SEAMLESS USAGE OF USER'S DATABASES IN ARCHAEOLOGICAL DATABASE SYSTEM

T. Hochin [a, *], F. Kobayashi [a], K. Tsuji [a], H. Nomiya [a]

[a] Division of Information Science, Kyoto Institute of Technology, Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto, 606-8585 Japan - hochin@kit.ac.jp

KEY WORDS: Archaeological database, User's database, View, PHP, Database system

ABSTRACT:

Archaeological database system is useful because the kind of relics wanted to be displayed could dynamically be changed by specifying the retrieval condition. The system, however, often enforces the usage of the database used in this system, which is called the system database, on the user. The user must store their data into the system database. This is a very cumbersome and heavy task because the users usually hold their archaeological data in their own databases, whose structures are different from that of the system database. In order to overcome this inconvenience, a method of using the user's database in the archaeological database system is proposed. We have to overcome two problems. The first is the variety of database systems. In order to address to this kind of variety, PHP Data Object (PDO) is used. The second problem is the variety of the structure of data. This problem is addressed to by using the view mechanism. A view is a virtual table derived from one or more ordinary and/or virtual tables. The proposed method enables users to use their own databases without re-storing data from their own database into the system database. This paper describes a prototype archaeological database system that has been constructed.

1. INTRODUCTION

Information Communication Technology has been used in the archaeological research area as well as the other areas. Archaeological data have also been managed by using computers. As the database management system is usually used in the management of data on computers, archaeological data have been managed by using the database management system. Many archaeological database systems have been reported (Oikawa 1997; Hachimura 1997; Yokoyama, Chiba 2002). Some systems have been in public on the Web.

Archaeological data have often been managed with the geographical information. The archaeological database system is required to manage the geographical information as well as the information of ruins and relics.

An archaeological database system, which retrieves archaeological data according to the retrieval condition, and displays the retrieval results in the list form and/or on maps, has been constructed (Hochin, 2009). This kind of archaeological database system is very useful. Archaeological data could be analyzed from various points of views only by changing retrieval condition. For example, relics appearing on a specific layer could be reported in the list form. Distribution of pieces of dishes could easily be obtained. Distribution of those of cups could also be obtained easily.

The system, however, often enforces a user to store archaeological data into a table in the system database. This is a very cumbersome and heavy task because he/she usually holds his/her archaeological data in the table of his/her own database, whose structure is different from that of the system database.

This paper proposes a method of using the user's database in the archaeological database system. The proposed method could overcome the inconvenience described above. To this end, two problems have to be addressed to. The first is the variety of database systems. In order to address to this kind of variety, PHP Data Object (PDO) is used. We could access the databases managed by any database system with the uniform way by using PDO. The second problem is the variety of the structures of data. This problem is addressed to by using the view mechanism. By using views, the user's data could be used as if they were in the system database.

The remaining of this paper is as follows: Section 2 describes archaeological database systems. Section 3 proposes a method of using user's data. Section 4 describes the prototype system. Section 5 concludes this paper.

2. ARCHAEOLOGICAL DATABASE SYSTEM

Archaeological data are often managed by using commercial or free database management system, e.g., ORACLE, MySQL, SQLite. One of the merits of the usage of database management system is the query functionality. Users can retrieve desired data by specifying retrieval condition. The data satisfying the retrieval condition are returned to the users. In this paper, the system managing archaeological data by using database management system is called the *archaeological database system*.

We have constructed an archaeological database system for the Ichijodani Asakura Clan Ruins in Japan. The Ichijodani Asakura Clan Ruins are the ruins of the castle town of five warring lords of the Asakura Family who ruled Echizen for 103 years (Fukui Prefectural Tourism Federation, 2009). The grand castle town was founded 530 years ago in 1471 and developed a graceful culture. However, when the Asakura Family was defeated by Oda Nobunaga in 1573, the town was burned and

* Corresponding author

its long history came to an end. In 1967, excavation and research started, uncovering the shape of the whole town, including a house belonging to the lord, samurai residences, temples, houses of merchants, houses of craft workers and streets. This area is considered to be very important in the Japanese archaeological research because this area remained as it was. The area of the Ichijodani Asakura Clan Ruins is a national historic site in Japan. About five thousands of remains and about two million relics have appeared until now.

