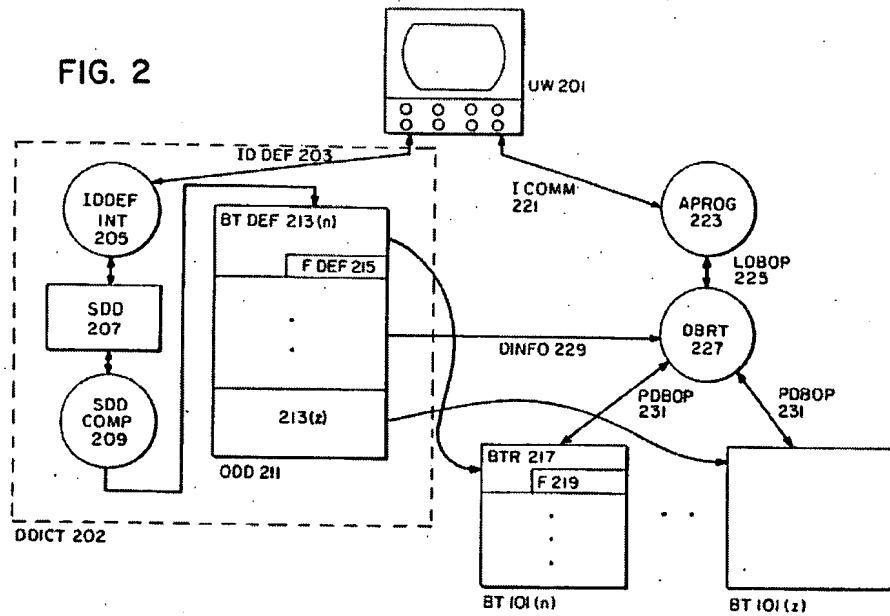


Fig. 2 provides of an overview of one type of relational data base system, as taught by Huber. It depicts a detailed layout of a data dictionary (DDICT 202) describes a relational data base accessed by the data base system, and explicitly includes relation definition tables (BT DEF 213). DDICT 202 contains both entity or relationship definition tables, as taught in both Huber and Teorey. Each BT DEF 213, e.g., BT DEF 213(n) .. 213(x), has a pointer to its relation instance table BT IOI, e.g., BT IOI(n) ... BT IOI(x). The BT DEF table, which may define both entities and relationships may point to both entity and relation instance tables, when Huber and Teorey are viewed in combination.

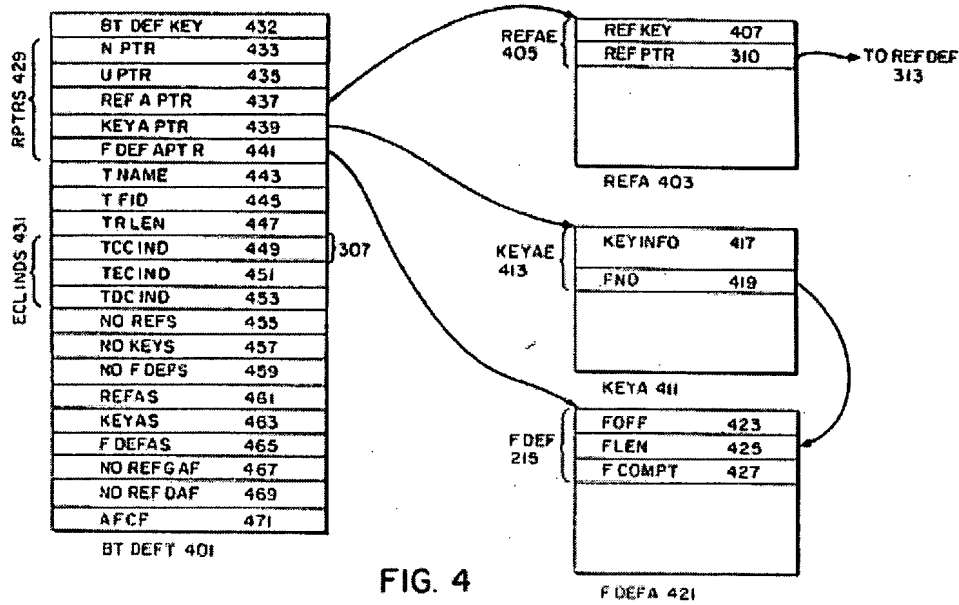


FIG. 4

Fig. 4 of Huber shows BT DEF 401, which is an expanded version of the BT DEF 213 of data dictionary 202 in Fig. 2 of Huber above. The “three fields [443, 445, and 447] relate BTDEF 213 to BT 101 which it defines. TNAME 443 is the name of BT 101; TFID 445 is the file identifier of the file which contains BT 101; TRLEN 447 is the length of the records which represent the rows of BT 101 in the file.” Huber, 15:17-21, 15:43-49. Thus, a data dictionary under Huber and Teorey contains the name of the entity or relation type and a pointer to an instance table, which contains the data for the entity or relation type.

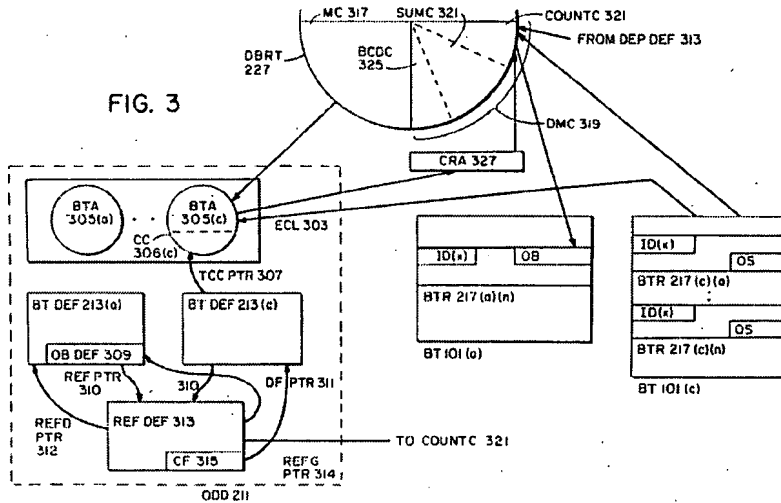


Fig. 3 of the Huber patent depicts another detailed layout of a data dictionary and further references a REFDEF table (REF DEF 313), which contains data defining the referenced-referencing relationship between base tables, such as the base tables which are used to create the relationship relation that itself is represented as a table. The REFDEF table 313 stores a pointer, REFDPTR 312 that points to a referenced base table (BT 101), and another pointer, REFGPTR 314, which points to a referencing base table (BT 101). These two pointers are entity table type identifiers. These entity tables contain entity records, which are retrievable by the user.

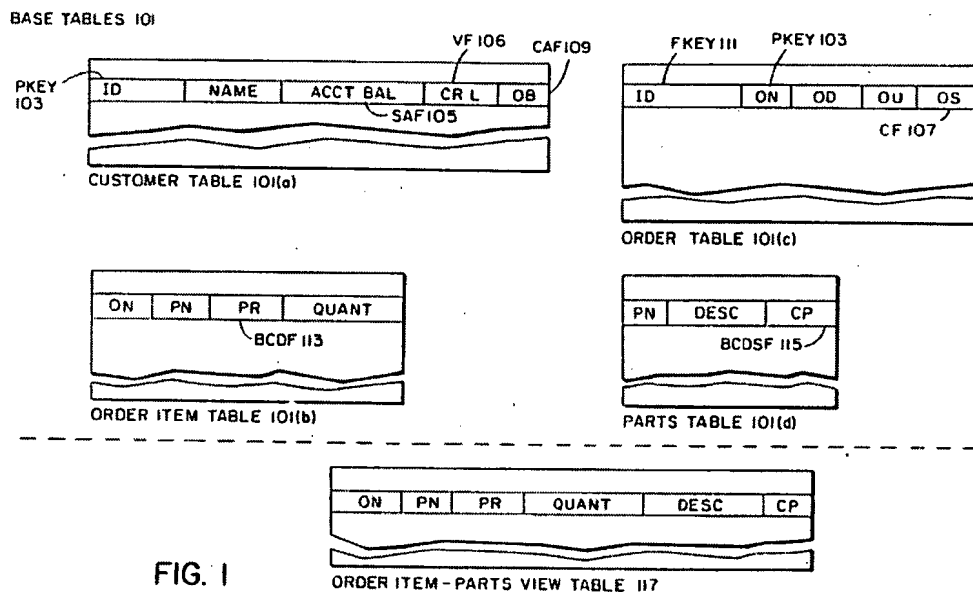


FIG. 1

Fig. 1 of the Huber patent is an example of the relational database. It depicts customer, order, and parts instance tables; the tables contain instances of customers, orders, and parts entities. Information about customers in the data base may be presented as shown in CUSTOMER TABLE 101(a). Huber, 5:5-10. The items of information CUST ID (customer ID), NAME (customer name), ACCT BAL (account balance), CRL (credit limit), and OB (orders booked) are associated with each customer and appear in a row belonging to the customer. Each row in the CUSTOMER TABLE 101(a) is uniquely identified by a primary key consisting of one or more fields. The primary key in the CUSTOMER TABLE would be CUST ID. On the other hand, a view table, unlike a base table, is a virtual table, i.e. "one whose rows do not correspond to a set of records in a file, but is instead made up by the relational data base system for items from one or more base tables." Huber, 5:21-25.

Thus, relation instance tables containing relation records are present under Huber. In the same way that a relation defines the relationship between two entity instances under the '259 patent, a relationship or dependence will define the relationship between two records, or between parent and child records under the Huber patent. Record identifiers, in the form of primary keys for a base table, are also present under Huber. Table identifiers for relation tables also exist under Huber, in the form of information about the location of base tables. Huber also teaches the use of record identifiers in the form of primary keys. In addition, it teaches the use of table identifiers in the form of location information that points to the files containing the actual base tables (BT).

C. Kumpati

Kumpati, in combination with (1) Teorey and (2) Zloof and/or Shaw teaches all of the claims set forth in the '259 patent⁶. Kumpati also describes in detail the use of a data dictionary to define and access the underlying data in a relational database. Kumpati pertains to a database management system, which has an active data dictionary component that a user can access. The user can make use of simple commands to query

⁶ Teorey teaches all elements of all of the claims set forth in the '259 patent. At a minimum, the definition table limitations in claims 1, 7, 8, 10, 15, 16, 17, and 18 are rendered obvious by Teorey, in combination with Kumpati, to the extent that Teorey does not anticipate these limitations. Further, the inquiry table limitations in claims 5, 6, and 14 are rendered obvious by Teorey, in combination with Zloof and Shaw, to the extent that Teorey does not anticipate these limitations.

the underlying data controlled by the database management system, as well as the contents of the data dictionary. The data dictionary provides the user with the definitions of the entity sets and relation (i.e., relationship) sets, of which the database is comprised, but also the identity and locations of the entity and relation (i.e., relationship) instances within the database management system.

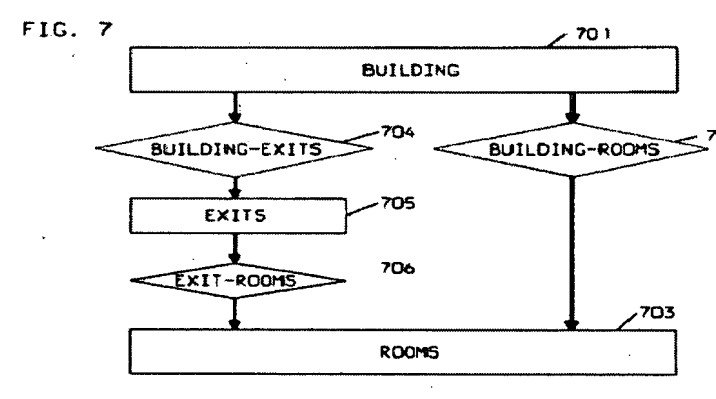


Fig. 7 of Kumpati shows a data model based upon relationships (diamonds 702, 704, and 706) and entities (rectangles 701, 703, and 705). This data model is part of the data dictionary. Kumpati at 8:30-39.

By means of a query processor and a database command processor, the system transfers the requested information - including the entity and relation records - to a buffer file, which an application program may access.

Under Kumpati, access to all the data stored in the database is controlled by a database command processor, which receives and processes requests for information from the operating system. The processed, decoded information is transmitted to query processor, which retrieves the identity of the various requested data elements from the schema stored in the data definition library. Using this information, the query processor ascertains the location of the requested data, which resides in the form of entity and relation instance tables. The tables themselves are comprised of entity and relation instance records. Kumpati also teaches the use of entity instance record identifiers, relation instance record identifiers, and relation instance table identifiers. It also teaches the use of a plurality of entity records and relation instance tables.

FIG. 2

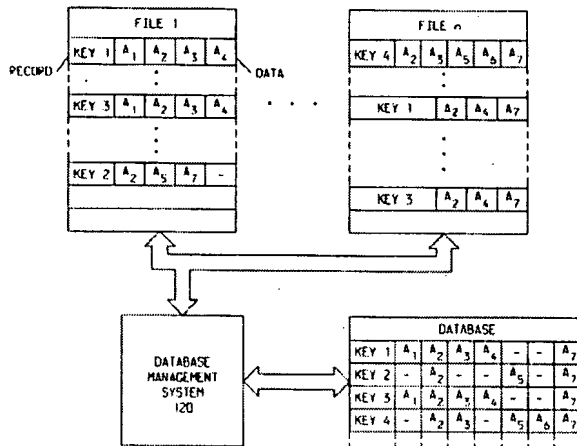


Fig. 2 of the Kumpati patent depicts sample instance tables with distinguishing keys for each record.

D. Dolk

Dolk, in combination with (1) Teorey and (2) Shaw and/or Zloof teaches all of the claims set forth in the '259 patent⁷. Dolk discloses a relational model of information resource dictionary system ("IRDS") used in database management. An important component of an IRDS is a data dictionary (D/D), which is a catalog of information about the logical and physical aspects of the underlying database. The D/D essentially describes the contents of the underlying operational database. The article also discloses a directory component, which may appear in an IRDS; the directory component describes where information resources are located and how they are accessed.

The IRDS architecture described by Dolk (depicted below) includes an IRD schema layer as well as a IRD data layer. It also includes an operational data layer, which is essentially the underlying E-R database system.

⁷ Dolk teaches all elements of all of the claims set forth in the '259 patent. At a minimum, the inquiry table limitations in claims 5, 6, and 14 are rendered obvious by Dolk, in combination with Zloof and Shaw, to the extent that Dolk does not anticipate these limitations.

IRD schema description layer	Entity-type	Relationship-type	Attribute-type
IRD schema layer	ELEMENT, RECORD, etc.	RECORD-CONTAINS-ELEMENT	DATE-ADDED, LENGTH, LOCATION, etc.
IRD data layer	Soc-Sec-No, Empl-Record, etc.	Empl-Record-CONTAINS-Soc-Sec-No	23Nov85, 12 (Char) Bldg A-Room 3
Operational data	555-23-6666 (Employee record for Kirk)	Empl-Record for Kirk-CONTAINS-(555-23-6666)	(Attributes do not appear as instances in operational databases)

FIGURE 1. IRDS Architecture

The IRD schema layer consists of instances of meta-entities, meta-relationships, and meta-attributes at the IRD schema description level. The schema layer describes various entity types (SYSTEM, FILE, RECORD, ELEMENT, USER, etc.), attribute types (ACCESS-NAME, ADDED-BY, CLASSIFICATION), and relationship types (CONTAINS, PROCESSES, RUNS, etc.), which comprise the layer underneath (the IRD data layer). See Fig. 2. The IRD data layer itself consists of entities, attributes, and the relationships that are the instances of the corresponding IRD schema entity-types, relationship-types, and attribute types. The IRD data layer maps the contents of the entity-relationship instance tables in the underlying operational layer. At all levels, including the organization layer, entities and relationships conform to a basic relational representation, as depicted in Figure 4.

```

ENTITY(ename, etype, dname, added-by,
       date-added, mod-by, last-mod, nmode,
       dur-value, dur-type, comments, descr,
       security, lang, lines-code, nrecs,
       rec-cat, data-class, doc-cat)
RELSHIP(rtype, e1name, e1type, e2name, e2type,
        access-method, frequency, rel_pos)

```

FIGURE 4. Basic Relational Representation of the IRDS Entity-Relationship Model

Entities and relationships within the Dolk IRDS architecture are “similar to a semantic network where the entities are the nodes, and relationships are the arcs that connect the nodes.” Dolk at 50. Thus, an ENTITY might have primary keys ename (entity name) and etype (entity type), and a RELSHIP (relationship) might have as primary keys rtype (relationship type), e1name (first entity name), e1type (first entity type), e2name (second entity type), and e2type (second entity type).

Dolk discloses that all three levels may be implemented in ORACLE by creating relations and views, as depicted in figure 4, 5, and 6, and by using SQL commands

CREATE TABLE and CREATE VIEW commands. Figure 7 provides examples of this process of creating ENTITY and RELSHIP tables.

```
(a) Create ENTITY and RELSHIP tables.
CREATE TABLE ENTITY
(ENAME CHAR(15) NOT NULL,
 ETYPE CHAR(8) NOT NULL,
 DNAME CHAR(30),
 ADDED_BY CHAR(15) NOT NULL,
 DATE_ADDED DATE NOT NULL,
 :
 :
 DATA_CLASS CHAR(8)
 DOC_CAT CHAR(8));

CREATE TABLE RELSHIP
(RTYPE CHAR(12) NOT NULL,
 E1NAME CHAR(15) NOT NULL,
 E1TYPE CHAR(8) NOT NULL,
 E2NAME CHAR(15) NOT NULL,
 E2TYPE CHAR(8) NOT NULL,
 ACC_METHOD CHAR(10),
 FREQUENCY CHAR(10),
 REL_POS NUMBER(5));

(b) Create entity and general relationship views.
CREATE VIEW PROGRAM AS
(SELECT ANAME, DNAME, ADDED_BY, DATE_ADDED, MOD_BY, LAST_MOD, RMODS, DUR_VALUE,
 DUR_TYPE, LANG, LINEG_CODE, COMMENTS, DESCR, SECURITY
 FROM ENTITY
 WHERE ETYPE='PROGRAM');

CREATE VIEW PROCESSES AS
(SELECT E1NAME, E1TYPE, E2NAME, E2TYPE
 FROM RELSHIP
 WHERE RTYPE='PROCESSES');

(c) Create specific relationship views.
CREATE VIEW PROGRAM-PROCESSED-FILE AS
(SELECT E1NAME, E2NAME, ACCESS_METHOD
 FROM RELSHIP
 WHERE RTYPE='PROCESSES' AND E1TYPE='PROGRAM' AND
 E2TYPE='FILE');
```

FIGURE 7. Creating R/RDS Tables and Views in ORACLE

E. Zloof

QBE and the use of inquiry tables were well know in the prior art before 1990. QBE is a high-level data base management language that provides a convenient and unified style to query, update, define, and control a relational data base. Zloof, which is cited in the Shaw reference described below, taught the use of an inquiry table, as claimed in claims 5, 6, and 14 of the '259 patent. Specifically, Zloof teaches use of the Query-by-Example language (QBE), which allows users to graphically query the underlying database. When a user performs an operation to query, update, define, or control the data base, the user can enter an example of a solution to a given operation in skeleton tables, which can be associated with actual tables in the database.

Figure 13 Qualified retrieval using links

TYPE	ITEM	COLOR	SIZE	SALES	DEPT	ITEM
	P. FRUIT	GREEN			TOY	MULTI

For illustration, suppose we want all green items sold in the toy department. Fig.

13 of Zloof below shows the two skeleton tables: "TYPE" and "SALES". A user would generate blank skeleton tables of these types and then fill out the headings and required entries. The same example element must be used in both tables, to indicate that if an example item such as NUT is green, that same item is also sold by the toy department. Only if these two qualifications are met simultaneously does the item qualify as a solution. Thus, Figure 13 shows an inquiry table consisting of two skeleton tables (i.e., inquiry records) linked together by the linking item NUT. They would be analogous to levels 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.

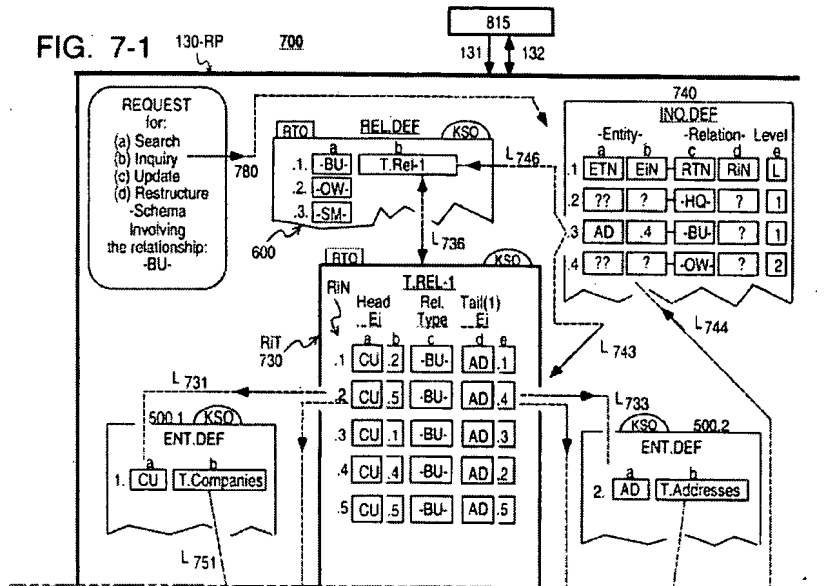


Figure 11 of Zloof (below) shows a skeleton table that prints the names of the employees who work in the toy department and earn more than \$10,000.

Figure 11 Qualified retrieval

EMP	NAME	SAL	MGR	DEPT
P		>10000		TOY

A similar example is shown Table 24 of Shaw (the employees who work in the San Jose Department, and earn more than \$20,000).

TABLE 24

DEPENDENT RETRIEVAL

```

0 SELECT ENAME,SAL, EMP ENAME SAL DNO
  DNO
FROM EMP          P.          >20000  _D
WHERE SAL > 20000
AND DNO = ANY
          DEPT  DNO  LOC
5 (SELECT DNO
  FROM DEPT
  WHERE LOC = 'SJ')

```

A snapshot can be created by giving the resultant table a name. A snapshot is a newly stored table that contains the data from the resultant table. For example, a resultant table can be created from the results of the search on the two skeleton tables in Fig. 13 of Zloof. This resultant table, once named, becomes a new stored table.

Thus, the linked retrieval in Figure 13 of Zloof is an inquiry table. And the individual skeleton tables, "TYPE" and "SALES", are the claimed inquiry records. The Zloof patent therefore discloses the portions of claims 5, 6, and 14 in the '259 patent which reference inquiry tables and records.

F. Shaw

Shaw teaches the use of an inquiry table as claimed in claims 5, 6, and 14 of the '259 patent. The Shaw patent teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal.

Shaw cites Zloof as a graphic query language which provides for defining, accessing, and modifying stored tables in a data base, and provides a particularly "user friendly" format for the terminal operator. Shaw converts the QBE skeleton tables into SQL queries, which are, in turn, used to retrieve the inquiry results from the relational database.

For the purposes of this reexamination application: Shaw teaches that an SQL query can be stored in an inquiry table for execution by a query processor on the database. The SQL query is translated from a QBE query input by a user, and the translated SQL query is stored in a table. According to one embodiment, the linear query is expressed in Structure Query Language (SQL) syntax, and the graphic query in Query by Example (QBE) syntax. Responsive to a QBE print query, an SQL command is generated which comprises the UNION of one or more select statements. Generated SQL queries are stored in DXEGFT tables. More specifically, the SQL query is generated into a buffer area (GFTSQL) in storage. The Shaw patent further teaches that the QBE query input from the user is stored in an example table, from which the query information is read in order to generate the SQL query during query processing.

Specifically, Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provide a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2. Shaw at 11:1-2. Thus, GFTTABLE 72 is an inquiry definition table, as contemplated by Doktor.

In Fig. 5 of Shaw, GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a GFTCNMPT

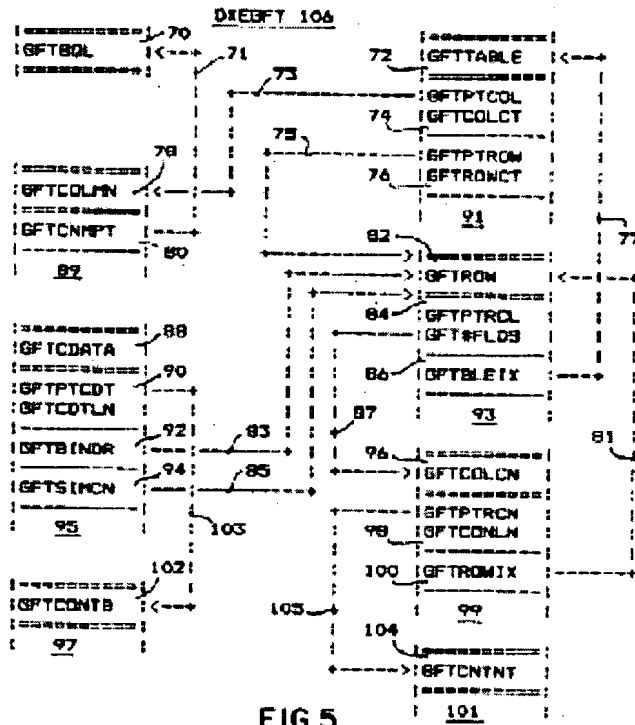


FIG. 5

field 80 providing a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL command, such as the SQL SELECT statement for the query for the employees who work in the San Jose Department, and earn more than \$20,000, as shown by Table 24 of Zloof above.

Thus, GFTTABLE 72 and the QBE skeleton or example tables constitute inquiry tables and records, as claimed. The Shaw patent therefore discloses the portions of claims 5, 6, and 14 in the '259 patent which reference inquiry tables and records.

V. DISCUSSION OF FST'S RESPONSE TO NOTICE OF NON-COMPLIANT AMENDMENT

FST filed a Response to Notice of Non-Compliant Amendment, filed on Sept. 21, 2006 (hereinafter the "Response"), attached as Exhibit PAT-A3. In this Response, FST makes numerous misleading characterizations of both the '259 patent and the cited prior art. The present proceedings are a merged re-examination and a re-issue. Therefore, as

part of Requester's presentation of the required "detailed explanation and pertinency and manner of applying the cited prior art to every claim for which re-examination is requested" in the reexamination (37 CFR 1.510; MPEP § 2214), Requester addresses FST's mischaracterizations of the '259 patent and the cited prior art.

The Examiner mailed an Office Communication on December 22, 2006, (hereinafter the "Second Office Action" or "2nd OA") in response to FST's Response. This Second Office Action is attached as **Exhibit PAT-A6**. In the Second Office Action, the Examiner has in many instances correctly noted FST's mischaracterizations and properly disregarded them. Requester appreciates the Examiner's careful attention to this case.

A. FST Identification of Alleged "Benefits Achieved by the Claimed Invention" is Unavailing as the "Benefits" are Unclaimed, and Because the "Benefits" are Disclosed by Prior Art

The '259 patent claims a relational database system that is based on the well-established entity-relation model. Under this well-established model, relation types associate entity types with each other. The database stores, until retrieval, entity and relation records within instance tables where each table holds records of a given entity or relation type. Entity and relation definition tables hold the entity and relation type definitions. The patent also mentions, in its dependent claims, entity and relation record identifiers, entity and relation table identifiers, and inquiry tables.

FST claims that the '259 "invention" discloses four additional "benefits", distinguishing it from the prior art. FST recites a number of supposed "real-world" results that the claimed inventions realize. (Response 17-18)

Interposing Metadata Between the Table Catalog and the Query: the '259 patent allegedly introduces meta tables (entity definition and relation definition tables) which sit above the table catalog (which lists the table identifiers and files locators for the entity and relation instance tables).

Two-Part Keys: the '259 patent allegedly introduces two-part keys where relation instances explicitly store the entity type of a related entity, as well as a record identifier of that entity instance record.

Inquiry Table: FST alleges that the '259 patent describes a feature whereby a relational database contains an inquiry table that stores relation types and/or entity types

to be queried.

Multi-Tailed Relation Types: the '259 patent allegedly claims relation instance records that may involve more than two entity instances. This may be known as a multi-tailed instance, where there is one head entity identifier and at least two tail entity identifiers in the relation instance record.

None of these supposed results, however, are recited in the language of the claims, nor is FST even able to identify any specific claim language that would lead to these supposed results. As the Examiner correctly notes, none of the claims recite any tangible result or function, and thus the claims are non-statutory. (2nd OA at 5, 31.)

