US005276870A

United States Patent [19]

Shan et al.

[11] Patent Number: 5,276,870

[45] Date of Patent: Jan. 4, 1994

[54] VIEW COMPOSITION IN A DATA BASE MANAGEMENT SYSTEM

[75] Inventors: **Ming-Chien Shan**, San Jose; **Peter Lyngbaek**, Mountain View, both of Calif.

[73] Assignee: **Hewlett-Packard Company**, Palo Alto, Calif.

[21] Appl. No.: **595,717**

[22] Filed: **Oct. 9, 1990**

Related U.S. Application Data

[63] Continuation of Ser. No. 131,876, Dec. 11, 1987, abandoned.

[51] **Int. Cl.**[5] **G06F 15/403**

[52] **U.S. Cl.** **395/600**; 364/974; 364/974.6; 364/978; 364/DIG. 2

[58] **Field of Search**..... 395/600; 364/DIG.1, DIG. 2

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,497,039	1/1985	Kitakami et al.	364/900
4,506,326	3/1985	Shaw et al.	364/300
4,631,673	12/1986	Haas et al.	364/300
4,644,471	2/1987	Kojima et al.	364/300
4,688,195	8/1987	Thompson et al.	364/300
4,769,772	9/1988	Dwyer	364/300
4,791,561	12/1988	Huber	395/600
4,805,099	2/1989	Huber	364/300
4,888,690	12/1989	Huber	395/600
4,918,593	4/1990	Huber	395/600
4,930,071	5/1990	Tou et al.	395/600
5,097,408	3/1992	Huber	395/600

FOREIGN PATENT DOCUMENTS

2172130 3/1986 Japan .

OTHER PUBLICATIONS

Roussopoulos, Nicholas: "View Indexing in Relational Databases", ACM Transactions on Database Systems, vol. 7, No. 2, Jun. 1982, pp. 258-290.

Rowe et al: "Data Abstraction, Views and Updates in RIGEL", *The Ingres Papers: Anatomy of a Relational Database System,* Addison-Wesley Publ., 1986, pp. 278-294.

Dayal et al. "On the Updatability of Relational Views", Proceedings of 4th Internatinoal Conf. on Very Large Databases, Sep. 1978, pp. 368-377.

K. C. Kinsley et al.: "A Generalized Method for Maintaining Views", AFIPS Conference Las Vegas, Nevada, Jul. 9-12, 1984, pp. 587-593.

S. Talbot: "An Investigation into Logical Optimization of Relational Query Languages" The Computer Journal, vol. 27, No. 4, No. 1984, pp. 301-309.

S. Hanara et al: "Conversational Database Query Language", Review of the Electrical Communication Laboratories, vol. 29, Nos. 1-2, Jan./Feb. 1981, pp. 32-50.

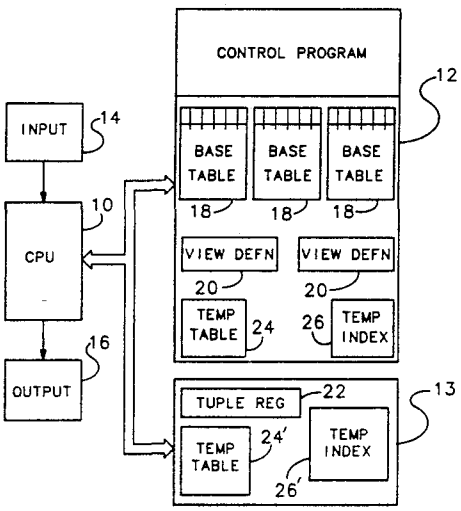
C. J. Date: "An Introduction to Data Base Systems", 3rd Edition, Section 9, The External Level of System R, Nov. 1974, pp. 159-168.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Maria N. Von Buhr

[57] **ABSTRACT**

In a database management system that operates according to the Structured Query Language standard, a query containing a reference to a view is processed by dynamically materializing the view through execution of the view definition. Once the view is materialized, it is treated as any other base table. To enable a query optimization plan to refer to the materialized view, a view node is introduced into the execution plan. The view node includes a subquery that results in the creation of the virtual table defined by the view. This created table is temporarily stored in a memory. The query optimizer can treat view nodes in the same manner as stored base tables, and thereby overcomes restrictions that were placed upon views by previous view decomposition approaches.