In the Ichijodani Asakura Clan Ruins, the information of relics includes ruin identifiers, excavation area names, excavation numbers, layer names, excavation dates, serial numbers of relic, branch numbers of relic, major types of objects, types of objects, and notes as follows:

1. ruin identifier: The ruin identifier is the identifier of a ruin. That of the Ichijodani Asakura Clan Ruins is "9MI."
2. excavation area name: This is a name of an excavation area.
3. excavation number: This is a serial number of excavation.
4. block identifier: The area of an excavation is managed through blocks. A block is 3x3 square meters. An example of this identifier is "NM23."
5. layer name: This is a name of the layer where a relic is found.
6. excavation date: This is a date of an excavation.
7. serial number of relic: This is the serial number of a relic. This serial number begins at every excavation.
8. branch number of relic: This is an auxiliary number put to the relics having the same serial number.
9. major type of object: Major types of objects are how objects are made.
10. type of object: Types of objects include a cup, a dish, a base, and so on.
11. note: Various notes on a relic could be described.

The information of relics described above is managed in the table, whose name is "relics_detail_t," in the archaeological database system for the Ichijodani Asakura Clan Ruins. This table has eleven columns, which correspond to the information of relics described above. The structure of the table *relics_detail_t* is as follows:

relics_detail_t (isekiID_c, chiku_c, jisu_c, kukaku_c, dosou_c, date_c, number_c, subnumber_c, taibetsu_c, kisyu_c, bikou_c)

The system includes other tables as well as this table. These are of the information on areas, maps, and so on. The details of them are omitted because these are not important in this paper.

Figure 1 is a window for specifying retrieval condition on relics in the archaeological database system for the Ichijodani Asakura Clan Ruins, which supports only Japanese. Users could specify retrieval condition on the desired relics as shown in Fig. 2. The retrieval result could be displayed on the map or in the form of list. Figure 3 is an example of the retrieval result displayed on the map. The colours of blocks change according to the numbers of relics retrieved. Figure 4 is an example of the retrieval result displayed in the form of list.

This system is useful because the kind of relics wanted to be displayed could dynamically be changed by specifying the retrieval condition. The system, however, enforces the usage of the database used in this system, which is called the system database, on the user. The user must store their data into the system database. This is a very cumbersome and heavy task because the users usually hold their archaeological data in their own databases, whose structures are different from that of the system database.

Figure 1. Window specifying retrieval condition.

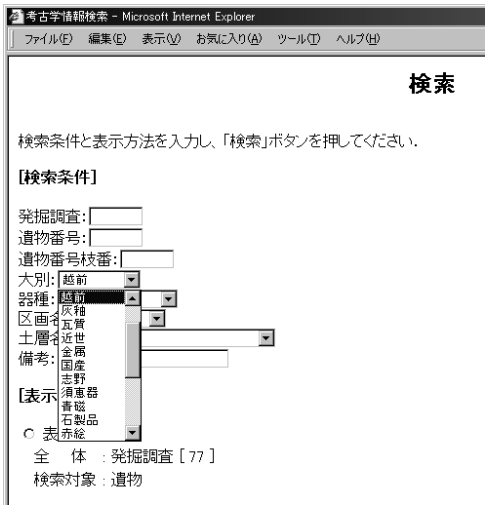


Figure 2. Specifying retrieval condition.

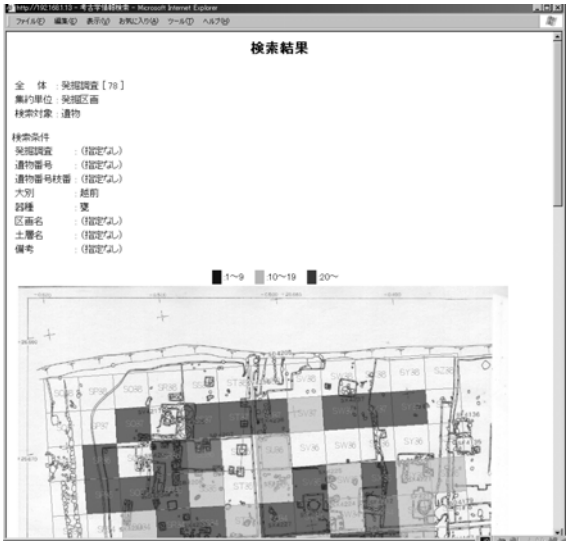
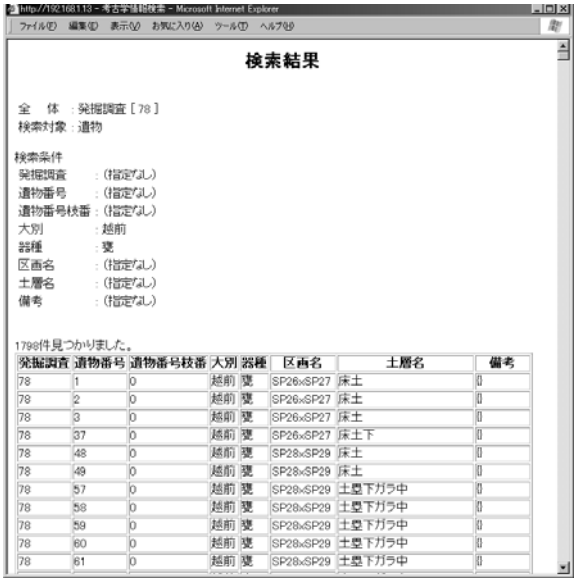