Further, all of these features are found in the prior art cited in either this request or in the presently pending proceedings. Accordingly, all of the claims of the '259 patent are either anticipated or obvious in light of the cited prior art.

1. Interposing Metadata

a. Munz Interposes Metadata

FST concedes that all relational databases have a catalog of system tables, called a "table catalog", which list table identifiers and file locators for the tables. FST contends that its inventive contribution was to add a second layer of meta-table information, that maps an entity type or relation type to a given table identifier in the table catalog. However, this mapping is precisely what the Munz references teach. In Munz, the meta-table information maps entity type codes and relationship codes into "tabcodes", and then uses the "tabcodes" in the "descriptors" discussed in Munz to locate the actual tables and table pages where the data is stored. (Munz II at 515-516.)

FST contends that the claimed "entity definition table" has two features. It identifies an entity type, and it specifies an entity instance table. (Response at 21.) However, Munz's schema table also identifies an entity type, and also specifies an entity instance table.⁸ The entity type is identified by the "entity type code" found in the

⁸ Recall that the Munz references refer to two different types of "relationships" as that term is used in Munz. The first type is a relationship between "entities" as that term is used in Munz. This type of relationship is referred to in the First Reexamination Request as an "entity-entity relationship", which corresponds to the term "relation type" as that term is used in the '259 patent. The second type is a relationship between an "entity" and an "attribute" as those terms are used in the Munz references. This type of relationship is referred to in the First Reexamination Request as an "entity-attribute relationship", which corresponds to the term "entity type" as that term is used in the '259 patent. See First Reexamination Request at 45 n.12.

schema table records, as shown in Munz II on page 515 (figure replicated below).

x'00'	entity type code	relationship code	length-field	relationships-shipname	card	type
1 byte	1 byte	1 byte	1 byte	variable length	1 byte	2 byte
ID-field				value field		

The entity instance table is specified by the “tabcode”, which is an encoded combination of the entity type code (name) and the relationship code (name).⁹ (Munz II at 515-16.)

Similarly, FST contends that the claimed “relation instance table” has two features; identifying a relation type and specifying a relation instance table. These same features are found in Munz’s schema table. The relation type is identified by the relationship code, and the relation instance table is specified by the “tabcode”, which is an encoded combination of the entity type code (name) and the relationship code (name). (Munz II at 515.)

b. Malhotra Interposes Metadata

The Malhotra reference also teaches this mapping. Malhotra’s ERLANG system is built on top of a standard relational database, which FST concedes has a “table catalog”. (Malhotra at 2.) Malhotra adds a second layer of meta-table information on top of the standard relational database, in particular the “RELATIONSHIPS” table and the “ENT.TYPE” table.

The ENT.TYPE table identifies an entity type, and it specifies an entity instance table, just like FST contends its “entity definition table” does. The entity type is identified by the NAME field of the ENT.TYPE table (Malhotra, p. 16, line 1). the entity instance table is also specified by the NAME field. (Id.)

The RELATIONSHIPS table identifies a relation type, and it specifies a relation instance table, just like FST contends its “relation definition table” does. The relation type is identified by the REL_TYPE field of the RELATIONSHIPS table, and the

⁹ The entity type name and relation type name noted by the Examiner as identifying the entity instance table and relation instance table, respectively, are each encoded as part of the tabcode.

relation instance table is specified by the REL_NAME field of the RELATIONSHIPS table. (Malhotra at 4.)

c. **Tsichritzis Interposes Metadata**

The Tsichritzis reference also teaches this mapping. Tsichritzis' LSL system "can be thought of as a relational system implemented on a network environment, or a network system with a relational interface for end users." (Tsichritzis at 123, ¶ 2) (emphasis added.) Thus, Tsichritzis is a relational database and therefore also has the "table catalog" that FST concedes is inherent in all relational databases. Tsichritzis adds a second layer of meta-table information on top of this "table catalog", in particular the RECORD DEFINITION TABLE (RDT) and the LINK DEFINITION TABLE (LDT).

The RDT identifies an entity type, and specifies an entity instance table, just like FST contends its "entity definition table" does. The entity type is identified by the name of the record type (e.g. "Employees" for the record type example in Tsichritzis, p. 124, ¶ 2). The entity instance table is also specified by this name, since the entity instance table is the loaded record type (Tsichritzis, pg. 125, ¶ 2.)

The LDT identifies a relation type, and it specifies a relation instance table, just like FST contends its "relation instance table" does. The relation type is identified by the name of the link (e.g. "own" for the link example in Tsichritzis, p. 124, ¶ 6). The relation instance table (which is the intermediate pointer structure in Tsichritzis) is also specified by the name of the link. For example, the "Create link own from Houses to Employees" command creates an intermediate pointer structure, associated with the name "own", that relates House entity instances to Employee entity instances. (Tsichritzis, p. 125, ¶ 6-7.)

2. **Two-Part Keys**

Initially, as the Examiner has noted, claims 2, 3 and 12 fail to reflect the particular interpretation that FST has advanced in its Response for this supposedly novel feature. Therefore, all of the Tsichritzis, Munz and Malhotra references anticipate or render obvious claims 2, 3 and 12 as presently presented, as demonstrated in the First Re-Examination Request.

a. **Munz Includes FST's New Interpretation of Two-Part Keys**

The WELL database system designed by Dr. Munz does contemplate two-part keys as described by FST in the Response. The previously-submitted Munz II reference, which discussed Dr. Munz's WELL System database, clearly disclosed the desired record identifiers being stored in each relation instance record, and disclosed that each relation instance record implicitly specified the desired entity type. (First Re-Examination Request at 47.)

Submitted with this Second Re-Examination Request is an additional reference, titled "Das WEB-Modell" (hereinafter "Munz III"), also authored by Dr. Munz, which further describes the WELL System database. This reference clearly shows that Dr. Munz's system contemplated that relation instance tables could store and use both the entity type and the record identifier. See Munz III (translated pages), attached as **Exhibit PA-K**, at 155-6, FIG. 10.2.10. These translated pages recite how the data structures of the WEB model can be implemented. Munz III at 154. In FIG. 10.2, reproduced below, the relation instances of an example relation instance table are shown.

Node	ID field	Edge	Node	ID field
Hans	1096	loves	cat	523
Hans	1096	pets	cat	523
Cat	523	name	Max	144
Cat	523	chases	mouse	27
Hans	1096	hates	cat	701
Cat	701	eats	mouse	27
Cat	523	eats	mouse	27
Cat	701	name	Moritz	49
Cat	701	eats	fish	87
Hans	1096	feeds	fish	87

Figure 10.2 Triple visual display of a WEB

b. **Malhotra Includes FST's New Interpretation of Two-Part Keys**

Additionally, two-part keys are anticipated by the Malhotra reference. As noted by the Examiner, a reference may be used not only for what it expressly teaches, but for

¹⁰ This reference was published in Germany, in 1976, and thus qualifies as prior art under 35 U.S.C. § 102(b). Since the reference is in German, Requester has supplied translations of the relevant pages of the reference.

what it fairly suggests, including fair suggestions to unpreferred embodiments and to structures not specifically chosen to illustrate concepts. Second Office Action at 34 (citing CCPA caselaw). FST claims that Malhotra only allows for a single entity type as source and a single entity type as target. Response at 44, citing to Malhotra at 4. However, FST fails to cite to the very next sentences in the Malhotra reference, which provides that “[t]his restriction is not necessary if relationships are stored by pointers or by system defined keys as, for example, in EAS-E (13). It is however, a minor restriction.” FST’s selective citation to a single sentence misrepresents the teachings of the Malhotra reference. This reference plainly teaches that the purported advantage realized by two-part keys is also realizable using the teachings of Malhotra, in an alternate embodiment. By explaining that the “single entity type” restriction, which is caused by the binding of a key attribute to a single table, is not necessary in the alternate embodiment, Malhotra fairly suggests a two-part key, for example a system-defined key, that can refer to more than one source or target entity type. Therefore, even assuming that claims 2, 3 and 12 properly recited this two-part key limitation as discussed by FST, those claims would still be anticipated by the Malhotra reference, either standing alone or in combination with the Munz references cited above.

3. Inquiry Table

FST’s argument in the Response discusses several purported additional features of the “inquiry tables” claimed in claims 5, 6, and 14. However, none of these additional features are actually claimed in claims 5, 6, and 14. Therefore, none of those purported additional features can serve as a basis for overcoming the cited prior art, which art includes all of the elements actually claimed in claims 5, 6, and 14.

Furthermore, as discussed in detail in the prior section, the Shaw reference includes an inquiry table which has the purported additional features that FST discusses in its Response. Thus the Shaw reference, alone or in combination with Teorey, Huber, Kumpati, Dolk, Zloof, Munz, Malhotra and Tschritzis references anticipates or renders obvious claims 5, 6 and 14, even if those claims are somehow interpreted differently in light of FST’s argument in the Response.

4. **Multi-Tailed Relation Types**

a. **FST's Argument is Inconsistent With The Claim Language**

FST's argument in the Response again discusses several purported features, in this instance of their "relation types", which features are not recited in their claims. FST contends that claims 7-8 are directed to "multi-tailed relation types." However, neither claim 7 nor claim 8 (nor any other claim of the '259 patent, for that matter) recites any "multi-tailed relation type". The disclosure of the '259 patent explains that "multi-tailed" relation types are implemented using additional fields in the relation type records for all of these additional tails, and a "tail-activation mask" or else NULL values in the additional fields. '259 patent, 22:56-23:20. Yet the claims include no mention of additional fields, of "tail activation masks" or any of the other features discussed in the '259 patent relating to "multi-tailed relation types". Since the claims make no mention of this supposed feature, FST's argument here has no foundation in the claim language, and thus fails to point out specific claim limitations alleged to be not met by the prior art, as required by MPEP § 2666 and 37 CFR § 1.111." The Examiner has properly rejected claims 7-8 on the basis of his interpretation of the scope of these claims. If FST now wishes to change the meaning of claims 7-8, then it must amend those claims to properly recite the limitations it seeks to use to distinguish these claims from the cited prior art, if it indeed can identify any such limitations.

b. **Claims 7-8 Are Still Anticipated, Even Under FST's New Position**

Even if claims 7-8, with or without amendment, were to be read as being limited to the "multi-tailed relation types" that FST now argues, those claims would still be anticipated by the art of record in this reexamination/reissue proceeding. FST contends that a "multi-tailed relation type" describes the feature "whereby a single relation instance record may involve more than two entity instances." (Response at 27.) FST further contends that this feature "allows for more complex relation types than simple 'binary relations.'" FST then contends that claim 7 is directed to the specific case where the relation instance record has two tails, and that claim 8 is directed to the specific case where the relation instance record has three tails.

c. **Malhotra Teaches Multi-Tailed Relationships**

This feature, as now argued by FST, is taught, or at least fairly suggested, by the Malhotra reference. It is true that Malhotra's preferred embodiment teaches that relationships (aka relation types) can only have a single entity type as a source and a single entity type as a target. (Malhotra, pg. 4.) However, Malhotra also teaches that an alternate embodiment can remove this "minor restriction" to its relationships, simply by changing the way that key values are used. As Malhotra explains, "if relationships are stored by pointers or by system defined keys" then the restriction to a single target entity type "is not necessary." Id. This teaching would instruct one of skill in the art that a relationship could have multiple target entity types (i.e. multiple tails), if the simple modification taught by Malhotra were made to the way that Malhotra's relationships store key values. Thus, Malhotra teaches or at least fairly suggests the use of relation types having multiple tails, and claims 7-8 are therefore invalidated by Malhotra, even if they are given the meaning FST now argues for these claims.

d. **Teorey Teaches Multi-Tailed Relationships**

Additionally, this feature, as now argued by FST, is expressly taught by the Teorey reference, which was discussed in detail in the prior section. Teorey teaches that relations can involve more than two entities. In particular, Teorey teaches that all relationships used in any Extended Entity-Relationship based database, which includes the relational database that Teorey itself is based on, can have any number of different entity types. (Teorey, p. 201, sec. 1.3.) Teorey teaches that "the degree of a relationship is the number of entities associated with the relationship." (Id.) Thus a relationship of degree 3 (a ternary relationship) would have two tails, as shown in FIG. 4(b). Teorey depicts two alternate ways of representing the relationship shown in FIG. 4(b). The first way of representing this relationship is the multi-tailed relationship labeled "SKILL-AVAILABLE", shown with one head entity signified by "EMP-NO", and two tail entities, signified by "SKILL-NO" and "PROJ-NAME". The alternate way of representing this multi-tailed relationship is by using two single-tailed relationships, "EMP-SKILL" and "EMP-PROJ", as shown in FIG. 4(b). A relationship of degree 4 would simply be the obvious extension of FIG. 4(b) to include a fourth column. Thus, Teorey teaches the use of relation types having multiple tails, and claims 7-8 are

invalidated by Teorey, even if they are given the special and uninstantiated meaning FST now argues for these claims.

B. The '259 Patent Claims are Non-Statutory

Regarding the Examiner's Section 101 rejections, FST argues that claim 10 recites a "computer-readable medium" and therefore is directed to statutory subject matter. Response at 11. However, as the Examiner correctly notes in the 2nd OA, the claims merely recite an arrangement of data, which is non-statutory. 2nd OA at 5, 31. FST attempts to support its interpretation of the claim by citing the definition of a term, "computerized database", from the specification that appears nowhere in claim 10. FST's citation thus misrepresents the language of claim 10. This citation, even if it could somehow be read to demonstrate the presence of statutory subject matter, cannot support the patentability of claim 10 because the recited term is not found in claim 10.

FST argues that claims 1 and 10 are statutory because they have the practical application of "transforming" prior art databases into improved databases. FST contends that "the steps of the method [of claim 1] transform relational databases of the prior art into a database that allows certain changes to be made to the database schemas..." (Response at 15), and that claim 10 "is directed to a relational database processing system which provides the same benefits...". (Response at 16.) This argument mischaracterizes the claim language. There is no step of "transforming" found anywhere in claim 1, nor is any "transformed" database claimed in claim 10. Claims 1 and 10 do not operate on prior art databases to somehow turn them into different databases. These claims simply recite abstract ideas or arrangements of data, without any tangible results, as noted by the Examiner. (2nd OA at 5, 31-2.) The point to the "transformation" test is to require that a claimed invention transforms an article into something different, not merely that the claim itself is directed to some abstract idea that may be different. Despite FST's mischaracterizations of claims 1 and 10, these claims transform nothing, they merely recite an abstract method and a non-functional arrangement of data.

C. The '259 Patent is Invalid Under 35 USC 102/103 Over the Munz, Malhotra and Tschritzis References

1. Despite FST's Misrepresentations, FST's Statements Regarding Claim Breadth are Inconsistent with FST's Response

FST leads off its arguments that the claims of the '259 patent are patentable over the cited references by contending that its Preliminary Infringement Contentions ("PICs") reflect an "appropriate scope" for the claims. (Response at 20.) Requestor disputes the validity and assertions of the PICs. Nevertheless, the PICs are a reflection of FST's view as to claim breadth. Indeed, FST contends that its analysis is "consistent with those Preliminary Infringement Contentions." (Id.) The inconsistency of these positions, however, can clearly be seen by a comparison of FST's positions on dependent claims 7-8, from the PICs and from the Response.

In FST's PICs, they simply repeat their interpretation of what an entity definition table is, from claim 1, and then add on a conclusory statement that a second (or third) record would be handled the same way. (Exh. B to First Re-Examination Request, PICs at 2, 8-9.) There is nothing in FST's PICs about any "multi-tailed relation instances," nor any showing that a single relation instance record involves more than two entity instances. However, in the Response FST for the first time contends that claims 7-8 describe this "multi-tailed relation instance" feature, and contends that these claims relate to a single relation instance record that has more than two tail entity identifiers. Notably, the language of claims 7-8 is totally devoid of any language about "multi-tailed relation instances". In fact, these claims do not even mention "relation instances" at all.

Statement in the PICS [Comparing claim 7 to accused Oracle product]	Statement in the Response
The OBJ\$ table serves as an entity definition table. A data type table (an entity type) is defined in the OBJ\$ table by way of a record that contains alternative identifiers of the data table type including the NAME, obj# and OID\$ fields in the record. <u>A second record would be handled in the same way as the first.</u> (Exhibit B to First Re-Examination Request, PICs at 8)	Claims 7 and 8 describe the feature whereby a single relation instance record may involve more than two entity instances. This is known as a multi-tailed relation instance, where there is one head entity identifier and at least two tail entity identifiers in the relation instance record. (Response at 27.)

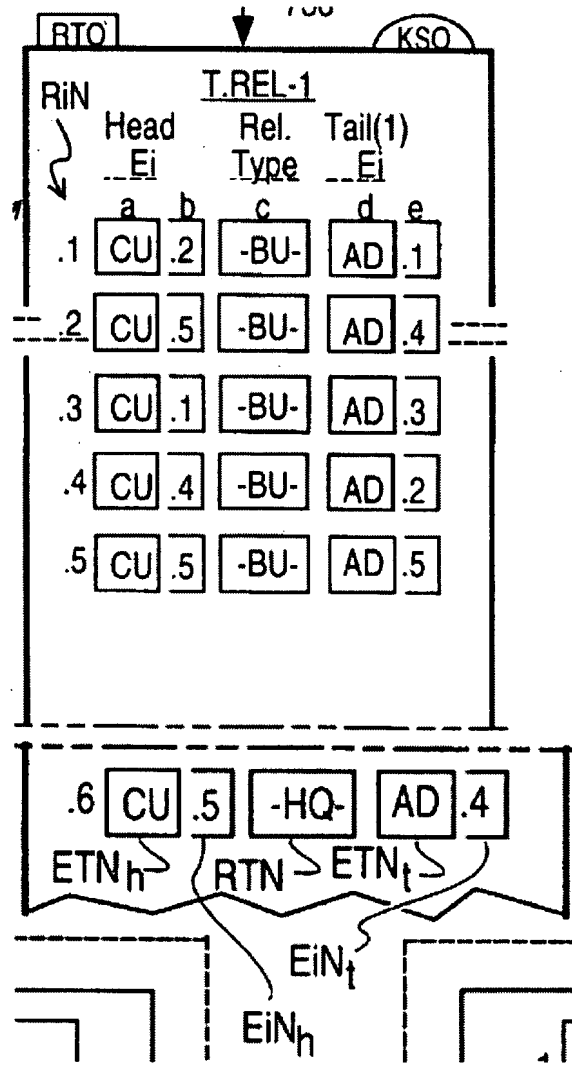
(emphasis added).

FST's obvious change in positions in the Response confirms that FST's PICs cannot be considered as an "appropriate scope" for any claim of the '259 patent, and that FST's apparently believes that its analysis can change at will, depending on who its audience is.

Node	ID field	Edge	Node	ID field
Hans	1096	loves	cat	523
Hans	1096	pets	cat	523
Cat	523	name	Max	144
Cat	523	chases	mouse	27
Hans	1096	hates	cat	701
Cat	701	eats	mouse	27
Cat	523	eats	mouse	27
Cat	701	name	Moritz	49
Cat	701	eats	fish	87
Hans	1096	feeds	fish	87

Figure 10.2 Triple visual display of a WEB (Munz III)

For example, looking at the fourth row of FIG. 10.2 (disregarding the header row), the entity coded with a record identifier (ID number) 523, with the entity type "Cat", is related to the entity coded with a record identifier (ID number) 27, with the entity type "mouse", by the relation type "chases". Similarly, looking at the fifth row, the entity coded as ID number 1096, with the entity type "Hans", is related to the entity coded as ID number 701, with the entity type "cat", by the relation type "hates". Thus this table includes relation instances which point to at least two different entity instance tables (namely the table for "cat" entities and the table for "mouse" entities). This table clearly depicts the use, as discussed in FST's Response, of two-part keys in Dr. Munz's WELL System database, which is the subject of all three of the Munz references cited to the Examiner.



Relation Instance Table (from FIG. 7-1 & 7-2 of '259 patent)

In fact, comparing Munz's FIG. 10.2 with the relation instance table depicted in FIG. 7 of the '259 patent (and replicated above), there is no apparent difference between FIG. 10.2 and the relation instance table 730 disclosed in the '259 patent. Both tables include an entity type and an ID number for the head and tail entities, as well as a field for the relation type. Therefore, even assuming that claims 2, 3 and 12 properly recited this two-part key limitation as discussed by FST, those claims would still be anticipated by Dr. Munz's database, as described in the various Munz references.

2. Tsichritzis Reference Has Relation Instance Tables

In the 2nd OA, the Examiner finds that the Tsichritzis reference "fails to disclose

the relation instance records which define a relation of a provided relation type between a provided entity and a desired entity.” On this basis, the Examiner has withdrawn his rejections under 35 USC 102(b) over the Tschritzis reference. (2nd OA at 33, ¶ 53.)

However, in FST’s Response which preceded the 2nd OA, the only argument FST presented to rebut the Examiner’s rejection in his First Office Action, was that “Tschritzis does not show the design of the intermediate pointer structures.” (Response at 30.) The Examiner’s rejection in the First Office Action supported by the facts and the discussion in the First Re-Examination Request (First Request at 8-12, 30, 39), that Tschritzis in fact teaches the claimed “relation instance tables.” Furthermore, as the Examiner has noted in maintaining other rejections in this proceeding, “it is a well settled rule that a reference must be considered not only for what it expressly teaches, but also for what it fairly suggests.” 2nd OA at 34 (citing to *In re Burckel*, 592 F.2d 1175, 201 USPQ 67 (CCPA 1979)). Furthermore, “it is an equally well settled rule that what a reference can be said to fairly suggest relates to the concepts fairly contained therein, and is not limited by the specific structure chosen to illustrate such concepts.” 2nd OA at 34 (citing to *In re Bascom*, 230 F.2d 612, 109 USPQ 98 (CCPA 1956)).

Tschritzis clearly discusses that the intermediate pointer structures are created, using the definitions found in the link definition table, when a “Create link” command is issued. (Tschritzis, p. 125, ¶ 6.) These links define connections between record types (recall that record types correspond to the entity types claimed in the ‘259 patent). Thus, the intermediate pointer structures, when created, plainly provide a connection (i.e. a relation) between the various instances of the relation between the provided entity and the desired entity. Additionally, Tschritzis at least fairly suggests a design of the intermediate pointer structures that anticipates the “relation instance tables” of the ‘259 patent.

For example, the link “own” is defined by the statement “Define link own between Houses and Employees”. This link (aka relation definition) defines a relation between the two record types (aka entity types) House and Employees. (Tschritzis, p. 124, ¶ 6.) When the “Create” command is executed, this command reads a previously-defined link, and uses that link to create the intermediate pointer structure (aka relation instance table) that corresponds to that link. For example, the command “Create link

Own from Houses to Employees” uses the previously defined link discussed above, where “the result of a create selector (link) is the construction of intermediate pointer structures which implement an access path according to the selector (link).” (Tsichritzis, p. 125, ¶ 6.) This implementation of an access path from the provided entity House to the desired entity Employee is a relation instance table. One of skill in the art would easily be able to implement a particular structure or design for the concepts fairly contained within the Tsichritzis reference.

Furthermore, these intermediate pointer structures are not merely temporary structures that disappear once a query has been executed. These structures can persist for a long time, including for the lifetime of the database if that is what is desired. For example, if the two participating record types are not locked (i.e. they are permitted to be modified), then the intermediate pointer structures are maintained. (Tsichritzis, p. 125, ¶ 6, 8.) Thus the intermediate pointer structures as taught by Tsichritzis are the same as the relation instance tables taught by the ‘259 patent. Therefore, the Tsichritzis reference was, and still is, properly asserted to invalidate all of the claims of the ‘259 patent, as the Examiner originally held in his First Office Action.

VI. DISCUSSION OF FST’S RESPONSE TO THE EXAMINER’S SECOND OFFICE ACTION

FST filed a “Response to Office Action” on March, 22 2007 (hereinafter “FST’s Response”), attached as **Exhibit PAT-A7**. This new response, yet again, appears to mischaracterize the prior art, and is mostly comprised of a re-hashing of arguments that FST made in its previous papers.

A. Wiederhold’s “Database Design Second Edition” Discloses Definition Tables

As the Examiner noted, FST cited reference CL, or Gio Wiederhold’s “Database Design Second Edition” (“Database Design”), attached as **Exhibit PA-L** in the IDS (received and stamped by the U.S.P.T.O. on October 23, 2006), but failed to provide adequate detail as to its relevance. The Examiner noted in his Second Office Action that the Database Design reference is approximately 700 pages. Second Office Action at 3. FST failed to give guidance as to the specific portions of the book that are relevant to patentability. In its Response to the Second Office Action, submitted on March 22, 2007

("FST's Response"), FST noted several sub-sections which it believed are relevant, along with explanations of those sub-sections. FST cited specifically to Sections 7-3-1, 7-3-7, 7-4-4, 7-4-5, and 9-7-6 and Figure 8-9.

FST focuses on the fact that the Database Design reference does not contain any entity definition tables or relation definition tables. To the contrary, the Database Design reference teaches a relation definition table. For example, Figure 8-5 on page 416 of the Wiederhold textbook depicts the schema for a relation definition in "STRUCTURE", within a given database under Wiederhold's model. As explained more thoroughly in the Wiederhold text, for every STRUCTURE, there is a relation.

```
PLEASE NAME YOUR DATABASE: EMPLOYEE >
I NEED TO KNOW THE STRUCTURE OF YOUR DATABASE.
PLEASE DESCRIBE EACH ITEM

ITEM  NAME  SIZE  TYPE
1     EMPLOYEE, 20,C >
2     JOB, 5,C >
3     SALARY, 8,N >
4     ADDRESS, 20,C >
5     CITY, 20,C >
6     >                                     Carriage Return terminates structure description.

EMPLOYEE CURRENTLY CONTAINS 0 72-CHARACTER RECORDS.
EMPLOYEE 'STR.D' NOW CONTAINS BASE STRUCTURE.
```

Figure 8-5 Schema creation.

Defined STRUCTUREs are also retrievable by the database user during a database session, as depicted in Figure 8-7 on page 418.

```

RETRIEVE >

PLEASE NAME YOUR DATABASE: PERSONNEL >
PERSONNEL CURRENTLY CONTAINS 848-CHARACTER RECORDS.
*STRUCTURE >
ITEM TYPE WIDTH NAME
1 C 20 EMPLOYEE
2 C 11 SOC SEC
3 N 6 SALARY
4 N 3 HRS
5 N 7 PAY
*APPEND >
EMPLOYEE SOC.SEC SALARY HRS PAY
MARSHALL MICHAEL, 347-72-6528, 283.00, 40.0 >
COLLINS WILLIAM, 402-90-3269, 6.70, 40.0 >
2 RECORDS PROCESSED
*SORT BY EMPLOYEE >
PERSONNEL 'OLD' CONTAINS YOUR UNSORTED DATABASE.
PERSONNEL IS NOW SORTED.
SHALL WE RETAIN PERSONNEL 'OLD'? NO >
*LIST >
RECNO EMPLOYEE SOC.SEC SALARY HRS PAY
1 ANDREWS KARL 403-20-0631 7.35 40 0
2 BRADFORD SUSAN 202-48-0277 4.90 40 0
3 COLLINS WILLIAM 402-90-3269 6.70 40 0
4 FRENCH MARK 519-45-0218 7.20 40 0
5 MARSHALL MICHAEL 347-72-6528 283.00 40 0
6 NELSON DONALD 311-61-2629 5.10 40 0
7 PALMER DAVID 357-48-3158 410.00 40 0
8 PARKER MARY 351-04-8200 4.10 40 0
9 RODRIGUES MARIA 373-75-7302 198.70 40 0
10 WINTON JOAN 421-98-7244 4.25 40 0
10 RECORDS PROCESSED

```

Figure 8-7 A database session.

The PERSONNEL STRUCTURE relates EMPLOYEE, SOCIAL SECURITY, SALARY, HOURS and PAY entities, each of which are defined within the STRUCTURE. Upon typing in “RETRIEVE” and entering the name of the database (i.e. “PERSONNEL”), the system indicates how many records exist in the database in question. The text also teaches entity definition tables indicating the size and type of variables in which the entity instances are encoded.

B. Requester Agrees with Examiner’s Section 101 Rejections

With respect to § 101 of the Patent Act, FST continues to re-assert a number of arguments which the Examiner considered and rejected in his Second Office Action. Second Office Action at 4-5. As described in greater detail below, FST continues to assert various uses for the subject matter described in the claims, or extensions of the subject matter in the claims. None of the language within the actual claims are directed to statutory subject matter, and thus the claims, in their current form, fail to satisfy § 101.