3 Claims, 3 Drawing Sheets



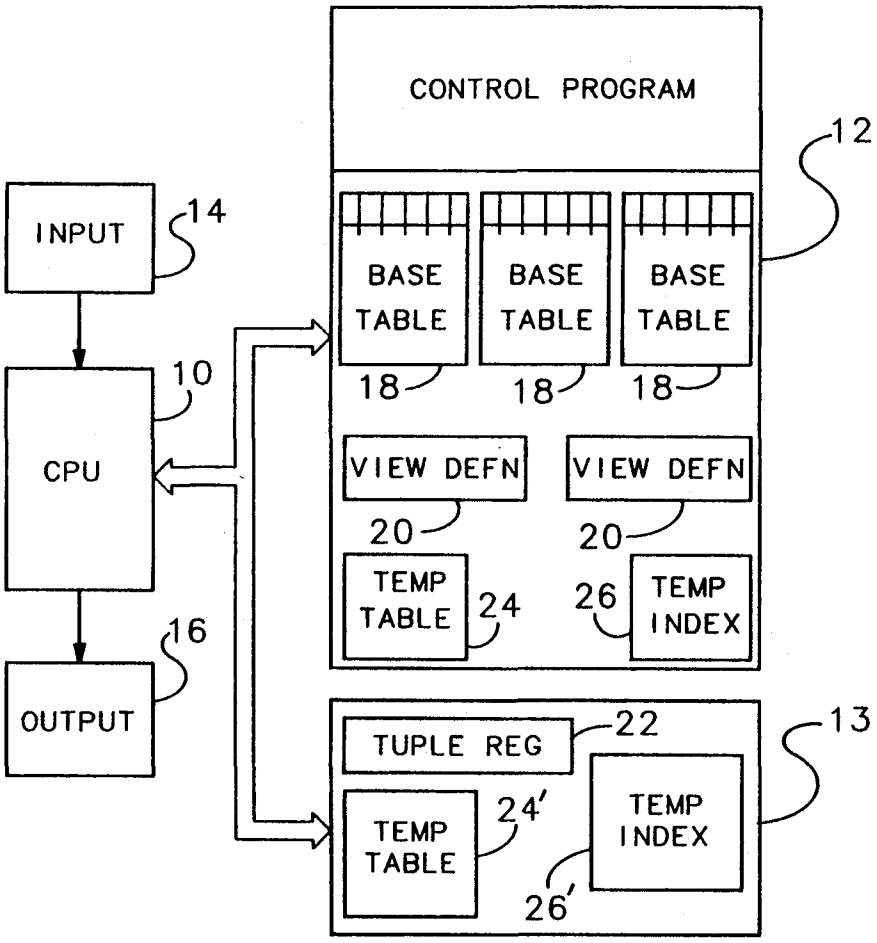


FIG 1

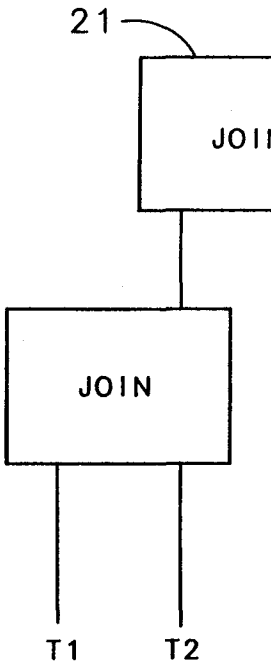


FIG 2

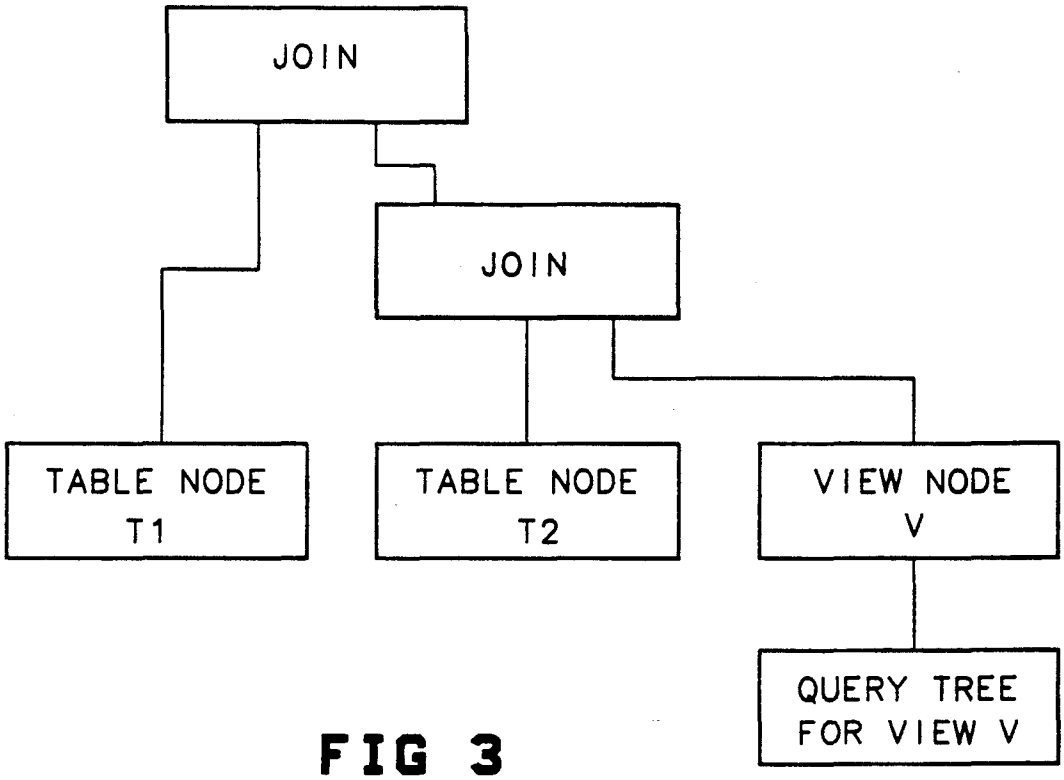