Figure 3. Window of retrieval result displayed on a map.



検索結果

全　体　：発掘調査［78］
検索対象：遺物

検索条件
発掘調査　：（指定なし）
遺物番号　：（指定なし）
遺物番号枝番：（指定なし）
大別　：越前
器種　：甕
区画名　：（指定なし）
土層名　：（指定なし）
備考　：（指定なし）

1799件見つかりました。

発掘調査	遺物番号	遺物番号枝番	大別	器種	区画名	土層名	備考
78	1	0	越前	甕	SP26xSP27	床土	
78	2	0	越前	甕	SP26xSP27	床土	
78	3	0	越前	甕	SP26xSP27	床土	
78	37	0	越前	甕	SP26xSP27	床土下	
78	48	0	越前	甕	SP28xSP29	床土	
78	49	0	越前	甕	SP28xSP29	床土	
78	57	0	越前	甕	SP28xSP29	土壁下ガラ中	
78	58	0	越前	甕	SP28xSP29	土壁下ガラ中	
78	59	0	越前	甕	SP28xSP29	土壁下ガラ中	
78	60	0	越前	甕	SP28xSP29	土壁下ガラ中	
78	61	0	越前	甕	SP28xSP29	土壁下ガラ中	

Figure 4. Window of retrieval result displayed in the list form.

3. METHOD OF USING USER'S DATA

3.1 Approaches

In order to overcome the inconvenience described in the last section, a method of using the user's database in the archaeological database system is proposed. We have to overcome two problems. The first is the variety of database management systems. In order to address to this kind of variety, PHP Data Object (PDO) is used. We could access the databases managed by any database management system in the uniform way by using PDO. Please refer to Appendix A for PDO. As described in Appendix A, a variety of database management systems could easily be used by using PDO. What we need is only the changes of text strings in creating an instance object of the class *PDO* as described in Appendix A.

The second problem is the variety of the structures of data. This problem is addressed to by using the view mechanism. A view is a virtual table derived from one or more ordinary and/or virtual tables. A view can be used only by defining the view with a kind of query. Storing data into a view is not required at all. By using views, the user's data could be used as if they were in the system database. Please refer to Appendix B for the view mechanism. In order to create views, specifying the correspondence between the structure of the user's data and that of the data in the system database is required.

3.2 Defining views

The method of defining a view is described here. The CREATE VIEW statement, which is the SQL statement for creating a view, is different depending on whether the number of the user tables having the information of the relics is one or more than one as described in Appendix B.

One table: This is the case that the information of the relics is stored in a table in a user's database. Let the name of a user table be *relics*, and let this table have the following structure:

relics (ruinID, area, exca_number, blockID, layer_name, date, number, subnumber, major, type, note)

where the columns *ruinD, area, exca_number, blockID, layer_name, date, number, subnumber, major, type,* and *note* correspond to the columns *isekiID_c, chiku_c, jisu_c, kukaku_c, dosou_c, date_c, number_c, subnumber_c, taibetsu_c, kisyu_c,* and *bikou_c* of the table *relics_detail_t*, respectively.