FST first responds to the Examiner’s rejection of claims 1-18 under 35 U.S.C. § 101 because the claimed invention was directed towards non-statutory subject matter.

FST's Response at 9. Essentially, FST appears to be rearguing that claims 1-18 are encoded in a "computer readable medium." *Id.* at 10. FST asserts that the "[t]he bit strings [of database information] are distributed spatially within a tangible medium of data storage such as an array of magnetic disks, optical disks or other information representing means capable of providing mass storage." *Id.* "Nonfunctional descriptive material' such as music, literary works, and a compilation or mere arrangement of data is nonstatutory" even if the non-functional descriptive matter is recurring on a computer readable medium. United States Patent & Trademark Office, Manual of Patent Examining Procedure § 2106.01 (8th ed. 2001). On the other hand, "[w]hen functional descriptive material is recorded on some computer-readable medium, it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized." *Id.* As Examiner correctly notes, Claims 1-18 do not explicitly recite computer readable media, and therefore do not satisfy statutory subject matter requirements.

FST next responds to the Examiner's argument that claims 1-9 fail "to recite a tangible result, a requirement for compliance with the provisions of 35 U.S.C. § 101 for a process that can be interpreted as being implemented through software." FST's Response at 11. As the Examiner notes, "[f]or a result to be tangible, it must be more than just a thought or a computation; it must have real-world value rather than an abstract result." Second Office Action at page 5. A closer examination reveals that FST's arguments regarding this issue are merely a re-hashing of its arguments from its previous papers. Requester agrees with the Examiner's conclusion that claims 1-9 fail to recite a tangible result. FST continues to assert, as it did in its first set of papers, that claim 1 does, in fact, recite steps involving the retrieval of various records, i.e. "said desired element from said desired entity instance table" in the final step. This same argument was made by FST, and considered and rejected by the Examiner before the Second Office Action. In the Second Office Action, the Examiner wrote:

Claim 1 recites a number of steps for retrieving data from a variety of tables. However, at no time is the retrieved data *written* to a table (which would constitute a tangible result), nor is there any recitation of the retrieved data being displayed to a user, nor transmitted to another computer (either of which would constitute a tangible result).

Second Office Action at 32. FST nonetheless continues to make this same argument without having amended any of its claims. Claim 1, as it is currently drafted, does not recite a tangible output visible to the user, however, and therefore fails the tangible result requirement.

FST also responds to the Examiner's conclusion by incorrectly arguing that some of the claims merely involve the rearrangement of data. Specifically, **the Examiner stated the following in** the second office action:

Claims 10-18 are claimed as a system in the preamble. However, all of the limitations comprise a mere rearrangement of data. In accordance with MPEP § 2106.01, mere arrangements of data constitute nonfunctional descriptive material, and is non-statutory under the provisions of 35 U.S.C. § 101.

Second Office Action at 31. FST responds by referring to the preamble, which specifies “[a] relational database processing system” and noting that such preamble gives life and vitality to the claims. FST's Response at 17. FST essentially concedes and agrees with the Examiner's conclusion that all of the *limitations of claim 10* merely recite the arrangement of data in tables and records.

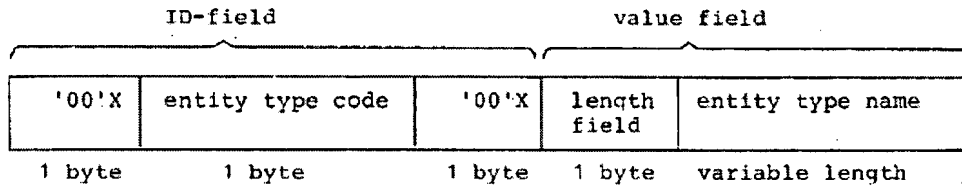
FST continues in its argument that the claimed invention does result in a “useful, concrete and tangible result.” FST's Response at 20. FST writes that “the claimed method and system allows the various schema to be changed without recompiling application queries and programs.” *Id.* In fact, this alleged improvement is not recited in any of the claims. FST also states that “[i]ndependent Claim 10 transforms an ordinary relational database processing system into one that has improved functionality and flexibility over Prior Art relational database systems.” *Id.* Requester notes that the “transformation” of which FST writes does not involve the *physical* transformation resulting from the claimed invention operating on something. Instead, the “transformation” that FST refers to¹¹ is the mere fact that the claimed arrangement of data

¹¹ To be statutory, a claimed computer-related process must either: (1) result in a physical transformation outside the computer for which a practical application in the technological arts is either disclosed in the specification or would have been known to a skilled artisan (discussed in (i) below), or (2) be limited by the language in the claim to a practical application within the technological arts. *See Diamond v. Diehr*, 450 U.S. 175, 183-84 (1981) (quoting *Cochrane v. Deener*, 94 U.S. 780, 787-88 (1877); *In re Alappat*, 33 F.3d 1526, 1543 (Fed. Cir. 1994).

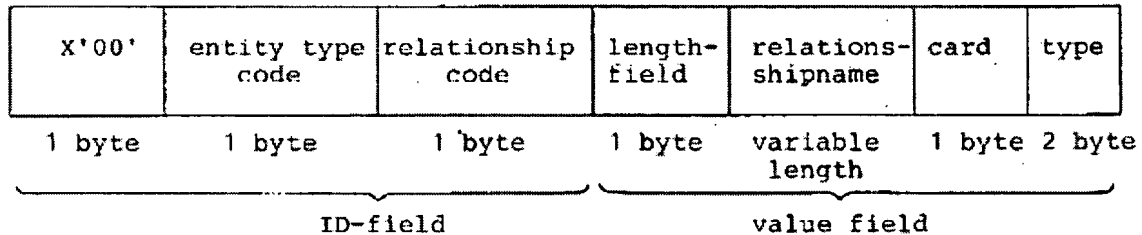
is allegedly different from prior arrangements of data. This is not the sort of “transformation” contemplated by the patent laws.

C. The Munz Reference is Anticipatory Prior Art

Turning to the Examiner’s rejection over Munz, FST first argues that entity type schema records¹², or Munz’s version of entity definition records, cannot provide valid tab codes (which designate an entity type name and a relationship name for each record). FST’s Response at 32. The Schema Record layouts are provided below.



First Schema Record Layout



Second Schema Record Layout

See Munz II at 515. According to FST, the only schema records that can have valid tabcodes are the ones in the second schema layout. FST’s Response at 32. However, Tabcodes under Munz II are merely 2 byte table identifiers. Munz II at 516. Specifically, “all tables can be uniquely designated by an entity type name and a relationship name. Internally an entity type name and a relationship name is encoded by using one byte for each name.” Munz II at 515. Contrary to what FST indicates, the first Schema Record Layout can have a valid tabcode and may function as an entity definition record. As a

¹² In the Second Office Action, the Examiner concluded that the first schema record layout corresponded to Entity type definition records, and that the second schema record layout corresponded to Relation type definition records.

tabcode is a two byte identifier containing an entity type name and a relationship name, the tabcode in the first schema record layout would be comprised of (1) the 1 byte entity type code and (2) the 1 byte hexadecimal code to the right being a ('00'X) if the record defines an entity, as opposed to a relationship. Essentially, the "relationship code" for entity definition records is always "0".

FST's second argument turns on the fact that all Munz tables only have two columns: one for 'ID' and another for 'value'. FST appears to be basing its argument on the fact that the columns do not disclose entity type names and relationship type names. Munz II does, in fact, contemplate the construction of multi-column tables, as evidenced by Figure 2-4 of the same.

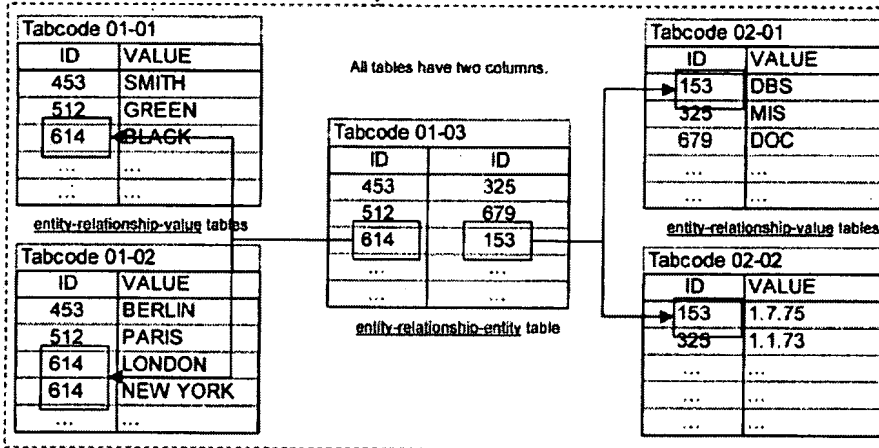
. Person (Name, Birthdate, Sex, living_at)

Black	410327	M	San Francisco
Smith	480514	UNDEF	San Diego
Green	491127	F	Salt Lake City
.	.	.	.
.	.	.	.
.	.	.	.

Figure 2-4: Record layout and organization by tables

In the above figure, entity type Person may have relationships Name, Birthdate, Sex, and Living_at.

Even if it were the case that all Munz tables contained only two columns, however, they do in fact disclose entity type names and relationship type names. Each Munz table contains ID and value columns, but each is also labeled by a tabcode, in the same manner depicted by FST (in its Diagram entitled "[Munz Expanded Fig. 22 – Drawn by FST.]") in its Response to the Examiner's Second Office Action. An excerpt of FST's Diagram (which models the tables found in Fig. 22 of Munz I at page 111) is depicted below:



FST contends that for each relationship of an entity, there exist tables like the ones depicted above. The very tables that FST depicts on page 34 of its Response to the Second Office Action disclose tabcodes representing entity type names and relation type names (i.e. 01-01, 01-02, 01-03, 02-01, and 02-02). Entity type tables would have tabcodes ending in '0', as the relationship type for entities might arbitrarily be set to '0'.

FST next incorrectly argues that entity types must be stored as data in the record under claim 2. In fact, claim 2, which is dependent on claim 1, requires that the subject relation instance record merely "specify" the desired entity by the subject entity type and subject entity record identifier. The Examiner stated that, under Munz, and in reference to claim 2:

The relation instance record also implicitly specifies the desired entity type, because Munz's relation instance tables contain records for only a single relation type, with a single desired entity type. The desired entity type is implicitly in each record. For Example, ... the desired record identifier in the first relation instance record of the "Person, works_in, (Department)" relation table [might be] "17", and the desired entity type [would be] "(Department)".

Second Office Action at 10. According to FST, Munz fails to satisfy this alleged requirement because, according to FST, "[d]ata that is implied by a query or its context is *not the same as* data being specified within the record itself." FST's Response at 3. It appears that FST construes the claim to require each record to *store* the entity type information. The word "store" does not, however, appear in claim 2.

FST argues that “each Munz ID (in a Munz relationship table) **does not** identify a single record in a single table, but rather identifies a multitude of records in a multitude of tables” FST’s Response at 36. Even if FST’s characterization of Munz were true, it would be irrelevant, because the ‘259 teaches precisely the same thing. See ‘259 Patent, Fig. 7-1 (re-using numerical identifiers in T.REL-1 and in other tables, to refer to other tables or rows). The identifiers the Patent teaches may identify a multitude of records in a multitude of tables either through their re-use **or** because they refer to sets of records or tables. Perhaps more importantly, however, FST’s mischaracterizes the use of a “record identifier.” A “record identifier” uniquely identifies a single record in a given table in which that identifier appears; under the claims, it is irrelevant whether a record identifier appears in other relationship tables. Under Munz, a desired record identifier does uniquely identify a single record in a given table in which that identifier appears. Munz therefore teaches “record identifiers.”

FST then challenges the Examiner’s conclusion that Munz “clearly teaches a non-preferred embodiment including a relational database on Figure 2.4 on page 513.” FST’s Response at 37. As the Examiner correctly noted, Figure 2.4 teaches a relational database. Thus, in Figure 2.4, each column could reflect entity instances, and the entity records within a row would be related to each other.

Person (Name, Birthdate, Sex, living_at)

Black	410327	M	San Francisco
Smith	480514	UNDEF	San Diego
Green	491127	F	Salt Lake City
.	.	.	.
.	.	.	.
.	.	.	.

Figure 2-4: Record layout and organization by tables

FST then incorrectly argues that the Munz system does not include entity instance tables. In order for the Munz system to work, tables storing entity ID numbers, which match up with different types of entities must exist.

Munz must have a way of storing the entity ID numbers in a manner such that it knows which ID numbers belong to which kind of entities. Otherwise the relationships wouldn't work. So it has to have entity instance tables so that it can keep track of which entity ID's are People, which ones are Projects, which are Departments. So it either has separate tables that do that or else the person_name, project_name, and department_name tables are entity instance tables.

Again, FST interprets the word "specify" to mean that the data, in this case, entity type information, must be stored within the relation instance record. Requester agrees with the Examiner's interpretation of the word "specify". While claims 2, 3, and 12 state that records in the relation instance table must define, relate and specify certain information, they do not explicitly claim that the relation instance table schema must include an entity type data field. Second Office Action at 36.

D. The Malhotra Reference is Anticipatory Prior Art

FST focuses again on the word "specify" in claim 2, and argues that the Examiner incorrectly concludes that Malhotra "additionally teaches ... the entity type as deduced from the key value and the context..." FST's Response at 41. There is no prohibition against specifying the entity type using pieces of data, or the query context, rather than the contents of the record itself. As noted above, the language in claim 2, as drafted, does not require the *contents* of a record to disclose the entity type.

FST then focuses on the word "associated" in claim 1, which states "...retrieving a desired entity type record containing said desired entity type from an entity definition table, wherein said desired entity type record specifies a desired entity instance table *associated* with said desired entity type..." FST's Response at 43 (emphasis added). Malhotra discloses the ENT.TYPE table, which has only one column, NAME, which acts as the table identifier. As the Examiner correctly notes, the entity type and table name can be one in the same. Second Office Action at 36. FST incorrectly construes the language in claim 1 to require that the entity type record in an entity definition table must contain both an entity type, as well as the entity instance table *associated* with the entity

type. Essentially, FST argues that both the entity type and the table identifier would have to be specified within the record in order for the association to exist. FST incorrectly assumes that the entity type record must record the *association* between the two in the claim. As in Munz, the entity type and table name can be one and the same. The claim merely requires that an association between the entity type and the table name exist, not that the association between the two be explicitly spelled out and manifest.

FST then addresses its argument that the ENT.TYPE table in Malhotra manages data only in main memory, and therefore fails to disclose a table in a relational database. FST now asserts that its claimed invention “*describes an entity definition table used to manage tables in a relational database, not simply records in main memory.*” FST’s Response at 45. FST’s statement is misleading in that it implies that references which describe records stored in main memory cannot be prior art. Whether the records are actually part of a relational database is irrelevant if they are in main memory, from FST’s perspective. In fact, there are no requirements in the claims, as currently written, that limit the claimed relational database to one that is stored in long term storage - such as a hard drive. If FST wishes to introduce this new limitation it must amend its claims.

E. The Claims Are Not Entitled to a Presumption of Validity

On a final note, FST incorrectly claims that it “is entitled to a presumption of validity for the claims issued in the original patent.” See FST’s Response at 48. It is well accepted that “[c]laims in a reissue application enjoy no “presumption of validity” whatsoever. *In re Doyle*, 482 F.2d 1385, 1392 (CCPA 1973); see also *Hewlett-Packard Co. v. Bausch & Lomb Inc.*, 882 F.2d 1556, 1563 (Fed. Cir. 1989), *In re Sneed*, 710 F.2d 1544, 1550 n.4 (Fed. Cir. 1983). As stated in 37 C.F.R. 1.176, a reissue application, including all the claims therein, is subject to “be examined in the same manner as a non-reissue, non-provisional application.” United States Patent & Trademark Office, Manual of Patent Examining Procedure § 1440, § 1445 (8th ed. 2001). “Accordingly, the claims in a reissue application are subject to any and all rejections which the examiner deems appropriate.” *Id.* This would include determinations as to utility under Section 101, or under any other condition of patentability, contrary to what FST indicates in its Response.

VII. APPLICATION OF PRIOR ART PATENTS AND PUBLICATIONS

As required by 37 C.F.R. § 1.510(b)(2), Oracle provides below a detailed explanation of how each reference renders the above-cited claims unpatentable, raising a substantial new question of patentability.

A. Teorey and Huber

Teorey and Huber together teach all limitations of claims 1-4, 7-13, 15-18 of the '259 patent as discussed in the chart below. Teorey, Huber, and Shaw together teach all limitations of claims 5-6 and 14 of the '259 patent, as discussed in the chart below:

Claim Language¹³	Teorey, Huber and Zloof/Shaw References
<p>1. A method for retrieving a desired entity of a desired entity type from a relational database, wherein said desired entity is related to a provided entity by a provided relation type associating an entity type of said provided entity with said desired entity type, said method comprising:</p>	<p>The primary aims of relational database design revolve around organizing and storing data so that the information within the database may later be accessed by the database user. According to an article by Peter Chen, "to design a database is to decide how to organize data into specific forms (record types, tables) and how to access them." Further, another related problem in database design is to make the "output of the database design process-the user schema (a description of the user view of the data)" more like the way humans represent the real world. Peter Pin-Shan Chen, <u>The entity-relationship model – A basis for the enterprise view of data</u> 77 (1977), attached as Exhibit PA-M.</p> <p>In addition an inherent component of databases is that they allow the retrieval of items, including relation type records. Front end user interfaces that enable users to easily retrieve the information in the underlying databases have been well known in the arts from at least from the mid 1980's, if not sooner. One type of front-end user interface is created using the QBE language. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990), attached as Exhibit PA-N.</p>

¹³ The claim language recited is inclusive of the amendments patentee included in its Response.

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>Teorey extends the Entity-Relationship model to a relational database (abstract, p.197). Teorey in Figure 8(f) shows that one can retrieve an engineer (desired entity) of the "ENGINEERS" (desired entity type) that "BELONGS-TO" (relation type) a particular professional association (provided entity) of the PRF-ASSOC (provided entity type).</p>
<p>retrieving a specific relation type record defining said provided relation type from a relation definition table;</p>	<p>Teorey teaches a data dictionary that includes a relation definition table that contains relation type records which define relation types. A user would supply such a relation type in making a search (query) on the database. Teorey, in combination with Huber, teaches the same things and offers different design choices.</p> <p>Teorey teaches that relation types (i.e. relationships) can be defined for a relational database. Teorey at 205 (Sec. 2.1, step 1.3). Each of these relation types defines a relationship between two or more entities. For example, FIG. 13 and Table 1 depict several relation types, including SKILL-USED, ASSIGNED-TO AND BELONGS-TO. Teorey at 216, FIG. 13, Table 1.</p> <p>Figure 13. Company personnel and project database candidate relations.</p>

Claim Language¹³

Teorey, Huber and Zloof/Shaw References

Table 1. Transformation of Entities and Relationships to Relations (Example)

- Step 2.1. Entities to relations
1. DIVISION(DIV-NO, ..., HEAD-EMP-NO)
 2. DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, ..., DIV-NO, MANAG-EMP-NO)
 3. EMPLOYER(EMP-NO, EMP-NAME, JOB-TITLE, ..., DEPT-NO, SPOUSE-EMP-NO, PC-NO)
 4. SKILL(SKILL-NO, ...)
 5. PROJECT(PROJ-NAME, ...)
 6. LOCATION(LOC-NAME, ...)
 7. EMP-MANAGER(EMP-NO, ...)
 8. EMP-ENGINEER(EMP-NO, ...)
 9. EMP-TECHNICIAN(EMP-NO, ...)
 10. EMP-SECRETARY(EMP-NO, ...)
 11. PC(PC-NO, ...)
 12. PRF-ASSOC(PA-NO, ...)
- Step 2.2. Binary or unary relationships to relations
13. BELONGS-TO(PA-NO, EMP-NO)
- Step 2.3. Ternary (or any n-ary) relationships to relations
14. SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME)
 15. ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME)

Teorey further teaches that each of the defined relation types can be transformed into a “relation” Teorey at 216, Table 1. The definitions of these relation types transformed into relations are stored in the data dictionary because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).

The use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was well known in the field. A data dictionary was considered “[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths.” They were also known as ‘catalogs’. . . *The IBM Dictionary of Computing Terms* defines “data dictionary” as “[a] list of all files, fields, and variables used in a data base management system. A “data dictionary” helps users remember what items they have to work with and how they have been defined. Particularly helpful when writing a large number of linked procedures or programs that share a data base.” *The IBM Dictionary of Computing Terms* 87 (8th Ed. 1987), attached as **Exhibit OTH-B**. *Webster’s New World Dictionary of Computer Terms* defines “data dictionary” as “1. A centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. It assists management, data base administrators, system analysts, and application programmers in planning, controlling, and evaluating the collection, storage, and use of data. 2. In the System/36 interactive data definition utility, a folder that contains field, format, and file definitions.” *Webster’s New World Dictionary of Computer Terms* 107 (3d Ed. 1988), attached as **Exhibit OTH-C**. Thus in light of Teorey, the

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p data-bbox="727 226 1317 260">data dictionary is the relation definition table.</p> <p data-bbox="727 300 1495 884">Huber further teaches that the definitions of these tables (i.e. the definitions of the relation types) are stored in the data dictionary. Huber, 6:68-7:8. Each of these table definitions contains field definitions defining the fields of each relation type, as well as location information that points to the file containing the actual base table (BT), which contains the relation data (i.e. the relation instance tables). Huber, 7:3-8. Thus, BTDEF 213 of Data Dictionary 202 specifies where BT 101 is located, the size and fields of the records representing the rows, the keys used to access the records. Huber, 15:43-49. BTDEF 213 is related in part to BT 101 by three fields: TNAME 443 is the name of BT 101; TFID 445 is the file identifier of the file which contains BT 101; and TRLEN 447 is the length of the records which represent the rows of BT 101 in the file. 15: 17-21.</p> <p data-bbox="727 930 1479 1178">Huber further teaches that the data dictionary is itself stored as a database, and thus the data dictionary itself includes tables that contain the table definitions. Huber, 7:10-13. Each table definition (BTDEF) is stored as a record in the table. Huber, 14:44-45. Thus Huber in conjunction with Teorey teach that the data dictionary can be a relation definition table.</p> <p data-bbox="727 1224 1495 1661">In addition, Huber further teaches that the REFDEF table 313 contains data defining the referenced-referencing relationship between base tables, such as the base tables which are used to create the relationship relation that is itself represented as a table. Huber, 10:43-49. The REFDEF table stores a pointer, REFDPTR 312, which points to the referenced base table, and another pointer, REFGPTR 314, which points to the referring base table. Huber, 11:1-4. These two pointers are entity table type identifiers. Thus Huber teaches that the REFDEF table 313 contains data defining the relationship relations defined in Teorey and can be a relation definition table.</p> <p data-bbox="727 1707 1490 1877">A critical component of databases is that they allow retrieval of items, including relation instance records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>

Claim Language¹³

Teorey, Huber and Zloof/Shaw References

retrieving a specific relation instance record defining a relation of said provided relation type between said provided entity and said desired entity from a relation instance table corresponding to said specific relation type record;

Teorey in combination with Huber teaches that relation instance records define relations between entities, and that these records are stored in relation instance tables, each of which corresponds to a particular relation type.

Teorey teaches that relationship relations store information that relates two or more entities to each other, and are represented as tables. Teorey at 203, FIG. 3; 212-13, FIG. 10; 216, FIG. 13.

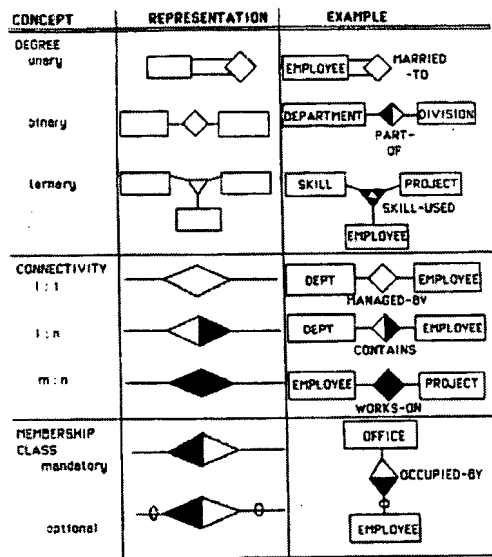
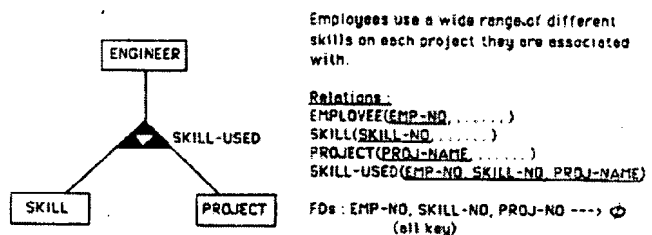


Figure 3. Fundamental EER constructs: relationship types.



SKILL-USED

EMP-NO	SKILL-NO	PROJ-NAME
3	A3	ALPHA
3	A5	BETA
3	B6	ALPHA
3	B6	BETA
4	G12	DELTA
4	G12	GAMMA
6	A3	BETA
8	C4	BETA
9	A5	ALPHA
9	G12	EPSILON
9	C6	ALPHA
9	C6	EPSILON

(b)

Figure 10. Ternary relationship transformation rules.

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References															
	<p>Huber teaches that each of the sets of related items, in this case the relationship relations (i.e. relation types) defined in Teorey, are stored in two-dimensional tables. Huber, 4:66-5:3. These tables are associated with the relation type information stored in the REFDEF table 313 and in the table definition for the relationship relation. REFDEF table 313 contains general information about the relationship (15:55-60), and the table definition (6:68-7:8).</p> <p>In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10.</p> <p>Fig. 10 depicts a plurality of relation instance records for the relation type SKILL-USED. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1.</p> <div style="text-align: center;"> <table border="1" data-bbox="919 1276 1295 1444"> <thead> <tr> <th data-bbox="919 1276 1027 1318">EMP-NO</th> <th data-bbox="1027 1276 1146 1318">SKILL-NO</th> <th data-bbox="1146 1276 1295 1318">PROJ-NAME.</th> </tr> </thead> <tbody> <tr> <td data-bbox="919 1318 1027 1350">38</td> <td data-bbox="1027 1318 1146 1350">27</td> <td data-bbox="1146 1318 1295 1350">GAMMA</td> </tr> <tr> <td data-bbox="919 1350 1027 1381">38</td> <td data-bbox="1027 1350 1146 1381">51</td> <td data-bbox="1146 1350 1295 1381">GAMMA</td> </tr> <tr> <td data-bbox="919 1381 1027 1413">38</td> <td data-bbox="1027 1381 1146 1413">27</td> <td data-bbox="1146 1381 1295 1413">DELTA</td> </tr> <tr> <td data-bbox="919 1413 1027 1444">38</td> <td data-bbox="1027 1413 1146 1444">3</td> <td data-bbox="1146 1413 1295 1444">DELTA</td> </tr> </tbody> </table> <p data-bbox="1013 1457 1040 1478">(a)</p> </div> <p>Fig. 4 depicts the relationship instance table for the relationship SKILL-USED.</p> <p>A critical component of databases is that they allow retrieval of items, including relation instance records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>	EMP-NO	SKILL-NO	PROJ-NAME.	38	27	GAMMA	38	51	GAMMA	38	27	DELTA	38	3	DELTA
EMP-NO	SKILL-NO	PROJ-NAME.														
38	27	GAMMA														
38	51	GAMMA														
38	27	DELTA														
38	3	DELTA														
retrieving a desired entity type record containing said desired	Teorey in combination with Huber teaches that entity type information is stored in entity type records in entity															

Claim Language¹³

entity type from an entity definition table, wherein said desired entity type record specifies a desired entity instance table associated with said desired entity type; and

Teorey, Huber and Zloof/Shaw References

definition tables, and that this data can be retrieved in order to operate on the database. The entity type records identify entity instance tables, including the entity instance tables containing entities that a user would desire to retrieve.

Teorey teaches that entity types (i.e. entities) can be defined for a relational database. Teorey at 204 (Sec. 2.1, step 1.1) For example, FIG. 13 and Table 1 depict a variety of entity types, including SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and others.

