A Correction Reflected Query Method of Database during Online Entry

Tsukasa Kudo†, Yui Takeda‡, Masahiko Ishino*, Kenji Saotome**, Kazuo Mutou***, and Nobuhiro Kataoka****

†Faculty of Comprehensive Informatics, Shizuoka Institute of Science and Technology, Japan
‡Mitsubishi Electric Information Systems Corporation, Japan
* Department of Management Information Science, Fukui University of Technology, Japan
** Hosei Business School of Innovation Management, Japan
*** Faculty of Science and Technology, Shizuoka Institute of Science and Technology, Japan
**** Department of Information Technology, Tokai University, Japan
kudo@cs.sist.ac.jp

Abstract - The database of the mission-critical systems is updated with entry data by transaction processing, and are queried to make statistics and so on by batch processing generally. Such a batch processing had been executed at the overtime to avoid the data entry service time, because it occupied the database for hours. On the other hand, in recent years, the entry service time is being rapidly extended with the development of the Internet business. So, the methods to execute the both concurrently have been put to practical use. However, there are some cases that cannot be supported by only the conventional methods, because there are various kinds of database query and operation in the actual mission-critical system. In this paper, to support such the case, we propose a query method to query the database as of designated time reflecting the correction entered after the time. Moreover, we implemented this method into a mission-critical system, and confirmed the effect to reduce the overtime batch processing in the actual operation.

Keywords: temporal database, transaction time database, mission-critical system, query, integrity, batch processing

1 INTRODUCTION

In the mission-critical system such as the retail, the finance, the manufacture, because data are entered by many online terminals concurrently (hereinafter "online entry"), concurrency controls are executed by the transaction processing [5]. On the other hand, a great deal of data processing, such as periodic sum of entered data, is processed by the batch processing [5]. For example, in the retail system, sales information at stores is reflected into its database immediately by the transaction processing; on the other hand, the settlement of accounts is calculated by the batch processing. Here, the batch processing had been executed in night to avoid the time zone of the online entry, because it occupy the database for hours to process a great deal of data. However, in recent years, this time zone was expanded by the development of the internet business and so on. As a result, it often caused a problem that the batch processing didn't complete in the given time.

So, the method to maintain the integrity of query result of database even during the online entry had been implemented. For example, the multiversion concurrency control of database [2], by which the integrity of query result is maintained during the online entry, is used widely. Here, in the batch processing, because the restriction of the execution time is looser than the online entry, strict examinations of the entry data are executed. Therefore, error data is often found. If the batch processing is executed while online entry isn't executed, it can be executed again after the correction of error data. However, if batch processing is executed concurrently with online entry, the newly entered data is also reflected into the batch processing result. That is, the corrected result of designated time, the cash total sum of the day and so on for example, can't be provided.

For this problem, authors showed that the integrity of the snapshot of the bitemporal database can be maintained during the online entry in the actual mission-critical system, even in the case that error data were detected, by reflecting its correction into the query result [9]. Here, the bitemporal database is a kind of temporal database [7], [13], which manages both of the transaction time and the valid time. The former is the time that data is valid in the database; the latter is the time that data is valid in real world [4], [6], [11], [13]. And, its query target was the data at the designated valid time.

However, in the case of the settlement of accounts and so on, the processing target is the data that was online entered by the deadline time, which is the database status as of this time. And, if error data are detected, they have to be corrected while the processing. In this case, the multiversion concurrency control has the problem that does not support the reflection of data correction after the deadline time; the bitemporal database also has the problem not being suitable for such the system that the status of real world was not entered instantly.

Our goal in this paper is to provide the query method that maintains the integrity of query result with reflecting the data correction, even in the above-mentioned case. We summarized this and showed it in the title as "A Correction Reflected Query Method". For this purpose, we propose the correction query method, which uses the transaction time. We show that the corrected data is queried without influences of the online entry by this method. Moreover, we implemented this method into an actual mission-critical system, and confirmed the effect to reduce the overtime batch processing.