In this case, the CREATE VIEW statement is as follows:

```
CREATE VIEW relics_detail_t (isekiID_c, chiku_c, jisu_c,
                kukaku_c, dosou_c, date_c, number_c,
                subnumber_c, taibetsu_c, kisyu_c, bikou_c)
AS SELECT ruinID, area, exca_number, blockID, layer_name,
                date, number, subnumber, major, type, note
    FROM relics;
```

Two or more tables: This is the case that the information of the relics is stored in more than one table in a user's database. Consider that a user has two tables, whose names are *excavation* and *relics.* The table *excavation* contains the information of excavation, while the table *relics* contains the information of the relics. Let these tables have the following structures:

excavation (ruinID, exca_number, area)
relics (ruinID, exca_number, blockID, layer_name, date, number, subnumber, major, type, note)

where the columns of the table *relics* having the same names as of the table *excavation*, which are *ruinID* and *exca_number*, have the values of the columns of the table *excavation*.

In this case, these two tables have to be joined in defining a view. The CREATE VIEW statement is as follows:

```
CREATE VIEW relics_detail_t (isekiID_c, chiku_c, jisu_c,
                kukaku_c, dosou_c, date_c, number_c,
                subnumber_c, taibetsu_c, kisyu_c, bikou_c)
  AS SELECT e.ruinID, e.area, e.exca_number, r.blockID,
                r.layer_name, r.date, r.number, r.subnumber,
                r.major, r.type, r.note
      FROM excavation e, relics r
     WHERE e.ruinID = r.ruinID
       AND e.exca_number = r.exca_number;
```

Please note that the CREATE VIEW clause, which is the part before "AS SELECT", is the same as in the case of one table. The difference between the two cases is the SELECT statement in the CREATE VIEW statement.

Missing columns: When the columns corresponding to those of the table *relics_detail_t* do not exist in a user's table, the system handles the values of these columns as NULL, which means that there is no value. For example, when the table *relics* in the example of one table described above does not have the columns *area, blockID, subnumber,* and *note*, so the table *relics* has the following structure:

relics (ruinID, exca_number, layer_name, date, number, major, type),

the CREATE VIEW statement becomes as follows:

```
CREATE VIEW relics_detail_t (isekiID_c, chiku_c, jisu_c,
                kukaku_c, dosou_c, date_c, number_c,
                subnumber_c, taibetsu_c, kisyu_c, bikou_c)
  AS SELECT ruinID, NULL, exca_number, NULL,
                layer_name, date, number, NULL, major, type,
                NULL FROM relics;
```

3.3 Implementation

We have introduced the following three tables: *user_t, userdb_t,* and *replace_t.* The table *user_t* is for the management of the information on users. The table *userdb_t* is for managing the information on users' databases. The table *replace_t* has the

information of the mappings between the table *relics_detail_t* and users' tables.

The table *user_t* has the following structure:

user_t (userID_c, username_c, password_c)

where the column *userID_c* is the primary key of this table, the column *username_c* is for a user's name, and the column *password_c* is for a user's password.

The table *userdb_t* has the following structure:

userdb_t (userdbID_c, userID_c, dbms_c, location_c, dbname_c, dbpass_c, dbuser_c, dbinfo_c)

where the column *userdbID_c* is the primary key of this table, the column *userID_c* is of a value of the column *userID_c* of the table *user_t*, and the columns *dbms_c*, *location_c*, *dbname_c*, *dbpass_c*, *dbuser_c*, and *dbinfo_c* are the type of the database management system, e.g., MySQL, SQLite, the location where a database exists, the name of the database, the password for the database, the user name for the database, and the note of the database, respectively. The information of this table is mainly used in using PDO.

The table *replace_t* has the following structure:

replace_t (replaceID_c, userdbID_c, isekiID_c, chiku_c, jisu_c, kukaku_c, dosou_c, date_c, number_c, subnumber_c, taibetsu_c, kisyu_c, bikou_c, table_c, join_c)

where the column *replaceID_c* is the primary key of this table, the column *userdbID_c* is of a value of the column *userdbID_c* of the table *userdb_t*. The columns *isekiID_c*, *chiku_c*, *jisu_c*, *kukaku_c*, *dosou_c*, *date_c*, *number_c*, *subnumber_c*, *taibetsu_c*, *kisyu_c*, and *bikou_c* are for describing the correspondence between the table *relics_detail_t* in the system database and the user's table in a user's database. The names of columns of the user's table are stored into these columns. For example, when the column *exca_date* of a user's table corresponds to the column *date_c* of the table *relics_detail_t*, the name "*exca_date*" is stored into the column *date_c* of the table *replace_t*. The column *table_c* is for the name of the user's table. When alias names to tables are wanted to be specified, the alias names could also be specified. For example, when the alias name "e" for the table *excavation* is watned to be set as described in **3.2**, the specification becomes "excavation e." When two or more tables have the information of the relics in a user's database, the names of all of the tables are specified by separating the comma, and stored into the column *table_c*. In this case, these tables must be joined. The join condition is specified as the text string in the column *join_c* of the table *replace_t*. In the example described in **3.2**, the specification becomes "e.ruinID = r.ruinID AND e.exca_number = r.exca_number."

