(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 October 2002 (03.10.2002)

PCT

(10) International Publication Number
WO 02/077879 A1

(51) International Patent Classification[7]: G06F 17/40

(21) International Application Number: PCT/KR02/00491

(22) International Filing Date: 22 March 2002 (22.03.2002)

(25) Filing Language: Korean

(26) Publication Language: English

(30) Priority Data:
2001/15453 24 March 2001 (24.03.2001) KR

(71) Applicant (*for all designated States except US*): **EXEM LTD.** [KR/KR]; 4Th Floor, Future-Flow Bldg., 1538-9 Seocho-dong, Seocho-gu, Seoul 137-070 (KR).

(72) Inventor; and
(75) Inventor/Applicant (*for US only*): **CHO, Chong-Arm** [KR/KR]; 105-1301 Seongdong-Maul-Gangnam-Village, 84 Seongbook-ri, Sooji-eup, Yongin-shi 449-840, Gyeonggi-do (KR).

(74) Agent: **PARK, Young-Il**; Hyundail Life Insurance Bldg., 5F., 649-14 Yoksam-dong, Gangnam-gu, Seoul 135-080 (KR).

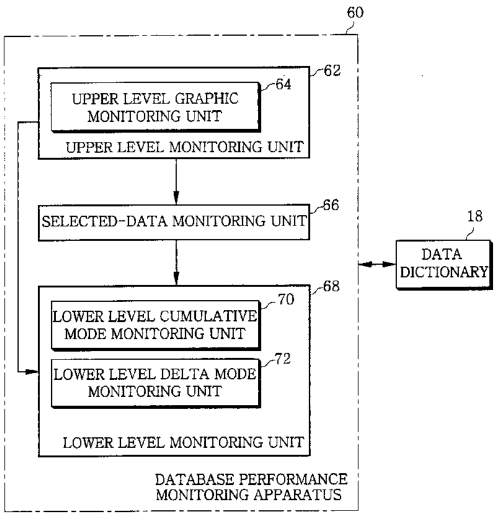
(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) **Title:** APPARATUS FOR MONITORING PERFORMANCE OF DATABASE AND METHOD THEREOF



(57) **Abstract:** An apparatus for monitoring the performance of a database included in an information processing system, and a method thereof are provided. The apparatus includes an upper level monitoring unit for calculating and displaying variations per unit time of the database performance data of the whole database level; a selected-data monitoring unit for accessing database performance data of each program corresponding to database performance data selected from among the database performance data, which are displayed by the upper level monitoring unit, calculating variations per unit time of the accessed database performance data, and displaying the calculated variations per unit time in descending order of size; and a lower level monitoring unit for displaying database performance data of the program level with respect to a program selected from among the programs displayed by the selected-data monitoring unit.

WO 02/077879 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

APPARATUS FOR MONITORING PERFORMANCE OF DATABASE AND METHOD THEREOF

Technical Field

The present invention relates to an information processing system including a database, and more particularly, to an apparatus for monitoring the performance of a database included in an information processing system, and a method thereof.

A database is a group of data which is configured to allow the content of data to be easily accessed, processed, and updated. A DataBase Management System (DBMS) is a program which allows a plurality of computer users to write data to a database or to access data within the database. Such a DBMS manages requests from users or other programs such that the users or the other programs can use particular data even if the users or the other programs do not know where the particular data is actually stored in a storage medium. A DBMS guarantees integrity and security of data in processing user requests. A relational DBMS (RDBMS) is a most general type of DBMS, and a standardized user and program interface of an RDBMS is referred to as a Structured Query Language (SQL). Hereinafter, the term “database” is used not only to indicate a group of data in a narrow sense but also as including a DBMS in a broad sense.

Whether an information processing system succeeds or not is depends on effective construction of a database and technology of managing the database. Accordingly, labor for developing and researching software tools for tuning the performance of a database included in an information processing system has been continued. However, since a tuning technique varies with the physical and logical status of data in a database as well as a variety of environmental factors of the database, a database performance tuning tool which can generally solve performance degradation problems, which are analyzed at a whole database level, at a programming level, and at an SQL level through monitoring, has not been proposed yet.

In such a status, studies have become to focus on a database performance monitoring technique which can support a database tuner, who is a professional consultant equipped with a variety of experiences and knowledge, such that the database tuner can effectively provide tuning solutions. Such a method requires a software tool for effectively monitoring fundamental information for tuning although persons tune a database.

Background Art

Generally, when application software accesses a database, a DBMS dynamically generates lots of data related to the performance of the database and stores the data at a specific area in memory. Particularly, the performance data (i.e., performance statistics data and waiting event data) which is generated by a DBMS of Oracle is classified into a whole database level, a programming (or a database session) level, and an SQL level and is then stored in individual data dictionaries, which are allocated in a memory space, for a predetermined period of time.

Conventional database performance monitoring tools do not apprehend the inter-relationships among a large number of database performance data which are produced by levels but just fractionally monitor data of each level. Accordingly, when using conventional database performance monitoring tools, it cannot be performed to check the performance of a database through general apprehension of performance statistics data and waiting event data, which are primarily detected in a broad range (i.e., at a whole database level), and there cannot be provided a function of narrowing the broad range of the problem of a database performance degradation.

Moreover, conventional database performance monitoring tools are limited to simply providing some selected data without properly processing database performance data generated by a DBMS. However, since database performance data generated by a DBMS is mostly a cumulative value after a particular time point, it is difficult to apprehend database performance at present time and at a past time point near to the present time, which a tuner is mainly interested in.

In addition, conventional database performance monitoring tools cannot provide a means for simultaneously monitoring a plurality of databases. Accordingly, when a single database is monitored, monitoring processes for other databases hide as backgrounds, so it is difficult to provide information for entirely tuning an information processing system including a plurality of databases.

Disclosure of the Invention

To solve the above problems, it is a first object of the present invention to provide an apparatus for monitoring database performance, which can easily monitor database performance data of a programming level related to the problem of database performance degradation which is apprehended at a whole database level by observing the inter-relationships among a large number of data generated by levels, and a method thereof.

It is a second object of the present invention to provide an apparatus for monitoring database performance, which can easily monitor database performance at present time and at a past time point near to the present time, which a tuner is mainly interested in, and a method thereof.

It is a third object of the present invention to provide an apparatus for monitoring database performance, which can provide information for entirely tuning an information processing system including a plurality of databases, and a method thereof.

To achieve one or more objects of the present invention, there is provide an apparatus for monitoring database performance using a plurality of database performance data which are generated and classified into a whole database level, a program level, and a Structured Query Language (SQL) level by a database management system installed in an information processing system. The apparatus includes an upper level monitoring unit for calculating and displaying variations per unit time of the database performance data of the whole database level; a selected-data monitoring unit for accessing database performance data of each program corresponding to database performance data selected from among the database performance data, which are displayed by the upper level monitoring unit, calculating variations per unit time of the accessed database performance data, and displaying the calculated variations per unit time in descending order of size; and a lower level monitoring unit for displaying database performance data of the program level with respect to a program selected from among the programs displayed by the selected-data monitoring unit.

The upper level monitoring unit includes a whole database graphic monitor for calculating variations per unit time in with respect to a predetermined number of database performance data among the database performance data of the whole database level and in relation to all databases installed in the information processing system so as to generate time-transition graphs for the predetermined number of database performance data and databases and displaying all of the generated time-transition graphs on a single screen.

The lower level monitoring unit includes a lower level cumulative mode monitor for displaying cumulative values of the database performance data of the program level; and a lower level delta mode monitor for calculating and displaying variations per unit time of the database performance data of the program level.

The database performance data includes a plurality of performance statistics data which indicate the degree of use of each resource provided in the information processing system and a plurality of waiting event data which indicate the amount of waiting time according to competition for the resource.

To achieve one or more objects of the present invention, there is also provide a method of monitoring database performance using a plurality of database performance data which are generated and classified into a whole database level, a program level, and an SQL level by a database management system installed in an information processing system. The method includes an upper level monitoring step of calculating and displaying variations per unit time of the database performance data of the whole database level; a performance data selecting step in which a user selecting one from among the database performance data which are displayed in the upper level monitoring step; a selected-data monitoring step of accessing database performance data of each program corresponding to the selected database performance data, calculating variations per unit time of the accessed database performance data, and displaying the calculated variations per unit time in descending order of size; a program selecting step in which the user selects one among the programs displayed in the selected-data monitoring step; and a lower level monitoring step of displaying database performance data of the program level with respect to selected program.

The upper level monitoring step includes calculating variations per unit time in with respect to a predetermined number of database performance data among the database performance data of the whole database level and in relation to all databases installed in the information processing system so as to generate time-transition graphs for the predetermined number of database performance data and databases and displaying all of the generated time-transition graphs on a single screen.

The lower level monitoring step includes a lower level cumulative mode monitoring step of displaying cumulative values of the database performance data of the program level; and a lower level delta mode monitoring step of calculating and displaying variations per unit time of the database performance data of the program level. Conversion between the lower level cumulative mode monitoring step and the lower level delta mode monitoring step is performed by the user's operation.

The database performance data includes a plurality of performance statistics data which indicate the degree of use of each resource provided in the information processing system and a plurality of waiting event data which indicate the amount of waiting time according to competition for the resource.

Brief Description of the Drawings

FIG. 1 is a diagram for explaining the connective relationship between an oracle database and a performance monitoring process according to the present invention.

FIG. 2 is a diagram of the hardware configuration of an information processing system in which an apparatus for monitoring database performance according to the present invention is embodied.

FIG. 3 is a diagram for explaining the functions of a first embodiment of an apparatus for monitoring database performance according to the present invention.

FIG. 4 shows an example of a screen on which delta values of database performance data of a whole database level are displayed by an upper level monitoring unit.

FIG. 5 shows an example of a screen on which time-transition graphs of delta values of database performance data of a whole database level are displayed by an upper level graphic monitoring unit.

FIG. 6 shows an example of a screen on which delta values of each program with respect to database performance data selected in FIG. 4 are displayed.

FIG. 7 shows an example of a screen on which delta values of each program with respect to database performance data selected in FIG. 5 are displayed.

FIG. 8 shows an example of a screen on which cumulative values of database performance data of a program level are displayed by a lower level cumulative mode monitoring unit.

FIG. 9 shows an example of a screen on which delta values of database performance data of a program level are displayed by a lower level delta mode monitoring unit.

FIG. 10 shows an example of a screen on which database performance data of an SQL level is displayed.

FIG. 11 is a flowchart of a method of monitoring database performance according to the present invention.

FIG. 12 is a diagram for explaining the functions of a second embodiment of an apparatus for monitoring database performance according to the present invention.

FIG. 13 shows an example of a screen on which time-transition graphs for individual database performance data and databases, based on delta values of the database performance data of a whole database level, are displayed.

FIG. 14 shows an example of a screen displayed when a user selects a particular database on a waiting event data graphic display area in FIG. 13.

FIG. 15 shows an example of a screen on which delta values for each program with respect to a database and database performance data, which are selected in FIG. 13, are displayed.

Best mode for carrying out the Invention

Hereinafter, preferred embodiment of the present invention will be described in detail with reference to the attached drawings.

For clarity of description, hereinafter, it is assumed that a database is one that is produced by Oracle. However, the present invention is not restricted to the monitoring of the performance of the Oracle databases, and it will be easily understood by those skilled in the art that the present invention can be applied to any type of DBMS in which database performance data is managed hierarchically without changing the essential idea of the present invention.

Referring to FIG. 1, an Oracle database is largely composed of processes, a memory space, and data files.

Here, the processes are programs which are executed in an information processing system and include a user process 14, a server process 12, and Oracle background processes. The user process 14 is used when a user executes an Oracle application program. The user process 14 transmits an SQL statement that is executed by a user to the server process 12 and receives the result of processing the SQL statement from the server process 12. The server process 12 processes an SQL statement, which is received from the user process 14, through parsing, execution, and fetching and transmits the result of processing to the user processor 14. The Oracle background processes are provided for supporting the server process 12 and include, for example, mandatory processes such as a process monitor (PMON), a system monitor (SMON), a database writer (DBWR), and a log writer (LGWR) and other processes such as a checkpoint (CKPT), an archiver (ARCH), a recoverer (RECO), a lock (LCKn), a parallel query (Pnnn), a snapshot refresh (SNPn), a shared server (Snnn), and a dispatcher

(Dnnn).

The memory space in the Oracle database is referred to as a system global area (SGA) 16. The SGA 16 is located in a random access memory (RAM) area of the information processing system and is a group of shared memory areas for all users of the Oracle database. The group of shared memory areas includes data and control information of an Oracle database system. A combination of the SGA 16 and the background processes is referred to as an Oracle instance 20.

The Oracle data file 22 is a file in which actual data stored by a user is stored. In the meantime, in addition to the data file 22, the Oracle database also includes a control file which stores the status and the physical structure of a database, a redo log file which records all changes occurring in the database, and a parameter file which the Oracle instance 20 refers to when starting.

While the Oracle database is in operation, database performance data which is classified into a whole database level, a programming (i.e., a database session) level, and an SQL level is stored in a data dictionary 18 of the SGA 16.

Database performance data of a whole database level is a start point for database monitoring and may be classified as follows:

as for input/output (I/O)-related information,

performance statistics data, i.e., logical read, physical read, and direct read, etc., and

waiting event data, i.e., DB file sequential read and DB file scattered read, etc.;

as for SQL execution performance information at a whole database level,

performance statistics data, i.e., user calls, recursive calls, parse count, and execution count, etc., and

waiting even data, i.e., latch free, library cache pin, and library cache lock, etc.;

as for lock-related information,

performance statistics data, i.e., enqueue waits and equeue deadlocks, and

waiting event data, i.e., enqueue;

as for sort-related information,

performance statistics data, i.e., sort (e.g., memory, disk, and rows), and

waiting event data, i.e., DB file scattered read and direct path read; and

as for response-time-related performance information,

performance statistics data, i.e., recursive CPU usage, CPU used by this session, parse time CPU, and parse time elapsed, and

waiting event data, i.e., all waiting information.