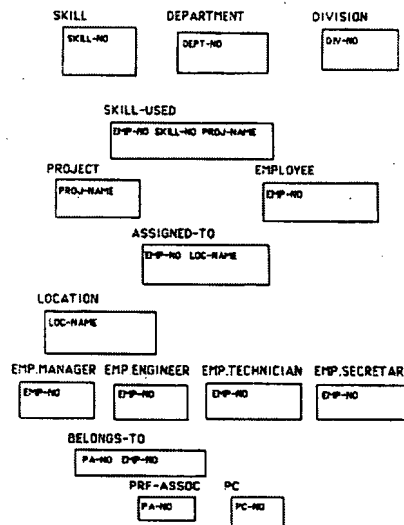


Figure 13. Company personnel and project database candidate relations.

Table 1. Transformation of Entities and Relationships to Relations (Example)

- Step 2.1. Entities to relations*
1. DIVISION(DIV-NO, ... HEAD-EMP-NO)
 2. DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, ... , DIV-NO, MANAG-EMP-NO)
 3. EMPLOYEE(EMP-NO, EMP-NAME, JOB-TITLE, ... , DEPT-NO, SPOUSE-EMP-NO, PC-NO)
 4. SKILL(SKILL-NO, ...)
 5. PROJECT(PROJ-NAME, ...)
 6. LOCATION(LOC-NAME, ...)
 7. EMP.MANAGER(EMP-NO, ...)
 8. EMP.ENGINEER(EMP-NO, ...)
 9. EMP.TECHNICIAN(EMP-NO, ...)
 10. EMP.SECRETARY(EMP-NO, ...)
 11. PC(PC-NO, ...)
 12. PRF.ASSOC(PA-NO, ...)
- Step 2.2. Binary or unary relationships to relations*
13. BELONGS-TO(PA-NO, EMP-NO)
- Step 2.3. Ternary (or any n-ary) relationships to relations*
14. SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME)
 15. ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME)

Teorey at 216, FIG. 13, Table 1. Teorey further teaches that its database system includes a data dictionary. Teorey at 217. ("If the EER constructs do not include nonkey attributes, the data requirements specification (or data dictionary) must be consulted."). Teorey further teaches that each of the defined entity types can be transformed to

Claim Language¹³**Teorey, Huber and Zloof/Shaw References**

a "relation." Teorey at 216, Table 1.

Table 1. Transformation of Entities and Relationships to Relations (Example)

Step 2.1. Entities to relations

1. DIVISION(DIV-NO, HEAD-EMP-NO)
2. DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, , DIV-NO, MANAG-EMP-NO)
3. EMPLOYEE(EMP-NO, EMP-NAME, JOB-TITLE, , DEPT-NO, SPOUSE-EMP-NO, PC-NO)
4. SKILL(SKILL-NO, . . .)
5. PROJECT(PROJ-NAME, . . .)
6. LOCATION(LOC-NAME, . . .)
7. EMP-MANAGER(EMP-NO, . . .)
8. EMP-ENGINEER(EMP-NO, . . .)
9. EMP-TECHNICIAN(EMP-NO, . . .)
10. EMP-SECRETARY(EMP-NO, . . .)
11. PC(PC-NO, . . .)
12. PRF-ASSOC(PA-NO, . . .)

Step 2.2. Binary or unary relationships to relations

13. BELONGS-TO(PA-NO, EMP-NO)

Step 2.3. Ternary (or any n-ary) relationships to relations

14. SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME)
15. ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME)

The definitions of these entity types transformed into relations are stored in the data dictionary, because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).

While Teorey clearly teaches that its relational database has and uses a data dictionary, Teorey does not describe the detailed layout of the data dictionary. However, the '593 patent to Huber does provide a detailed layout of a data dictionary for a relational database. Huber, FIG 2 (DDICT 202); FIG. 3 (ODD 211).

Huber teaches that a relational database, such as the relational database depicted in Teorey, includes sets of related items (i.e. the entity types of Teorey), which are stored in two-dimensional tables. Huber, 4:66-5:3. Huber further teaches that the definitions of these tables (i.e. the definitions of the entity types) are stored in the data dictionary. Huber, 6:68-7:8. Each of these table definitions contains field definitions defining the fields of each entity type, as well as location information that points to the file containing the actual base table (BT), which contains the entity data (i.e. the entity instance tables). Huber, 7:3-8. Thus BTDEF 213 of Data Dictionary 202 specifies where BT 101 is located, the size and fields of the records representing the rows, the keys used to access the records. Huber, 15:43-49. BTDEF 213 is related in part to BT 101 by three fields: TNAME 443 is the name of BT 101; TFID 445 is the file identifier of the file which contains BT 101; and TRLEN 447 is the length of the records which represent the rows of BT 101 in the file. 15:17-21.

Claim Language¹³

Teorey, Huber and Zloof/Shaw References

Huber teaches that the data dictionary is itself stored as a database, and thus the data dictionary itself includes tables that contain the table definitions. Huber, 7:10-13. Each table definition (BTDEF) is stored as a record in the table. Huber, 14:44-45. Thus Huber, in combination with Teorey teaches, that the data dictionary can be an entity definition table.

As indicated above, while Huber teaches the use of a entity definition table, the use of data dictionaries to map the contents of the underlying database, including the entities within it, was already common in the field. A data dictionary was considered “[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths.”

These entity definition records specify the location information that identifies the files containing the entity instance tables. The entity instance tables shown in Teorey are examples of tables which contain entity data. Teorey, FIG. 13 (SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and other tables).

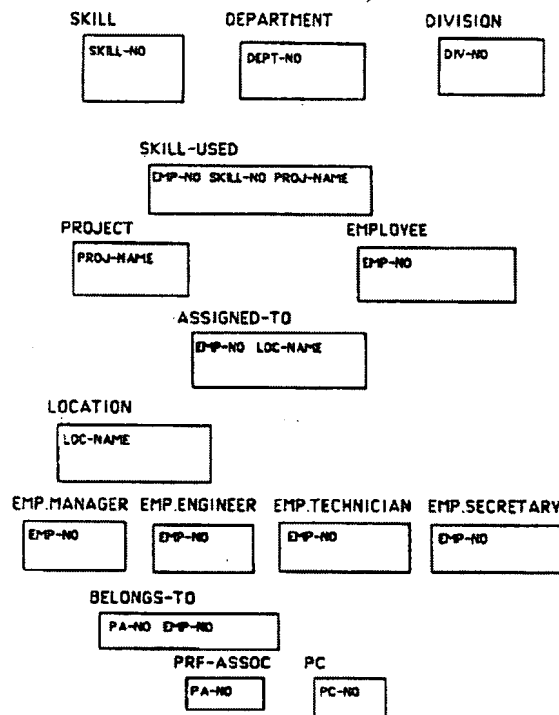


Figure 13. Company personnel and project database candidate relations.

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p style="text-align: center;">Table 1. Transformation of Entities and Relationships to Relations (Example)</p> <hr/> <p><i>Step 2.1. Entities to relations</i></p> <ol style="list-style-type: none"> 1. DIVISION(DIV-NO, . . . , HEAD-EMP-NO) 2. DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, . . . , DIV-NO, MANAG-EMP-NO) 3. EMPLOYEE(EMP-NO, EMP-NAME, JOB-TITLE, . . . , DEPT-NO, SPOUSE-EMP-NO, PC-NO) 4. SKILL(SKILL-NO, . . .) 5. PROJECT(PROJ-NAME, . . .) 6. LOCATION(LOC-NAME, . . .) 7. EMP-MANAGER(EMP-NO, . . .) 8. EMP-ENGINEER(EMP-NO, . . .) 9. EMP-TECHNICIAN(EMP-NO, . . .) 10. EMP-SECRETARY(EMP-NO, . . .) 11. PC(PC-NO, . . .) 12. PRF-ASSOC(PA-NO, . . .) <p><i>Step 2.2. Binary or unary relationships to relations</i></p> <ol style="list-style-type: none"> 13. BELONGS-TO(PA-NO, EMP-NO) <p><i>Step 2.3. Ternary (or any n-ary) relationships to relations</i></p> <ol style="list-style-type: none"> 14. SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME) 15. ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME) <hr/> <p>These entity instance tables are each associated with a corresponding entity type, as shown in FIG. 13 and Table 1. Teorey, FIG. 13, Table 1. Further entity instance tables are shown in FIG. 1 of Huber, for example the Customer Table 101(a), the Order Table 101(c), and the Parts Table 101(d). Huber, FIG. 1.</p> <p>A critical component of databases is that they allow retrieval of items, including entity type records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>
<p>retrieving said desired entity from said desired entity instance table.</p>	<p>Desired entities, just like all entities, are stored in entity instance tables, and can be retrieved therefrom.</p> <p>In Teorey, each entity instance table contains a plurality of entity instance records. The entities in Teorey are transformed into entity relations. Teorey at 208, Sec. 3.1(1). Examples of these entity relations are: EMPLOYEE (with identifier EMP-NO and descriptors EMP-NAME, HOME-ADDRESS, DATE-OF-BIRTH, JOB-TITLE, SALARY, SKILL), ENGINEER (with identifier EMP-NO and descriptors EMP-NAME, HOME-ADDRESS, SPECIALTY), SECRETARY (with identifiers EMP-NO and descriptors EMP-NAME, DATE-OF-BIRTH, SALARY, SPEED-OF-TYPING), TECHNICIAN (with identifier EMP-NO and descriptors EMP-NAME, SKILL, YEARS-OF-EXPERIENCE). Teorey, Step 1.2 at 205.</p> <p>Thus each entity relation (i.e. entity instance table) in Teorey contains a plurality of entity instance records. The entity instance tables in Huber, for example the Customer</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>Table 101(a), the Order Table 101(c), and the Parts Table 101(d), also each contain entity instance records which correspond to the instances of the entities Customer, Order and Parts. Huber, 4:66-5:13, FIG. 1.</p> <p>A critical component of databases is that they allow retrieval of items, including entity instance records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>
<p>2. The method of claim 1, wherein said relation instance record specifies said desired entity by said desired entity type and a desired record identifier.</p>	<p>Teorey teaches that each relation instance record contains a record identifier that corresponds to the desired entity instance record. For example, in the relation instance table of FIG. 4, the SKILL-USED relations instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 203, FIG. 4; 216, FIG. 13. Each relation instance record also contains a desired entity type, reflected in the column header, for example "SKILL" in the column header SKILL-NO in FIG. 4.</p> <p>Teorey further teaches that a key (i.e. a record identifier) can be a composite identifier, that is, an identifier composed of two or more attributes. Teorey at 204 (Step 1.1(5)). These two attributes could include the entity type and a record identifier, and thus anticipate this claim. Teorey teaches two alternate treatments for composite identifiers, one of which is to eliminate them where possible, but the second treatment is to retain the identifier where it is reasonably natural. Id. It would be reasonably natural to retain a composite identifier where it permitted the overloading of the record identifier column to designate two or more different target tables, as suggested by FST.</p> <p>Huber also teaches the use of two-part keys, by providing that "[e]ach row in a base table 101(x) must be uniquely identified by a primary key consisting of one or more fields." Huber, 5:28-30. Similarly, foreign keys may include "fields which are a primary key in a different base table 101(y)." Huber, 5:31-33. Therefore, a relationship relation as taught in Teorey, which is stored as a base table in Huber, would include two foreign keys to the two entity relations that it referenced, and each of those two foreign keys would be two-part keys.</p>
<p>3. The method of claim 2, wherein</p>	<p>Teorey teaches that each entity instance record contains a</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
said desired entity is identified by said desired record identifier in said desired entity instance table.	record identifier that corresponds to the desired entity. For example, in the SKILL entity instance table of FIG. 13, the SKILL entity instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 216, FIG. 13.
4. (once amended) The method of claim 1, wherein said retrieving a specific relation instance record comprises:	
retrieving a table identifier for said relation instance table from said specific relation type record; and	Teorey teaches that each relationships relation (i.e. relation type record) includes the name of the relationship. Teorey at 216, Table 1. This relationship relation is stored in the data dictionary as a table definition BT DEF (i.e. specific relation type record). Huber, 7:3-8. Huber further teaches that every table definition includes location information that permits the database to locate the file containing the corresponding table data (i.e. the relation instance table). Huber 7:3-8, 15:17-21, 15:43-45 This location information is the table identifier, as claimed.
retrieving said specific relation instance record from said relation instance table based on said specific relation type record and said provided entity.	<p>In Huber, the base table (BT) identified by the table identifier is the table (i.e. the relation instance table), which contains the specific relation instance records that are based on the specific relation type record and provided entity.</p> <p>In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1. These relation instance tables are represented in Huber as base tables (BT). Huber, 4:66-5:3; 5:17-20, 15:17-21, 15:43-45 .</p> <p>A critical component of databases is that they allow retrieval of items, including relation instance records. One type of front-end user interface which eases access to the</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References														
<p>5. The method of claim 1, further comprising retrieving data specifying said provided relation type from an inquiry table.</p>	<p>information stored in the database is one created using the QBE language, as noted above.</p> <p>Teorey teaches that relationships (relation types) are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Each of these relationships bears a name, for example SKILL_USED is the name of a relationship between the EMPLOYEE entity type and the SKILL entity type. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1. The relation instance table for SKILL_USED is represented in the database systems of Huber as a base table (BT). Huber, 4:66-5:3; 5:17-20.</p> <p>The use of inquiry tables such as ones based on the QBE language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990). While a QBE interface is pictorial in nature, a SQL's interface is more linear and textual. Id. "The user begins by specifying which table is needed for a particular query. Once a table(s) is chosen, the system displays an outline of that table, showing the table name and the names of its fields. To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Id.</p> <p>Zloof teaches Query-by-Example (QBE). When a user performs an operation, e.g., query, update, define, or control, against the data base, the user fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the database. For illustration, suppose we want all green items sold in the toy department.</p> <p>Figure 13 Qualified retrieval using links</p> <table border="1" data-bbox="737 1717 1224 1789"> <thead> <tr> <th>TYPE</th> <th>ITEM</th> <th>COLOR</th> <th>SIZE</th> <th>SALES</th> <th>DEPT</th> <th>ITEM</th> </tr> </thead> <tbody> <tr> <td></td> <td>P-FRUIT</td> <td>GREEN</td> <td></td> <td></td> <td>TOY</td> <td>PLUM</td> </tr> </tbody> </table>	TYPE	ITEM	COLOR	SIZE	SALES	DEPT	ITEM		P-FRUIT	GREEN			TOY	PLUM
TYPE	ITEM	COLOR	SIZE	SALES	DEPT	ITEM									
	P-FRUIT	GREEN			TOY	PLUM									

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p data-bbox="727 302 1490 520">Fig. 13 of Zloof above shows the two skeleton tables: "TYPE" and "SALES". These two tables are linked by the word "NUT". Thus Figure 13 shows an inquiry table consisting of two skeleton tables linked together like the level 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.</p> <p data-bbox="727 562 1490 814">The '326 patent to Shaw teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal, including relation records. Shaw takes the QBE skeleton tables , for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p data-bbox="727 856 1490 1184">Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL , such as the SQL select statement in Table 24.</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p style="text-align: center;">FIG. 5</p>
6. The method of claim 1, further comprising retrieving data	As noted above, the use of inquiry tables such as ones based on the QBE language was an inherent part of

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
<p>specifying said provided entity from an inquiry table.</p>	<p>managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to The <u>Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL for the query.</p> <p>One element of the SQL query stored in the GFTSQL 70 is the entity provided as a parameter to the query. For example, Table 24 of Shaw depicts an example of a query, in both SQL and QBE formats. Shaw, 26:59-68. In this example, the query is seeking the employees who work in the San Jose Department, and who meet a salary condition of > \$20,000. Id. The provided entity is "DEPT", and the desired entity is "EMP". Id.</p> <p>The QBE example table shown in Table 24 is another example of an inquiry table as claimed. This inquiry table contains the same DEPT provided entity information, which is retrieved from the table in order to create the SQL statement SELECT DNO FROM DEPT shown in Table 24. Shaw, 15:38-42, 26:59-68.</p>
<p>7. The method of claim 1, further comprising retrieving a second desired entity type record containing a second desired entity type from said entity definition table, wherein said second desired entity type record specifies a second desired entity instance table associated with said second desired entity type.</p>	<p>This limitation requires the presence of two entity type records, and thus two entity instance tables. Otherwise it is the same as the "retrieving a desired entity type record" element of Claim 1.</p> <p>Teorey clearly teaches relation types which create relation instance tables that involve more than two entity instances. Such relationships are disclosed as "ternary" or more generally "n-ary" relationships. For example, see the ternary relationships of FIG. 10. Teorey at 212-13, FIG.</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>10. When executing queries on these relationships, the entity type records of, for example, the EMPLOYEE, PROJECT and SKILL entities would be retrieved, for the ternary relationship SKILL-USED of FIG. 10(b). Teorey at 212; see also id at 216, Fig. 13.</p> <p>A critical component of databases is that they allow retrieval of items, including entity type records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>
<p>8. The method of claim 7, further comprising retrieving a third desired entity type record containing a third desired entity type from said entity definition table, wherein said third desired entity type record specifies a third desired entity instance table associated with said third desired entity type.</p>	<p>This limitation requires the presence of three entity type records, and thus three entity instance tables. Otherwise it is the same as the “retrieving a desired entity type record” element of Claim 1.</p> <p>Teorey clearly teaches relation types which create relation instance tables that involve more than two entity instances. Such relationships are disclosed as “ternary” or more generally “n-ary” relationships. For example, see the ternary relationships of FIG. 10.</p> <div data-bbox="760 1119 1208 1436" data-label="Diagram"> <pre> graph TD ENGINEER[ENGINEER] --- SKILL_USED{SKILL-USED} SKILL_USED --- SKILL[SKILL] SKILL_USED --- PROJECT[PROJECT] </pre> <p>The diagram shows a central inverted triangle labeled 'SKILL-USED'. A vertical line connects the top vertex of the triangle to a rectangular box labeled 'ENGINEER'. Two diagonal lines extend from the bottom-left and bottom-right vertices of the triangle to rectangular boxes labeled 'SKILL' and 'PROJECT' respectively.</p> </div> <p>Fig. 10 depicting a ternary relation.</p> <p>Teorey at 212-13, FIG. 10. When executing queries on these relationships, the entity type records of, for example, the EMPLOYEE, PROJECT and SKILL entities would be retrieved, for the ternary relationship SKILL-USED of FIG. 10(b). Teorey at 212; see also id at 216, Fig. 13.</p> <p>A critical component of databases is that they allow retrieval of items, including entity type records. One type of front-end user interface which eases access to the information stored in the database is one created using the</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
<p>9. The method of claim 1, further comprising retrieving a second specific relation instance record defining a relation of a second provided relation type between said provided entity and said desired entity from a second relation instance table corresponding to said second provided relation type record.</p>	<p>QBE language, as noted above.</p> <p>This claim as now amended requires retrieval of a second specific relation instance record of a second relation type, between the same two entities. Otherwise this claim is the same as the “retrieving a relation instance record defining a relation of said provided relation type” element of claim 1.</p> <p>Teorey expressly permits the existence of two different relationships, of two different relation types, between the same two entities. Teorey at 205 (Step 1.3(1)) (“Note that two or more relationships are allowed between the same two entities as long as the two relationships have different meanings.”). Thus a query could provide two relation types between the two entities, and both relation instance records defining the two relations between the two entity instance records would be retrieved.</p> <p>A critical component of databases is that they allow retrieval of items, including relation instance records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language, as noted above.</p>
<p>10. A relational database processing system comprising:</p>	
<p>an entity definition table containing a first entity type record defining a first entity type;</p>	<p>Teorey teaches that entity types (i.e. entities) can be defined for a relational database. Teorey at 204 (Sec. 2.1, step 1.1) For example, FIG. 13 and Table 1 depict a variety of entity types, including SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and others. Teorey at 216, FIG. 13, Table 1.</p>

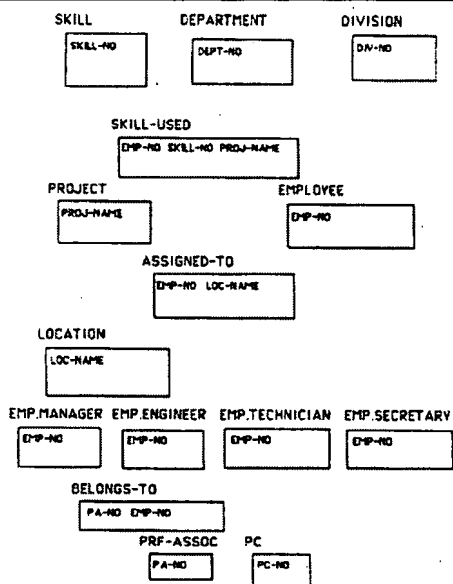


Figure 13. Company personnel and project database candidate relations.

Teorey further teaches that its database system includes a data dictionary. Teorey at 217. (“If the EER constructs do not include nonkey attributes, the data requirements specification (or data dictionary) must be consulted.”). Teorey further teaches that each of the defined entity types can be transformed to a “relation.” Teorey at 216, Table 1. The definitions of these entity types transformed into relations are stored in the data dictionary, because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).

Huber teaches that a relational database, such as the relational database depicted in Teorey, includes sets of related items (i.e. the entity types of Teorey), which are stored in two-dimensional tables. Huber, 4:66-5:3. Huber further teaches that the definitions of these tables (i.e. the definitions of the entity types) are stored in the data dictionary. Huber, 6:68-7:8. Each of these table definitions contains field definitions defining the fields of each entity type, as well as location information that points to the file containing the actual base table (BT), which contains the entity data (i.e. the entity instance tables). Huber, 7:3-8. Thus BTDEF 213 of Data Dictionary 202 specifies where BT 101 is located, the size and fields of the records representing the rows, the keys used to access the records. Huber, 15:43-49. BTDEF 213 is related in part to

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>BT 101 by three fields: TNAME 443 is the name of BT 101; TFID 445 is the file identifier of the file which contains BT 101; and TRLEN 447 is the length of the records which represent the rows of BT 101 in the file. 15: 17-21.</p> <p>Huber teaches that the data dictionary is itself stored as a database, and thus the data dictionary itself includes tables that contain the table definitions. Huber, 7:10-13. Each table definition (BTDEF) is stored as a record in the table. Huber, 14:44-45. Thus Huber, in combination with Teorey teaches, that the data dictionary can be an entity definition table.</p>
<p>a first entity instance table associated with said first entity type;</p>	<p>The entity instance tables shown in Teorey are examples of tables which contain entity data. Teorey, FIG. 13 (SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and other tables). These entity instance tables are each associated with a corresponding entity type, as shown in FIG. 13 and Table 1. Teorey, FIG. 13, Table 1. Further entity instance tables are shown in FIG. 1 of Huber, for example the Customer Table 101(a), the Order Table 101(c), and the Parts Table 101(d). Huber, FIG. 1.</p>
<p>a plurality of entity instance records stored in said first entity instance table;</p>	<p>In Teorey, each entity instance table contains a plurality of entity instance records. The entities in Teorey are transformed into entity relations. Teorey at 208, Sec. 3.1(1). These entity relations are similar in layout to the relationship relations shown in FIGS. 4 and 10, except they contain entity instance records rather than relationship instance records. Teorey at 203, FIG. 4; 212-13, FIG. 10.</p> <p>Examples of these entity relations are: EMPLOYEE (with identifier EMP-NO and descriptors EMP-NAME, HOME-ADDRESS, DATE-OF-BIRTH, JOB-TITLE, SALARY, SKILL), ENGINEER (with identifier EMP-NO and descriptors EMP-NAME, HOME-ADDRESS, SPECIALTY), SECRETARY (with identifiers EMP-NO and descriptors EMP-NAME, DATE-OF-BIRTH, SALARY, SPEED-OF-TYPING), TECHNICIAN (with identifier EMP-NO and descriptors EMP-NAME, SKILL, YEARS-OF-EXPERIENCE). Teorey, Step 1.2 at 205: Thus each entity relation (i.e. entity instance table) in Teorey contains a plurality of entity instance records.</p> <p>The entity instance tables in Huber, for example the Customer Table 101(a), the Order Table 101(c), and the</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>Parts Table 101(d), also each contain entity instance records which correspond to the instances of the entities Customer, Order and Parts. Huber, 4:66-5:13, FIG. 1.</p>
<p>a relation definition table containing a first relation type record defining a provided relation type;</p>	<p>Teorey also teaches that relation types (i.e. relationships) can be defined for a relational database. Teorey at 205 (Sec. 2.1, step 1.3). For example, FIG. 13 and Table 1 depict several relation types, including SKILL-USED, ASSIGNED-TO AND BELONGS-TO. Teorey at 216, FIG. 13, Table 1. Teorey further teaches that each of the defined relation types can be transformed into a "relation" Teorey at 216, Table 1. The definitions of these relation types transformed into relations are stored in the data dictionary because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).</p> <p>Huber further teaches that the definitions of these tables (i.e. the definitions of the relation types) are stored in the data dictionary. Huber, 6:68-7:8. Each of these table definitions contains field definitions defining the fields of each relation type, as well as location information that points to the file containing the actual base table (BT), which contains the relation data (i.e. the relation instance tables). Huber, 7:3-8. Thus, BTDEF 213 of Data Dictionary 202 specifies where BT 101 is located, the size and fields of the records representing the rows, the keys used to access the records. Huber, 15:43-49.</p> <p>Huber further teaches that the data dictionary is itself stored as a database, and thus the data dictionary itself includes tables that contain the table definitions. Huber, 7:10-13. Each table definition (BTDEF) is stored as a record in the table. Huber, 14:44-45. Thus Huber in conjunction with Teorey teach that the data dictionary can be a relation definition table.</p> <p>As indicated above, while Huber teaches the use of a relation definition table, the use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was already common in the field. A data dictionary was considered "[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths."</p>

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References																																										
	<p>Huber further teaches that the REFDEF table 313 contains data defining the referenced-referencing relationship between base tables, such as the base tables which are used to create the relationship relation that is itself represented as a table. Huber, 10:43-49. The REFDEF table stores a pointer, REFDPTR 312, which points to the referenced base table, and another pointer, REFGPTR 314, which points to the referring base table. Huber, 11:1-4. These two pointers are entity table type identifiers. Thus Huber teaches that the REFDEF table 313 contains data defining the relationship relations defined in Teorey.</p>																																										
<p>a first relation instance table associated with said provided relation type; and</p>	<p>Teorey teaches that relationship relations are represented as tables. Teorey at 203, FIG. 4; 212-13, FIG. 10; 216, FIG. 13. For example, in FIG. 10 SKILL-USED is a relation type associated with the following relation instance table:</p> <table border="1" data-bbox="756 863 1256 1146"> <thead> <tr> <th colspan="3">SKILL-USED</th> </tr> <tr> <th>EMP-NO</th> <th>SKILL-NO</th> <th>PROJ-NAME</th> </tr> </thead> <tbody> <tr><td>3</td><td>A3</td><td>ALPHA</td></tr> <tr><td>3</td><td>A5</td><td>BETA</td></tr> <tr><td>3</td><td>B6</td><td>ALPHA</td></tr> <tr><td>3</td><td>B6</td><td>BETA</td></tr> <tr><td>4</td><td>G12</td><td>DELTA</td></tr> <tr><td>4</td><td>G12</td><td>GAMMA</td></tr> <tr><td>8</td><td>A3</td><td>BETA</td></tr> <tr><td>8</td><td>C4</td><td>BETA</td></tr> <tr><td>9</td><td>A5</td><td>ALPHA</td></tr> <tr><td>9</td><td>G12</td><td>EPSILON</td></tr> <tr><td>9</td><td>C6</td><td>ALPHA</td></tr> <tr><td>9</td><td>C6</td><td>EPSILON</td></tr> </tbody> </table> <p>(b)</p> <p>Figure 10. Ternary relationship transformation rules.</p> <p>Huber teaches that each of the sets of related items, in this case the relationship relations (i.e. relation types) defined in Teorey, are stored in two-dimensional tables. Huber, 4:66-5:3. These tables are associated with the relation type information stored in the REFDEF table 313 and in the table definition for the relationship relation. REFDEF table 313 contains general information about the relationship (15:55-60), and the table definition (6:68-7:8).</p>	SKILL-USED			EMP-NO	SKILL-NO	PROJ-NAME	3	A3	ALPHA	3	A5	BETA	3	B6	ALPHA	3	B6	BETA	4	G12	DELTA	4	G12	GAMMA	8	A3	BETA	8	C4	BETA	9	A5	ALPHA	9	G12	EPSILON	9	C6	ALPHA	9	C6	EPSILON
SKILL-USED																																											
EMP-NO	SKILL-NO	PROJ-NAME																																									
3	A3	ALPHA																																									
3	A5	BETA																																									
3	B6	ALPHA																																									
3	B6	BETA																																									
4	G12	DELTA																																									
4	G12	GAMMA																																									
8	A3	BETA																																									
8	C4	BETA																																									
9	A5	ALPHA																																									
9	G12	EPSILON																																									
9	C6	ALPHA																																									
9	C6	EPSILON																																									
<p>a first relation instance record of said provided relation type, said first relation instance record relating a desired entity in one of said entity instance records to a provided entity.</p>	<p>In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED</p>																																										