4. PROTOTYPE SYSTEM

As the three tables *user_t*, *userdb_t*, and *replace_t* have been introduced, the procedures managing the information in these tables are required. The windows for these management procedures could easily be implemented.

The windows for the management of the information in the tables *user_t*, *userdb_t*, and *replace_t* are shown in Fig. 5, Fig. 6, and Fig. 7, respectively. The button "display" is for the display of the information registered. After appropriate values are specified into the text fields in the window, pressing the button "register" results in the registration of the specified values. The button "delete" is for the deletion of the information.

A user could use several databases. The system provides the functionality of selecting a database that the user wants to use. The window for this purpose is shown in Fig. 8. This window appears after a user enters into the system or after following the link named "change DB" in the main menu appearing at the left frame. A database used is decided by pressing the button "use the select DB" after selecting a desired database. After the database used is decided, the window for specifying the retrieval condition as shown in Fig. 1 appears. When another database or another table is wanted to be used, a user could follow the link named "change DB" as described above. More than one set of tables in a database could be used. For example, a table *relics_A* and another one *relics_B* in a database could be used, whereas the name of the link is "change DB." The *Note* information is convenient in identifying the table because the current implementation shows only the information of databases in the window shown in Fig. 8. We could not distinguish the tables in a database.

Figure 5. Window for inserting the user information.

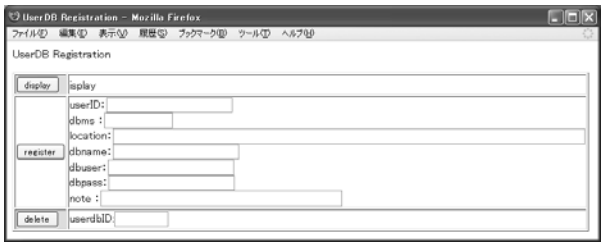


Figure 6. Window for inserting the user's database information.

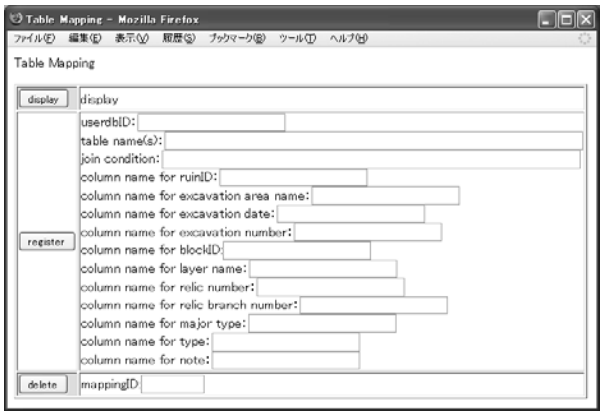


Figure 7. Window for inserting the mapping information.

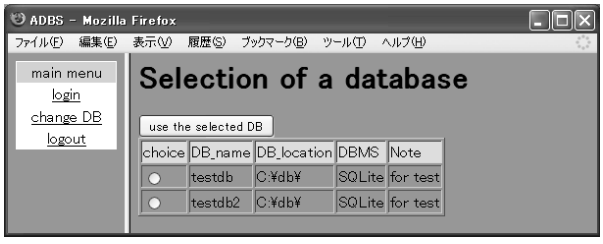


Figure 8. Window for selecting a database.

A table in a database and a table in another database, however, could not be registered into the system in the window shown in Fig. 5. That is, a view could not be defined by using the tables under the management of different database management systems. Addressing to these issues is in the future work.

5. CONCLUDING REMARK

This paper proposed a method of using the user's database in the archaeological database system. Two kinds of varieties, which are the variety of database management systems and that of the structures of data, are addressed to. In order to address to the variety of database management systems, PHP Data Object (PDO) is used. We could access the databases managed by any database management system in the uniform way by using PDO. The problem of the variety of the structures of data is addressed to by using the view mechanism. By using views, the user's data could be used as if they were in the system database. A prototype archaeological database system that has been constructed was described.
The prototype system is mainly for the Ichijodani Asakura Clan Ruins. The application of this system to the other ruins is in the future work. A view could not be defined by using the tables managed by different database management systems. Addressing to these issues is also in the future work.