As described above, the database performance data of a whole database level is classified into performance statistics data and waiting event data. The performance statistics data is an index which the Oracle database provides to track and estimate the performance of a database and shows detailed performance information related to a variety of resources (such as an input/output unit, a central processing unit (CPU), and memory). In other words, the performance statistics data is "a direct referential index about use of resources", and overuse or non-overuse of a particular resource can be estimated according to an increase or decrease in a value of each item. The waiting event data can be referred to as "an indirect referential index about use of resources". Since a resource of any item is limited, competition occurs inevitably. Although occurrence of a waiting event does not inevitably mean that a particular resource is overused, it mostly means that there is competition for the particular resource and warns that the problem of performance degradation likely occurs due to overuse of the resource throughout the database. Accordingly, it is necessary to complimentarily apprehend the performance statistics data and the waiting event data in order to identify a problem area during database performance monitoring.

Database performance data of a program level can be tracked only while a relevant program is being executed and disappears from the SGA 18 when the program ends. The kinds of database performance data of a program level are almost similar to the kinds of database performance data of a whole database level.

Database performance data of an SQL level is information of the lowest level. For the database performance data of an SQL level, in order to apprehend the performance of particular SQL in a particular program, there is the difficulty that the database performance data of a program level is mapped to the SQL which changes in the program while the database performance data of a program level is being monitored.

An apparatus for monitoring database performance according to the present invention functions in the form of combination of a hardware, i.e., an information processing system, in which a database is installed, and a software, i.e., performance monitoring process 10, which operates in the information processing system. Similarly to the user process 14, the performance monitoring process 10 can access database performance data stored in the data dictionary 18 of the SGA 18 with the support of the server process 12.

Referring to FIG. 2, an information processing system 30 in which an apparatus for monitoring database performance according to the present invention is embodied includes a computer 31 having at least one CPU 38 and a memory system 32, an input unit 46, and an output unit 48. These elements are connected to one another through at least one bus structure 50.

The CPU 38 includes an arithmetic logic unit (ALU) 40 for performing arithmetic operations and logic operations, a register set 42 for temporarily storing data and a command, and a controller 44 for controlling the operations of the information processing system 30. The CPU 38 used in the present invention is not restricted to particular structures made by particular companies but may be any type of processor having the basic structure as described above.

The memory system 32 includes a high-speed main memory unit 34 and an auxiliary memory unit 36 for storing data for a long term. The main memory unit 34 is composed of RAM and read only memory (ROM) semiconductor chips, and the auxiliary memory unit 36 is manifested as a device for storing data using a floppy disc, hard disc, CD-ROM, flash memory, electrical recording medium, magnetic recording medium, optical recording medium, or other recording medium. In addition, the main memory unit 34 may include video display memory for displaying images through a display device. It will be understood by those skilled in the art of the present invention that the memory system 32 may include a variety of replaceable elements having various storage capacities.

The input unit 46 may include a keyboard, a mouse, a physical converter (e.g., a microphone), etc. The output unit 48 may include a display unit, a printer, a physical converter (e.g., a speaker), etc. A device such as a network interface or modem may be used as an input/output unit.

The information processing system 30 is provided with an operating system and at least one application program. The operating system is software which controls the operations of the information processing system 30 and allocation of resources. The application program is software which performs jobs requested by a user by using available computer resources through the operating system. The operating system and the application program are stored in the memory system 32.

Hereinafter, embodiments of the present invention will be described with reference to the operations performed in the information processing system 30 and symbolic expressions of the operations which are practically used by those skilled in the art of the present invention.

Referring to FIG. 3, a database performance monitoring apparatus 60 according to a first embodiment of the present invention includes an upper level monitoring unit 62, a selected-data monitoring unit 66, and a lower level monitoring unit 68.

The upper level monitoring unit 62 fetches database performance data of a whole database level from the data dictionary 18 and calculates and displays variation per unit time.

The following is an example of a pseudo code for explaining a procedure in which the upper level monitoring unit 62 monitors performance statistics data (v$sysstat) of a whole database level at predetermined time intervals, calculates a current cumulative value, divides the variation between the current cumulative value and the previous cumulative value by a time interval calculated using a second as a time scale, and displays the result of division.

```
INPUT PARAMETER: SECOND
DECLARE
  V_NAME( )            STRING ARRAY;
  V_VALUE_INIT( )      NUMBER ARRAY;
  V_VALUE_CURR( )      NUMBER ARRAY;
BEGIN /* SET INITIAL VALUE FOR INITIALIZATION */
  CURSOR_A AS
  SELECT STATISTIC#, NAME, VALUE
  FROM V$SYSSTAT;
    CURSOR_A FETCH LOOP
      V_NAME(CURSOR_A.STATISTIC#): = CURSOR_A.NAME;
```

```
        V_VALUE_INIT(CURSOR_A.STATISTIC#): =
                                       CURSOR_A.VALUE;
      END FETCH LOOP;
   LOOP (WHEN PROGRAM CLOSES EVENT, THEN EXIT)
     /* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER
   AND DISPLAY DELTA VALUE PER SECOND */
     SLEEP(:SECOND);
     CURSOR_B AS
     SELECT STATISTIC#, NAME, VALUE
     FROM V$SYSSTAT;
       CURSOR_B FETCH LOOP
         V_VALUE_CURR(CURSOR_B.STATISTIC#): =
                                          CURSOR_B.VALUE;
         DISPLAY(V_NAME(CURSOR_B.STATISTIC#),
                 (V_VALUE_CURR(CURSOR_B.STATISTIC#)–
                 V_VALUE_INIT(CURSOR_B.STATISTIC#))/:SECOND);
         /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA VALUE
   PER SECOND IN NEXT TIME INTERVAL */
         V_VALUE_INIT(CURSOR_B.STATISTIC#): =
                      V_VALUE_CURR(CURSOR_B.STATISTIC#);
       END FETCH LOOP;
   END LOOP
   END
```

The following is an example of a pseudo code for explaining a procedure in which the upper level monitoring unit 62 monitors waiting event data (v$system_event) of a whole database level at predetermined time intervals, calculates a current cumulative value, divides the variation between the current cumulative value and the previous cumulative value by a time interval calculated using a second as a time scale, and displays the result of division.

```
INPUT PARAMETER: SECOND
DECLARE
  V_NAME( )           STRING ARRAY;
  V_VALUE_INIT( )     NUMBER ARRAY;
  V_VALUE_CURR( )     NUMBER ARRAY;
BEGIN /* SET INITIAL VALUE FOR INITIALIZATION */
  CURSOR_A AS
  SELECT B.EVENT#, B.NAME, NVL(A.TIME_WAITED,0) VALUE
  FROM V$SYSTEM_EVENT A, V$EVENT_NAME B
  WHERE B.NAME = A.EVENT(+);
    CURSOR_A FETCH LOOP
      V_NAME(CURSOR_A.EVENT#): = CURSOR_A.NAME;
      V_VALUE_INIT(CURSOR_A.EVENT#): = CURSOR_A.VALUE;
    END FETCH LOOP;
LOOP (WHEN PROGRAM CLOSES EVENT, THEN EXIT)
  /* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER
AND DISPLAY DELTA VALUE PER SECOND */
  SLEEP(:SECOND);
  CURSOR_B AS
  SELECT B.EVENT#, B.NAME, NVL(A.TIME_WAITED,0) VALUE
  FROM V$SYSTEM_EVENT A, V$EVENT_NAME B
  WHERE B.NAME = A.EVENT(+);
    CURSOR_B FETCH LOOP
      V_VALUE_CURR(CURSOR_B.EVENT#): = CURSOR_B.VALUE;
      DISPLAY(V_NAME(CURSOR_B.EVENT #),
              (V_VALUE_CURR(CURSOR_B.EVENT#)–
              V_VALUE_INIT(CURSOR_B.EVENT#))/:SECOND);
      /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA PER
SECOND IN NEXT TIME INTERVAL */
      V_VALUE_INIT(CURSOR_B.EVENT#): =
```

```
                V_VALUE_CURR(CURSOR_B.EVENT#);
        END FETCH LOOP;
    END LOOP
    END
```

Database performance data recorded in the data dictionary 18 indicates cumulative values after particular time (for example, a database beginning point). However, as for the whole database level, information necessary for tuning database performance is database performance at current time and at near past time. Accordingly, the upper level monitoring unit 62 cuts out cumulative values of pouring performance data in a delta mode and displays them so that transition of performance can be easily apprehended. FIG. 4 shows an example of a screen displayed by the upper level monitoring unit 62.

Referring to FIG. 3, the upper level monitoring unit 62 includes an upper level graphic monitoring unit 64. The upper level graphic monitoring unit 64 calculates variation per unit time with respect to the predetermined number of database performance data among the database performance data of the whole database level to produce time-transition graphs of the predetermined number of database performance data. All of the time-transition graphs of the predetermined number of database performance data, which are generated by the upper level graphic monitoring unit 64, are displayed on a single screen, as shown in FIG. 5.

In a state in which the screen shown in FIG. 4 or 5 is displayed, if it is determined that particular performance statistics data or particular waiting event data has an excessive value, a user can click the display location of the particular performance statistics data or the particular waiting event data, that is, the location of a graph corresponding to the particular performance statistics data on the screen shown in FIG. 4 or the location of a graph corresponding to the particular waiting event data on the screen shown in FIG. 5. Then, the selected-data monitoring unit 66 shown in FIG. 3 operates. The selected-data monitoring unit 66 accesses database performance data of each program, which corresponds to the database performance data selected by the user from among the database performance data displayed by the upper level monitoring unit 62, from the data dictionary 18 and calculates variations per unit time with respect to the database performance data of the program. The calculated variations are displayed in descending order of size by the selected-data monitoring unit 66, as shown in FIG. 6 or 7.

The following is an example of a pseudo code for explaining a procedure in which the selected-data monitoring unit 66 monitors performance statistics data (v$sesstat) of a program (or session) level at predetermined time intervals, calculates a current cumulative value, calculates a delta value for each program by dividing the variation between the current cumulative value and the previous cumulative value by a time interval calculated using a second as a time scale, and arranges and displays the delta values for individual programs in descending order of size.

```
INPUT PARAMETER: USER_CLICK_STATISTIC#, SECOND
DECLARE
   V_NAME                STRING;
   V_VALUE_INIT( )       NUMBER ARRAY;
   V_VALUE_CURR( )       NUMBER ARRAY;
BEGIN
   /* BRING THE NAME OF PERFORMANCE STATISTICS DATA
CLICKED BY USER */
   SELECT NAME INTO V_NAME
   FROM V$STATNAME
   WHERE STATISTIC# = : USER_CLICK_STATISTIC#;
   /* SET INITIAL VALUE FOR INITIALIZATION */
```

```
 CURSOR_A AS
 SELECT SID, VALUE
 FROM V$SESSTAT A
   WHERE STATISTIC# = : USER CLICK_STATISTIC#
   CURSOR_A FETCH LOOP
     V_VALUE_INIT(CURSOR_A.SID): = CURSOR_A.VALUE;
   END FETCH LOOP;
LOOP (WHEN USER CLOSES EVENT, THEN EXIT)
 /* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER,
ARRANGE IT IN DESCENDING ORDER OF SIZE, AND DISPLAY
DELTA VALUE PER SECOND */
 SLEEP(:SECOND);
 CURSOR_B AS
 SELECT SID, VALUE, (VALUE-V_VALUE_INIT(SID)) DELTA_VAL
 FROM V$SESSTAT A
 WHERE STATISTIC# = :USER_CLICK_STATISTIC#
 ORDER BY DELTA_VL DESCENDING;
 CURSOR_B FETCH LOOP
     DISPLAY(V_NAME, CURSOR_B.SID, CURSOR_B.DELTA_VAL
                                            /: SECOND);
     /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA VALUE
PER SECOND IN NEXT TIME INTERVAL */
     V_VALUE_INIT(CURSOR_B.SID): = CURSOR_B.VALUE;
   END FETCH LOOP;
END LOOP
END
```

The following is an example of a pseudo code for explaining a procedure in which the selected-data monitoring unit 66 monitors waiting event data (v$session_event) of a program (or session) level at predetermined time intervals, calculates a current cumulative value, calculates a delta value for each program by dividing the variation between the current cumulative value and the previous cumulative value by a time interval calculated using a second as a time scale, and arranges and displays the delta values for individual programs in descending order of size.

```
INPUT PARAMETER: USER_CLICK_EVENT#, SECOND
DECLARE
  V_NAME              STRING;
  V_VALUE_INIT( )     NUMBER ARRAY;
  V_VALUE_CURR( )     NUMBER ARRAY;
BEGIN
  /* BRING THE NAME OF WAITING EVENT DATA CLICKED BY
USER */
  SELECT NAME INTO V_NAME
  FROM V$EVENT_NAME
  WHERE EVENT# = : USER_CLICK_EVENT#;
  /* SET INITIAL VALUE FOR INITIALIZATION */
  CURSOR_A AS
  SELECT SID, TIME_WAITED VALUE
  FROM V$SESSION_EVENT
    WHERE NAME# = V_NAME;
    CURSOR_A FETCH LOOP
      V_VALUE_INIT(CURSOR_A.SID): = CURSOR_A.VALUE;
    END FETCH LOOP;
LOOP (WHEN USER CLOSES EVENT, THEN EXIT)
  /* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER,
ARRANGE IT IN DESCENDING ORDER OF SIZE, AND DISPLAY
DELTA VALUE PER SECOND */
  SLEEP(:SECOND);
  CURSOR_B AS
```

```
  SELECT SID, TIME_WAITED VALUE, (TIME_WAITED –
NVL(V_VALUE_INIT(SID),0)) DELTA_VAL
  FROM V$SESSION_EVENT
  WHERE NAME = V_NAME
  ORDER BY DELTA_VL DESCENDING;
  CURSOR_B FETCH LOOP
      DISPLAY(V_NAME, CURSOR_B.SID, CURSOR_B.DELTA_VAL
                                             /: SECOND);
      /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA VALUE
PER SECOND IN NEXT TIME INTERVAL */
      V_VALUE_INIT(CURSOR_B.SID): = CURSOR_B.VALUE;
    END FETCH LOOP;
END LOOP
END
```