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1.
11. The relational database processing system of claim 10, wherein each of said entity instance records is identified by a record identifier.	Teorey teaches that the entity instance records have key fields, which uniquely identify the entity instances. Teorey at 198 (“The major interattribute dependencies are between the entity keys (unique identifiers) of different entities that are captured in the ER modeling process.”). Huber also teaches that the entity instance records, stored in the base tables, have primary key fields. Huber, 5:28-31.
12. The relational database processing system of claim 10, wherein said first relation instance record contains a desired record identifier and a desired entity type corresponding to a desired entity instance record containing said desired entity.	<p>Teorey teaches that each relation instance record contains a record identifier that corresponds to the desired entity instance record. For example, in the relation instance table of FIG. 4, the SKILL-USED relations instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 203, FIG. 4; 216, FIG. 13. Each relation instance record also contains a desired entity type, reflected in the column header, for example “SKILL” in the column header SKILL-NO in FIG. 4.</p> <p>Teorey further teaches that a key (i.e. a record identifier) can be a composite identifier, that is, an identifier composed of two or more attributes. Teorey at 204 (Step 1.1(5)). These two attributes could include the entity type and a record identifier, and thus anticipate this claim. Teorey teaches two alternate treatments for composite identifiers, one of which is to eliminate them where possible, but the second treatment is to retain the identifier where it is reasonably natural. Id. It would be reasonably natural to retain a composite identifier where it permitted the overloading of the record identifier column to designate two or more different target tables, as suggested by FST.</p> <p>Huber also teaches the use of two-part keys, by providing that “[e]ach row in a base table 101(x) must be uniquely identified by a primary key consisting of one or more fields.” Huber, 5:28-30. Similarly, foreign keys may include plural “fields which are a primary key in a different base table 101(y).” Huber, 5:31-33. Therefore, a relationship relation as taught in Teorey, which is stored as a base table in Huber, would include two foreign keys to the two entity relations that it referenced, and each of those two foreign keys would be two-part keys.</p>
13. The relational database	Teorey teaches that relationship relations are stored in

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
<p>processing system of claim 10, wherein said first relation type record comprises a table identifier identifying said first relation instance table.</p>	<p>tables. Teorey at 203, FIG. 4; 216, FIG. 13. Huber further teaches that the definitions of these tables (i.e. the definitions of the relation types) are stored in the data dictionary. Huber, 6:68-7:8. Each of these table definitions contains field definitions defining the fields of each relation type, as well as location information that points to the file containing the actual base table (BT), which contains the relation data (i.e. the relation instance tables). Huber, 7:3-8. Thus the relation type records include an identifier that points to the relation instance table.</p>
<p>14. The relational database processing system of claim 10, further comprising an inquiry table containing an inquiry record, wherein said inquiry record specifies said provided relation type and said provided entity.</p>	<p>Teorey teaches that relationships (relation types) are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Each of these relationships bears a name, for example SKILL_USED is the name of a relationship between the EMPLOYEE entity type and the SKILL entity type. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1. The relation instance table for SKILL_USED is represented in the database systems of Huber as a base table (BT). Huber, 4:66-5:3; 5:17-20.</p> <p>Shaw takes the QBE skeleton tables, for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL, such as the SQL select statement in Table 24 (inquiry record).</p> <p>Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1. Thus for example, SKILL_USED is the name of a relationship between a</p>

Claim Language¹³	Teorey, Huber and Zloof/Shaw References
	provided EMPLOYEE entity and a desired SKILL entity.
15. The relational database processing system of claim 10 further comprising:	
a second entity instance table associated with a second entity type; and	Teorey teaches two entity instance tables, for example the SKILL and DEPARTMENT entity instance tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.
wherein said entity definition table contains a second entity type record containing said second entity type and associating said second entity instance table with said second entity type.	<p>Teorey teaches that all entities, including both SKILL and DEPARTMENT entities, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Huber teaches that the data dictionary contains a set of table definitions. Huber, 6:68-7:3. A set, by its very nature, can contain two (or more) entity type records.</p> <p>As indicated above, while Huber teaches the use of a relation definition table, the use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was already common in the field. A data dictionary was considered “[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths.”</p>
16. The relational database processing system of claim 15 further comprising:	
a third entity instance table associated with a third entity type; and	Teorey teaches three entity instance tables, for example the SKILL, DEPARTMENT and DIVISION entity tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
	<p>Figure 13. Company personnel and project database candidate relations.</p>
<p>wherein said entity definition table contains a third entity type record containing said third entity type and associating said third entity instance table with said third entity type.</p>	<p>Teorey teaches that all entities, including SKILL, DEPARTMENT and DIVISION entities, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Huber teaches that the data dictionary contains a set of table definitions. Huber, 6:68-7:3. A set, by its very nature, can contain three (or more) entity type records.</p> <p>As indicated above, while Huber teaches the use of a relation definition table, the use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was already common in the field. A data dictionary was considered “[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths.”</p>
<p>17. The relational database processing system of claim 10 further comprising:</p>	

Claim Language ¹³	Teorey, Huber and Zloof/Shaw References
a second relation instance table associated with a second relation type; and	Teorey teaches two relation instance tables, for example the SKILL-USED and ASSIGNED-TO relation instance tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.
wherein said relation definition table contains a second relation type record containing said second relation type and associating said second relation instance table with said second relation type.	Teorey teaches that all relationships, including both SKILL-USED and ASSIGNED-TO relationships, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Huber teaches that the data dictionary contains a set of table definitions. Huber, 6:68-7:3. A set, by its very nature, can contain two (or more) relation type records. Huber further teaches that the REFDEF table used to define relationships contains a separate record for each relationship. Huber, 10:43-51; 15:50-55. Thus the relation definition table can contain two (or more) relation definition records, each of which are associated to a different relation instance table (e.g. SKILL-USED and ASSIGNED-TO relation instance tables).
18. The relational database processing system of claim 17 further comprising:	
a third relation instance table associated with a third relation type; and	Teorey teaches three relation instance tables, for example the SKILL-USED, ASSIGNED-TO and BELONGS-TO relation instance tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.
wherein said relation definition table contains a third relation type record containing said third relation type and associating said third relation instance table with said third relation type.	Teorey teaches that all relationships, including SKILL-USED, ASSIGNED-TO and BELONGS-TO relationships, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Huber teaches that the data dictionary contains a set of table definitions. Huber, 6:68-7:3. A set, by its very nature, can contain three (or more) relation type records. Huber further teaches that the REFDEF table used to define relationships contains a separate record for each relationship. Huber, 10:43-51; 15:50-55. Thus the relation definition table can contain three (or more) relation definition records, each of which are associated to a different relation instance table (e.g. SKILL-USED, ASSIGNED-TO and BELONGS-TO relation instance tables).

B. Teorey and Kumpati

Teorey and Kumpati together teach all limitations of claims 1-4, 7-13, 15-18 of the '259 patent as discussed in the chart below. Teorey, Kumpati and Zloof and/or Shaw

together teach all limitations of claims 5-6 and 14 of the '259 patent, as discussed in the chart below:

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
<p>1. A method for retrieving a desired entity of a desired entity type from a relational database, wherein said desired entity is related to a provided entity by a provided relation type associating an entity type of said provided entity with said desired entity type, said method comprising:</p>	<p>Data dictionaries were well-known in the arts in 1990. The IBM Dictionary of Computing Terms defines "data dictionary" as "[a] list of all files, fields, and variables used in a data base management system. A "data dictionary" helps users remember what items they have to work with and how they have been defined. Particularly helpful when writing a large number of linked procedures or programs that share a data base." <u>The IBM Dictionary of Computing Terms 87 (8th Ed. 1987).</u></p> <p>Kumpati teaches that the active data dictionary of Kumpati is used to process queries for desired data, including the entities stored in the data files of Kumpati. Kumpati, 5:52-68.</p> <p>It would have been obvious to combine the data dictionary of Kumpati with the relational database of Teorey to help users remember what items they have to work with and how they have been defined.</p> <p>Teorey in Figure 8(f) shows that one can retrieve an engineer (desired entity) of the "ENGINEERS" (desired entity type) that "BELONGS-TO" (relation type) a particular professional association (provided entity) of the PRF-ASSOC (provided entity type).</p> <div data-bbox="743 1381 1230 1591"> <p>Every professional association could have many members who are engineers, or no engineers. Every engineer could belong to many professional associations, or none.</p> <p>Relations: PRF-ASSOC(A-NO, ...) ENGINEER(EMP-NO, ...) BELONGS-TO(A-NO, EMP-NO)</p> <p>(f)</p> </div>
<p>retrieving a specific relation type record defining said provided relation type from a relation definition table;</p>	<p>Teorey in combination with Kumpati teaches a data dictionary that includes a relation definition table that contains relation type records which define relation types. A user would supply such a relation type in making an inquiry to the query module 124 of Kumpati. Kumpati, 5:63-68. The query processor 124</p>

¹⁴ The claim language recited is inclusive of the amendments patentee included in its Response.

Claim Language¹⁴

Teorey, Kumpati, and Zloof/Shaw References

retrieves the identity of the various requested data elements from the data dictionary. Kumpati, 3:66-4:2.

Teorey teaches that relation types (i.e. relationships) can be defined for a relational database. Teorey at 205 (Sec. 2.1, step 1.3). Each of these relation types defines a relationship between two or more entities. For example, FIG. 13 and Table 1 depict several relation types, including SKILL-USED, ASSIGNED-TO AND BELONGS-TO. Teorey at 216, FIG. 13, Table 1.

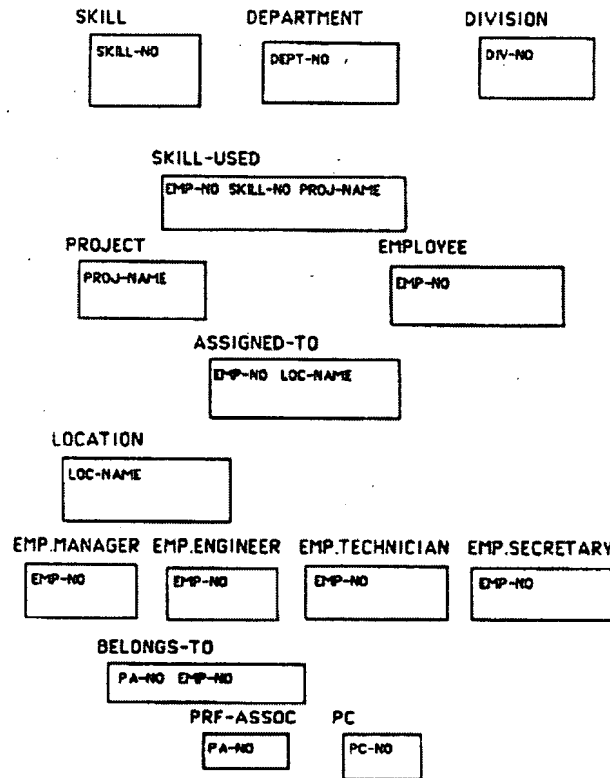


Figure 13. Company personnel and project database candidate relations.

Claim Language¹⁴

Teorey, Kumpati, and Zloof/Shaw References

Table 1. Transformation of Entities and Relationships to Relations (Example)

- Step 2.1. Entities to relations*
1. DIVISION(DIV-NO, ..., HEAD-EMP-NO)
 2. DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, ..., DIV-NO, MANAG-EMP-NO)
 3. EMPLOYEE(EMP-NO, EMP-NAME, JOB-TITLE, ..., DEPT-NO, SPOUSE-EMP-NO, PC-NO)
 4. SKILL(SKILL-NO, ...)
 5. PROJECT(PROJ-NAME, ...)
 6. LOCATION(LOC-NAME, ...)
 7. EMP-MANAGER(EMP-NO, ...)
 8. EMP-ENGINEER(EMP-NO, ...)
 9. EMP-TECHNICIAN(EMP-NO, ...)
 10. EMP-SECRETARY(EMP-NO, ...)
 11. PC(PC-NO, ...)
 12. PRF-ASSOC(PA-NO, ...)
- Step 2.2. Binary or unary relationships to relations*
13. BELONGS-TO(PA-NO, EMP-NO)
- Step 2.3. Ternary (or any n-ary) relationships to relations*
14. SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME)
 15. ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME)

Teorey further teaches that each of the defined relation types can be transformed into a “relation” Teorey at 216, Table 1. The definitions of these relation types transformed into relations are stored in the data dictionary because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).

Kumpati further teaches that the definitions of these tables (i.e. the definitions of the relation types) are stored in the data dictionary. Kumpati, 8:35-59. The data dictionary contains the entity set (i.e. relation definition table) ERSET, which is the set of all relationship sets (i.e. relation types) contained in the database. Id. The table ERSET includes, for example, records that define the relation types Building-Rooms and Exits-Rooms. Id.

While Kumpati teaches the use of a relation definition table, the use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was already common in the field. The IBM Dictionary of Computing Terms defines “data dictionary” as “[a] list of all files, fields, and variables used in a data base management system. A “data dictionary” helps users remember what items they have to work with and how they have been defined. Particularly helpful when writing a large number of linked procedures or programs that share a data base.” The IBM Dictionary of Computing Terms 87 (8th Ed. 1987). Webster’s NewWorld Dictionary of Computer Terms defines “data dictionary” as “1. A centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. It assists management, data base

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>administrators, system analysts, and application programmers in planning, controlling, and evaluating the collection, storage, and use of data. 2. In the System/36 interactive data definition utility, a folder that contains field, format, and file definitions.” <u>Webster’s NewWorld Dictionary of Computer Terms</u> 107 (3d Ed. 1988).</p> <p>Under Kumpati, the user can make “use of simple commands to control, order and query not only the underlying data controlled by the database management system but also the contents of the data dictionary.” Kumpati, Abstract. Thus, this “capability enables the user to write generic application programs which are logically independent of the data since the subject database management system enables the user/application program to access all data in the database independent of each application program’s data model.” Id.</p> <p>Further, a critical component of databases is that they allow retrieval of items, including relation type records. One type of front-end user interface which eases access to the information stored in the database is one created using the QBE language. According to <u>The Database Step-by-Step</u> textbook, “The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility.” To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information.” Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p>
<p>retrieving a specific relation instance record defining a relation of said provided relation type between said provided entity and said desired entity from a relation instance table corresponding to said specific relation type record;</p>	<p>Teorey in combination with Kumpati teaches that relation instance records define relations between entities, and that these records are stored in relation instance tables, each of which corresponds to a particular relation type. The query processor 124 retrieves the identity of the various requested data elements from the data dictionary. Kumpati, 3:66-4:2.</p> <p>Teorey teaches that relationship relations store information that relates two or more entities to each other, and are represented as tables. Teorey at 202,</p>

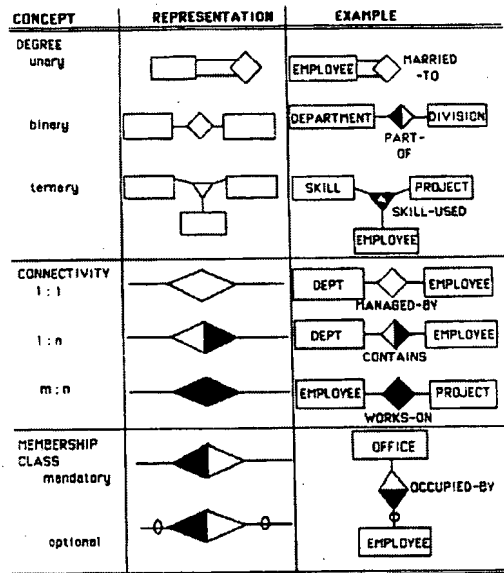


Figure 3. Fundamental EXR constructs: relationship types.

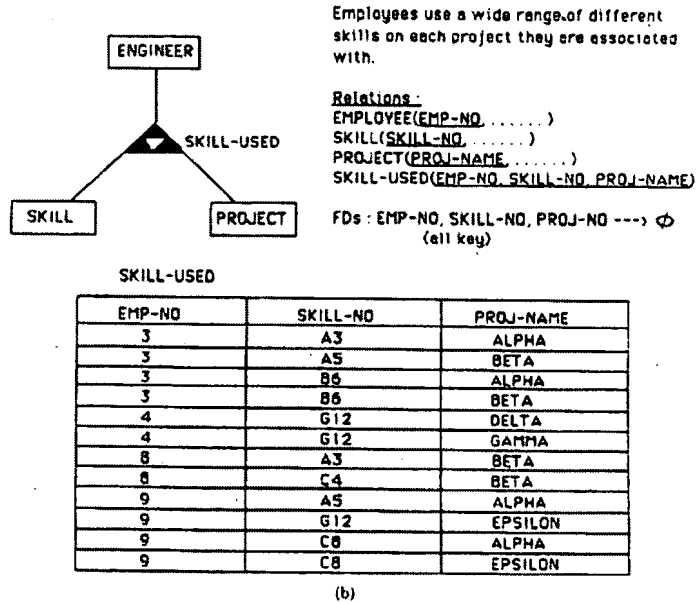


Figure 10. Ternary relationship transformation rules.

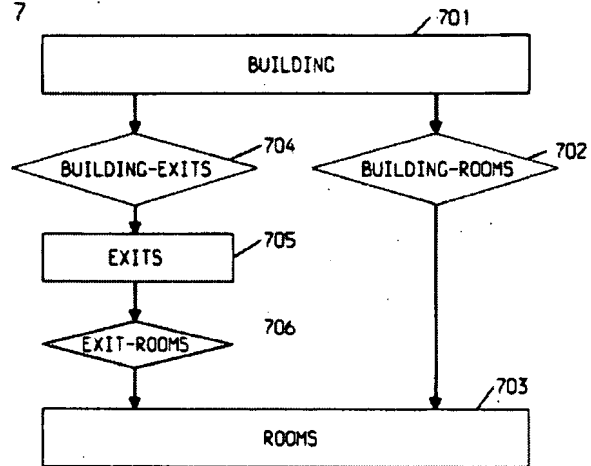
Kumpati teaches that each of the sets of related items, in this case the relationship relations (i.e. relation types) defined in Teorey, are stored in two-dimensional tables in the data files. Kumpati, 4:44-66. For example, the relation instance table Building-

Claim Language¹⁴

Teorey, Kumpati, and Zloof/Shaw References

Rooms 702 stores the set of all relationships between buildings and rooms. Kumpati, 9:43-44, FIG. 7.

FIG. 7



This relation instance table is associated with the relationship set (i.e. relation type) “Buildings-Rooms” that is stored in the relation definition table ERSET. Kumpati, 8:35-39.

In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10.

SKILL-USED	EMP-NO	SKILL-NO	PROJ-NAME
	38	27	GAMMA
	38	51	GAMMA
	38	27	DELTA
	38	3	DELTA

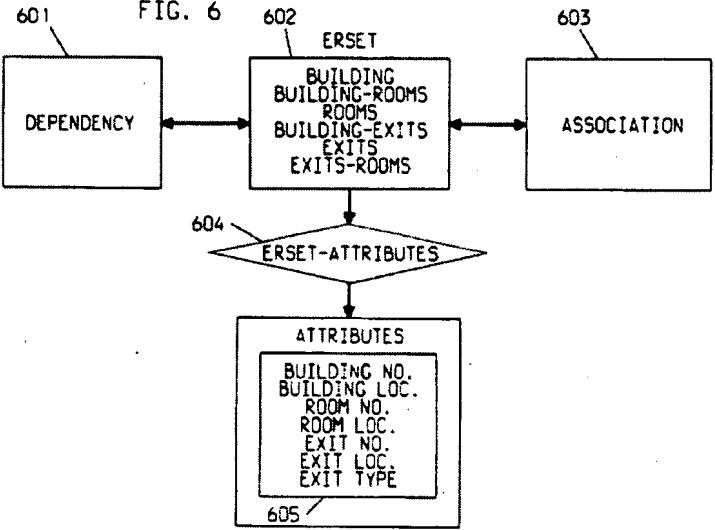
(a)

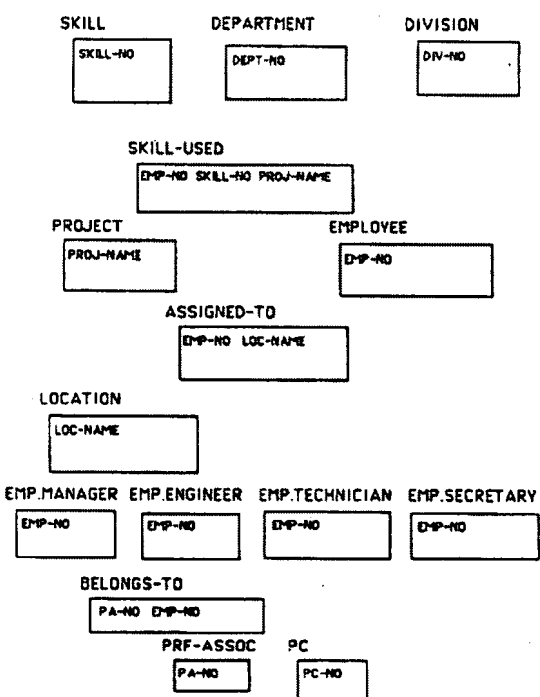
Fig. 4 of Teorey depicts the relationship instance table for the relationship SKILL-USED.

Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table.

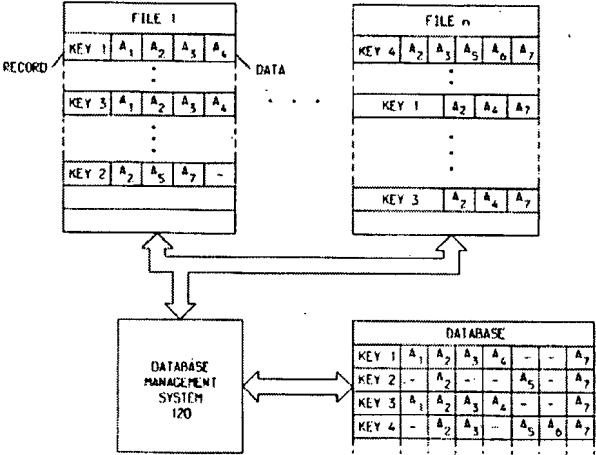
Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>Teorey at 203, FIG. 4; 216, FIG. 13, Table 1.</p> <p>In Kumpati, the relation instance records also contain information that relates the two entities in the relationship defined by the relation instance record. For example, attributes of the Buildings-Rooms relationship include information that identifies what rooms are located in what buildings. Kumpati, 9:43-46.</p>
<p>retrieving a desired entity type record containing said desired entity type from an entity definition table, wherein said desired entity type record specifies a desired entity instance table associated with said desired entity type; and</p>	<p>Teorey in combination with Kumpati teaches that entity type information is stored in entity type records in entity definition tables, and that this data can be retrieved in order to operate on the database. The entity type records identify entity instance tables, including the entity instance tables containing entities that a user would desire to retrieve. The query processor 124 retrieves the identity of the various requested data elements from the data dictionary. Kumpati, 3:66-4:2.</p> <p>Teorey teaches that entity types (i.e. entities) can be defined for a relational database. Teorey at 204 (Sec. 2.1, step 1.1) For example, FIG. 13 and Table 1 depict a variety of entity types, including SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and others. Teorey at 216, FIG. 13, Table 1.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>The diagram illustrates various database entities and their attributes:</p> <ul style="list-style-type: none"> SKILL: SKILL-NO DEPARTMENT: DEPT-NO DIVISION: DIV-NO SKILL-USED: EMP-NO, SKILL-NO, PROJ-NAME PROJECT: PROJ-NAME EMPLOYEE: EMP-NO ASSIGNED-TO: EMP-NO, LOC-NAME LOCATION: LOC-NAME EMP.MANAGER: EMP-NO EMP.ENGINEER: EMP-NO EMP.TECHNICIAN: EMP-NO EMP.SECRETARY: EMP-NO BELONGS-TO: PA-NO, EMP-NO PRF-ASSOC: PA-NO PC: PC-NO <p>Figure 13. Company personnel and project database candidate relations.</p> <p>Table 1. Transformation of Entities and Relationships to Relations (Example)</p> <p><i>Step 2.1. Entities to relations</i></p> <ol style="list-style-type: none"> DIVISION(DIV-NO, ..., HEAD-EMP-NO) DEPARTMENT(DEPT-NO, DEPT-NAME, ROOM-NO, PHONE-NO, ..., DIV-NO, MANAG-EMP-NO) EMPLOYEE(EMP-NO, EMP-NAME, JOB-TITLE, ..., DEPT-NO, SPOUSE-EMP-NO, PC-NO) SKILL(SKILL-NO, ...) PROJECT(PROJ-NAME, ...) LOCATION(LOC-NAME, ...) EMP.MANAGER(EMP-NO, ...) EMP.ENGINEER(EMP-NO, ...) EMP.TECHNICIAN(EMP-NO, ...) EMP.SECRETARY(EMP-NO, ...) PC(PC-NO, ...) PRF.ASSOC(PA-NO, ...) <p><i>Step 2.2. Binary or unary relationships to relations</i></p> <ol style="list-style-type: none"> BELONGS-TO(PA-NO, EMP-NO) <p><i>Step 2.3. Ternary (or any n-ary) relationships to relations</i></p> <ol style="list-style-type: none"> SKILL-USED(EMP-NO, SKILL-NO, PROJ-NAME) ASSIGNED-TO(EMP-NO, LOC-NAME, PROJ-NAME) <p>Teorey further teaches that its database system includes a data dictionary. Teorey at 217. (“If the EER constructs do not include nonkey attributes, the data requirements specification (or data dictionary) must be consulted.”). Teorey further teaches that each of the defined entity types can be transformed to a “relation.” Teorey at 216, Table 1. The definitions of these entity types transformed into relations are stored in the data dictionary, because that is the location</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).</p> <p>While Teorey clearly teaches that its relational database has and uses a data dictionary, Teorey does not describe the detailed layout of the data dictionary. However, the '661 patent to Kumpati does provide a detailed layout of a data dictionary for a relational database. Kumpati, 8:30-9:27, FIG. 6.</p>  <pre> graph TD 601[601 DEPENDENCY] <--> 602[602 ERSET BUILDING BUILDING-ROOMS ROOMS BUILDING-EXITS EXITS EXITS-ROOMS] 602 <--> 603[603 ASSOCIATION] 602 --> 604{604 ERSET-ATTRIBUTES} 604 --> 605[605 ATTRIBUTES BUILDING NO. BUILDING LOC. ROOM NO. ROOM LOC. EXIT NO. EXIT LOC. EXIT TYPE] </pre> <p>Kumpati further provides that “[i]t is well known in the art to provide a data dictionary in an database management system.” Kumpati, 5:51-52. The data dictionary contains the entity set (i.e. entity definition table) ERSET, which is the set of all entity sets (i.e. entity types) contained in the database. Kumpati, 8:35-39. The table ERSET includes, for example, records that define the entity types Building, Rooms and Exits. Id.</p> <p>Kumpati teaches the entity definition table having entity type records. These entity definition records specify the location information that identifies the files containing the entity instance tables. The entity instance tables shown in Teorey are examples of tables which contain entity data. Teorey, FIG. 13 (SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and other tables).</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	 <p data-bbox="771 1302 1315 1333">Figure 13. Company personnel and project database candidate relations.</p> <p data-bbox="727 1333 1429 1648">These entity instance tables are each associated with a corresponding entity type, as shown in FIG. 13 and Table 1. Teorey, FIG. 13, Table 1. Further entity instance tables are shown in FIG. 7 of Kumpati, for example the Building Table 701, the Exits Table 705, and the Rooms Table 703. Kumpati, 9:29-42, FIG. 7. These entity instance tables are associated with the corresponding entity types "Building", "Rooms" and "Exits".</p>
retrieving said desired entity from said desired entity instance table.	Desired entities, just like all entities, are stored in entity instance tables, and can be retrieved therefrom. In Kumpati, the database update processor 122 retrieves the requested data from database 130 and stores it in a designated segment of a file buffer 121, where it is buffered for use by the application program