The reminder of this paper is organized as follows. In section 2, we show the problem to intend for, and in section 3,

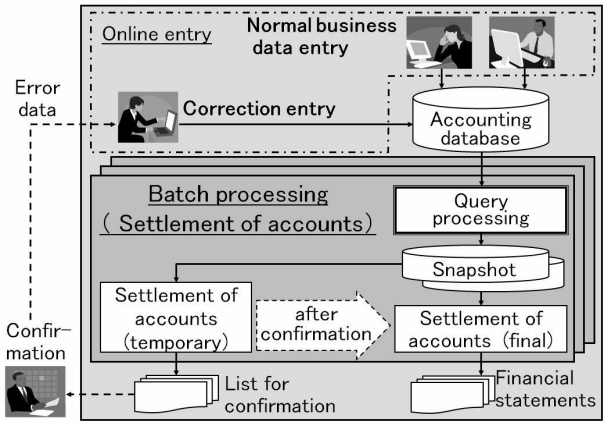

Figure 1: An example of batch processing constitution.

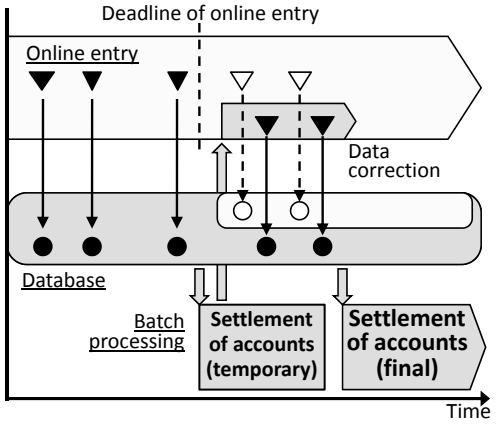

Figure 2: Settlement of accounts data by time series

we propose the query method to solve this problem. In section 4, we show an implementation case of this method in a mission-critical system, and in section 5, we evaluate the method based on the implementation result. Finally, we consider this method in section 6.

2 PROBLEM WITH BATCH PROCESSING

2.1 Constitution of Batch Processing

In the mission-critical system, a certain level integrity of online entered data is maintained by the integrity control of the database management system and the transaction processing, and by the checking function of the business application program. In this paper, we define the integrity as what the state of the real world is reflected in the database with validity and completeness [10]. By the way, the integrity confirmation with querying a large quantity of data needs to be executed by batch processing. For example, the calculations of total for the collation with the actual cash or the actual articles, or the consistency check among some tables and so on. So, in the batch processing, the first process is usually the integrity confirmation of its target data.

Figure 1 shows the example of the batch processing about the accounting system. Accounting data is accumulated in the database by the online entry, and the settlement of accounts processing is executed regularly. In this processing, temporary processing is executed first to prevent errors of the processing, in which various kinds of data check is done. And, when error data is detected, it is corrected by the online entry. In this way, after all confirmation is complete, final processing is executed to make the financial statements.

Here, the query processing in the batch processing (hereinafter "batch query processing") of Figure 1 has to be executed without undergoing influence of the online entry, though it is executed concurrently with the online entry. So, even if the correction data is entered by the online entry, it must be distinguished from the normal business entry data entered after the deadline time. Figure 2 shows the state of data of the settlement of accounts processing of Figure 1 by the time series. In figure 2, "▼" shows both of the online entry before the deadline time, and its correction entry; "●" shows the data corresponding to them. Also, "▽" shows the new online entry after the deadline time; "◯" shows the data corresponding to this. In the settlement of accounts processing, the temporary processing is executed for the confirmation about the data entered by the deadline time of Figure 2. And, the final settlement of accounts processing is executed after correction of the data error. Therefore, the target data of the settlement of accounts processing is the query result as of the the deadline time, in which only the correction entered after the time is reflected. That is, in Figure 2, only the data shown by "●" is the target for the final processing.

2.2 Problem about Conventional Database Query Method

We show the problem about the conventional database query method in the case of batch query processing accompanied by the data correction. In the multiversion concurrency control, the version of the database is managed with the time series. That is, the data entered after the deadline time for the correction cannot be distinguished from the normal business entry. Therefore, in the case shown in Figure 2, there is the problem that even the normal business entry data shown by "◯" become the processing target, too.