In the above pseudo codes, the database performance data is expressed as a delta value and thus includes information related to the near past, but in the case of the waiting event data of the program level, the current state excluding the past may be very important. Accordingly, it is necessary for the selected-data monitoring unit 66 to find programs (or sessions) in a state of the waiting event selected by the user from the waiting event data of the program (or session) level and display them in descending order of "waiting time (seconds_in_wait)". The following is an example of a pseudo code for explaining such a procedure.

```
INPUT PARAMETER: USER_CLICK_EVENT#, SECOND
DECLARE
  V_NAME              STRING;
BEGIN
  SELECT NAME INTO V_NAME
  FROM V$EVENT_NAME
  WHERE EVENT# = : USER_CLICK_EVENT#;
```

```
LOOP (WHEN USER CLOSES EVENT, THEN EXIT
  /* SINCE IT IS CURRENT INFORMATION, IT WILL BE OK JUST TO
QUERY IT AND ARRANGE AND DISPLAY IT IN DESCENDING
ORDER */
  CURSOR_A AS
  SELECT SID, SECONDS_IN_WAIT VALUE
  FROM V$SESSION_WAIT
  WHERE NAME = V_NAME AND WAIT_TIME=0;
      ORDER BY SECONDS_IN_WAIT DESCENDING;
  CURSOR_A FETCH LOOP
      DISPLAY(V_NAME, CURSOR_A.SID, CURSOR_A.VALUE);
  END FETCH LOOP;
  SLEEP(:SECOND);
END LOOP
END
```

The user can monitor database performance data of a program level with respect to a particular program by clicking a location where the particular program is displayed on a window denoted by reference numeral 80 of FIG. 6 or reference numeral 82 of FIG. 7. When the user clicks the location of the particular program, the lower level monitoring unit 68 shown in FIG. 3 operates. Accordingly, the lower level monitoring unit 68 displays database performance data of a program level with respect to a particular program selected from the programs displayed by the selected-data monitoring unit 66.

Referring to FIG. 3, the lower level monitoring unit 68 includes a lower level cumulative mode monitoring unit 70 and a lower level delta mode monitoring unit 72. The lower level cumulative mode monitoring unit 70 displays database performance data of a program level with respect to a selected program as a cumulative value, and the lower level delta mode monitoring unit 72 calculates and displays variations per unit time of database performance data of a program level with respect to a selected program.

FIG. 8 shows an example of a screen displayed by the lower level cumulative mode monitoring unit 70, and FIG. 9 shows an example of a screen displayed by the lower level delta mode monitoring unit 72.

The followings are pseudo codes for explaining examples of procedures on which the lower level delta mode monitoring unit 72 displays four kinds of information provided by database performance data of a program level with respect to a particular program. The four kinds of information are denoted by reference numerals 90, 92, 94, and 96 in FIG. 9.

<Calculation and Presentation of Delta Value per Second with respect to Cumulative Program Performance Statistics Data>

```
/* PASS OVER SESSION ID AND USER REFRESH INTERVAL AS
INPUT PARAMETERS */
INPUT PARAMETER: SID, SECOND
DECLARE
  V_NAME              STRING;
  V_VALUE_INIT( )     NUMBER ARRAY;
  V_VALUE_CURR( )     NUMBER ARRAY;
BEGIN
  /* SET INITIAL VALUE FOR INITIALIZATION */
  CURSOR_A AS
  SELECT B.STATISTIC#, B.NAME, A.VALUE
  FROM V$SESSTAT A, V$STATNAME B
  WHERE A.STATISTIC# = B.STATISTIC# AND SID =: SID;
    CURSOR_A FETCH LOOP
        V_NAME(CURSOR_A.STATISTIC#): = CURSOR_A.NAME;
        V_VALUE_INIT(CURSOR_A.STATISTIC#): =
                                        CURSOR_A.VALUE;
```

```
    END FETCH LOOP;
LOOP (WHEN PROGRAM CLOSES EVENT, THEN EXIT)
  /* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER
AND DISPLAY DELTA VALUE PER SECOND */
  SLEEP(:SECOND);
  CURSOR_B AS
  SELECT B.STATISTIC#, B.NAME, A.VALUE
  FROM V$SESSTAT A, V$STATNAME B
  WHERE A.STATISTIC# = B.STATISTIC# AND SID =: SID;
  CURSOR_B FETCH LOOP
      V_VALUE_CURR(CURSOR_B.STATISTIC#): =
                                          CURSOR_B.VALUE
      DISPLAY(V_NAME(CURSOR_B.STATISTIC#),
              (V_VALUE_CURR(CURSOR_B.STATISTIC#) –
              V_VALUE_INIT(CURSOR_B.STATISTIC#))/: SECOND);
      /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA VALUE
PER SECOND IN NEXT TIME INTERVAL */
      V_VALUE_INIT(CURSOR_B.STATISTIC#): =
                    V_VALUE_CURR(CURSOR_B.STATISTIC#);
      END FETCH LOOP;
END LOOP
END
```

<Calculation and Presentation of Delta Value per Second with respect to Cumulative Program Waiting Event Data>

```
/* PASS OVER SESSION ID AND USER REFRESH INTERVAL AS
INPUT PARAMETERS */
INPUT PARAMETER: SID, SECOND
DECLARE
  V_NAME              STRING;
  V_VALUE_INIT( )     NUMBER ARRAY;
```

```
  V_VALUE_CURR( )      NUMBER ARRAY;
BEGIN
/* SET INITIAL VALUE FOR INITIALIZATION */
  CURSOR_A AS
  SELECT B.EVENT#, B.NAME, NVL(A.TIME_WAITED,0) VALUE
  FROM V$SESSION_EVENT A, V$EVENT_NAME B
  WHERE B.NAME = A.EVENT(+) AND A.SID(+) =: SID;
  CURSOR_A FETCH LOOP
        V_NAME(CURSOR_A.EVENT#): = CURSOR_A.NAME;
        V_VALUE_INIT(CURSOR_A.EVENT#): = CURSOR_A.VALUE;
   END FETCH LOOP;
LOOP (WHEN PROGRAM CLOSES EVENT, THEN EXIT)
/* CALCULATE DELTA VALUE IN TIME INTERVAL SET BY USER
AND DISPLAY DELTA VALUE PER SECOND */
    SLEEP(:SECOND);
    CURSOR_B AS
    SELECT B.EVENT#, B.NAME, NVL(A.TIME_WAITED,0) VALUE
    FROM V$SESSION_EVENT A, V$EVENT_NAME B
    WHERE B.NAME = A.EVENT AND A.SID =: SID;
    CURSOR_B FETCH LOOP
        V_VALUE_CURR(CURSOR_B.EVENT#): =
                                      CURSOR_B.VALUE;
        DISPLAY(V_NAME(CURSOR_B.EVENT#),
                    (V_VALUE_CURR(CURSOR_B.EVENT#) –
                V_VALUE_INIT(CURSOR_B.EVENT#))/: SECOND);
        /* INITIALIZE INITIAL VALUE TO CALCULATE DELTA VALUE
PER SECOND IN NEXT TIME INTERVAL */
        V_VALUE_INIT(CURSOR_B.EVENT#): =
                  V_VALUE_CURR(CURSOR_B.EVENT#);
    END FETCH LOOP;
```

```
END LOOP
END
```

<Presentation of Value of Current Program Waiting Event Data>

```
SELECT * FROM V$SESSION_WAIT WHERE SID =: SID;
```

<Presentation of Current SQL>

```
SELECT ST.SQL_TEXT
FROM V$SESSION S, V$SQLTEXT_WITH_NEWLINES ST
WHERE S.SQL_HASH_VALUE = ST.HASH_VALUE
  AND S.SQL_ADDRESS = ST.ADDRESS
  AND S.SID =: SID
ORDER BY ST.HASH_VALUE, ST.PIECE;
```

The lower level monitoring unit 68 has a function of simultaneously showing two connected sessions in two related databases. In other words, the lower level monitoring unit 68 simultaneously provides windows which display database performance data of a program level for related sessions, respectively, in two different databases (for example, two sessions connected through a dblink in the case of an Oracle database) on a single screen. In addition, the lower level monitoring unit 68 reflashes displayed values at predetermined time intervals and preferably provides a function of recording data before reflash in a predetermined log file at the user's request and replaying the data recorded in the log file at the user's request.

FIG. 10 shows an example of a screen on which the database performance monitoring apparatus 60 according to the first embodiment of the present invention displays database performance data of an SQL level. Accordingly, a user can recognize performance data of SQL performed by a program in a cumulative mode and monitor performance data of SQL which is being performed in terms of a current and delta mode.

Hereinafter, a method of monitoring database performance according to a first embodiment of the present invention will be described in detail with reference to FIG. 11.

Variations of database performance data of a whole database level per unit time are calculated and displayed in step S1100. Here, it is possible to calculate variations per unit time with respect to the predetermined number of database performance data, generate time-transition graphs of the predetermined number of database performance data, and display all of the time-transition graphs generated for the predetermined number of database performance data on a single screen.

Thereafter, in step S1110 a user selects one from among the database performance data displayed in step S1100. Here, when particular performance statistic data or particular waiting event data has a value exceeding a predetermined reference value, the user can select the data as problem data.

If the problem data is selected, in step S1120 database performance data of different programs corresponding to the selected database performance data is accessed, variations per unit time are calculated, and the variations calculated with respect to the different programs are arranged and displayed in descending order of size.

Next, in step S1130 the user selects one of the programs displayed in step S1120. Here, the user can select a program that is determined as causing a problem.

If the problem program is selected, in step S1140 database performance data of a program level with respect to the selected program is displayed. Here, a procedure of displaying cumulative values of the database performance data of a program level with respect to the selected program can alternate with a procedure of calculating and displaying variations per unit time of the database performance data of a program level with respect to the selected program according to the user's operation. In other words, the user can convert a mode between a cumulative mode and a delta mode to apprehend database performance.

Thereafter, in step S1150 the user can operate to make database performance data of an SQL level displayed, so the user can recognize the performance data of SQL which has been performed by the program in the cumulative mode and can monitor the performance data of SQL which is being performed in a current and delta mode.

Referring to FIG. 12, a database performance monitoring apparatus 120 according to a second embodiment of the present invention includes an upper level monitoring unit 122, a selected-data monitoring unit 126, and a lower level monitoring unit 126 like the first embodiment shown in FIG. 3.

The lower level monitoring unit 128 includes a lower level cumulative mode monitor 130 and a lower level delta mode monitor 132. Accordingly, descriptions of the same elements of the second embodiment shown in FIG. 12 as those of the first embodiment shown in FIG. 3 will be omitted.

Referring to FIG. 12, the upper level monitoring unit 122 includes a whole database graphic monitor 124. The whole database graphic monitor 124 calculates variations per unit time in relation to all databases installed in the information processing system with respect to the predetermined number of database performance data among database performance data of a whole database level and generates time-transition graphs for the predetermined number of database performance data and databases. All of the time-transition graphs for the predetermined number of database performance data and databases which are generated by the whole database graphic monitor 124 are displayed on a single screen.

Referring to FIG. 13, six databases can be simultaneously monitored on a single screen, and performance statistics data, waiting event data, and an SGA status are displayed with respect to each database. Time-transition graphs for each database with respect to pre-selected nine kinds of data are displayed on a performance statistics data display area 130. A time-transition graph of each database with respect to waiting event data is displayed on a waiting event data graphic display area 132. Details about a database ("DEV5" in FIG. 13) having a largest variation of a waiting event are displayed by default on a waiting event data text display area 134 and an SGA status display area 136.

Here, a user can view details about the waiting event of another database by clicking the graph of the database on the waiting event data graphic display area 132. FIG. 14 shows a waiting event data text display area 144 and an SGA status display area 146 on which the details of FIG. 13 are changed when the user selects the database "PPP" on the waiting event data graphic display area 132 of FIG. 13.

When the user clicks the item "Kbytes thru DBLINK" of the database "SDTEST" in FIG. 13 in order to track the sessions of the item in a top-down manner, the selected-data monitoring unit 126 displays the sessions, as illustrated in FIG. 15.

A method of monitoring database performance according to a second embodiment of the present invention is almost the same as that according to the first embodiment shown in FIG. 11 with the exception that the step S1100 of FIG. 11 is replaced with a step of monitoring all databases.

In the step of monitoring all databases, with respect to each of the predetermined number of database performance data among database performance data of a whole database level, variations per unit time in relation to all databases which are installed in the information processing system 30 are calculated, all time-transition graphs for database performance data and databases are generated, and all of the generated time-transition graphs for database performance data and databases are displayed on a single screen.

Since the same steps as those after the step S1100 in the first embodiment shown in FIG. 11 are performed in the second embodiment, descriptions thereof will be omitted.

While this invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the art that various changes may be made therein without departing from the scope of the invention. Therefore, the above-described embodiments will be considered not in restrictive sense but in descriptive sense only. The scope of the invention will be defined not by the above description but by the appended claims, and it will be construed that all differences made within the scope defined by the claims are included in the present invention.