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>that requested it. Kumpati, 4:8-17.</p> <p>In Teorey, each entity instance table contains a plurality of entity instance records. The entities in Teorey are transformed into entity relations. Teorey at 208, Sec. 3.1(1). These entity relations are similar in layout to the relationship relations shown in FIGS. 4 and 10, except they contain entity instance records rather than relationship instance records. Teorey at 203, FIG. 4; 212-13, FIG. 10, at 205 Step 1.2. Thus each entity relation (i.e. entity instance table) in Teorey contains a plurality of entity instance records. The entity instance tables in Kumpati, for example the Building Table 701, the Exits Table 705, and the Rooms Table 703, also each contain entity instance records which correspond to the instances of the entity types "Building", "Rooms" and "Exits". An example of the records contained within a typical file representing an instance table is shown in FIG. 2. Kumpati, 4:44-56.</p>
<p>2. The method of claim 1, wherein said relation instance record specifies said desired entity by said desired entity type and a desired record identifier.</p>	<p>Teorey teaches that each relation instance record contains a record identifier that corresponds to the desired entity instance record. For example, in the relation instance table of FIG. 4, the SKILL-USED relations instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 203, FIG. 4; 216, FIG. 13. Each relation instance record also contains a desired entity type, reflected in the column header, for example "SKILL" in the column header SKILL-NO in FIG. 4. Kumpati also teaches that the relation instance records have key fields. Kumpati, 7:66-68. Kumpati further teaches that the relation instance records identify the desired entity type. For example, the Building-Rooms table includes attributes that specify the entity type "Building" and the entity type "Room" for each relation instance. Kumpati, 9:43-45.</p> <p>Even under FST's new argument, advanced in their Response, Teorey anticipates this claim. Response, 22-23. Teorey further teaches that a key (i.e. a record identifier) can be a composite identifier, that is, an identifier composed of two or more attributes. Teorey at 204 (Step 1.1(5)). These two attributes could include the entity type and a record identifier, and thus</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>anticipate this claim. Teorey teaches two alternate treatments for composite identifiers, one of which is to eliminate them where possible, but the second treatment is to retain the identifier where it is reasonably natural. Id. It would be reasonably natural to retain a composite identifier where it permitted the overloading of the record identifier column to designate two or more different target tables, as suggested by FST. Therefore, a relationship relation as taught in Teorey, which is stored as a relationship table in Kumpati, would include two foreign keys to the two entity relations that it referenced, and each of those two foreign keys would be two-part keys, as discussed in FST's Response.</p>
<p>3. The method of claim 2, wherein said desired entity is identified by said desired record identifier in said desired entity instance table.</p>	<p>Teorey teaches that each entity instance record contains a record identifier that corresponds to the desired entity. For example, in the SKILL entity instance table of FIG. 13, the SKILL entity instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 216, FIG. 13. Kumpati also teaches that the entity instance records have key fields (i.e. Key 1, Key 2, ...) that are used to identify the entities stored in the records of the entity instance tables. Kumpati, 4:51-56, FIG. 2.</p> <p>FIG. 2</p>  <p>The diagram illustrates a data management system. At the top, two files are shown: FILE 1 and FILE n. FILE 1 contains records with key fields (KEY 1, KEY 2, KEY 3) and attribute fields (A1, A2, A3, A4). FILE n contains records with key fields (KEY 1, KEY 3) and attribute fields (A2, A3, A4, A7). Below these files is a DATABASE MANAGEMENT SYSTEM 120, which is connected to a DATABASE table. The DATABASE table contains records with key fields (KEY 1, KEY 2, KEY 3, KEY 4) and attribute fields (A1, A2, A3, A4, A5, A6, A7). Arrows indicate data flow from the files to the system and between the system and the database table.</p>
<p>4. (once amended) The method of claim 1, wherein said retrieving a specific relation instance record comprises:</p>	
<p>retrieving a table identifier for said</p>	<p>Teorey teaches that each relationships relation (i.e.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
relation instance table from said specific relation type record; and	<p>relation type record) includes the name of the relationship. Teorey at 216, Table 1. This relationships relation is stored in the data dictionary as a table definition (i.e. relation type record). Kumpati, 5:58-63. These definitions include information (i.e. a table identifier) that is used to ascertain the physical location in the database of the requested data. Kumpati, 3:66-4:2. This location information is the table identifier as claimed.</p>
retrieving said specific relation instance record from said relation instance table based on said specific relation type record and said provided entity.	<p>In Kumpati, the data table identified by the table identifier is the table which contains the specific relation instance records that are based on the specific relation type record and entity provided, for example the Building-Rooms, Building-Exits, and Exit-Rooms relationship sets in FIG. 7. Kumpati, 9:43-58, FIG. 7.</p> <p>FIG. 7</p> <p>In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1.</p>

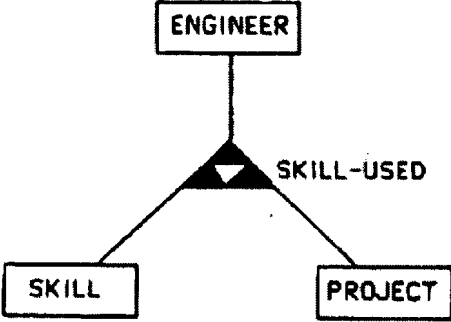
Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References														
<p>5. The method of claim 1, further comprising retrieving data specifying said provided relation type from an inquiry table.</p>	<p>Teorey teaches that relationships (relation types) are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Each of these relationships bears a name, for example SKILL_USED is the name of a relationship between the EMPLOYEE entity type and the SKILL entity type. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1. The relation instance table for SKILL_USED is represented in the database systems of Huber as a base table (BT). Huber, 4:66-5:3; 5:17-20.</p> <p>The use of inquiry tables such as ones based on the QBE language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to <u>The Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990). While a QBE interface is pictorial in nature, a SQL's interface is more linear and textual. Id. "The user begins by specifying which table is needed for a particular query. Once a table(s) is chosen, the system displays an outline of that table, showing the table name and the names of its fields. To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Id.</p> <p>Zloof teaches Query-by-Example (QBE). When a user performs an operation, e.g., query, update, define, or control, against the data base, the user fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the database. For illustration, suppose we want all green items sold in the toy department.</p> <p>Figure 13 Qualified retrieval using links</p> <table border="1" data-bbox="740 1751 1227 1818"> <thead> <tr> <th>TYPE</th> <th>ITEM</th> <th>COLOR</th> <th>DEPT</th> <th>SALES</th> <th>DEPT</th> <th>ITEM</th> </tr> </thead> <tbody> <tr> <td></td> <td>PLANT</td> <td>GREEN</td> <td></td> <td></td> <td>TOY</td> <td>TRUCK</td> </tr> </tbody> </table>	TYPE	ITEM	COLOR	DEPT	SALES	DEPT	ITEM		PLANT	GREEN			TOY	TRUCK
TYPE	ITEM	COLOR	DEPT	SALES	DEPT	ITEM									
	PLANT	GREEN			TOY	TRUCK									

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p data-bbox="727 449 1433 667">Fig. 13 of Zloof above shows the two skeleton tables: "TYPE" and "SALES". These two tables are linked by the word "NUT". Thus Figure 13 shows an inquiry table consisting of two skeleton tables linked together like the level 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.</p> <p data-bbox="727 709 1433 961">The '326 patent to Shaw teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal, including relation records. Shaw takes the QBE skeleton tables , for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p data-bbox="727 1003 1433 1360">Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL , such as the SQL select statement in Table 24.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p style="text-align: center;">FIG. 5</p>
	<p>Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1.</p> <p>. The query translator of Shaw receives as an input a QBE query, which is a "graphic language query expressed as one or more elements . . . appearing in rows and columns of an example table including one or more source and target tables." Shaw, 2:38-42. Where the source table included in the example table is a relationship relation, as discussed in Teorey, this example table contains the name of the source table, which is "data specifying said provided relation type" as claimed. When the example query is translated as part of the query processing, this relation type data is retrieved in order to form part of the SQL query submitted to the DBMS. Shaw, 15:29-31.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
<p>6. The method of claim 1, further comprising retrieving data specifying said provided entity from an inquiry table.</p>	<p>Shaw takes the QBE skeleton tables , for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL statement.</p> <p>One element of the SQL query stored in the GFTSQL 70 table of the DXEGFT tables 106 is the entity provided as a parameter to the query. For example, Table 24 of Shaw depicts an example of a query, in both SQL and QBE formats. Shaw, 26:59-68. In this example, the query is seeking the employees who work in the San Jose Department, and who meet a salary condition of > \$20,000. Id. The provided entity is "DEPT", and the desired entity is "EMP". Id.</p> <p>The QBE example table shown in Table 24 is another example of an inquiry table as claimed. This inquiry table contains the same DEPT provided entity information, which is retrieved from the table in order to create the SQL statement SELECT DNO FROM DEPT shown in Table 24. Shaw, 15:38-42, 26:59-68.</p>
<p>7. The method of claim 1, further comprising retrieving a second desired entity type record containing a second desired entity type from said entity definition table, wherein said second desired entity type record specifies a second desired entity instance table associated with said second desired entity type.</p>	<p>This limitation requires the presence of two entity type records, and thus two entity instance tables. Otherwise it is the same as the "retrieving a desired entity type record" element of Claim 1.</p> <p>FST's Response advances a new construction for this claim, the scope of which appears to be inconsistent with the claim language. Under FST's new construction, which relates to "multi-tailed" relation types, this limitation is still anticipated by Teorey. FST contends that this claim "describe[s] the feature whereby a single relation instance record may involve more than two entity instances." Response at 27. Furthermore, FST recites that these instances "allow[]</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>for more complex relation types than simple binary relationships.” Id.</p> <p>Teorey clearly teaches relation types which create relation instance tables that involve more than two entity instances. Such relationships are disclosed as “ternary” or more generally “n-ary” relationships. For example, see the ternary relationships of FIG. 10. Teorey at 212-13, FIG. 10. When executing queries on these relationships, the entity type records of, for example, the EMPLOYEE, PROJECT and SKILL entities would be retrieved, for the ternary relationship SKILL-USED of FIG. 10(b). Teorey at 212; see also id at 216, Fig. 13.</p>
<p>8. The method of claim 7, further comprising retrieving a third desired entity type record containing a third desired entity type from said entity definition table, wherein said third desired entity type record specifies a third desired entity instance table associated with said third desired entity type.</p>	<p>This limitation requires the presence of three entity type records, and thus three entity instance tables. Otherwise it is the same as the “retrieving a desired entity type record” element of Claim 1.</p> <p>FST’s Response advances a new construction for this claim, the scope of which appears to be inconsistent with the claim language. Under FST’s new construction, which relates to “multi-tailed” relation types, this limitation is still anticipated by Teorey. FST contends that this claim “describe[s] the feature whereby a single relation instance record may involve more than two entity instances.” Response at 27. Furthermore, FST recites that these instances “allow[] for more complex relation types than simple binary relationships.” Id.</p> <p>Teorey clearly teaches relation types which create relation instance tables that involve more than two entity instances. Such relationships are disclosed as “ternary” or more generally “n-ary” relationships. For example, see the ternary relationships of FIG. 10. Teorey at 212-13, FIG. 10.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	 <p data-bbox="727 648 1232 682">Fig. 10 depicting a ternary relationship.</p> <p data-bbox="727 722 1421 905">When executing queries on these relationships, the entity type records of, for example, the EMPLOYEE, PROJECT and SKILL entities would be retrieved, for the ternary relationship SKILL-USED of FIG. 10(b): Teorey at 212; see also id at 216, Fig. 13.</p>
<p data-bbox="232 909 695 1272">9. (Once amended) The method of claim 1, further comprising retrieving a second specific relation instance record defining a relation of a second provided relation type between said provided entity and said desired entity from a second relation instance table corresponding to said second provided relation type record.</p>	<p data-bbox="727 909 1398 1125">This claim as now amended requires retrieval of a second specific relation instance record of a second relation type, between the same two entities. Otherwise this claim is the same as the “retrieving a relation instance record defining a relation of said provided relation type” element of claim 1.</p> <p data-bbox="727 1167 1437 1524">Teorey expressly permits the existence of two different relationships, of two different relation types, between the same two entities. Teorey at 205 (Step 1.3(1)) (“Note that two or more relationships are allowed between the same two entities as long as the two relationships have different meanings.”). Thus a query could provide two relation types between the two entities, and both relation instance records defining the two relations between the two entity instance records would be retrieved.</p>
<p data-bbox="232 1533 703 1600">10. A relational database processing system comprising:</p>	
<p data-bbox="232 1608 695 1717">an entity definition table containing a first entity type record defining a first entity type;</p>	<p data-bbox="727 1608 1437 1854">Teorey teaches that entity types (i.e. entities) can be defined for a relational database. Teorey at 204 (Sec. 2.1, step 1.1) For example, FIG. 13 and Table 1 depict a variety of entity types, including SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and others. Teorey at 216, FIG. 13, Table 1.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>The diagram illustrates various database candidate relations. At the top, three relations are shown: SKILL (with attribute SKILL-NO), DEPARTMENT (with attribute DEPT-NO), and DIVISION (with attribute DIV-NO). Below these is SKILL-USED (with attributes EMP-NO, SKILL-NO, and PROJ-NAME). The next row contains PROJECT (with attribute PROJ-NAME) and EMPLOYEE (with attribute EMP-NO). Below these are ASSIGNED-TO (with attributes EMP-NO and LOC-NAME) and LOCATION (with attribute LOC-NAME). The next row contains four relations: EMP.MANAGER (EMP-NO), EMP.ENGINEER (EMP-NO), EMP.TECHNICIAN (EMP-NO), and EMP.SECRETARY (EMP-NO). Below these are BELONGS-TO (with attributes PA-NO and EMP-NO), PRF-ASSOC (with attribute PA-NO), and PC (with attribute PC-NO).</p>
	<p>Figure 13. Company personnel and project database candidate relations.</p> <p>Teorey further teaches that its database system includes a data dictionary. Teorey at 217. (“If the EER constructs do not include nonkey attributes, the data requirements specification (or data dictionary) must be consulted.”). Teorey further teaches that each of the defined entity types can be transformed to a “relation.” Teorey at 216, Table 1. The definitions of these entity types transformed into relations are stored in the data dictionary, because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).</p> <p>While Teorey clearly teaches that its relational database has and uses a data dictionary, Teorey does not describe the detailed layout of the data dictionary. However, the ‘661 patent to Kumpati does provide a detailed layout of a data dictionary for a relational database. Kumpati, 8:30-9:27, FIG. 6.</p>

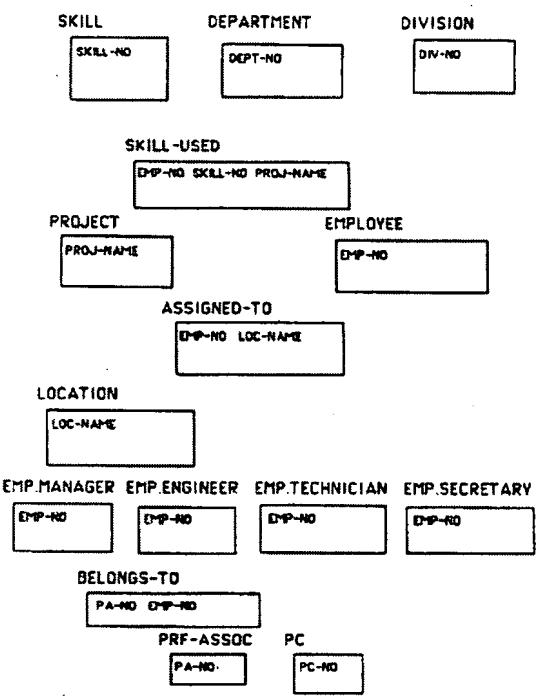
Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p data-bbox="747 226 1446 814"> </p> <p data-bbox="727 835 1421 1161"> Kumpati further provides that “[i]t is well known in the art to provide a data dictionary in an database management system.” Kumpati, 5:51-52. The data dictionary contains the entity set (i.e. entity definition table) ERSET, which is the set of all entity sets (i.e. entity types) contained in the database. Kumpati, 8:35-39. The table ERSET includes, for example, records that define the entity types Building, Rooms and Exits. Id. </p>
<p data-bbox="233 1169 638 1272"> a first entity instance table associated with said first entity type; </p>	<p data-bbox="727 1169 1421 1640"> The entity instance tables shown in Teorey are examples of tables which contain entity data. Teorey, FIG. 13 (SKILL, DEPARTMENT, DIVISION, PROJECT, EMPLOYEE and other tables). These entity instance tables are each associated with a corresponding entity type, as shown in FIG. 13 and Table 1. Teorey, FIG. 13, Table 1. Further entity instance tables are shown in FIG. 7 of Kumpati, for example the Building Table 701, the Exits Table 705, and the Rooms Table 703. Kumpati, 9:29-42, FIG. 7. These entity instance tables are associated with the corresponding entity types “Building”, “Rooms” and “Exits”. </p>
<p data-bbox="233 1648 703 1751"> a plurality of entity instance records stored in said first entity instance table; </p>	<p data-bbox="727 1648 1437 1896"> In Teorey, each entity instance table contains a plurality of entity instance records. The entities in Teorey are transformed into entity relations. Teorey at 208, Sec. 3.1(1). These entity relations are similar in layout to the relationship relations shown in FIGS. 4 and 10, except they contain entity instance records rather than relationship instance records. Teorey at </p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>203, FIG. 4; 212-13, FIG. 10. Thus each entity relation (i.e. entity instance table) in Teorey contains a plurality of entity instance records. The entity instance tables in Kumpati, for example the Building Table 701, the Exits Table 705, and the Rooms Table 703, also each contain entity instance records which correspond to the instances of the entity types "Building", "Rooms" and "Exits". An example of the records contained within a typical file representing an instance table is shown in FIG. 2. Kumpati, 4:44-56.</p>
<p>a relation definition table containing a first relation type record defining a provided relation type;</p>	<p>Teorey also teaches that relation types (i.e. relationships) can be defined for a relational database. Teorey at 205 (Sec. 2.1, step 1.3). For example, FIG. 13 and Table 1 depict several relation types, including SKILL-USED, ASSIGNED-TO AND BELONGS-TO. Teorey at 216, FIG. 13, Table 1. Teorey further teaches that each of the defined relation types can be transformed into a "relation" Teorey at 216, Table 1. The definitions of these relation types transformed into relations are stored in the data dictionary because that is the location where Teorey teaches that attributes of candidate relations are stored and can be retrieved from. Teorey at 217 (Step 3.2).</p> <p>Kumpati further teaches that the definitions of these tables (i.e. the definitions of the relation types) are stored in the data dictionary. Kumpati, 8:35-59. The data dictionary contains the entity set (i.e. relation definition table) ERSET, which is the set of all relationship sets (i.e. relation types) contained in the database. Id. The table ERSET includes, for example, records that define the relation types Building-Rooms and Exits-Rooms. Id.</p>
<p>a first relation instance table associated with said provided relation type; and</p>	<p>Teorey teaches that relationship relations are represented as tables. Teorey at 203, FIG. 4; 212-13, FIG. 10; 216, FIG. 13. Kumpati teaches that each of the sets of related items, in this case the relationship relations (i.e. relation types) defined in Teorey, are stored in two-dimensional tables in the data files. Kumpati, 4:44-66. For example, the relation instance table Building-Rooms 702 stores the set of all relationships between buildings and rooms. Kumpati, 9:43-44, FIG. 7. This relation instance table is associated with the relationship set (i.e. relation type) "Buildings-Rooms" that is stored in the relation</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
<p>a first relation instance record of said provided relation type, said first relation instance record relating a desired entity in one of said entity instance records to a provided entity.</p>	<p>definition table ERSET. Kumpati, 8:35-39.</p> <p>In Teorey, each relation instance table contains a plurality of relation instance records. The relationships in Teorey are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Examples of these relationship relations are shown in FIGS. 4 and 10, clearly showing that each relationship relation includes a plurality of rows, each of which is a relation instance record. Teorey at 203, FIG. 4; 212-13, FIG. 10. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity table. For example, the SKILL_USED relationship instance table relates entities in the EMPLOYEE table to entities in the SKILL table. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1.</p> <p>In Kumpati, the relation instance records also contain information that relates the two entities in the relationship defined by the relation instance record. For example, attributes of the Buildings-Rooms relationship include information that identifies what rooms are located in what buildings. Kumpati, 9:43-46.</p>
<p>11. The relational database processing system of claim 10, wherein each of said entity instance records is identified by a record identifier.</p>	<p>Teorey teaches that the entity instance records have key fields, which uniquely identify the entity instances. Teorey at 198 (“The major interattribute dependencies are between the entity keys (unique identifiers) of different entities that are captured in the ER modeling process.”) Kumpati also teaches that the entity instance records have key fields that are used to identify the entities stored in the records of the entity instance tables.. Kumpati, 4:51-56, FIG. 2.</p>
<p>12. The relational database processing system of claim 10, wherein said first relation instance record contains a desired record identifier and a desired entity type corresponding to a desired entity instance record containing said desired entity.</p>	<p>Teorey teaches that each relation instance record contains a record identifier that corresponds to the desired entity instance record. For example, in the relation instance table of FIG. 4, the SKILL-USED relations instances each contain a SKILL-NO record identifier that identifies the desired SKILL entity instance record. Teorey at 203, FIG. 4; 216, FIG. 13. Each relation instance record also contains a desired entity type, reflected in the column header, for example “SKILL” in the column header SKILL-NO in FIG. 4.</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>Even under FST's new argument, advance in their Response, Teorey anticipates this claim. Response, 22-23. Teorey further teaches that a key (i.e. a record identifier) can be a composite identifier, that is, an identifier composed of two or more attributes. Teorey at 204 (Step 1.1(5)). These two attributes could include the entity type and a record identifier, and thus anticipate this claim. Teorey teaches two alternate treatments for composite identifiers, one of which is to eliminate them where possible, but the second treatment is to retain the identifier where it is reasonably natural. Id. It would be reasonably natural to retain a composite identifier where it permitted the overloading of the record identifier column to designate two or more different target tables, as suggested by FST.</p> <p>Therefore, a relationship relation as taught in Teorey, which is stored as a relation instance table in Kumpati, would include two foreign keys to the two entity relations that it referenced, and each of those two foreign keys would be two-part keys, as discussed in FST's Response.</p>
<p>13. The relational database processing system of claim 10, wherein said first relation type record comprises a table identifier identifying said first relation instance table.</p>	<p>Teorey teaches that relationship relations are stored in tables. Teorey at 203, FIG. 4; 216, FIG. 13. Kumpati further teaches that the definitions of these tables (i.e. the relation type records defining the relation types) are stored in the data dictionary. Kumpati, 5:58-63. These definitions include information (i.e. a table identifier) that is used to ascertain the physical location in the database of the requested data. Kumpati, 3:66-4:2.</p>
<p>14. The relational database processing system of claim 10, further comprising an inquiry table containing an inquiry record, wherein said inquiry record specifies said provided relation type and said provided entity.</p>	<p>Teorey teaches that relationships (relation types) are transformed into relationship relations. Teorey at 208, Sec. 3.1(3). Each of these relationships bears a name, for example SKILL_USED is the name of a relationship between the EMPLOYEE entity type and the SKILL entity type. Teorey at 203, FIG. 4; 216, FIG. 13, Table 1. Kumpati teaches that each of the sets of related items, in this case the relationship relations (i.e. relation types) defined in Teorey, are stored in two-dimensional tables in the data files. Kumpati, 4:44-66. For example, the relation instance table Building-Rooms 702 stores the set of all relationships between buildings and rooms. Kumpati,</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	<p>9:43-44, FIG. 7. Similarly, the SKILL_USED relation would also be stored as a table in the Kumpati database.</p> <p>As noted above, in claim 5, the '326 patent to Shaw provides that an SQL query can be stored in an inquiry table, for execution by a query processor on the database. The SQL query is translated from a QBE query input by a user, and the translated SQL query is stored in GFTSQL 70 (inquiry record), which is pointed to by GFTTABLE (inquiry table). Shaw, 10:59-68, 11:1-12.</p> <p>Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement, which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1. Thus for example, SKILL_USED is the name of a relationship between a provided EMPLOYEE entity and a desired SKILL entity.</p> <p>One element of the SQL query stored in the GFTSQL 70 table is the entity provided as a parameter to the query. For example, Table 24 of Shaw depicts an example of a query, in both SQL and QBE formats. Shaw, 26:59-68. In this example, the query is seeking the employees who work in the San Jose Department, and who meet a salary condition of > \$20,000. Id. The provided entity is "DEPT", and the desired entity is "EMP". Id.</p> <p>The QBE example table shown in Table 24 is another example of an inquiry table as claimed. This inquiry table contains the same DEPT provided entity information, which is retrieved from the table in order to create the SQL statement SELECT DNO FROM DEPT shown in Table 24. Shaw, 15:38-42, 26:59-68.</p>
15. The relational database processing system of claim 10	

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
further comprising:	
a second entity instance table associated with a second entity type; and	Teorey teaches two entity instance tables, for example the SKILL and DEPARTMENT entity instance tables of FIG. 13. Teorey at 216, FIG. 13, Table 1. Kumpati also teaches two entity instance tables, for example the Building and Rooms tables of FIG. 7. Kumpati, 9:29-42, FIG. 7.
wherein said entity definition table contains a second entity type record containing said second entity type and associating said second entity instance table with said second entity type.	Teorey teaches that all entities, including both SKILL and DEPARTMENT entities, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Kumpati teaches that the data dictionary contains a set of entity definitions. Kumpati, 8:35-39. A set, by its very nature, can contain two (or more) entity type records, for example the entity types for Building and Rooms. Kumpati, 8:35-39.
16. The relational database processing system of claim 15 further comprising:	
a third entity instance table associated with a third entity type; and	<p>Teorey teaches three entity instance tables, for example the SKILL, DEPARTMENT and DIVISION entity tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.</p>  <p>Figure 13. Company personnel and project database candidate relations.</p> <p>Kumpati also teaches three entity instance tables, for</p>

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	example the Building, Rooms and Exits tables of FIG. 7. Kumpati, 9:29-42, FIG. 7.
wherein said entity definition table contains a third entity type record containing said third entity type and associating said third entity instance table with said third entity type.	Teorey teaches that all entities, including SKILL, DEPARTMENT and DIVISION entities, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. H Kumpati teaches that the data dictionary contains a set of entity definitions. Kumpati, 8:35-39. A set, by its very nature, can contain two (or more) entity type records, for example the entity types for Building, Rooms and Exits. Kumpati, 8:35-39.
17. The relational database processing system of claim 10 further comprising:	
a second relation instance table associated with a second relation type; and	Teorey teaches two relation instance tables, for example the SKILL-USED and ASSIGNED-TO relation instance tables of FIG. 13. Teorey at 216, FIG. 13, Table 1. Kumpati also teaches two relation instance tables, for example the Building-Rooms and Exit-Rooms tables of FIG. 7. Kumpati, 9:29-42, FIG. 7.
wherein said relation definition table contains a second relation type record containing said second relation type and associating said second relation instance table with said second relation type.	Teorey teaches that all relationships, including both SKILL-USED and ASSIGNED-TO relationships, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Kumpati teaches that the data dictionary contains a set of relation definitions. Kumpati, 8:35-39. A set, by its very nature, can contain two (or more) relation type records, for example the relation types for Building-Rooms and Exits-Rooms. Kumpati, 8:35-39. Thus the relation definition table can contain two (or more) relation definition records, each of which are associated to a different relation instance table (e.g. Building-Rooms and Exits-Rooms relation instance tables).
18. The relational database processing system of claim 17 further comprising:	
a third relation instance table associated with a third relation type; and	Teorey teaches three relation instance tables, for example the SKILL-USED, ASSIGNED-TO and BELONGS-TO relation instance tables of FIG. 13.