For this problem, we showed a solution utilizing the bitemporal database and confirmed that we could execute the batch processing even while the online entry in the actual mission-critical system [9]. In the bitemporal database, both histories of the valid time and the transaction time are managed, and the state of data, which once existed in the database, is accumulated as the records. That is, the both records of the state of the database and the real world are accumulated [4], [8]. For example, in the personnel management system of the company, the period that a person was in office for one duty position is shown with the valid time; on the other side, the period that its data was valid in the database is shown with the transaction time. Incidentally, the database that manages none of these times is called the snapshot database [12].

Figure 3 shows the application example of the bitemporal database to the travel expense checkout of the accounting system, in which correction data is queried on the condition that the deadline time is April 20th. In Figure 3, "$[V_a, V_d)$" shows

(1) Case of normal query

ID	Va	Vd	Ta	Td	Amount	Result
001	4/19	4/20	4/20	4/21	1,000	
001	4/19	4/20	4/21	now	1,500	●
002	4/20	4/21	4/21	now	3,000	

(2) Case of wrong query

ID	Va	Vd	Ta	Td	Amount	Result
001	4/19	4/20	4/20	4/21	1,000	
001	4/19	4/20	4/21	now	1,500	●
002	4/19	4/20	4/21	now	3,000	●

Figure 3: Query of correction data by bitemporal database

the period of the valid time, i.e. one business trip period, and "$[T_a, T_d)$" shows the period of the transaction time, i.e. the period that its slip data was valid in the database of the system. Incidentally, the time is expressed by the unit of a day. And, "●" of the column " Result" shows the queried data for the query condition explained below. On April 20th, the data $ID = 001$ was entered, and on April 21st, the correction entry of the data $ID = 001$ and the new entry of the data $ID = 002$ was done. Here, when making a travel expense checkout data aggregate $D = \{d\}$ and designating the valid time t_v and the transaction t_t, the following data is queried as the snapshot as of the above-mentioned time.

$$D_1 = \{d | d \in D, t_v \in [d[V_a], d[V_d]) \wedge t_t \in [d[T_a], d[T_d])\} \quad (1)$$

Here, $d[V_a]$ shows the instance of the attribute V_a in d, and the others are same, too.

Therefore, as shown in (1) of Figure 3, when time were designated as $t_v = April\ 19th$ and $t_t = April\ 21st$, the data $ID = 001$ after correction is queried; the data $ID = 002$ is not queried. Here, the time "now" of T_d shows the corresponding data is valid at the time to query [1], [14].

However, in the actual business, the state of the real world isn't always reflected into the database immediately. (2) of Figure 3 shows the case that the entry of the travel expense checkout has been late. Though the valid time period of the trip is $[4/19, 4/20)$, its data was entered on April 21th. In this case, there is a problem that the query result includes the data $ID = 002$, because it satisfies the condition of equation (1). But nevertheless it is the normal business entry data after the deadline time.

Moreover, there is the problem that some businesses don't need to manage the valid time. For example, the slips of the purchase and the payment of the accounting system are managed by the system, so their valid time as for the real world isn't managed usually. That is, the split table of the database doesn't need to take the composition of bitemporal database.

3 PROPOSAL OF QUERY METHOD TO REFLECT DATA CORRECTION

We propose a query method,"correction query", for the problem shown in section 2.

3.1 Correction Query

The correction query is the query method which result of time t_1 reflects only its correction entered by time t_2. We call the time t_1 "query time", and t_2 "correction query tim", and it becomes $t_1 < t_2$. Incidentally, in the case of Figure 2, t_1 corresponds to the deadline, and t_2 corresponds to the start time of settlement of accounts (final).

The correction query deals with the database that manages the transaction time, i.e. the transaction time database. The relation [3] of the transaction time database R is expressed as following.

$$R(K, T, A) \quad (2)$$

We show each attribute as follows.

- $K = \{K_1, ..., K_m\}$
  This expresses the set of attributes constituting the primary key of the snapshot queried by the designated transaction time.