Industrial Applicability

According to the present invention, first, taking notice of the relationships among database performance data, performance statistics data and waiting event data which are primarily detected in a broad range (i.e., at a whole database level) are generally apprehended to check the performance of each database, and there is provided a function of narrowing the broad range of the problem of a database performance degradation.

Second, database performance data generated by a DBMS is processed such that database performance at present time and at a past time point nearest to the present time, which a tuner is mainly interested in, can be apprehend.

Third, a plurality of databases can be effectively monitored on a single screen.

Accordingly, the present invention allows a tuner to quickly and easily analyze problems, which cause the performance of databases to deteriorate, using the above-described functions.

What is claimed is:

1. An apparatus for monitoring database performance using a plurality of database performance data which are generated and classified into a whole database level, a program level, and a Structured Query Language (SQL) level by a database management system installed in an information processing system, the apparatus comprising:

an upper level monitoring unit for calculating and displaying variations per unit time of the database performance data of the whole database level;

a selected-data monitoring unit for accessing database performance data of each program corresponding to database performance data selected from among the database performance data, which are displayed by the upper level monitoring unit, calculating variations per unit time of the accessed database performance data, and displaying the calculated variations per unit time in descending order of size; and

a lower level monitoring unit for displaying database performance data of the program level with respect to a program selected from among the programs displayed by the selected-data monitoring unit.

2. The apparatus of claim 1, wherein the upper level monitoring unit comprises an upper level graphic monitoring unit for calculating variations per unit time with respect to a predetermined number of database performance data among the database performance data of the whole database level so as to generate time-transition graphs for the predetermined number of database performance data and displaying all of the generated time-transition graphs on a single screen.

3. The apparatus of claim 1, wherein the upper level

monitoring unit comprises a whole database graphic monitoring unit for calculating variations per unit time in with respect to a predetermined number of database performance data among the database performance data of the whole database level and in relation to all databases installed in the information processing system so as to generate time-transition graphs for the predetermined number of database performance data and databases and displaying all of the generated time-transition graphs on a single screen.

4. The apparatus of claim 1, wherein the lower level monitoring unit comprises:

a lower level cumulative mode monitoring unit for displaying cumulative values of the database performance data of the program level; and

a lower level delta mode monitoring unit for calculating and displaying variations per unit time of the database performance data of the program level.

5. The apparatus of claim 4, wherein the lower level monitoring unit provides windows on which database performance data of the program level with respect to corresponding sessions between two related databases are displayed, respectively, on a single screen.

6. The apparatus of claim 4, wherein the lower level monitoring unit records data before reflash in a predetermined log file at the write request of a user and replays the data recorded in the log file at the replay request of the user.

7. The apparatus of any one of claims 1 through 6, wherein the database performance data comprises a plurality of performance

statistics data which indicate the degree of use of each resource provided in the information processing system and a plurality of waiting event data which indicate the amount of waiting time according to competition for the resource.

8. A method of monitoring database performance using a plurality of database performance data which are generated and classified into a whole database level, a program level, and a Structured Query Language (SQL) level by a database management system installed in an information processing system, the method comprising:

an upper level monitoring step of calculating and displaying variations per unit time of the database performance data of the whole database level;

a performance data selecting step in which a user selecting one from among the database performance data which are displayed in the upper level monitoring step;

a selected-data monitoring step of accessing database performance data of each program corresponding to the selected database performance data, calculating variations per unit time of the accessed database performance data, and displaying the calculated variations per unit time in descending order of size;

a program selecting step in which the user selects one among the programs displayed in the selected-data monitoring step; and

a lower level monitoring step of displaying database performance data of the program level with respect to selected program.

9. The method of claim 8, wherein the upper level monitoring step comprises calculating variations per unit time with respect to a predetermined number of database performance data among the database performance data of the whole database level so as to

generate time-transition graphs for the predetermined number of database performance data and displaying all of the generated time-transition graphs on a single screen.

10. The method of claim 8, wherein the upper level monitoring step comprises calculating variations per unit time in with respect to a predetermined number of database performance data among the database performance data of the whole database level and in relation to all databases installed in the information processing system so as to generate time-transition graphs for the predetermined number of database performance data and databases and displaying all of the generated time-transition graphs on a single screen.

11. The method of claim 8, wherein the lower level monitoring step comprises:

a lower level cumulative mode monitoring step of displaying cumulative values of the database performance data of the program level; and

a lower level delta mode monitoring step of calculating and displaying variations per unit time of the database performance data of the program level, and

conversion between the lower level cumulative mode monitoring step and the lower level delta mode monitoring step is performed by the user's operation.

12. The method of claim 11, wherein in the lower level monitoring step, there are provided windows on which database performance data of the program level with respect to corresponding sessions between two related databases are displayed, respectively, on a single screen.

13. The method of claim 11, wherein in the lower level monitoring step, data before reflash is recorded in a predetermined log file at the write request of the user and is replayed at the replay request of the user.

14. The method of any one of claims 8 through 13, wherein the database performance data comprises a plurality of performance statistics data which indicate the degree of use of each resource provided in the information processing system and a plurality of waiting event data which indicate the amount of waiting time according to competition for the resource.

FIG.1

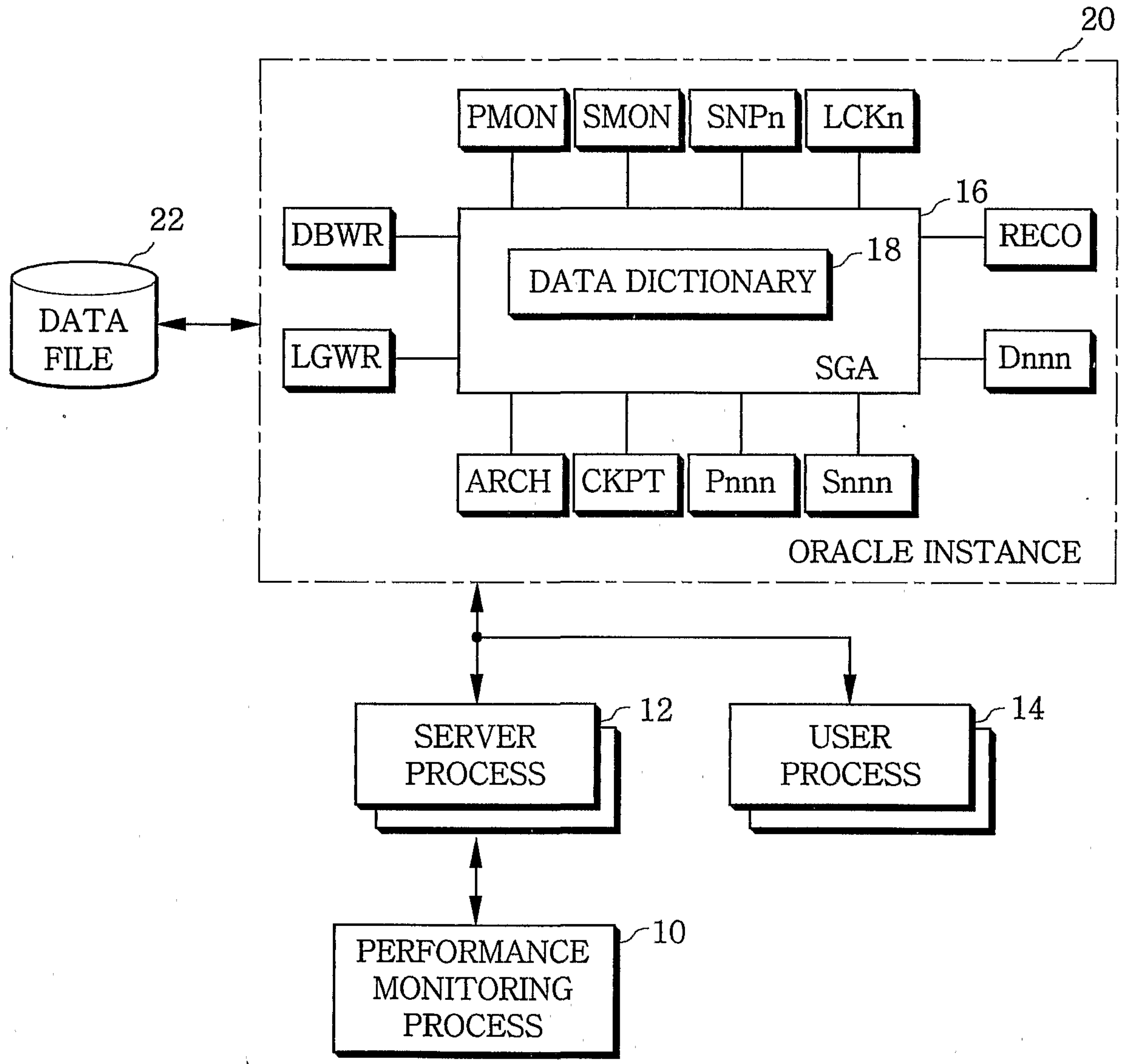

20
PMON
SMON
SNPn
LCKn
16
22
DBWR
RECO
DATA DICTIONARY
18
DATA FILE
LGWR
SGA
Dnnn
ARCH
CKPT
Pnnn
Snnn
ORACLE INSTANCE
SERVER PROCESS
12
USER PROCESS
14
PERFORMANCE MONITORING PROCESS
10

FIG.2

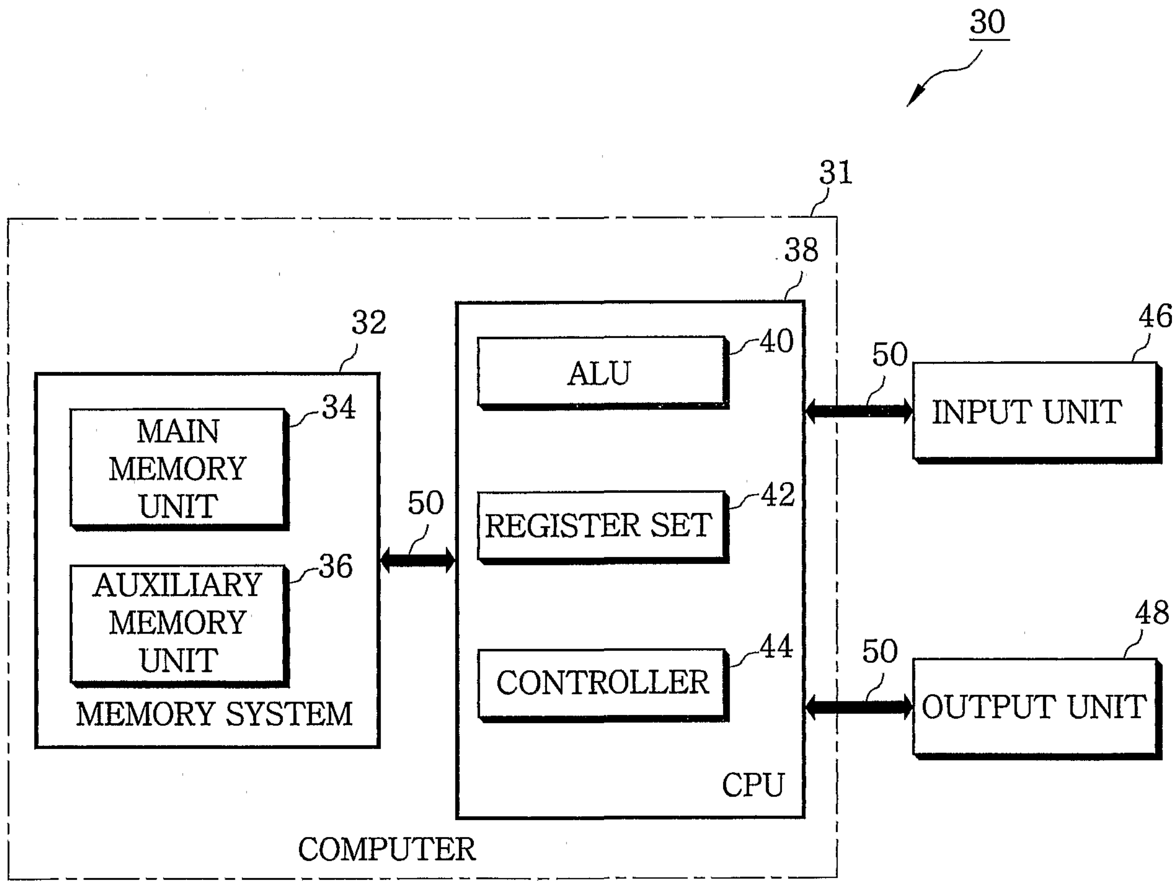

30
31
38
32
MAIN MEMORY UNIT
34
AUXILIARY MEMORY UNIT
36
MEMORY SYSTEM
50
ALU
40
REGISTER SET
42
CONTROLLER
44
CPU
COMPUTER
50
INPUT UNIT
46
50
OUTPUT UNIT
48