FIG 3

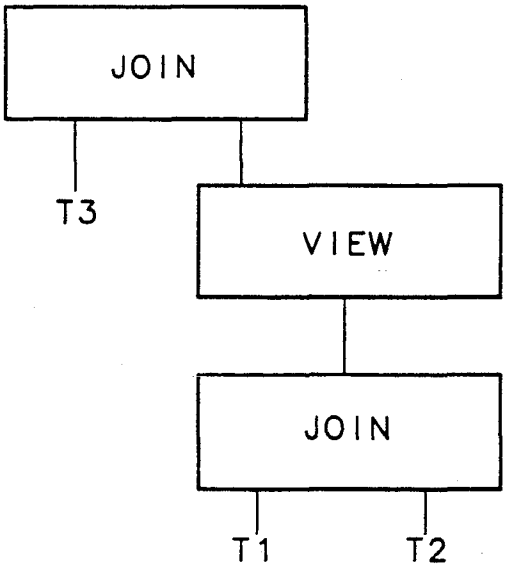


FIG 4

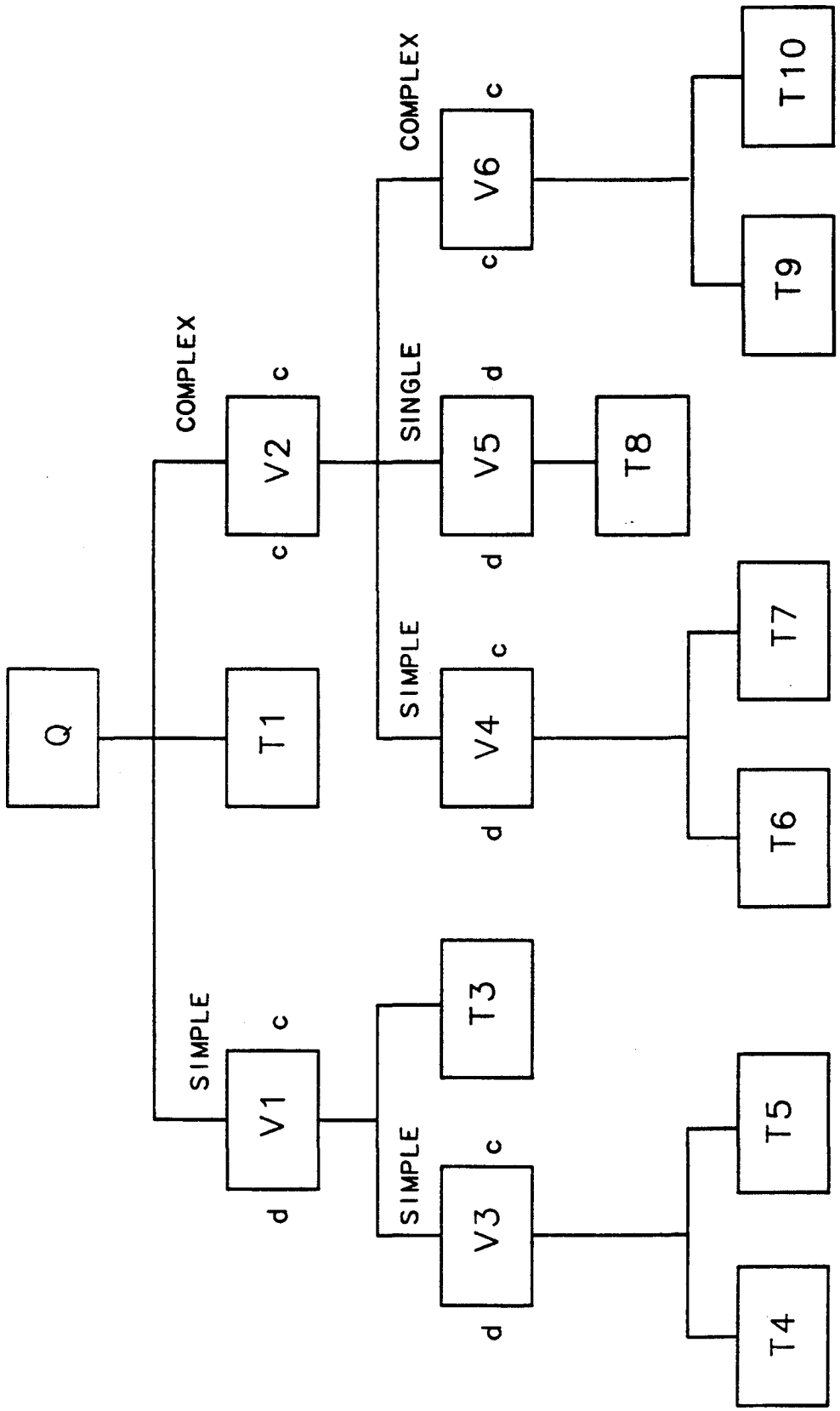


FIG 5

VIEW COMPOSITION IN A DATA BASE MANAGEMENT SYSTEM

CROSS REFERENCE TO RELATED APPLICATIONS

This is a continuation of copending application 7/131,876 filed on Dec. 11, 1987, now abandoned.