Claim Language ¹⁴	Teorey, Kumpati, and Zloof/Shaw References
	Teorey at 216, FIG. 13, Table 1. Kumpati also teaches three relation instance tables, for example the Building-Rooms, Exit-Rooms and Buildings-Exits tables of FIG. 7. Kumpati, 9:29-42, FIG. 7.
wherein said relation definition table contains a third relation type record containing said third relation type and associating said third relation instance table with said third relation type.	<p>Teorey teaches that all relationships, including SKILL-USED, ASSIGNED-TO and BELONGS-TO relationships, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Kumpati teaches that the data dictionary contains a set of relation definitions. Kumpati, 8:35-39. A set, by its very nature, can contain three (or more) relation type records, for example the relation types for Building-Rooms, Exits-Rooms and Buildings-Exits. Kumpati, 8:35-39.</p> <p>Thus the relation definition table can contain three (or more) relation definition records, each of which are associated to a different relation instance table (e.g. Building-Rooms, Exits-Rooms and Buildings-Exits relation instance tables).</p>

C. Dolk, Teorey, Zloof and/or Shaw

Dolk and Teorey together teach all limitations of claims 1-4, 7-13, 15-18 of the '259 patent as discussed in the chart below. Dolk, Teorey, and Zloof/Shaw together teach all limitations of claims 5-6 and 14 of the '259 patent, as discussed in the chart below:

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
1. A method for retrieving a desired entity of a desired entity type from a relational database, wherein said desired entity is related to a provided entity by a provided relation type associating an entity type of said provided entity with said desired entity type, said method comprising:	The primary aims of relational database design revolve around organizing and storing data so that the information within the database may later be accessed by the database user. According to an article by Peter Chen, "to design a database is to decide how to organize data into specific forms (record types, tables) and how to access them." Further, another related problem in database design is to make the "output of the database design process-the user schema (a description of the user view of the data)" more like the way humans represent the real world. Peter Pin-Shan Chen, <u>The entity-relationship model – A basis for</u>

¹⁵ The claim language recited is inclusive of the amendments patentee included in its Response.

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p data-bbox="727 226 1209 262"><u>the enterprise view of data</u> 77 (1977).</p> <p data-bbox="727 317 1495 867">In addition an inherent component of databases is that they allow the retrieval of items, including relation type records. Front end user interfaces that enable users to easily retrieve the information in the underlying databases have been well known in the arts from at least from the mid 1980's, if not sooner. One type of front-end user interface is created using the QBE language. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p data-bbox="727 947 1490 1457">A critical component of databases is that they allow the retrieval of items, including relation type records. Front end user interfaces that enable users to access the information in the underlying databases are inherent to database management systems. One type of front-end user interface is created using the QBE language. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p data-bbox="727 1535 1487 1717">"A DBMS-dependent IRDS uses an existing DBMS to implement the description, manipulation, and control of its metadata, and therefore can avail itself of the underlying query processor, security, backup/recovery, and other features." Dolk at 49.</p>
<p data-bbox="235 1759 667 1892">retrieving a specific relation type record defining said provided relation type from a relation definition table;</p>	<p data-bbox="727 1759 1495 1892">Dolk teaches a data definition table in the form of a information resource definition system (IRDS) that includes a relation definition table that contains relation type records which define relationship types. A user would</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>supply such a relation type in making a search (query) on the database.</p> <p>Dolk teaches that relation types (i.e. relationships) can be defined for a relational database. Dolk at 51, Fig. 4. Fig. 4 depicts a basic relational representation of the IRDS Entity-Relationship Model. Specifically, one standard template for defining a relationship would be as follows:</p> <pre style="text-align: center;"> RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, access-method, frequency, rel_pos) </pre> <p>Fig. 4. According to Dolk, relationships have certain core attributes, as depicted below in Fig. 5, below. They associate two entities with names e1name and e2name.</p> <p style="text-align: center;">Relationships</p> <p style="text-align: center;">All relationships have the same attributes and keys:</p> <pre style="text-align: center;"> REL(<u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>) </pre> <p style="text-align: center;">where e1name, e2name are the entity instances</p> <p style="text-align: center;">e1type, e2type are the entity-types of which e1name, e2name are instances, respectively</p> <p style="text-align: center;">REL is any of the relationships CONTAINS, PROCESSES, RUNS, RESP_FOR, CALLS, GOES_TO, DERIVED_FROM, ALIAS, and KWIC</p> <p style="text-align: center;">Integrity constraints</p> <p style="text-align: center;">See Figure 3</p> <p style="text-align: center;">FIGURE 5. Relational IRDS (RIRDS)</p> <p>(in pertinent part).</p> <p>All relationships are binary, relate entities to each other, and are named self-descriptively according to the entity-types that participate in them. Dolk at 50, 51. Under the model proffered by Dolk, different relationship types connect various entity types. For example, Fig. 2 listing various entity types and relationship types; the relationship types may relate two entities. Fig. 1 provides an example of a relationship Empl – Record for Kirk – CONTAINS – (555-23-6666) that falls within the Relationship type RECORD-CONTAINS-ELEMENT.</p> <p>The definitions of these relationship types are stored in the</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>IRD data layer of a relational information resource dictionary system (RIRDS). Dolk, Fig. 1, p. 50. These definition of these relation types must have the following basic relational representation:</p> <pre style="text-align: center;">RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, access-method, frequency, rel_pos)</pre> <p>Dolk, fig. 4 at p. 53.</p> <p>The IRDS is itself of an implementation of a ORACLE database and by using the SQL CREATE TABLE and CREATE VIEW commands. Dolk at pp. 49, 55 (“The RIRDS is implemented very straightforwardly in ORACLE by creating the relations and views given in Figures 4, 5, and 6 using the SQL CREATE TABLE and CREATE VIEW commands.”) Therefore, the data dictionary component of the IRDS itself includes tables that contain relationship table definitions. Fig. 7 depicts how RIRDS Tables and Views may be created in ORACLE.</p> <pre style="text-align: center;">RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, access-method, frequency, rel_pos)</pre> <p>Further, in extending the RIRDS, relationship tuples, or definition rows, may be added to the data dictionary by the process mentioned p. 56-57 of Dolk.</p>
<p>retrieving a specific relation instance record defining a relation of said provided relation type between said provided entity and said desired entity from a relation instance table corresponding to said specific relation type record;</p>	<p>Dolk teaches that relation instance records define relations between entities, and that these records are stored in relation instance tables, each of which corresponds to a particular relation type.</p> <p>Dolk teaches that relationship relations store information that relates two entities to each other and are represented as tables. “All relationships are binary, and entities may be related to themselves.” Dolk at 50. The IRDS network contains entities, or nodes, which are connected to each other by relationships, or arcs. Id. As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the relationship instances may be stored in a relation instance table, corresponding to a specific relationship type record. <u>See i.e.</u> p. 60. Emp-Record for Kirk- CONTAINS (555-23-6666) is one relationship instance record the relationship Emp-Record-CONTAINS-Soc-Sec-No between Soc-Sec-No and Empl-Rec entities. Fig. 1 (depicting IRDS architecture). Relationship instance records in the underlying Oracle</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>database may be retrieved using SQL queries (Dolk at 58, 60), regardless of whether the RIRDS system is integrated with the host RDBMS. [see also textbooks on QBE and Chen articles]</p> <p>Dolk further teaches that the RIRDS may contain a directory component “describing where information resources [i.e. relationship tables] are located and how they can be accessed.” Dolk at 49. Put another way, the directory is analogous to a physical description of where the relationship instance table is located in computer memory. Dolk at 49. The “directory description would contain data on the machine, operating system, and the file structure under which the file is stored.” Dolk at 49.</p>
<p>retrieving a desired entity type record containing said desired entity type from an entity definition table, wherein said desired entity type record specifies a desired entity instance table associated with said desired entity type; and</p>	<p>Dolk teaches that entity type information is stored in entity records in entity definition tables, and that this data can be retrieved in order to operate on the underlying database. The entity type records identify entity instance tables, including the entity instance tables containing entities that a user would desire to retrieve.</p> <p>Dolk teaches that entity types can be defined for a relational database. Dolk at 50, Fig. 4. Fig. 4 depicts a basic relational representation of the IRDS Entity-Relationship Model. Specifically, one standard template for defining an entity would be as follows:</p> <pre style="text-align: center;"> ENTITY(<u>ename</u>, <u>etype</u>, dname, added-by, date-added, mod-by, last-mod, nmods, dur-value, dur-type, comments, descr security, lang, lines-code, nrecs, rec-cat, data-class, doc-cat) </pre> <p>Fig. 4. <u>ename</u> would represent the name of the entity; <u>etype</u> would represent the entity type. Fig. 2 lists various entity types such as SYSTEM, PROGRAM, MODULE, USER, FILE, RECORD, ELEMENT, etc.</p> <p>The definitions of these relationship types are stored in the IRD data layer of a relational information resource dictionary system (RIRDS). Dolk, Fig. 1, p. 50.</p> <p>The IRDS is itself of an implementation of a ORACLE database and by using the SQL CREATE TABLE and CREATE VIEW commands. Dolk at 49, 55. Therefore, the data dictionary component of the IRDS itself includes</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>tables that contain entity table definitions. Fig. 7 depicts how RIRDS Tables and Views may be created in ORACLE.</p> <p>Further, in extending the RIRDS, entity tuples, or definition rows, may be added to the data dictionary by the process mentioned on pp. 56-57 of Dolk.</p> <p>Dolk further teaches that the RIRDS may contain a directory component “describing where information resources [i.e. entity instance tables] are located and how they can be accessed.” Dolk at 49. Put another way, the directory is analogous to a physical description of where the entity instance table is located in computer memory. Dolk at 49. The “directory description would contain data on the machine, operating system, and the file structure under which the file is stored.” Dolk at 49.</p>
<p>retrieving said desired entity from said desired entity instance table.</p>	<p>Dolk teaches that desired entities, just like all entities, are stored in entity instance tables, and can be retrieved therefrom.</p> <p>The IRDS network contains entities of various entity types. P. 50. As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the entity instances may be stored in entity instance tables, corresponding to a specific entity type record. <i>See i.e.</i> p. 60. Fig. 1 provides an example of an entity record 555-23-6666 (Employee record for Kirk), in the form of a social security number. This entity record may be stored in and retrieved from a Soc-Sec-No entity instance table. Entity instance records in the underlying Oracle database may be retrieved using SQL queries at pp. 58, 60 of Dolk, regardless of whether the RIRDS system is integrated with the host RDBMS.</p>
<p>2. The method of claim 1, wherein said relation instance record specifies said desired entity by said desired entity type and a desired record identifier.</p>	<p>Dolk teaches that each relation instance record contains a record identifier that corresponds to the desired relation instance record.</p> <p>Specifically, Dolk teaches that a record identifier for an entity may be made of composite keys. Fig. 4 depicts a generic representation of a relationship, where <u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, and <u>e2type</u> are keys to a given relationship of relationship type <u>rtype</u>, which associates entities <u>e1name</u> and <u>e2name</u> of entity types <u>e1type</u> and <u>e2type</u>.</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, <u>access-method</u>, <u>frequency</u>, <u>rel_pos</u>) Fig. 4 of Dolk.</p>
<p>3. The method of claim 2, wherein said desired entity is identified by said desired record identifier in said desired entity instance table.</p>	<p>Dolk teaches that each entity instance record contains a record identifier that corresponds to the desired entity instance record.</p> <p>Specifically, Dolk teaches that a record identifier for an entity may be made of composite keys. Fig. 4 depicts a generic representation of an entity, where <u>ename</u> and <u>etype</u> are keys to a given entity by entity name <u>ename</u> and entity type <u>etype</u>.</p> <p>ENTITY(<u>ename</u>, <u>etype</u>, <u>dname</u>, <u>added-by</u>, <u>date-added</u>, <u>mod-by</u>, <u>last-mod</u>, <u>nmods</u>, <u>dur-value</u>, <u>dur-type</u>, <u>comments</u>, <u>descr</u>, <u>security</u>, <u>lang</u>, <u>lines-code</u>, <u>nrecs</u>, <u>rec-cat</u>, <u>data-class</u>, <u>doc-cat</u>) Fig. 4 of Dolk.</p>
<p>4. The method of claim 1, wherein said retrieving a specific relation instance record comprises:</p>	
<p>retrieving a table identifier for said relation instance table from said specific relation type record; and</p>	<p>Dolk teaches that each relationship relation (i.e. relation type record) includes the name of the relationship. Fig. 1. Relationships are named self-descriptively according to the entity-types that participate in them. Therefore the relationship that associates Empl-Record and Soc-Sec-No would be named Empl-Record-CONTAINS-Soc-Sec-No. Dolk at 50, 51.</p>
<p>retrieving said specific relation instance record from said relation instance table based on said specific relation type record and said provided entity.</p>	<p>Under Dolk, the table identified by a specific table identifier is the table which contains the specific relation instance records that are based on the specific relation type record and entity provided.</p> <p>Each relation instance table contains a plurality of relation instance records. The relationships in Dolk are transformed into relationship relations. As the organizational data underneath is stored in an Oracle database, p. 50 of Dolk, the relationship relations are stored in tables, containing a plurality of rows, each of which is a relation instance record. Dolk at 59-60. Each relation instance record relates an entity instance in one entity table to an entity instance in a second entity instance table.</p>
<p>5. The method of claim 1, further</p>	<p>The use of inquiry tables such as ones based on the QBE</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
<p>comprising retrieving data specifying said provided relation type from an inquiry table.</p>	<p>language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to <u>The Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990). While a QBE interface is pictorial in nature, a SQL's interface is more linear and textual. Id. "The user begins by specifying which table is needed for a particular query. Once a table(s) is chosen, the system displays an outline of that table, showing the table name and the names of its fields. To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Id.</p> <p>Zloof teaches Query-by-Example (QBE). When a user performs an operation , e.g., query, update, define, or control, against the data base, the user fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the database. For illustration, suppose we want all green items sold in the toy department.</p> <p>Fig. 13 of Zloof above shows the two skeleton tables: "TYPE" and "SALES". These two tables are linked by the word "NUT". Thus Figure 13 shows an inquiry table consisting of two skeleton tables linked together like the level 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.</p> <p>The '326 patent to Shaw teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal, including relation records. Shaw takes the QBE skeleton tables , for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2. Shaw, 11:1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in</p>

Claim Language¹⁵

Dolk, Teorey, and Zloof/Shaw References

Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL, such as the SQL select statement in Table 24.

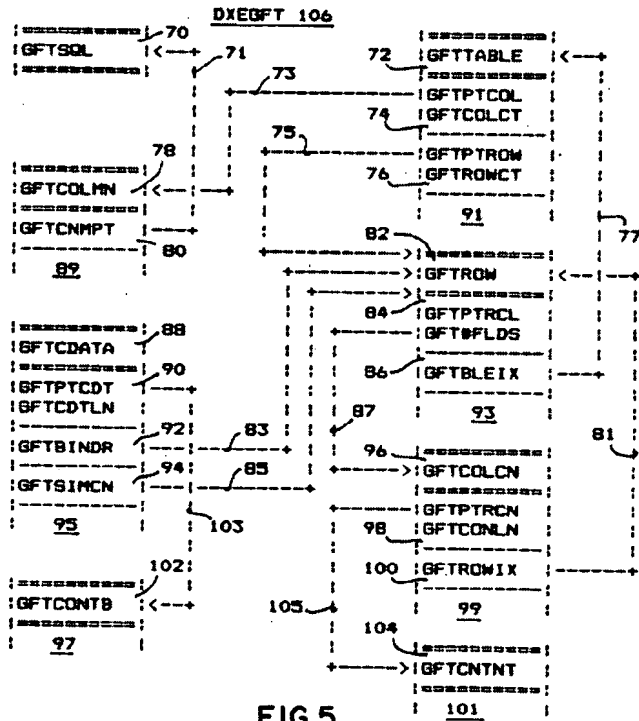


FIG. 5

Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1.

The query translator of Shaw receives as an input a QBE query, which is a "graphic language query expressed as one or more elements . . . appearing in rows and columns of an example table including one or more source and target tables." Shaw, 2:38-42. Where the source table included in the example table is a relationship relation, as discussed in Teorey, this example table contains the name of the source table, which is "data specifying said provided relation type" as claimed. When the example query is

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	translated as part of the query processing, this relation type data is retrieved in order to form part of the SQL query submitted to the DBMS. Shaw, 15:29-31.
6. The method of claim 1, further comprising retrieving data specifying said provided entity from an inquiry table.	<p>The use of inquiry tables such as ones based on the QBE language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to <u>The Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990). While a QBE interface is pictorial in nature, a SQL's interface is more linear and textual. Id. "The user begins by specifying which table is needed for a particular query. Once a table(s) is chosen, the system displays an outline of that table, showing the table name and the names of its fields. To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Id.</p> <p>Zloof teaches Query-by-Example (QBE). When a user performs an operation , e.g., query, update, define, or control, against the data base, the user fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the database. For illustration, suppose we want all green items sold in the toy department.</p> <p>Fig. 13 of Zloof above shows the two skeleton tables: "TYPE" and "SALES". These two tables are linked by the word "NUT". Thus Figure 13 shows an inquiry table consisting of two skeleton tables linked together like the level 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.</p> <p>The '326 patent to Shaw teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal, including relation records. Shaw takes the QBE skeleton tables , for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p>

Claim Language¹⁵

Dolk, Teorey, and Zloof/Shaw References

Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL, such as the SQL select statement in Table 24.

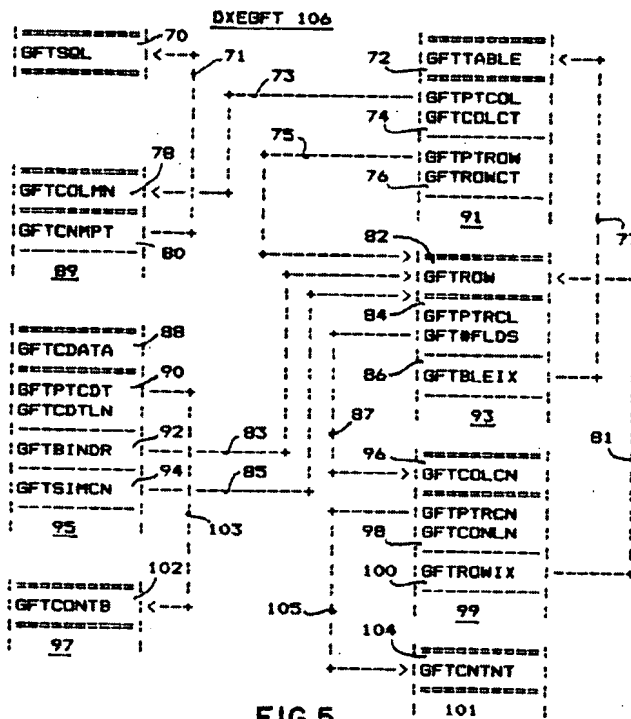


FIG.5

Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1. Thus for example, SKILL_USED is the name of a relationship between a provided EMPLOYEE entity and a desired SKILL entity.

One element of the SQL query stored in the GFTSQL 70 is the entity provided as a parameter to the query. For

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>example, Table 24 of Shaw depicts an example of a query, in both SQL and QBE formats. Shaw, 26:59-68. In this example, the query is seeking the employees who work in the San Jose Department, and who meet a salary condition of > \$20,000. Id. The provided entity is "DEPT", and the desired entity is "EMP". Id.</p> <p>The QBE example table shown in Table 24 is another example of an inquiry table as claimed. This inquiry table contains the same DEPT provided entity information, which is retrieved from the table in order to create the SQL statement SELECT DNO FROM DEPT shown in Table 24. Shaw, 15:38-42, 26:59-68.</p>
<p>7. The method of claim 1, further comprising retrieving a second desired entity type record containing a second desired entity type from said entity definition table, wherein said second desired entity type record specifies a second desired entity instance table associated with said second desired entity type.</p>	<p>This limitation requires the presence of two entity type records, and thus two entity instance tables. Otherwise, it is the same as "retrieving a desired entity type record" of Claim 1. Fig. 1 of Dolk depicts the association of two different entity types, Soc-Sec-No and Empl-Record to generate a relationship Empl-Record-CONTAINS-Soc-Sec-No. Two entity instance records appear in Fig. 1: Soc-Sec-No 555-23-6666 and Empl-Record Kirk.</p>
<p>8. The method of claim 7, further comprising retrieving a third desired entity type record containing a third desired entity type from said entity definition table, wherein said third desired entity type record specifies a third desired entity instance table associated with said third desired entity type.</p>	<p>This limitation requires the presence of three entity type records, and thus three entity instance tables. Otherwise, it is the same as "retrieving a desired entity type record" of Claim 1.</p> <p>Teorey provides another example ternary relationships relating three entities. Specifically, Teorey clearly teaches relation types which create relation instance tables that involve more than two entity instances. Such relationships are disclosed as "ternary" or more generally "n-ary" relationships. For example, see the ternary relationships of FIG. 10. Teorey at 212-13, FIG. 10. When executing queries on these relationships, the entity type records of, for example, the EMPLOYEE, PROJECT and SKILL entities would be retrieved, for the ternary relationship SKILL-USED of FIG. 10(b). Teorey at 212; see also id at 216, Fig. 13.</p>
<p>9. The method of claim 1, further comprising retrieving a second specific relation instance record defining a relation of a second</p>	<p>This claim merely requires the retrieval of a second specific relation instance record of a second relation type, between the same two entities. Otherwise, it is the same as "retrieving a desired entity type record" of Claim 1.</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References																																
<p>provided relation type between said provided entity and said desired entity from a second relation instance table corresponding to said second provided relation type record.</p>	<p>Dolk expressly permits the existence of two different relationships, of two different relation types, between the same two entities. Fig. 2 indicates that not only can an IRDS contain entities of different Entity-types, i.e. FILE, RECORD, or ELEMENT, but an IRDS may also contain different Relationship-types, i.e. CONTAINS, PROCESSES, RESPONSIBLE-FOR, or RUNS. Specifically, Fig. 2 provides in pertinent part, as follows:</p> <div style="text-align: center;"> <p>Entity-types</p> <table border="0"> <tr> <td>SYSTEM</td> <td>FILE</td> <td>BIT-STRING</td> </tr> <tr> <td>PROGRAM</td> <td>RECORD</td> <td>CHARACTER-STRING</td> </tr> <tr> <td>MODULE</td> <td>ELEMENT</td> <td>FIXED-POINT</td> </tr> <tr> <td>USER</td> <td>DOCUMENT</td> <td>FLOAT</td> </tr> </table> <p>Relationship-types</p> <table border="0"> <tr> <td>CONTAINS</td> <td>GOES-TO</td> </tr> <tr> <td>PROCESSES</td> <td>CALLS</td> </tr> <tr> <td>RESPONSIBLE-FOR</td> <td>DERIVED-FROM</td> </tr> <tr> <td>RUNS</td> <td>REPRESENTED-AS</td> </tr> </table> </div> <p>Fig. 2, entitled "Core System-Standard Schema Types"</p> <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">IRD schema description layer</th> <th style="text-align: center;">Entity-type</th> <th style="text-align: center;">Relationship-type</th> </tr> </thead> <tbody> <tr> <td>IRD schema layer</td> <td style="text-align: center;">ELEMENT, RECORD, etc.</td> <td style="text-align: center;">RECORD-CONTAINS-ELEMENT</td> </tr> <tr> <td>IRD data layer</td> <td style="text-align: center;">Soc-Sec-No, Empl-Record, etc.</td> <td style="text-align: center;">Empl-Record-CONTAINS-Soc-Sec-No</td> </tr> <tr> <td>Operational data</td> <td style="text-align: center;">555-23-6666 (Employee record for Kirk)</td> <td style="text-align: center;">Empl-Record for Kirk-CONTAINS-(555-23-6666)</td> </tr> </tbody> </table> <p style="text-align: center;">FIGURE 1. IRDS Architecture</p>	SYSTEM	FILE	BIT-STRING	PROGRAM	RECORD	CHARACTER-STRING	MODULE	ELEMENT	FIXED-POINT	USER	DOCUMENT	FLOAT	CONTAINS	GOES-TO	PROCESSES	CALLS	RESPONSIBLE-FOR	DERIVED-FROM	RUNS	REPRESENTED-AS	IRD schema description layer	Entity-type	Relationship-type	IRD schema layer	ELEMENT, RECORD, etc.	RECORD-CONTAINS-ELEMENT	IRD data layer	Soc-Sec-No, Empl-Record, etc.	Empl-Record-CONTAINS-Soc-Sec-No	Operational data	555-23-6666 (Employee record for Kirk)	Empl-Record for Kirk-CONTAINS-(555-23-6666)
SYSTEM	FILE	BIT-STRING																															
PROGRAM	RECORD	CHARACTER-STRING																															
MODULE	ELEMENT	FIXED-POINT																															
USER	DOCUMENT	FLOAT																															
CONTAINS	GOES-TO																																
PROCESSES	CALLS																																
RESPONSIBLE-FOR	DERIVED-FROM																																
RUNS	REPRESENTED-AS																																
IRD schema description layer	Entity-type	Relationship-type																															
IRD schema layer	ELEMENT, RECORD, etc.	RECORD-CONTAINS-ELEMENT																															
IRD data layer	Soc-Sec-No, Empl-Record, etc.	Empl-Record-CONTAINS-Soc-Sec-No																															
Operational data	555-23-6666 (Employee record for Kirk)	Empl-Record for Kirk-CONTAINS-(555-23-6666)																															
<p>10. A relational database processing system comprising:</p>	<p>The primary aims of relational database design revolve around organizing and storing data so that the information within the database may later be accessed by the database user. According to an article by Peter Chen, "to design a database is to decide how to organize data into specific forms (record types, tables) and how to access them." Further, another related problem in database design is to make the "output of the database design process-the user schema (a description of the user view of the data)" more like the way humans represent the real world. Peter Pin-Shan Chen, <u>The entity-relationship model – A basis for the enterprise view of data</u> 77 (1977).</p>																																