FIG.3

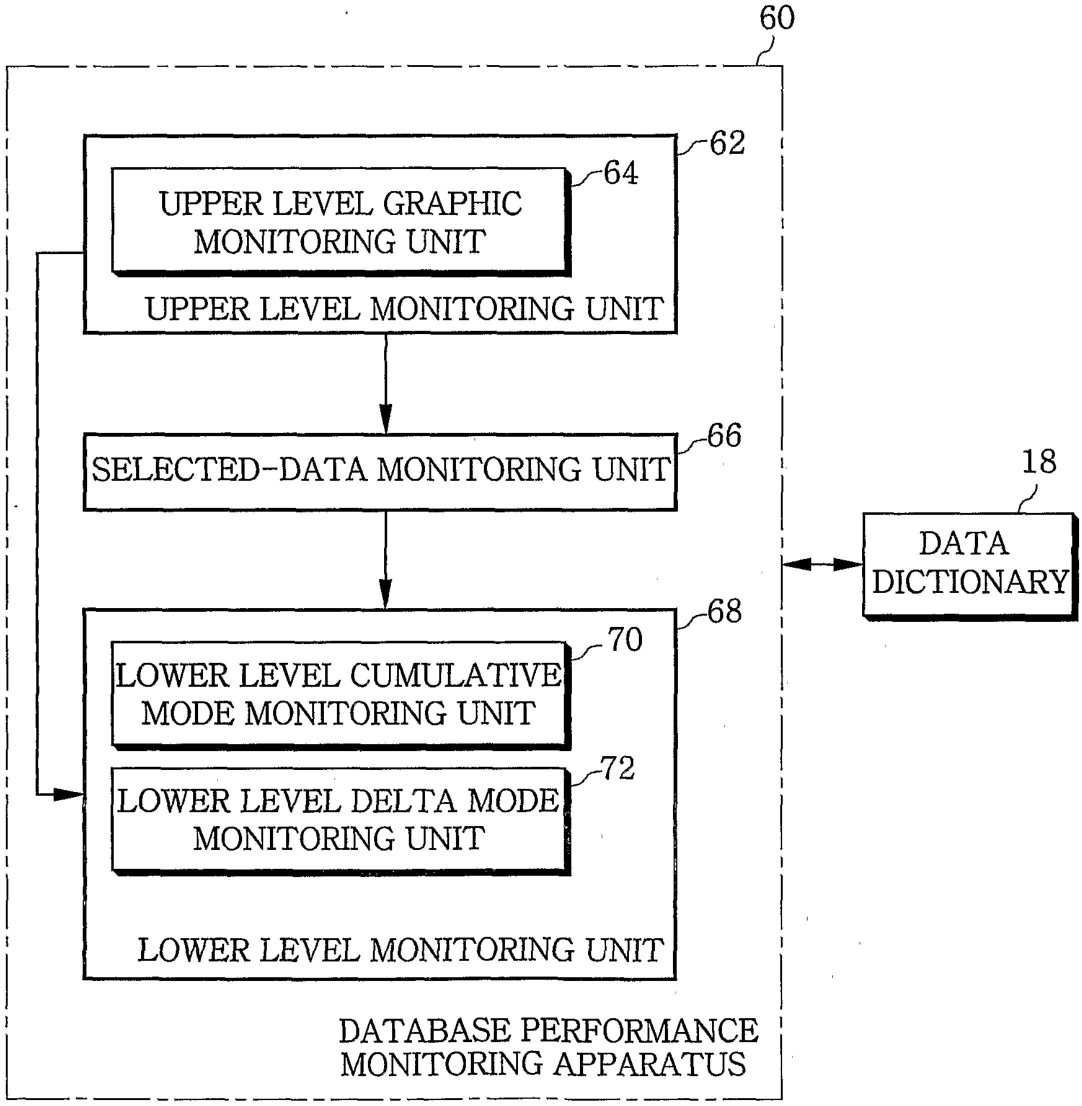

60
62
UPPER LEVEL GRAPHIC MONITORING UNIT
64
UPPER LEVEL MONITORING UNIT
SELECTED-DATA MONITORING UNIT
66
18
DATA DICTIONARY
68
LOWER LEVEL CUMULATIVE MODE MONITORING UNIT
70
LOWER LEVEL DELTA MODE MONITORING UNIT
72
LOWER LEVEL MONITORING UNIT
DATABASE PERFORMANCE MONITORING APPARATUS

FIG.4

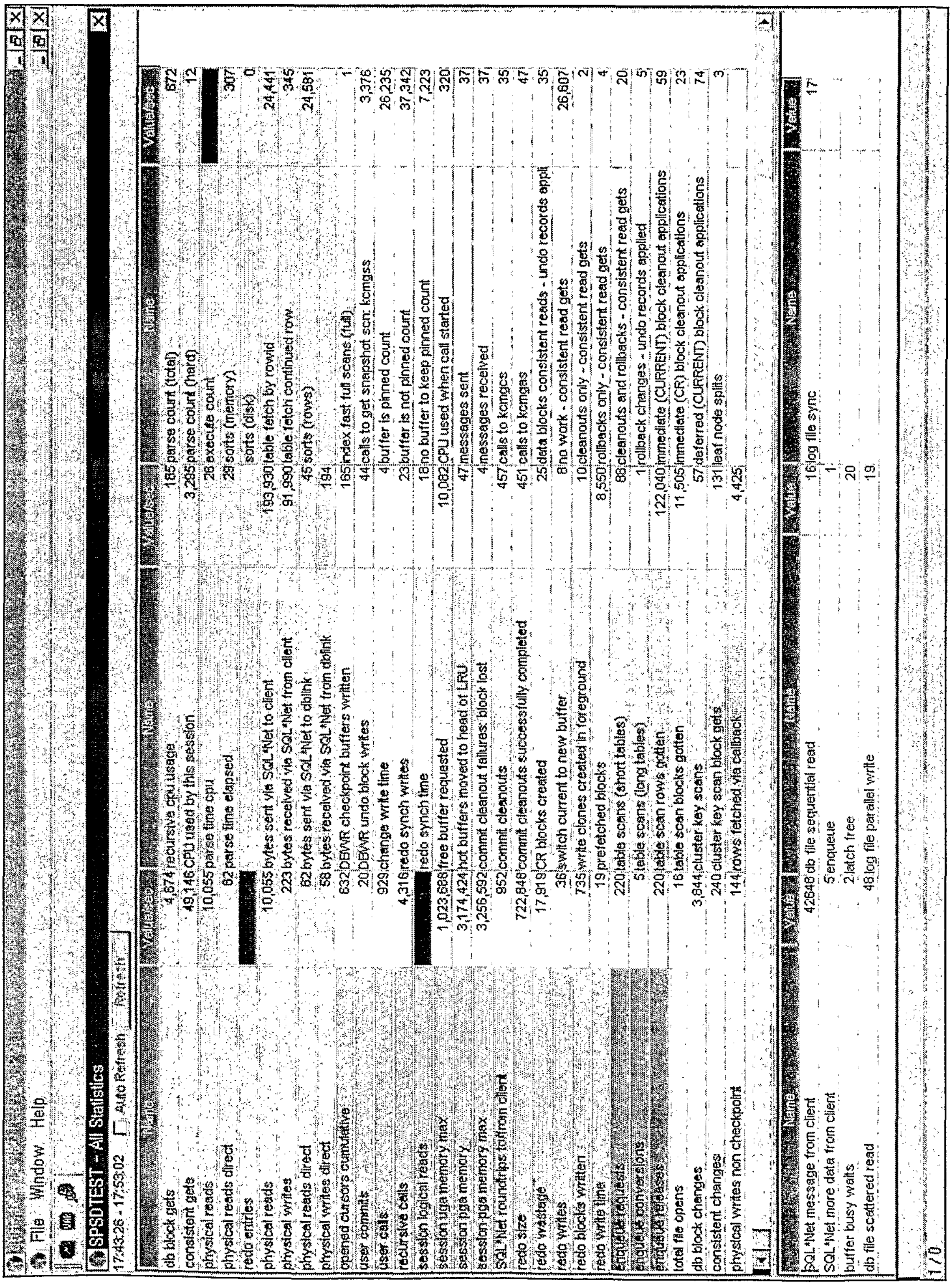
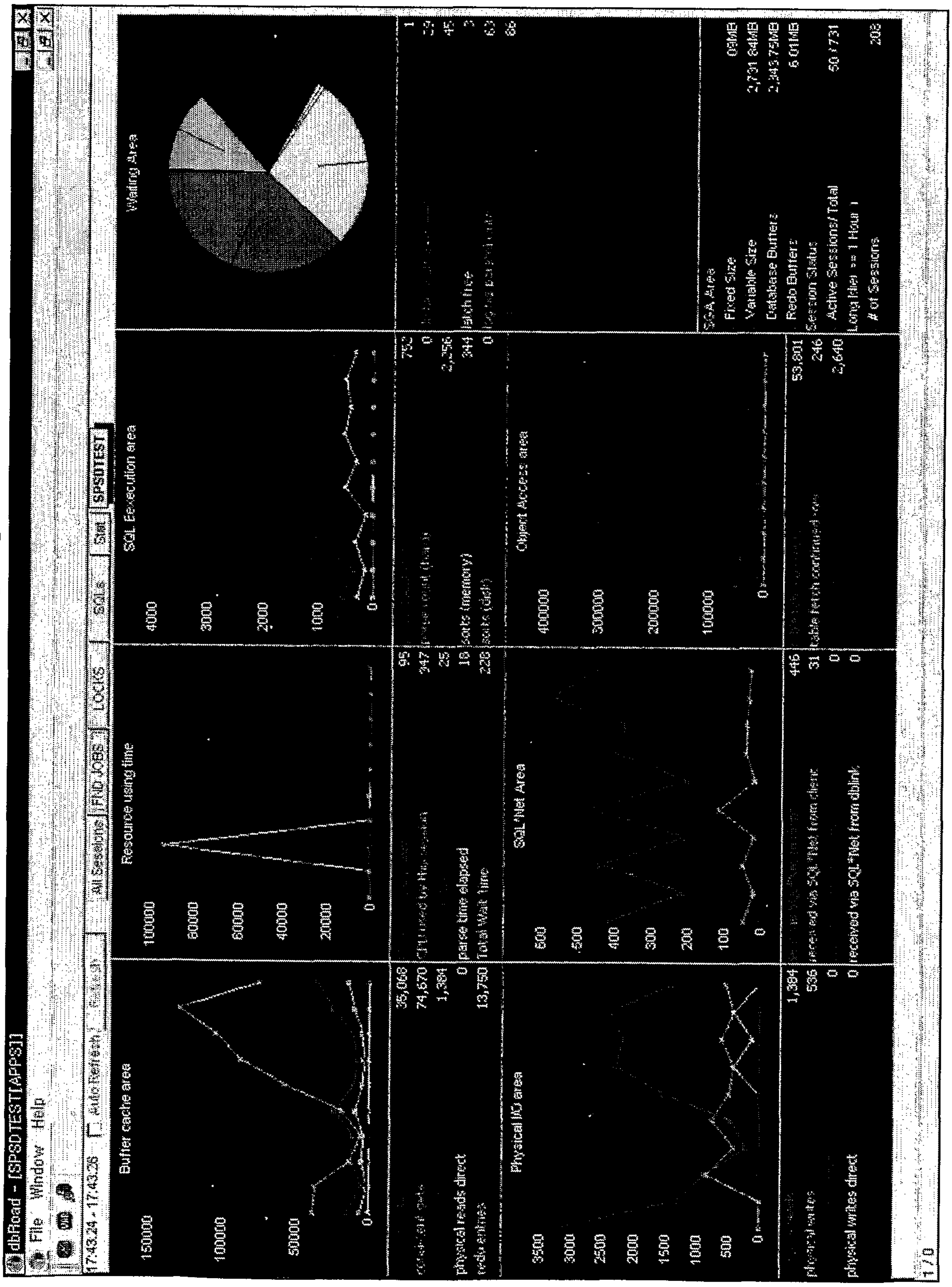

File Window Help
SPSDTEST - All Statistics
17:43:26 - 17:53:02
Auto Refresh
Refresh
Name
Value/sec
Value
db block gets
consistent gets
physical reads
physical reads direct
redo entries
physical writes
physical reads direct
physical writes direct
opened cursors cumulative
user commits
user calls
recursive calls
session logical reads
session uga memory max
session pga memory
session pga memory max
SQL*Net roundtrips to/from client
redo size
redo wastage
redo writes
redo blocks written
redo write time
enqueue requests
enqueue conversions
enqueue releases
total file opens
db block changes
consistent changes
physical writes non checkpoint
SQL*Net message from client
SQL*Net more data from client
buffer busy waits
db file scattered read
db file sequential read
enqueue
latch free
log file parallel write
log file sync

FIG.5



dbRoad - [SPSDTEST[APPS]]
File Window Help
Auto Refresh
All Sessions
LOCKS
SQLs
Stat
SPSDTEST
Buffer cache area
Resource using time
SQL Execution area
Waiting Area
Physical I/O area
SQL*Net Area
Object Access area
physical reads direct
redo entries
parse time elapsed
Total Wait time
sorts (memory)
sorts (disk)
latch free
physical writes
physical writes direct
received via SQL*Net from dblink
table fetch continued row
SGA Area
Fixed Size
Variable Size
Database Buffers
Redo Buffers
Session Status
Active Sessions/ Total
# of Sessions