- $T = \{T_a, T_d\}$
  This expresses the time period attribute of the transaction time, which is generated by system and isn't made public to the users. Here, T_a shows the time that the data was added to the database (hereinafter "addition time"), and T_d shows the time that the data was logically deleted from the database (hereinafter "deletion time"). As long as the data hasn't been deleted yet, the instance of attribute T_d is expressed by the above-mentioned "now".

- $A = \{A_1, ..., A_n\}$
  This expresses the other attributes.

We can query the snapshot at any designated transaction time, which is the state of the database at the time. When making the designated time t, the relation of this snapshot is expressed by the following equation.

$$Q(t) = \{q | q \in R \wedge q[T_a] \leq t \wedge t < q[T_d]\} \quad (3)$$

Here, $q[T_a]$ shows the instance of the attribute T_a of q, and $q[T_d]$ is similar, too. In the correction query, both of the snapshot at above-mentioned t_1 and t_2 are queried. And, the correction query result is the data that reflected the corrections entered by the time t_2 into the snapshot of t_1.

The relation of the correction query for R is expressed by the union of the following S_1 and S_2, i.e. $S = S_1 \cup S_2$. Here, S_1 shows the data not being changed or deleted between t_1 and t_2. So, the correction query result is the same as the snapshot of t_1. The corresponding data is expressed by the following equation, because it exists at the both of t_1 and t_2.

$$S_1 = \{s | s \in Q(t_1) \wedge s \in Q(t_2)\} \quad (4)$$

On the other hand, S_2 shows the data being changed or deleted between t_1 and t_2. So, the correction query result is the data after the change or delete. As for the change, it is expressed by the following equation, because the data before and after change is connected by the primary key attributes

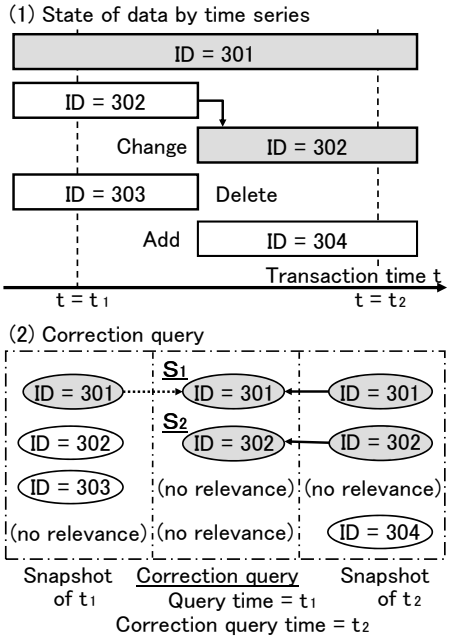

Figure 4: An example of correction query

$r[K]$ and $s[K]$. And, by this definition, the data deleted by the time t_2 isn't the target of the correction query.

$$S_2 = \{s | s \notin Q(t_1) \wedge \in Q(t_2) \wedge \exists r \in Q(t_1); r[K] = s[K]\} \quad (5)$$

Incidentally, the data of the correction query result is the subset of the snapshot $Q(t_2)$, which is entered by the usual transaction, so the consistency of the data is maintained.

Figure 4 shows the example of the correction query, of which query time is the transaction time $t = t_1$ and correction query time is $t = t_2$. In the entered data $ID = 301, 302\ and\ 303$, $ID = 302$ was changed, $ID = 303$ was deleted, on the other hand $ID = 304$ was added newly after the time t_1. (2) of Figure 4 shows the correction query result for these data. First, the data $ID = 301$ is queried based on the equation (4); $ID = 302$ after correction is queried based on (5). Second, $ID = 303$ that was deleted and $ID = 304$ that was newly added don't become the target.