BACKGROUND OF THE INVENTION

The present invention is directed to data base systems, and more particularly to a technique which enables data to be efficiently retrieved with the use of views having complex definitions. Although not limited thereto, the invention is particularly concerned with the standard for relational data bases known as "Structured Query Language" (SQL) and the constraints imposed thereby. For further detailed information relating to this standard, reference is made to Date, *A Guide To The SQL Standard* (1987) and Date, *A Guide To DB2* (1985).

In relational data base management systems of the type to which the SQL standard pertains, data is stored in the form of base tables. Each base table essentially consists of a series of fields which define columns of the table. Each row of the table comprises a single record, also referred to as a "tuple." For each row of data in a table, a counterpart of that data is physically stored in the data base. When the data base user accesses a base table, the appropriate portion of the stored data is retrieved and presented to him in the form of a table.

In addition to base tables, data can also be accessed by means of "views." In contrast to a base table, a view is a "virtual" table in that the table does not directly exist in the data base as such, but appears to the user as if it did. In fact, the view is stored in the data base only in terms of its definition. For example, a view might comprise a subset of a base table, such as those tuples of the table which have a predetermined value in a particular field. As another example, a view could comprise a join operation, e.g. union, of two or more tables, where these other tables can be base tables, other views or a combination of both.

In relational data base systems that comply with the SQL standard, desired information is searched for and retrieved by means of expressions known as queries. For example, if a table named "Employee" contains the fields "Name", "Dept", "Age" and "Salary", and a user desires to find the subset of employees who work in the toy department, the following query can be used:

```
SELECT  Name, Salary, Age
FROM    Employee
WHERE   Dept = "Toy"
```

If it is known that the information produced by this query will be used several times, a view can be defined, to thereby avoid the need to reformulate the query each time the information is desired. A view named "Emptoy" which produces the same information can be defined by means of the following statement:

```
CREATE VIEW   Emptoy (Name, Salary, Age) AS
SELECT        Name, Salary, Age
FROM          Employee
WHERE         Dept = "Toy"
```


With this view created, the original query would then become:

```
SELECT *
FROM Emptoy
```

(where the character "*" is a universal character to designate all fields in the table or view). As can be seen, the prior definition of the view makes the query much easier to formulate.

In addition, this view can be the subject of a further query. For example, to obtain the salary of all employees named Johnson in the toy department, the following query can be entered:

```
SELECT  Salary
FROM    Emptoy
WHERE   Name = "Johnson"
```

As noted above, the data which makes up the virtual table Emptoy is not actually stored as such in the data base. Rather, only the definition of the view is stored. Accordingly, when a query which references a view is entered, a process known as view decomposition is carried out to translate the query into an equivalent query which references only stored base tables. In essence, the view decomposition process comprises the steps of replacing the reference to the view in the query with the definition of that view. Thus, in the example given above, the query for obtaining the salary of the employees named Johnson in the toy department would be modified to produce the following equivalent query:

```
SELECT  Salary
FROM    Employee
WHERE   Name = "Johnson" and Dept = "Toy"
```

As can be seen, the modified query refers only to base tables and fields within that table.

In principle, it should be possible for a query to reference any view defined by an arbitrary query, and to translate that query into an equivalent operation on the underlying base tables. In practice, however, the applicability of this procedure is quite limited because of restrictions imposed by the SQL standard. For example, a view labeled "Avgsal" can be defined from two base tables Dept and Emp. The table Dept contains the fields "Dno" (department number), "Dname", "Mgrname" and "Loc", and the table Emp contains the fields "Eno" (employee number), "Dno", "Salary" and "Jobtitle". The definition of the view Avgsal which describes a department by its department number, department name and the average salary of its employees could appear as follows:

```
CREATE VIEW Avgsal   (Dno, Dname, Avgsalary) AS
SELECT               Dno, Dname, AVG(Salary)
FROM                 Dept, Emp
WHERE                Dept.Dno = Emp.Dno
GROUP BY             Dno
```

A query that asks for the location of all departments whose employees have an average salary greater than $30,000 would require a join operation on the base table

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.