FIG.6

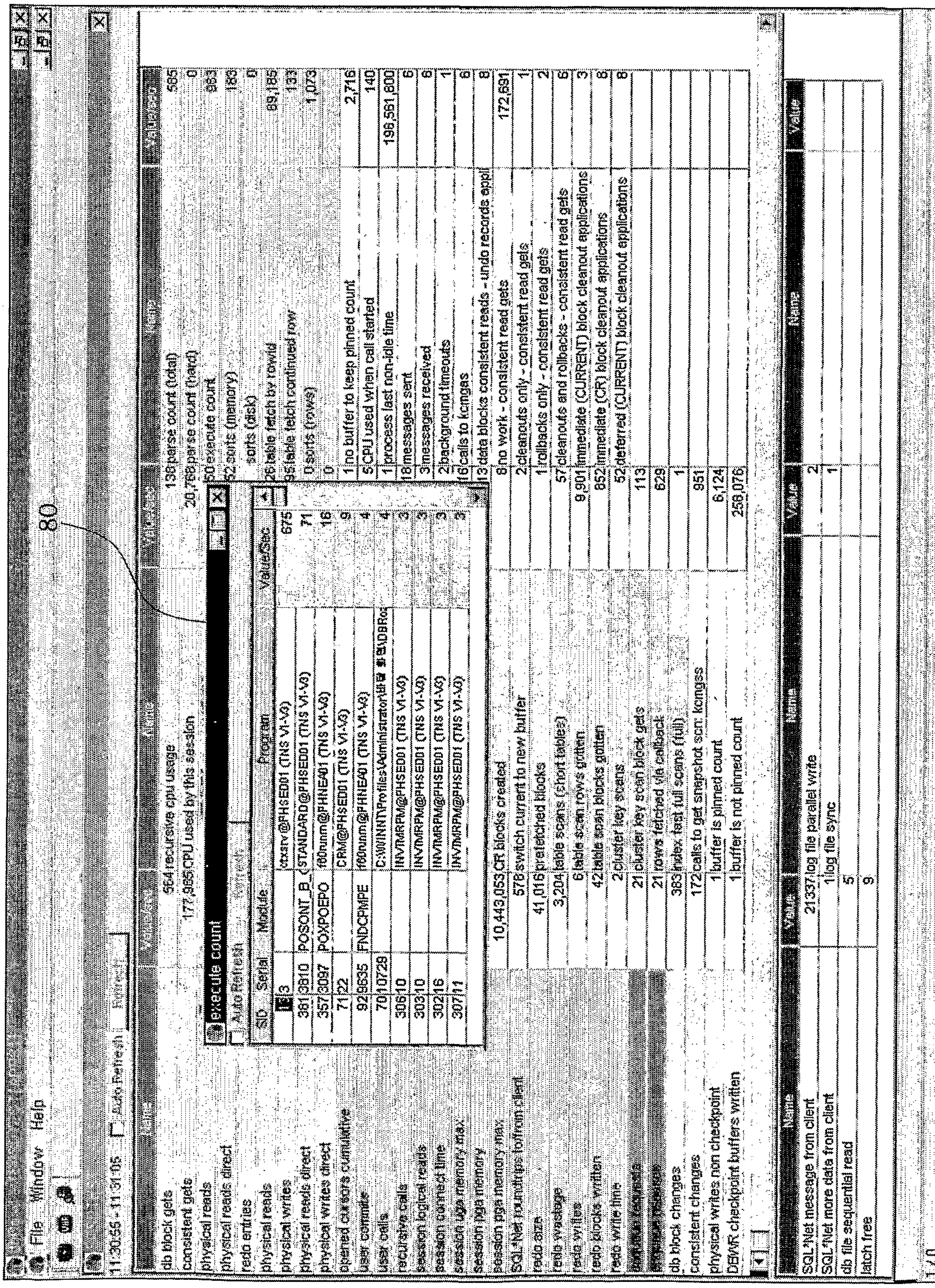


FIG.7

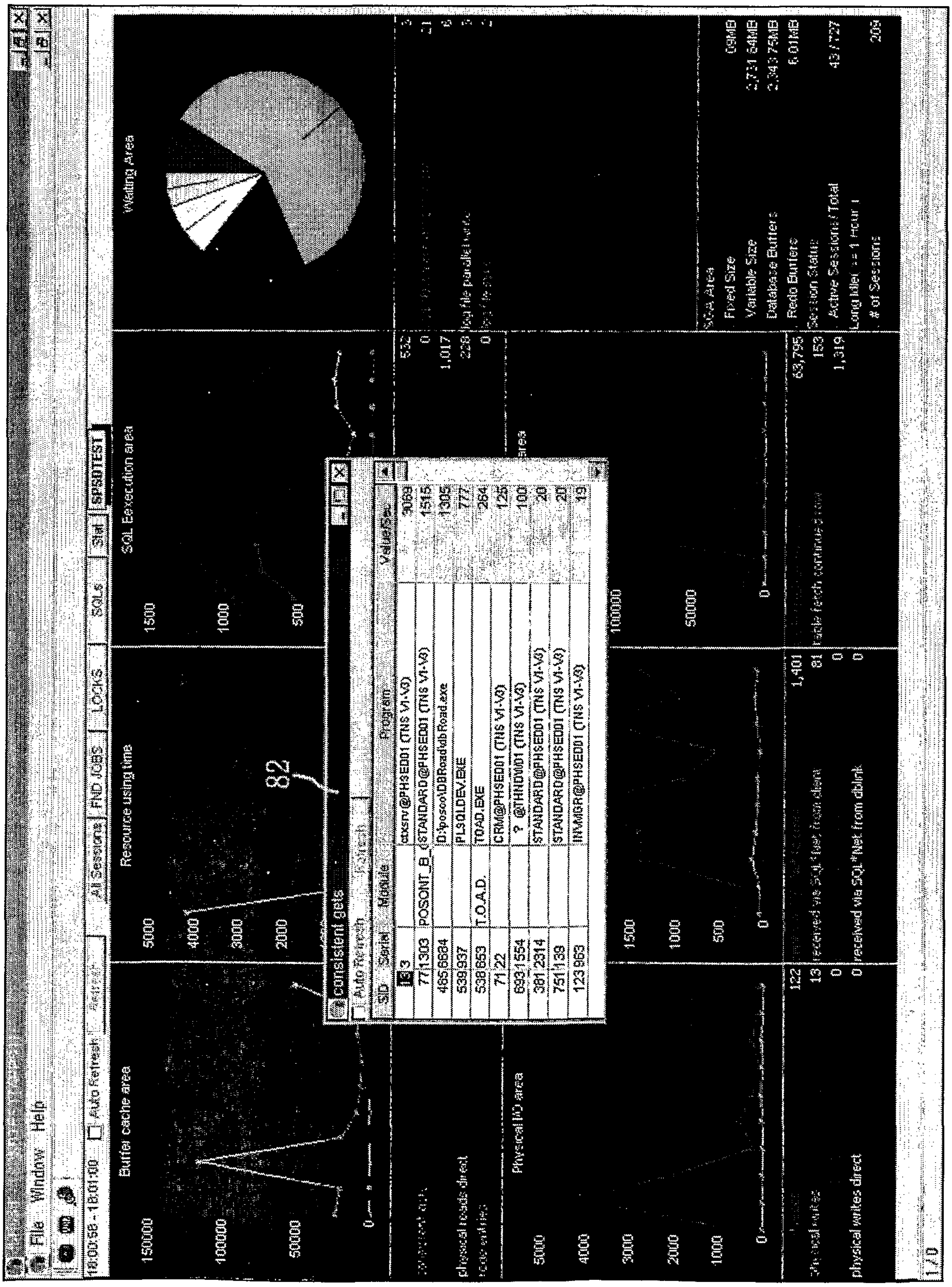

File Window Help
Auto Refresh
All Sessions
FND JOBS
LOCKS
SQLs
Stat
SPSOTEST
Buffer cache area
Resource using time
SQL Eexecution area
Waiting Area
Physical I/O area
consistent gets
82
SID
Serial
Module
Program
Value/Sec
SGA Area
Fixed Size
Variable Size
Database Buffers
Redo Buffers
Session Status
Active Sessions/Total
# of Sessions
physical reads direct
physical writes direct

FIG.8

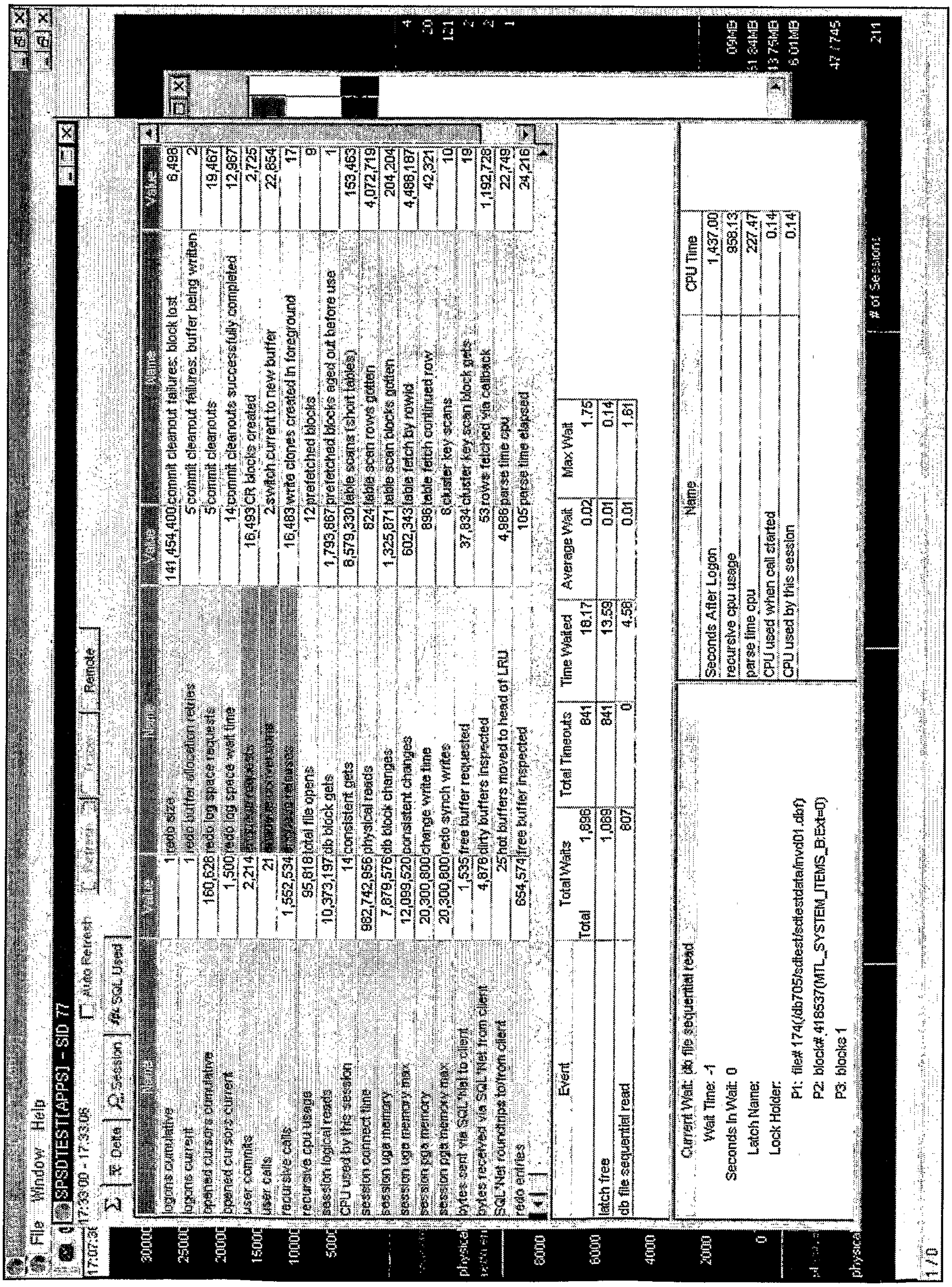

File Window Help
SPSDTEST[APPS] - SID 77
17:33:00 - 17:33:06
Auto Refresh
Delta
Session
SQL Used
Remote
Name
Value
logons cumulative
logons current
opened cursors cumulative
opened cursors current
user commits
user calls
recursive calls
recursive cpu usage
session logical reads
CPU used by this session
session connect time
session uga memory
session uga memory max
session pga memory
session pga memory max
bytes sent via SQL*Net to client
bytes received via SQL*Net from client
SQL*Net roundtrips to/from client
redo entries
redo size
redo buffer allocation retries
redo log space requests
redo log space wait time
total file opens
db block gets
consistent gets
physical reads
db block changes
consistent changes
change write time
redo synch writes
free buffer requested
dirty buffers inspected
hot buffers moved to head of LRU
free buffer inspected
commit cleanout failures: block lost
commit cleanout failures: buffer being written
commit cleanouts
commit cleanouts successfully completed
CR blocks created
switch current to new buffer
write clones created in foreground
prefetched blocks
prefetched blocks aged out before use
table scans (short tables)
table scan rows gotten
table scan blocks gotten
table fetch by rowid
table fetch continued row
cluster key scans
cluster key scan block gets
rows fetched via callback
parse time cpu
parse time elapsed
Event
Total Waits
Total Timeouts
Time Waited
Average Wait
Max Wait
Total
latch free
db file sequential read
Current Wait: db file sequential read
Wait Time: -1
Seconds In Wait: 0
Latch Name:
Lock Holder:
P1: file# 174(/db705/sdtest/sdtestdata/invd01.dbf)
P2: block# 418537(MTL_SYSTEM_ITEMS_B:Ext=0)
P3: blocks 1
CPU Time
Seconds After Logon
recursive cpu usage
parse time cpu
CPU used when call started
CPU used by this session
# of Sessions