References

References from Journals:
Hachimura, K., 1997. Databases in the Humanities. *Information Processing*, 38(5), pp. 377-382 (in Japanese).

Hochin, T., Tsuji T., 2002. The Present and Future of Archaeological Databases, *The Journal of the Institute of Electronics, Information, and Communication Engineers*, 85(3), pp.171-175 (in Japanese).

Hochin, T., 2009. An Implementation Method for Various Aggregations of Archaeological Databases. *Journal of Computer Archaeology*, 15(1-2), pp. 1-12 (in Japanese).

Oikawa, A. 1997. Archaeological Database -Multimedia Technology for Re-emergence of the Past-, *Information Processing*, 38(5), pp. 388-391 (in Japanese).

Yokoyama, R., Chiba, F., 2002. Construction of an Archaeological Site Database Using Geographical Information System, *The Journal of the Institute of Electronics, Information, and Communication Engineers* , 85(3), pp.176-180 (in Japanese).

References from Books:
Date, C. J., 2003. An Introduction to Database Systems (8th Edition). Addison Wesley.

References from websites:
Fukui Prefectural Tourism Federation 2009. "Ichijodani Asakura Clan Ruins", http://www.fuku-e.com/lang/english/culture_s.html (accessed 27 July, 2009)

PHP Group 2009. "PHP: PDO – Manual 2009", http://jp.php.net/manual/en/book.pdo.php (accessed 17 July, 2009)

Acknowledgements

We would like to give great thanks to Mr. Nobuyuki Mizumura. He is an archaeological researcher of the Ichijodani Asakura Clan Ruins. He gave us a lot of assistance, help, and cooperation.

APPENDIX A. PHP DATA OBJECT (PDO)

We could access the databases managed by a variety of database management systems in the uniform way by using PHP Data Object (PDO) (PHP Group, 2009).
An instance object of the class PDO is created in order to prepare the access to a database. The following is an example of the creation of a PDO instance object for accessing a database, whose name is "relics_db," managed by the database management system MySQL, which is on the "localhost" computer.

```
$link = new PDO('mysql:host=localhost;dbname=relics_db',
                DB_USER, DB_PASSWORD) ;
```

The example described above is for the case of using the database management system MySQL. When the character string "mysql" is changed to "pgsql," the database management system PostgreSQL can be used.
The instance object created, which is stored into the variable named "link" in our example, is used for accessing the database. An example of obtaining all of the tuples in a table named "T" is as follows:

```
$result = $link->query("SELECT * FROM  T ;");
```

The *query* method of the object is used in obtaining tuples in the table. The character string *"SELECT * FROM T ;"* is an SQL statement. SQL is a standard database language. Please note that the same operation could be used in obtaining tuples from the table managed by the different database management system by using PDO.

APPENDIX B. VIEW

A view is a virtual table derived from one or more ordinary and/or virtual tables (Date, 2003). A view can be used only by defining the view with a kind of query. Storing data into a view is not required at all. By using views, the data could be handled as if they had different structure.
Let a table *T* have five columns: *C1*, *C2*, *C3*, *C4*, and *C5*. Let consider the situation that the columns *C1*, *C2*, and *C5* are handled as *X1*, *X2*, and *X3*, respectively, in a view *V1*, and the value of column *X1* of each tuple in *V1* is larger than 10. This view *V1* is defined in SQL as follows:

```
CREATE VIEW V1(X1, X2, X3)
    AS SELECT C1, C2, C5 FROM T WHERE C1 > 10 ;
```

The set of tuples retrieved by executing the SELECT statement in the view definition is that of tuples in a view table.
The example described above is for one table. For two or more tables, a view can similarly be defined. Let a table *S* have three columns: *D1*, *D2*, and *D3*. Let consider another situation that the columns *C1* and *C2* of the table *T* are handled as *Y1* and *Y2*, respectively, and the column *D2* of the table *S* is handled as *Y3*, and the value of the column *C1* of the table *T* is equal to that of column *D1* of the table *S* and is larger than 10 in a view *V2*. This view *V2* is defined as follows:

```
CREATE VIEW V2(Y1, Y2, Y3)
    AS SELECT T.C1, T.C2, S.D2  FROM T, S
        WHERE T.C1 = S.D1 AND T.C1 > 10 ;
```