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>In addition an inherent component of databases is that they allow the retrieval of items, including relation type records. Front end user interfaces that enable users to easily retrieve the information in the underlying databases have been well known in the arts from at least from the mid 1980's, if not sooner. One type of front-end user interface is created using the QBE language. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p>A critical component of databases is that they allow the retrieval of items, including relation type records. Front end user interfaces that enable users to access the information in the underlying databases are inherent to database management systems. One type of front-end user interface is created using the QBE language. According to <u>The Database Step-by-Step textbook</u>, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p>The Dolk article discloses an IRDS which contains an enhanced data dictionary D/D. Specifically, Dolk discloses a relational model of a passive IRDS (i.e. a "stand-alone" IRDS, or "DBMS independent") that is consistent with the subset of the FIPS specifications and can easily be implemented and used with existing RDBMS products. Dolk at 49.</p>
<p>an entity definition table containing a first entity type record defining a first entity type;</p>	<p>Dolk teaches that entity types (i.e. entities) can be defined for a relational database. P.50. As noted above, Fig. 4 provides a template for how an entity may be defined with various attributes:</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p data-bbox="735 233 1425 394">ENTITY(<u>ename</u>, <u>etype</u>, <u>dname</u>, <u>added-by</u>, <u>date-added</u>, <u>mod-by</u>, <u>last-mod</u>, <u>nmods</u>, <u>dur-value</u>, <u>dur-type</u>, <u>comments</u>, <u>descr</u>, <u>security</u>, <u>lang</u>, <u>lines-code</u>, <u>nrecs</u>, <u>rec-cat</u>, <u>data-class</u>, <u>doc-cat</u>)</p> <p data-bbox="735 436 1495 688">Fig. 2 depicts two example of entity types, Soc-Sec-No and Empl-Record. p. 50. As noted above, Dolk further teaches that an IRDS contains an enhanced data dictionary D/D. p.49. The entity definitions are contained in the IRD data and schema layers, which are separated from the operational data layer (which contain the entity and relationship instance records). Dolk at 50.</p> <p data-bbox="735 730 1495 1014">Finally, the data dictionary containing the entity definitions are stored in a table. As the data and scheme layers are based on the E-R model, Dolk teaches the implementation of these layers using ORACLE tables and SQL tables. Dolk at 55 (“The RIRDS is implemented very straightforwardly in ORACLE by creating the relations and views given in Figures 4, 5, and 6 using the SQL CREATE TABLE and CREATE VIEW commands.”)</p>
<p data-bbox="237 1026 634 1129">a first entity instance table associated with said first entity type;</p>	<p data-bbox="735 1026 1451 1167">Dolk discloses entity instance tables as part of the IRDS architecture. The IRD schema and data layers map the contents of the underlying database, including entity instance tables.</p> <p data-bbox="735 1209 1479 1388">Entity instance definitions can be retrieved in order to operate on the underlying database. The entity type definition records identify entity instance tables, including the entity instance tables containing entities that a user would desire to retrieve.</p> <p data-bbox="735 1430 1463 1608">Dolk teaches that entity types can be defined for a relational database. Dolk, Fig. 4, p. 50. Fig. 4 depicts a basic relational representation of the IRDS Entity-Relationship Model. Specifically, one standard template for defining an entity would be as follows:</p> <p data-bbox="837 1650 1507 1812">ENTITY(<u>ename</u>, <u>etype</u>, <u>dname</u>, <u>added-by</u>, <u>date-added</u>, <u>mod-by</u>, <u>last-mod</u>, <u>nmods</u>, <u>dur-value</u>, <u>dur-type</u>, <u>comments</u>, <u>descr</u>, <u>security</u>, <u>lang</u>, <u>lines-code</u>, <u>nrecs</u>, <u>rec-cat</u>, <u>data-class</u>, <u>doc-cat</u>)</p> <p data-bbox="735 1850 1425 1877">Fig. 4. <u>ename</u> would represent the name of the entity;</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p><u>etype</u> would represent the entity type. Fig. 2 lists various entity types such as SYSTEM, PROGRAM, MODULE, USER, FILE, RECORD, ELEMENT, etc.</p> <p>The definitions of these relationship types are stored in the IRD data layer of a relational information resource dictionary system (RIRDS). Dolk, Fig. 1, p. 50.</p> <p>The IRDS is itself of an implementation of a ORACLE database and by using the SQL CREATE TABLE and CREATE VIEW commands. P.49, 55. Therefore, the data dictionary component of the IRDS itself includes tables that contain entity table definitions. Fig. 7 depicts how RIRDS Tables and Views may be created in ORACLE. Further, in extending the RIRDS, entity tuples, or definition rows, may be added to the data dictionary by the process mentioned on p. 56-57 of Dolk.</p> <p>Dolk further teaches that the RIRDS may contain a directory component “describing where information resources [i.e. entity instance tables] are located and how they can be accessed.” P. 49. Put another way, the directory is analogous to a physical description of where the entity instance table is located in computer memory. P. 49. The “directory description would contain data on the machine, operating system, and the file structure under which the file is stored.” Dolk at 49.</p>
<p>a plurality of entity instance records stored in said first entity instance table;</p>	<p>Dolk teaches that desired entities, or entity instance records, are stored in entity instance tables.</p> <p>The IRDS network contains entities of various entity types. Dolk at 50. As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the entity instances may be stored in entity instance tables, corresponding to a specific entity type record. <i>See i.e.</i> p. 60. Fig. 1 provides an example of a entity record 555-23-6666 (Employee record for Kirk), in the form of a social security number. This entity record may be stored in and retrieved from a Soc-Sec-No entity instance table. Entity instance records in the underlying Oracle database may be retrieved using SQL queries at p. 58, 60 of Dolk, regardless of whether the RIRDS system is integrated with the host RDBMS.</p>
<p>a relation definition table</p>	<p>Dolk teaches a data definition table in the form of a</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
<p>containing a first relation type record defining a provided relation type;</p>	<p>information resource definition system (IRDS) that includes a relation definition table which contains relation type records which define relationship types.</p> <p>Dolk teaches that relation types (i.e. relationships) can be defined for a relational database. Dolk at 51, Fig. 4. Fig. 4 depicts a basic relational representation of the IRDS Entity-Relationship Model. Specifically, one standard template for defining a relationship would be as follows:</p> <p style="text-align: center;">RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, access-method, frequency, rel_pos)</p> <p>Fig. 4. According to Dolk, relationships have certain core attributes, as depicted below in Fig. 5, below. They associate two entities with names e1name and e2name.</p> <p style="text-align: center;">Relationships</p> <p style="text-align: center;">All relationships have the same attributes and keys:</p> <p style="text-align: center;">REL(<u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>)</p> <p style="text-align: center;">where e1name, e2name are the entity instances</p> <p style="text-align: center;">e1type, e2type are the entity-types of which e1name, e2name are instances, respectively</p> <p style="text-align: center;">REL is any of the relationships CONTAINS, PROCESSES, RUNS, RESP_FOR, CALLS, GOES_TO, DERIVED_FROM, ALIAS, and KWIC</p> <p style="text-align: center;">Integrity constraints</p> <p style="text-align: center;">See Figure 3</p> <p style="text-align: center;">FIGURE 5. Relational IRDS (RIRDS)</p> <p>(in pertinent part).</p> <p>All relationships are binary, relate entities to each other, and are named self-descriptively according to the entity-types that participate in them. Dolk at 50, 51.</p> <p>The definitions of these relationship types are stored in the IRD data layer of a relational information resource dictionary system (RIRDS). Fig. 1, p. 50. These definition of these relation types must have the following basic relational representation:</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p data-bbox="743 233 1446 296">RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, <u>access-method</u>, <u>frequency</u>, <u>rel_pos</u>)</p> <p data-bbox="732 327 1024 359">Fig. 4 at p. 53 of Dolk.</p> <p data-bbox="732 401 1479 768">The IRDS is itself of an implementation of a ORACLE database and by using the SQL CREATE TABLE and CREATE VIEW commands. P.49, 55 (“The RIRDS is implemented very straightforwardly in ORACLE by creating the relations and views given in Figures 4, 5, and 6 using the SQL CREATE TABLE and CREATE VIEW commands.”) Therefore, the data dictionary component of the IRDS itself includes tables that contain relationship table definitions. Fig. 7 depicts how RIRDS Tables and Views may be created in ORACLE.</p> <p data-bbox="792 768 1300 821">RELSHIP(<u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>, <u>access-method</u>, <u>frequency</u>, <u>rel_pos</u>)</p> <p data-bbox="732 852 1487 957">Further, in extending the RIRDS, relationship tuples, or definition rows, may be added to the data dictionary by the process mentioned on pp. 56-57 of Dolk.</p>
<p data-bbox="237 968 618 1073">a first relation instance table associated with said provided relation type; and</p>	<p data-bbox="732 968 1487 1115">Dolk teaches that relation instance records define relations between entities, and that these records are stored in relation instance tables, each of which corresponds to a particular relation type.</p> <p data-bbox="732 1146 1495 1766">Dolk teaches that relationship relations store information that relates two entities to each other and are represented as tables. The IRDS network contains entities, or nodes, which are connected to each other by relationships, or arcs. Dolk at 50. As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the relationship instances may be stored in a relation instance table, corresponding to a specific relationship type record. <i>See i.e.</i> p. 60 of Dolk. Emp-Record for Kirk-CONTAINS (555-23-6666) is one relationship instance record. The relationship Emp-Record-CONTAINS-Soc-Sec-No between Soc-Sec-No and Empl-Rec entities. Fig. 1 (depicting IRDS architecture). Relationship instance records in the underlying Oracle database may be retrieved using SQL queries at pp. 58, 60 of Dolk, regardless of whether the RIRDS system is integrated with the host RDBMS.</p>
<p data-bbox="237 1776 708 1877">a first relation instance record of said provided relation type, said first relation instance record relating</p>	<p data-bbox="732 1776 1438 1877">Dolk teaches that a relationship instance record, which corresponds to a particular relationship type, defines relations between entities.</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
<p>a desired entity in one of said entity instance records to a provided entity.</p>	<p>Dolk teaches that relationship relations store information that relates two entities to each other. "All relationships are binary, and entities may be related to themselves." P. 50. The IRDS network contains entities, or nodes, which are connected to each other by relationships, or arcs. P. 50. As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the relationship instances may be stored in a relation instance table, corresponding to said relationship type record. <i>See i.e.</i> p. 60 of Dolk. Emp-Record for Kirk- CONTAINS (555-23-6666) is one relationship instance record the relationship Emp-Record-CONTAINS-Soc-Sec-No between Soc-Sec-No and Empl-Rec entities. Fig. 1 (depicting IRDS architecture). Relationship instance records in the underlying Oracle database may be retrieved using SQL queries at p. 58, 60 of Dolk, regardless of whether the RIRDS system is integrated with the host RDBMS.</p>
<p>11. The relational database processing system of claim 10, wherein each of said entity instance records is identified by a record identifier.</p>	<p>Dolk teaches that each entity instance record contains a record identifier that corresponds to the desired entity instance record.</p> <p>Specifically, Dolk teaches that a record identifier for an entity may be made of composite keys. Fig. 4 depicts a generic representation of an entity, where <u>ename</u> and <u>etype</u> are keys to a given entity by entity name <u>ename</u> and entity type <u>etype</u>.</p> <pre> ENTITY(<u>ename</u>,<u>etype</u>,dname,added-by, date-added,mod-by,last-mod,nmods, dur-value,dur-type,comments,descr, security,lang,lines-code,nrecs, rec-cat,data-class,doc-cat) </pre> <p>Fig. 4 of Dolk.</p>
<p>12. The relational database processing system of claim 10, wherein said first relation instance record contains a desired record identifier and a desired entity type corresponding to a desired entity instance record containing said desired entity.</p>	<p>Dolk teaches that each relation instance record contains a record identifier that corresponds to the desired relation instance record. The relation instance record also contains desired entity type corresponding to a desired entity instance record containing said desired entity.</p> <p>Specifically, Dolk teaches that a record identifier for an entity may be made of composite keys. Fig. 4 depicts a generic representation of a relationship, where <u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, and <u>e2type</u> are keys to a given</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>relationship of relationship type <u>rtype</u>, which associates entities <u>e1name</u> and <u>e2name</u> of entity types <u>e1type</u> and <u>e2type</u>.</p> <p><u>RELSHIP(rtype, e1name, e1type, e2name, e2type, access-method, frequency, rel_pos)</u></p> <p>Fig. 4 of Dolk.</p>
<p>13. The relational database processing system of claim 10, wherein said first relation type record comprises a table identifier identifying said first relation instance table.</p>	<p>Dolk teaches that each relation instance record contains a record identifier that corresponds to the desired relation instance record.</p> <p>Fig. 4 depicts a generic representation of a relationship, where <u>rtype</u>, <u>e1name</u>, <u>e1type</u>, <u>e2name</u>, and <u>e2type</u> are keys to a given relationship of relationship type <u>rtype</u>, which associates entities <u>e1name</u> and <u>e2name</u> of entity types <u>e1type</u> and <u>e2type</u>.</p> <p><u>RELSHIP(rtype, e1name, e1type, e2name, e2type, access-method, frequency, rel_pos)</u></p> <p>Fig. 4 of Dolk. One of the attributes to the relationship may identify which table the relationship may be found in the underlying database.</p> <p>Dolk further teaches that the RIRDS may contain a directory component “describing where information resources [i.e. relationship tables] are located and how they can be accessed.” Dolk at 49. Put another way, the directory is analogous to a physical description of where the relationship instance table is located in computer memory. Dolk at 49. The “directory description would contain data on the machine, operating system, and the file structure under which the file is stored.” Dolk at 49.</p>
<p>14. The relational database processing system of claim 10, further comprising an inquiry table containing an inquiry record, wherein said inquiry record specifies said provided relation type and said provided entity.</p>	<p>As noted above under claim 1, the use of running inquiries on the underlying databases was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations.</p> <p>Shaw takes the QBE skeleton tables, for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11,</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
	<p>lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL statement (inquiry record).</p> <p>Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement, which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1. Thus for example, SKILL_USED is the name of a relationship between a provided EMPLOYEE entity and a desired SKILL entity.</p>
<p>15. The relational database processing system of claim 10 further comprising:</p>	
<p>a second entity instance table associated with a second entity type; and</p>	<p>Dolk discloses the existence of a second entity instance tables as part of the IRDS architecture. The IRD schema and data layers map the contents of the underlying database, including entity instance tables. As relationships associate two entities, a second entity instance table associated with a second entity type is part of the IRDS architecture, and is mapped by the IRD schema and data layers.</p> <p>A relationship associates two entities with (names e1name and e2name)</p>

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References								
	<p style="text-align: center;">Relationships</p> <p>All relationships have the same attributes and keys: REL(<u>e1name</u>, <u>e1type</u>, <u>e2name</u>, <u>e2type</u>) where e1name, e2name are the entity instances e1type, e2type are the entity-types of which e1name, e2name are instances, respectively</p> <p>REL is any of the relationships CONTAINS, PROCESSES, RUNS, RESP_FOR, CALLS, GOES_TO, DERIVED_FROM, ALIAS, and KWIC</p> <p style="text-align: center;">Integrity constraints</p> <p>See Figure 3</p> <p style="text-align: center;">FIGURE 5. Relational IRDS (AIRDS)</p> <p>(in pertinent part).</p> <p>Thus, an entity instance table associated with a second entity type is taught by Dolk. Thus, a second entity instance table might be one for Empl-Record, where the first entity instance table was one for Soc-Sec-No. The two entity types might be associated by the relationship Empl-Record-CONTAINS-Soc-Sec-No.</p>								
<p>wherein said entity definition table contains a second entity type record containing said second entity type and associating said second entity instance table with said second entity type.</p>	<p>Dolk teaches that an entity type record of a second entity type may be contained in a second instance table of said second entity instance type. Further, the entity definition table may contain the second entity type as a record.</p> <p>As noted in Fig. 1 of Dolk, the IRD Layer defines both entity type records Soc-Sec-No and Employee-Rec.</p> <table border="0" style="width: 100%;"> <tr> <td style="width: 50%;">IRD schema description layer</td> <td style="width: 50%; text-align: right;">Entity-type</td> </tr> <tr> <td>IRD schema layer</td> <td style="text-align: right;">ELEMENT, RECORD, etc.</td> </tr> <tr> <td>IRD data layer</td> <td style="text-align: right;">Soc-Sec-No, Empl-Record, etc.</td> </tr> <tr> <td>Operational data</td> <td style="text-align: right;">555-23-6666 (Employee record for Kirk)</td> </tr> </table>	IRD schema description layer	Entity-type	IRD schema layer	ELEMENT, RECORD, etc.	IRD data layer	Soc-Sec-No, Empl-Record, etc.	Operational data	555-23-6666 (Employee record for Kirk)
IRD schema description layer	Entity-type								
IRD schema layer	ELEMENT, RECORD, etc.								
IRD data layer	Soc-Sec-No, Empl-Record, etc.								
Operational data	555-23-6666 (Employee record for Kirk)								
<p>16. The relational database</p>									

Claim Language ¹⁵	Dolk, Teorey, and Zloof/Shaw References
<p>processing system of claim 15 further comprising:</p> <p>a third entity instance table associated with a third entity type; and</p>	<p>Teorey teaches three entity instance tables, for example the SKILL, DEPARTMENT and DIVISION entity tables of FIG. 13. Teorey at 216, FIG. 13, Table 1.</p> <p>Figure 13. Company personnel and project database candidate relations.</p>
<p>wherein said entity definition table contains a third entity type record containing said third entity type and associating said third entity instance table with said third entity type.</p>	<p>Teorey teaches that all entities, including SKILL, DEPARTMENT and DIVISION entities, are transformed into relations that are stored in the data dictionary. Teorey at 216, Table 1. Huber teaches that the data dictionary contains a set of table definitions. Huber, 6:68-7:3. A set, by its very nature, can contain three (or more) entity type records.</p> <p>As indicated above, while Huber teaches the use of a relation definition table, the use of data dictionaries to map the contents of the underlying database, including the entities and relationships within it, was already common in the field. A data dictionary was considered “[a] system database that contain[ed] information about a user database, such as location of data, lists of fields and tables, and data types and lengths.”</p>

Claim Language¹⁵	Dolk, Teorey, and Zloof/Shaw References
17. The relational database processing system of claim 10 further comprising:	
a second relation instance table associated with a second relation type; and	<p>Dolk teaches that a plurality of relationship types may exist within a given IRDS system and within a given relation instance table. As the IRDS in Dolk maps the contents of the underlying ORACLE E-R database, a second relation instance table may exist.</p> <p>A second relation type and an associated relation instance table may be part of the relational database. More than one relation type may exist - "The IRD architecture is based on the entity-relationship [E-R] model The IRD consists of entities, attributes, and relationships that are instances of the corresponding IRD schema entity-types, relation-types, and attribute-types." Dolk at 50. The IRDS may be implemented using Oracle tables and the contents of the same may be accessed using SQL commands. Dolk at 55. Finally, as the IRDS maps the contents of the operational data in the underlying database, the second relationship type in the IRDS is associated with a relationship table in the underlying database.</p>
wherein said relation definition table contains a second relation type record containing said second relation type and associating said second relation instance table with said second relation type.	<p>Dolk teaches that a relationship instance record, which corresponding to the second relationship type may be found in a second relation instance table.</p> <p>As the underlying database upon which the freestanding IRDS is based may be programmed in Oracle, the relationship instances may be stored in a relation instance table, corresponding to said relationship type record.</p>
18. The relational database processing system of claim 17 further comprising:	
a third relation instance table associated with a third relation type; and	Dolk teaches that a plurality of relationship types may exist within a given IRDS system and within a given relation instance table. As the IRDS in Dolk merely maps the contents of the underlying ORACLE E-R database, a third relation instance table, to which the IRDS maps, may exist.
wherein said relation definition table contains a third relation type record containing said third relation type and associating said third relation instance table with said third relation type.	Dolk teaches that a relationship instance record, which corresponding to the third relationship type may be found in a second relation instance table.

D. Tsichritzis, Munz, Zloof, and Shaw References

The Tsichritzis, Munz I, and Munz II references, in combination with Zloof and/or Shaw, as taught in the above charts and in the below, teach all limitations of claims 5-6 and 14 of the '259 patent.

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
<p>5. The method of claim 1, further comprising retrieving data specifying said provided relation type from an inquiry table.</p>	<p><u>Tsichritzis</u> '<u>259 specification description of INQ.DEF or inquiry table</u> The '259 specification describes path selection using the inquiry table as <starting entity type> <connecting relationship type><intermediate entity type> <connecting relationship type><intermediate entity type>...'259 patent, col. 29, lines 57-65.</p> <p><u>Tsichritzis description of inquiry table</u> In Tsichritzis, a user constructs a relation by selecting a record type, then linking on a different type record type and possibly selecting and linking again. See Tsichritzis, pp 126, para 6. E.g.: <u>Define relation</u> relation-name <u>from A select SA</u> (This is <starting entity type>) <u>Link with LAB to B select SB</u> (This is <connecting relationship type><intermediate entity type>) <u>Link with LBC to C select SC</u> (This is <connecting relationship type><intermediate entity type>) The above relation (path) is stored in a RELATION TABLE. See Tsichritzis, pp. 126, para. 5. The RELATION TABLE is the claimed inquiry table. When a relation is created, the relation definition in the RELATION TABLE is retrieved, including the link name, which specifies the link (provided relation type).¹⁶</p> <p><u>Munz</u> According to Munz I, Munz queries are supplied as a comma-separated values table (pattern string) to a procedure call. This pattern string is inherently stored by the PL/I procedure in a data structure (such as an array)</p>

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
	<p>which is a table. See Munz I, sec. 4.3.3, p. 107. The example pattern strings in Munz I are laid out in a row and column style. This layout corresponds to the layout of the rows of the inquiry table of this claim. Compare Munz I, sec. 4.3.3 (example pattern string) with '259 patent, 25:44-26:10, FIG. 7-1, element 740. This pattern string is simply a textual representation of the same underlying data structure as claimed.</p> <p>The provided relation type data is retrieved from the table the query is stored in, since this how such data is provided to the Munz system by the procedure call. See Munz I, p. 108, sec. 4.3.4. (FIND_DB procedure is how patterns are provided to the database; pattern string is a parameter to FIND_DB). For example, using the first pattern string discussed in sec. 4.3.3 of Munz I, the relation type data would be "WORKS_IN" in either of the first two lines. See id at 4.3.3.</p> <p><u>Zloof/Shaw</u></p> <p>The use of inquiry tables such as ones based on the QBE language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to <u>The Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990). While a QBE interface is pictorial in nature, a SQL's interface is more linear and textual. Id. "The user begins by specifying which table is needed for a particular query. Once a table(s) is chosen, the system displays an outline of that table, showing the table name and the names of its fields. To issue a query, the user gives an example of the required information, which amounts to specifying a variable name in the column of the desired information." Id.</p> <p>Zloof teaches Query-by-Example (QBE). When a user performs an operation, e.g., query, update, define, or control, against the data base, the user fills in an example of a solution to that operation in skeleton tables that can be associated with actual tables in the database. For illustration, suppose we want all green items sold in the toy</p>

Claim Language

Tsichritzis, Munz, Zloof, and Shaw References

department.

Figure 13 Qualified retrieval using links

TYPE	ITEM	COLOR	SIZE	SALES	DEPT	VIEW
	P. NUT	GREEN			TOY	NUT

Fig. 13 of Zloof above shows the two skeleton tables: "TYPE" and "SALES". These two tables are linked by the word "NUT". Thus Figure 13 shows an inquiry table consisting of two skeleton tables linked together like the level 1 and 2 rows of the INQ.DEF table 740 in Figure 7-1 of Doktor.

The '326 patent to Shaw teaches how to synthesize a linear query for accessing the contents of a relational database from a graphic query input at a user terminal, including relation records. Shaw takes the QBE skeleton tables, for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.

Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL, such as the SQL select statement in Table 24.

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
	<p style="text-align: center;">FIG. 5</p>
6. The method of claim 1, further comprising retrieving data	<p>Tsichritzis See claim 5 above. When a relation is created, the</p>

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
<p>specifying said provided entity from an inquiry table.</p>	<p>provided entity is also retrieved from the RELATION TABLE. Tsichritzis, p. 128, para. 2.</p> <p><u>Munz</u> According to Munz I, Munz queries are supplied as a comma-separated values table (pattern string) to a procedure call. This pattern string is inherently stored by the PL/1 procedure in a data structure (such as an array) which is a table. See Munz I, sec. 4.3.3, p. 107. The example pattern strings in Munz I are laid out in a row and column style. This layout corresponds to the layout of the rows of the inquiry table of this claim. Compare Munz I, sec. 4.3.3 (example pattern string) with '259 patent, 25:44-26:10, FIG. 7-1, element 740. This pattern string is simply a textual representation of the same underlying data structure as claimed.</p> <p>The provided entity data is retrieved from the table the query is stored in, since this how such data is provided to the Munz system by the procedure call. See Munz I, p. 108, sec. 4.3.4. (FIND_DB procedure is how patterns are provided to the database; pattern string is a parameter to FIND_DB). For example, using the first pattern string discussed in sec. 4.3.3 of Munz I, the provided entity data would be "P=PROJECT" in the second line, or "U" in the last line. See id at 4.3.3. and FIG. 18.</p> <p><u>Zloof/Shaw</u> As noted above, the use of inquiry tables such as ones based on the QBE language was an inherent part of managing E-R databases. A QBE table was one form of a graphical front-end interface that allowed a user to access the underlying database containing entities and relations. According to The <u>Database Step-by-Step</u> textbook, "The [QBE] user interface, designed for technical and nontechnical people alike, is a two-dimensional, on-line, video display terminal oriented query facility." Mark L. Gillenson, <u>Database Step-by-Step</u> 141-42 (2d Ed. 1990).</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content</p>

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
	<p>89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL for the query.</p> <p>One element of the SQL query stored in the GFTSQL 70 is the entity provided as a parameter to the query. For example, Table 24 of Shaw depicts an example of a query, in both SQL and QBE formats. Shaw, 26:59-68. In this example, the query is seeking the employees who work in the San Jose Department, and who meet a salary condition of > \$20,000. Id. The provided entity is "DEPT", and the desired entity is "EMP". Id.</p> <p>The QBE example table shown in Table 24 is another example of an inquiry table as claimed. This inquiry table contains the same DEPT provided entity information, which is retrieved from the table in order to create the SQL statement SELECT DNO FROM DEPT shown in Table 24. Shaw, 15:38-42, 26:59-68.</p>
<p>14. The relational database processing system of claim 10, further comprising an inquiry table containing an inquiry record, wherein said inquiry record specifies said provided relation type and said provided entity.</p>	<p><u>Tsichritzis</u> The RELATION TABLE is the inquiry definition table. See claim 5. The relation table contains the definition of a relation. <u>Specific example:</u> <u>Define relation</u> relation-name <u>from</u> A <u>select</u> SA <u>Link with</u> LAB <u>to</u> B <u>select</u> SB</p> <p>See Tsichritzis, pp 127, para 6. Each relation definition is a inquiry record. Each relation definition specifies the provided relation type (LAB) and said provided entity (A select SA).</p> <p><u>Munz</u> According to Munz I, Munz queries are supplied as a comma-separated values table (pattern string) to a procedure call. This pattern string is inherently stored by the PL/1 procedure in a data structure (such as an array) which is a table. See Munz I, sec. 4.3.3, p. 107. The example pattern strings in Munz I are laid out in a row and column style. This layout corresponds to the layout of the rows of the inquiry table of the this claim. Compare Munz I, sec. 4.3.3 (example pattern string) with '259 patent, 25:44-26:10, FIG. 7-1, element 740. This pattern string is simply a textual representation of the same underlying data structure as claimed.</p>

Claim Language	Tsichritzis, Munz, Zloof, and Shaw References
	<p>The provided relation type data and provided entity data is retrieved from the table the query is stored in, since this how such data is provided to the Munz system by the procedure call. See Munz I, p. 108, sec. 4.3.4. (FIND_DB procedure is how patterns are provided to the database; pattern string is a parameter to FIND_DB). For example, using the first pattern string discussed in sec. 4.3.3 of Munz I, the relation type data would be "WORKS_IN" in either of the first two lines, and the provided entity data would be "P=PROJECT" in the second line, or "U" in the last line. See id at 4.3.3. and FIG. 18.</p> <p><u>Zloof/Shaw</u></p> <p>Shaw takes the QBE skeleton tables, for example, from Zloof, and generates an SQL query, which is then used to retrieve the inquiry results from the relational database.</p> <p>Figure 5 of Shaw shows a collection of tables DXTGTF 106 that provides a table GFTTABLE 72 having one entry for each skeleton or example table in a query 2 [col. 11, lines 1-2]. Thus GFTTABLE 72 is the Inquiry table 740 in Doktor Fig. 7-1. GFTCOLMN 78 provides one entry for each column of an example table in a query Q. The content 89 of GFTCOLMN 78 includes a pointer 71 to GFTSQL 70. GFTSQL 70 contains the SQL , such as the SQL select statement in Table 24 (inquiry record).</p> <p>Where the SQL query operates on a relationship relation as described in Teorey, the SQL query will include an SQL FROM statement which identifies the name of relationship relation being queried. Shaw, 15:28-31. This name is the relation type, since the relationship relation is named with it's type, e.g. the SKILL_USED relationship type is also the name of the relationship relation containing the relation instances of that type. Teorey, 203 FIG. 4; 212-13 FIG. 10; 216, FIG. 13, Table 1. Thus for example, SKILL_USED is the name of a relationship between a provided EMPLOYEE entity and a desired SKILL entity.</p>

VIII. CONCLUSION

The prior art references attached hereto as Exhibits PA-A through PA-J, considered in view of the admissions presented in this application, raise substantial new questions of patentability of claims 1-18 of the '259 patent. These references render

these claims unpatentable under 35 U.S.C. §§ 102 and/or 103. Accordingly it is respectfully requested that the Request for Reexamination be granted and that the PTO give due consideration to the prior art discussed herein.

Respectfully submitted,

Dated: May 9, 2007



William L. Anthony, Jr.
Reg. No. 24771
Attorney for Petitioner Oracle
Corporation

IN THE UNITED STATES PATENT OFFICE

Request For *Ex Parte* Reexamination Of:

U.S. Patent No. 5,826,259

Inventor: Karol Doktor

Assignee: Financial Systems Technology
Pty. Ltd.

Filed: May 22, 1997

Issued: October 20, 1998

For: Easily Expandable Data
Processing Systems and
Method

INDEX FOR *EX PARTE*
REEXAMINATION OF U.S. PATENT
NO. 5,826,259

CERTIFICATE OF SERVICE

I, Stephanie C. Hart, hereby certify that on May 9, 2007, true and correct copies of the following documents were served on the following counsel of record at the addresses and in the manner indicated:

- 1. REQUEST FOR EX PARTE REEXAMINATION TRANSMITTAL FORM 1465**
- 2. PTO-1449 AND REFERENCES CITED THEREON**
- 3. REQUEST FOR EX PARTE REEXAMINATION OF U.S. PATENT NO.259 ATTACHMENT TO FORM 1465**
- 4. CERTIFICATE OF SERVICE**

I hereby certify that the attached associated documents are being deposited with the United States Postal Service on this date in an envelope as "Express Mail Post Office to Addressee" addressed to the following:

EDWARD KWOK, ESQ,
MACPHERSON KWOK CHEN & HEID LLP
2033 GATEWAY PLACE
SUITE 400
SAN JOSE, CA 95110

ALLEN, DYER, DOPPELT, MILBRATH & GILCHRIST, P.A.
1401 CITRUS CENTER
25 SOUTH ORANGE AVENUE
P.O. BOX 3791
ORLANDO, FL 32802-3791

SAM BAXTER, ESQ.
McKOOL SMITH P.C.
505 EAST TRAVIS, SUITE 105
P.O. BOX O
MARSHALL, TEXAS 75670
TELEPHONE: 903-927-2111
FACSIMILE: 903-927-2622

Date of Mailing: May 9, 2007


Stephanie C. Hart