FIG.9

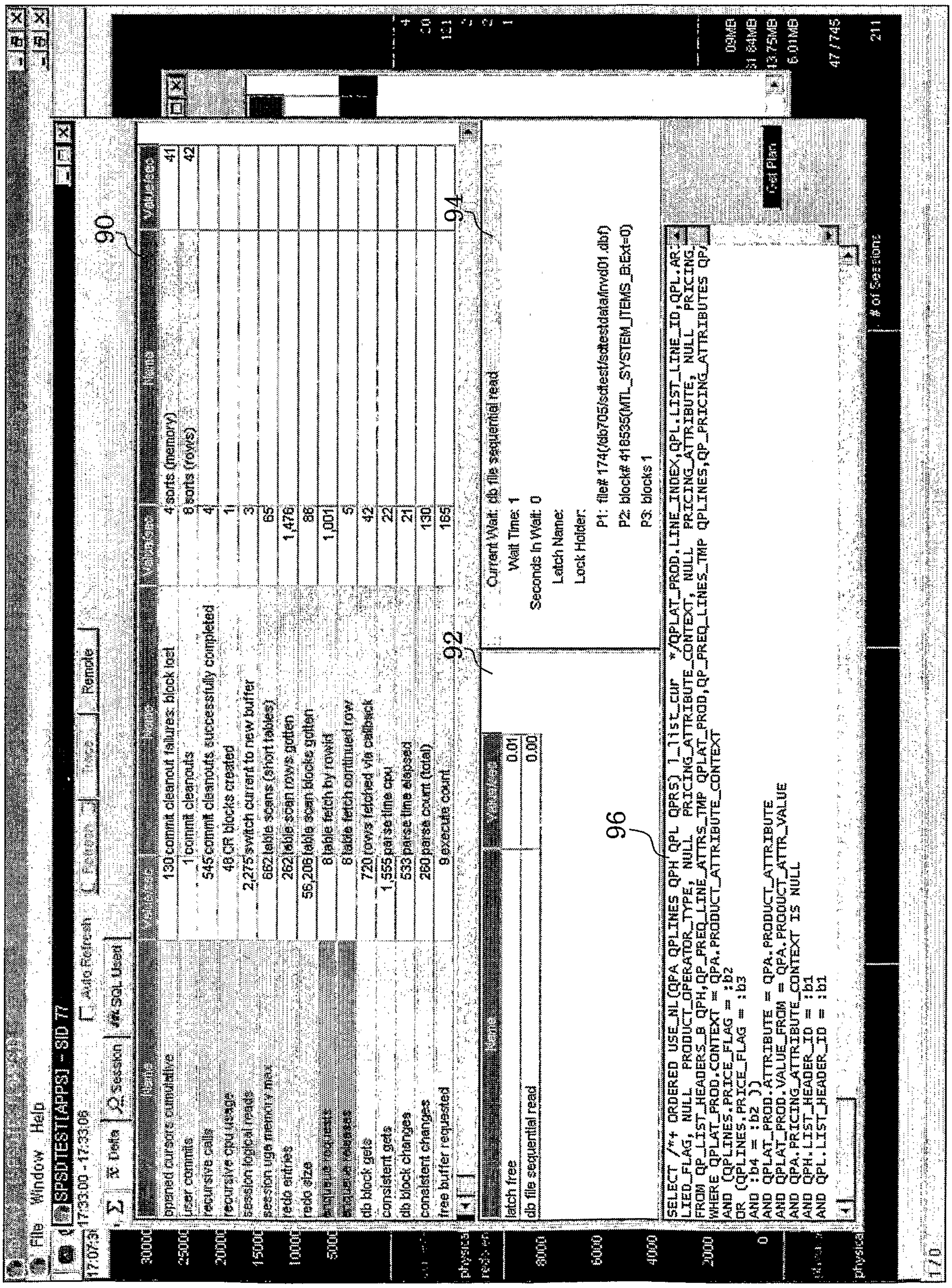


FIG.10

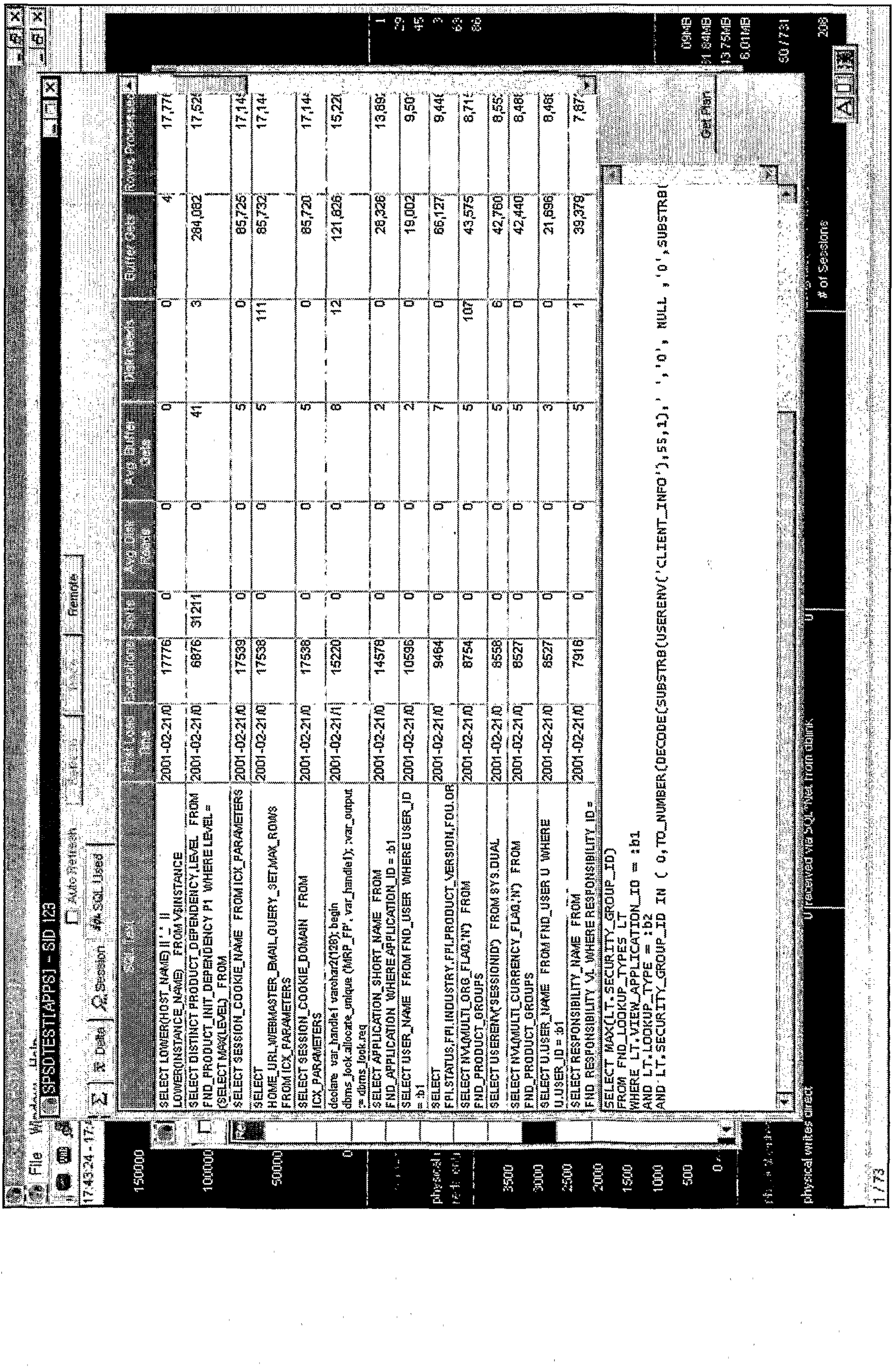


FIG.11

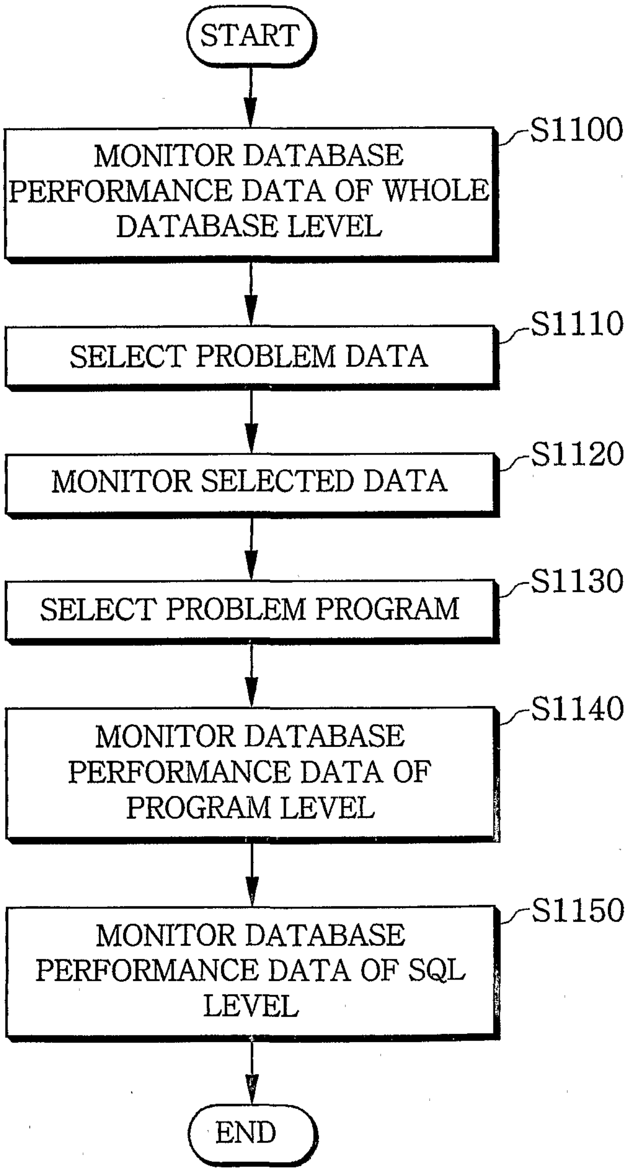
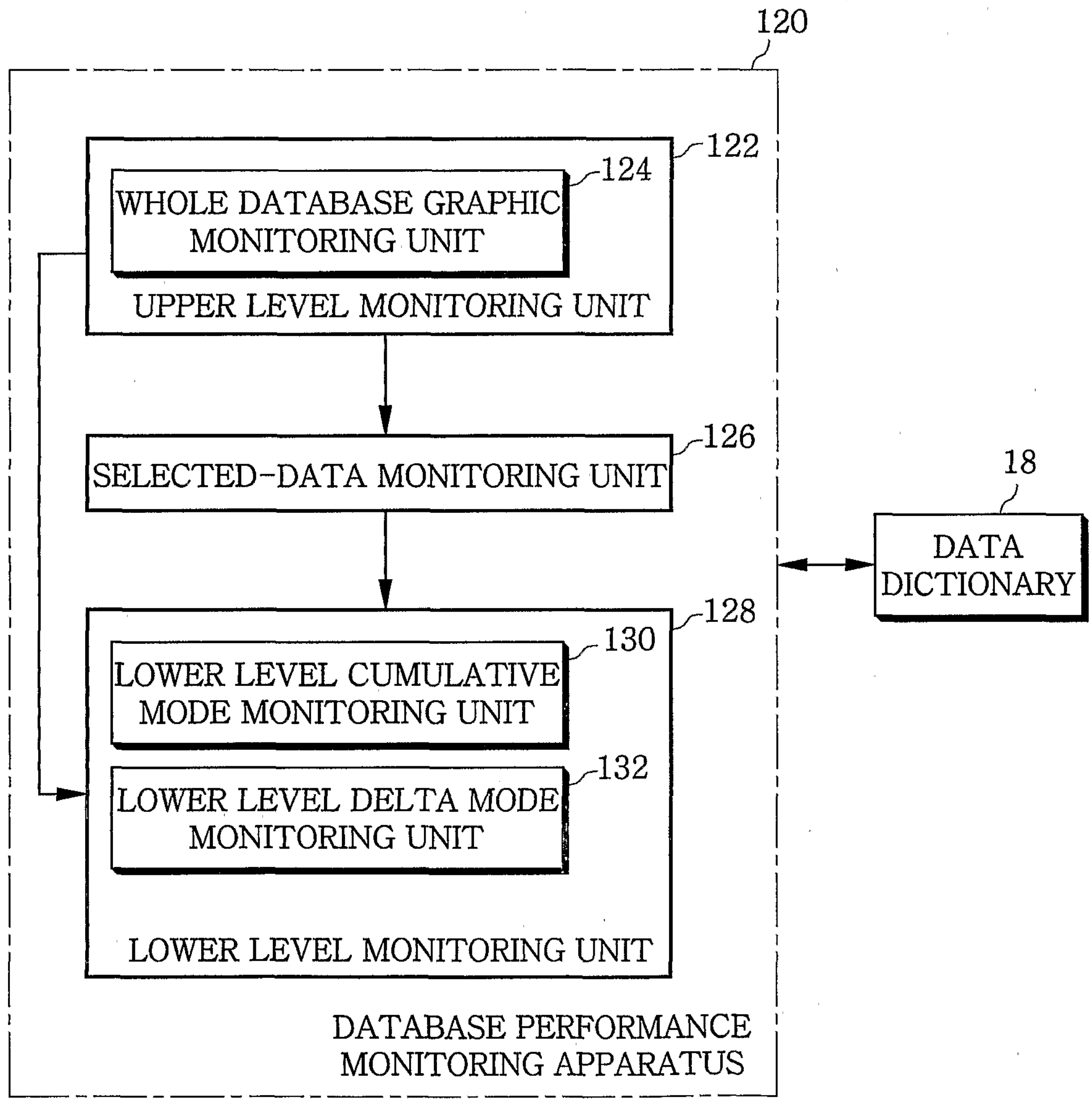

START
MONITOR DATABASE PERFORMANCE DATA OF WHOLE DATABASE LEVEL
S1100
SELECT PROBLEM DATA
S1110
MONITOR SELECTED DATA
S1120
SELECT PROBLEM PROGRAM
S1130
MONITOR DATABASE PERFORMANCE DATA OF PROGRAM LEVEL
S1140
MONITOR DATABASE PERFORMANCE DATA OF SQL LEVEL
S1150
END

FIG.12


120
122
124
WHOLE DATABASE GRAPHIC MONITORING UNIT
UPPER LEVEL MONITORING UNIT
126
SELECTED-DATA MONITORING UNIT
18
DATA DICTIONARY
128
130
LOWER LEVEL CUMULATIVE MODE MONITORING UNIT
132
LOWER LEVEL DELTA MODE MONITORING UNIT
LOWER LEVEL MONITORING UNIT
DATABASE PERFORMANCE MONITORING APPARATUS

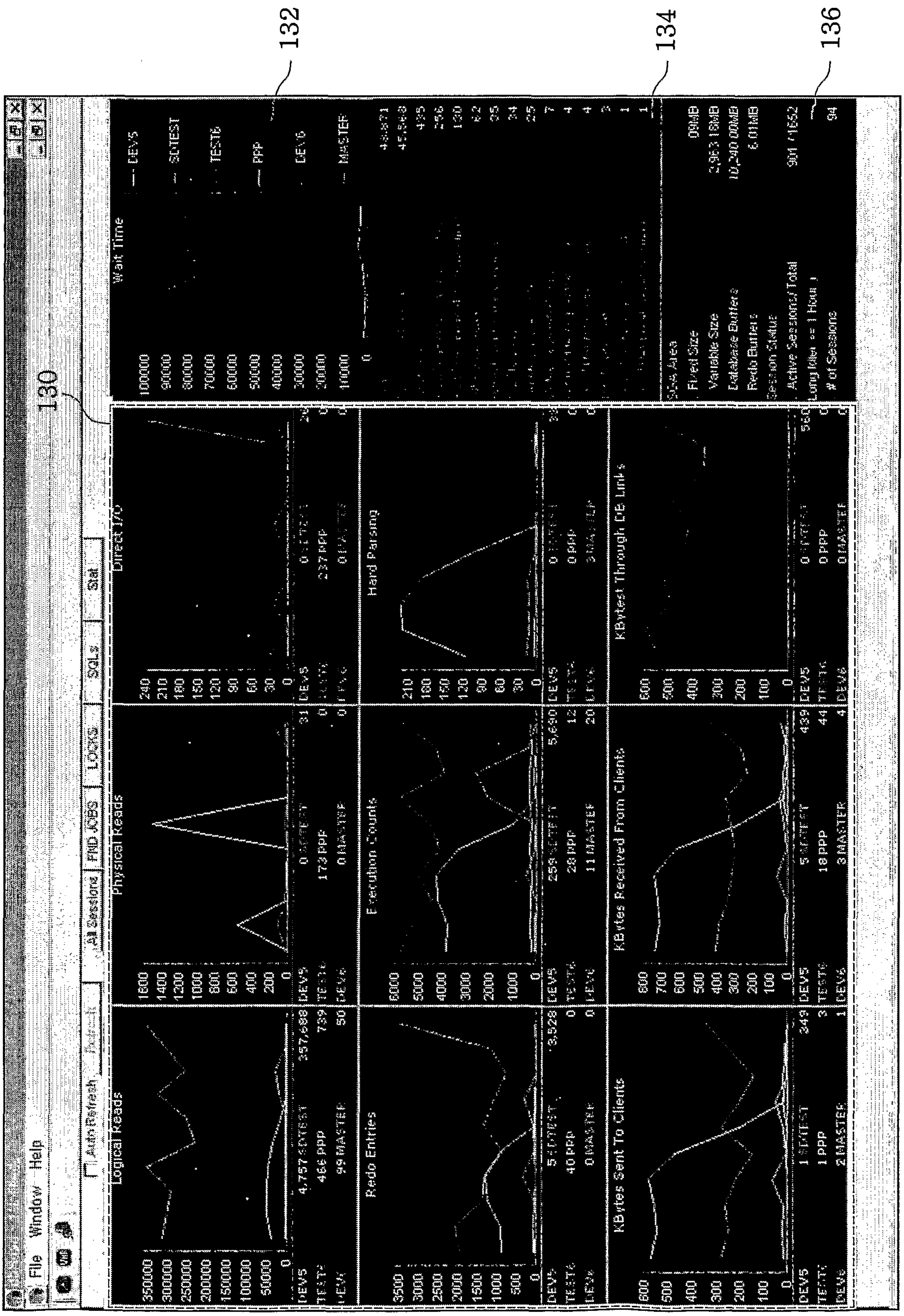


FIG.13

FIG.14

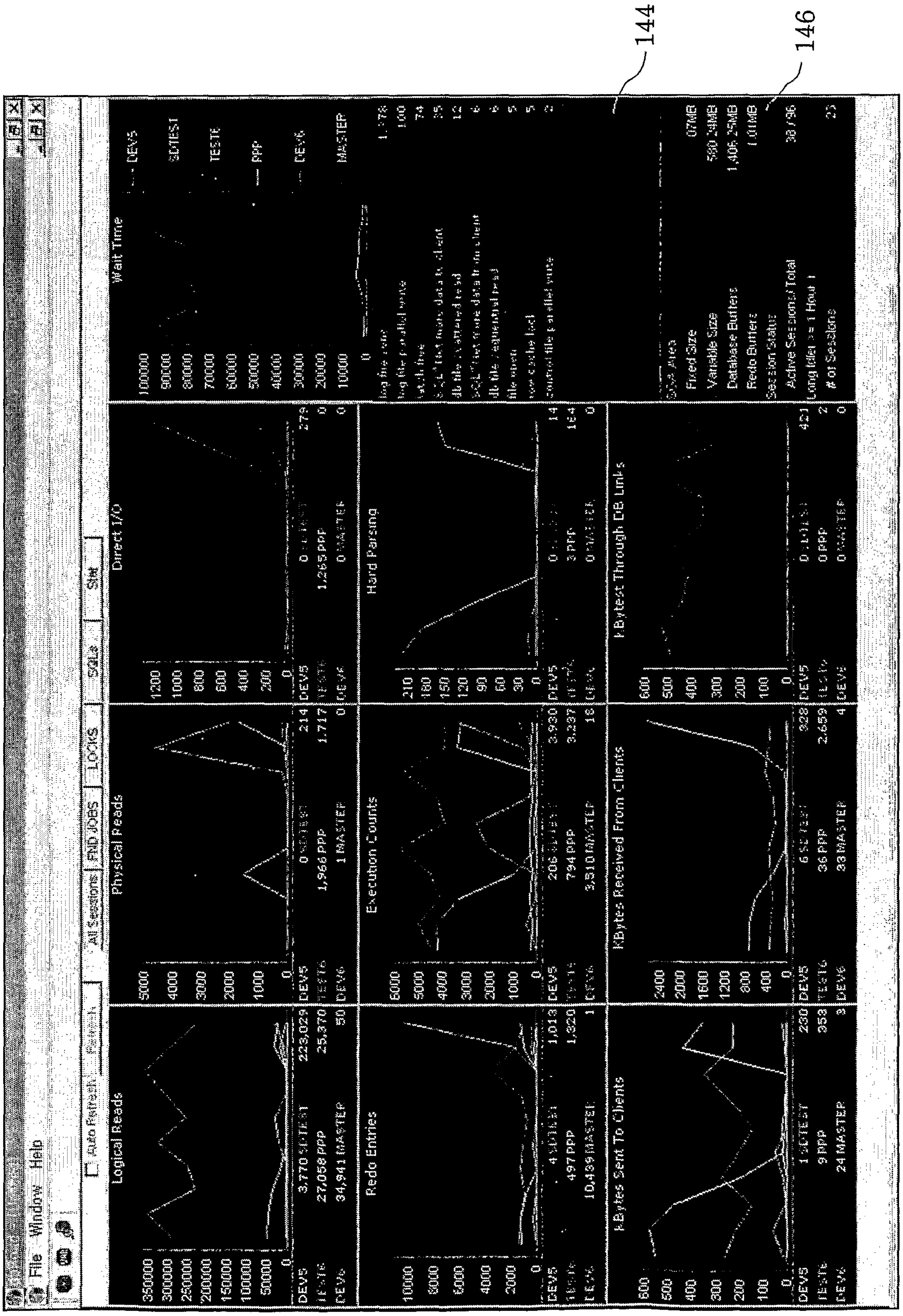
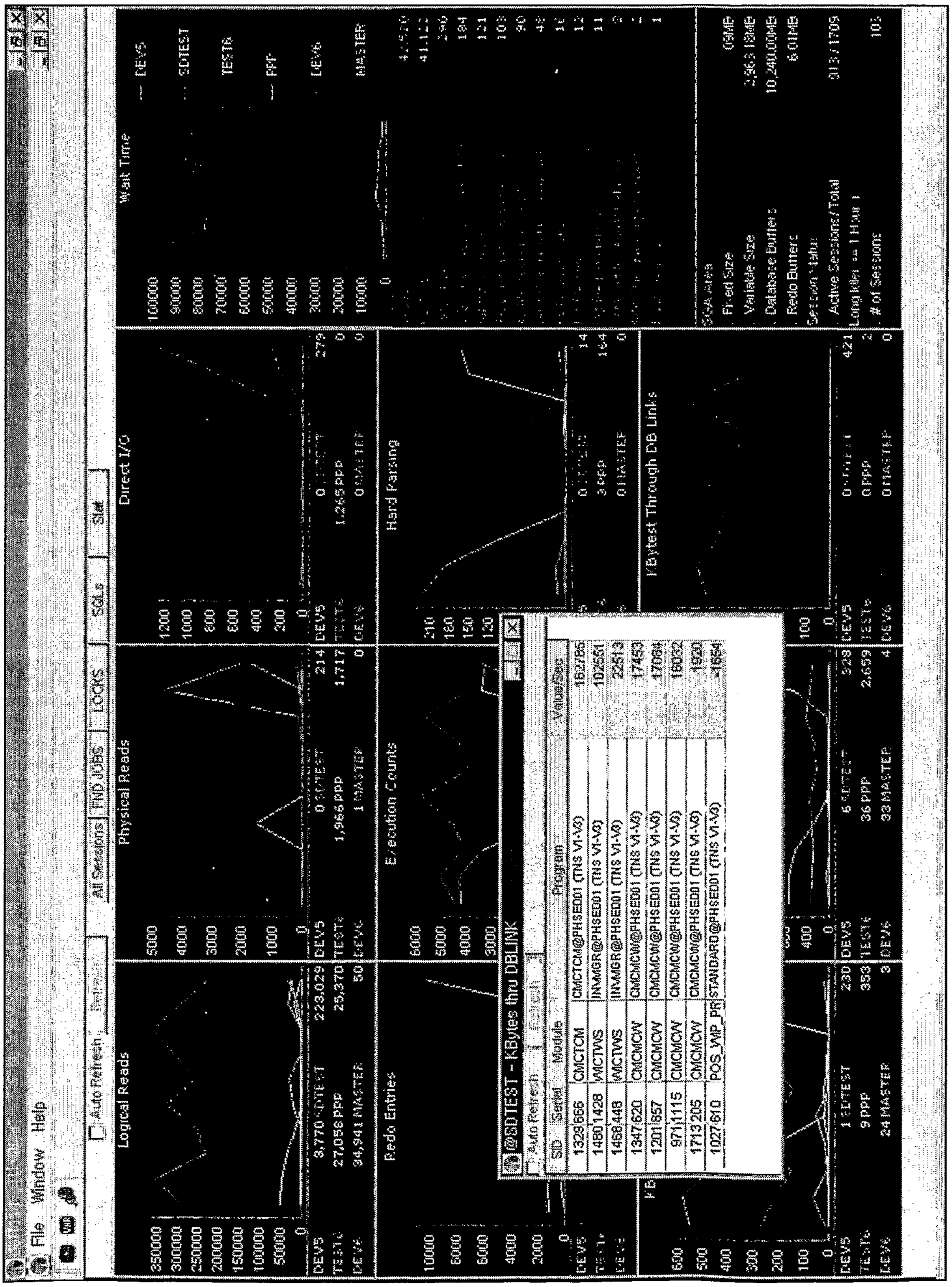

144
146
File Window Help
Auto Refresh
Refresh
All Sessions
FND JOBS
LOCKS
SQLs
Stat
Logical Reads
Physical Reads
Direct I/O
Wait Time
Redo Entries
Execution Counts
Hard Parsing
KBytes Sent To Clients
KBytes Received From Clients
KBytest Through DB Links
DEV5
SDTEST
TEST6
PPP
DEV6
MASTER
SGA Area
Fixed Size
Variable Size
Database Buffers
Redo Buffers
Session Status
Active Sessions/Total
# of Sessions

FIG.15


File Window Help
Auto Refresh
All Sessions
FND JOBS
LOCKS
SQLs
Stat
Logical Reads
Physical Reads
Direct I/O
Wait Time
Redo Entries
Execution Counts
Hard Parsing
KBytest Through DB Links
@SDTEST - KBytes thru DBLINK
SID
Serial
Module
Program
Value/Sec

INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR02/00491

A. CLASSIFICATION OF SUBJECT MATTER

IPC7 G06F 17/40

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Patents and applications for inventions since 1975

Electronic data base consulted during the intertnational search (name of data base and, where practicable, search terms used)
FPD, PAJ, PATROM

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,035,306 A(TERASCAPE SOFTWARE INC.,)7.MAR.2000 * see abstracts & claims	1 - 14
A	US 5,701,471 A(SUN MICROSYSTEMS, INC.,)23.DEC.1997(Family None) * see abstracts & claims	1 - 14
A	JP 1997-305461 A(TOSHIBA CORP)28.NOV.1997(Family None) * see abstracts & claims	1 - 14

☐ Further documents are listed in the continuation of Box C.   ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevence
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevence; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevence; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 JUNE 2002 (10.06.2002)	10 JUNE 2002 (10.06.2002)

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
920 Dunsan-dong, Seo-gu, Daejeon 302-701,
Republic of Korea
Facsimiie No. 82-42-472-7140

Authorized officer
CHO, Young Kab
Telephone No. 82-42-481-5781

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/KR02/00491

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6,035,306	07.03.2000	WO 9927451 A1	03.06.1999