3.2 Effect of Correction Query

We show that the problem shown in section 2.2 can be solved by the correction query. Figure 5 shows the application example of the correction query to the settlement of accounts processing, in the case of Figure 2. Here, we show the change of data of database by the time series like (1) of Figure 4. The temporary processing of the settlement of account had been executed for the data entered by the deadline time, and to correct the data, the change of $ID = 302$ and the deletion of $ID = 303$ were executed by the online entry based on the confirmation result of the temporary processing. On the one hand, the online entry of the normal business data were continued after the deadline time as same as before the time. In this example, the result of correction query, of which

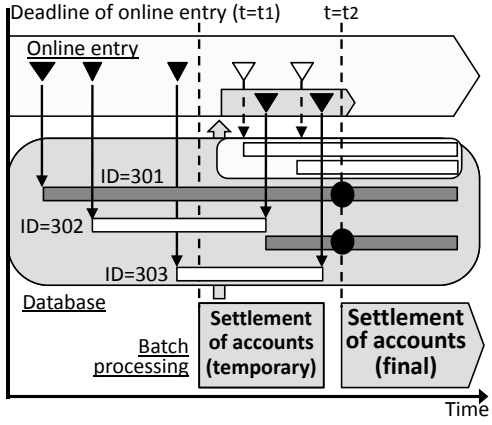

Figure 5: Correction query for settlement of accounts

the query time is the deadline time $t = t_1$ and the correction query time is the start time of the "final" settlement of account processing $t = t_2$, is the data shown by "●" in Figure 5. That is, the state of database as of the deadline time with reflecting the corrections entered after the time can be queried even during the normal business online entry without undergoing influence of this.

4 APPLICATION TO A MISSION-CRITICAL SYSTEM

In this section, we show the application result of the correction query to a mission-critical system, the local government system.

4.1 Overview of Local Government System

The local government system is a mission-critical system for the public administration business of the local government like a city hall. And, as shown in Figure 6, it consisted of various kinds of subsystems to assist the local government business. They were classified by business contents as follows.

(a) **Subsystems about Resident information**
They were used for the business, such as management and certificate of the residents who live in the city.

(b) **Subsystems about Local Tax**
They were used for the business of the local tax, such as levy and certificate about tax.

(c) **Subsystems about Welfare**
They were used for the business of welfare, such as qualification management, levy and grant.

(d) **Subsystems about City Office**
They were used for the business of the office work of local government, such as personnel management, salary computation and financial accounting.

In each subsystem, the reports were accepted at the report windows and online entered to accumulate in the database. And, the processing to query a large quantity of data was executed as the batch processing regularly or at any time. In the

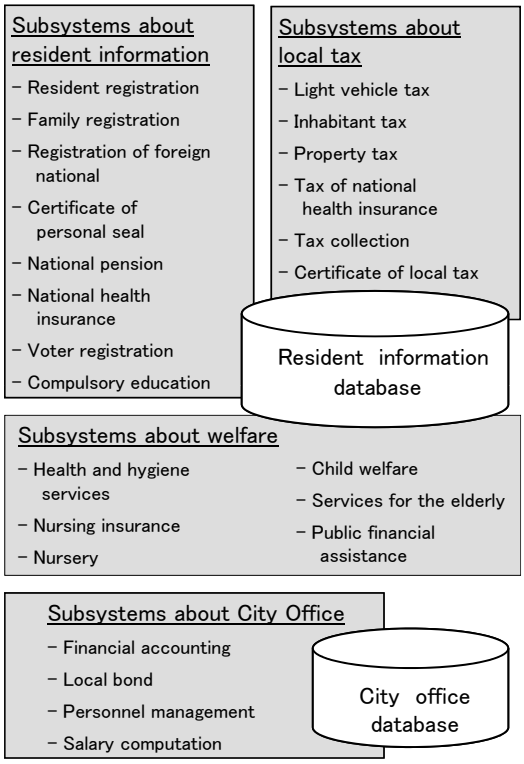

Figure 6: Composition of local government system

batch processing, the state of database as of the designated time was often queried. We show the example of batch processing like this below.

- **Population statistics:** based on the resident transfer reports, the statistics of such as the population and the number of households was made as of the end time of the first day of every month.
- **Taxation processing:** based on the reports about the local tax, the taxation processing was executed. It used the state of database as of the individually designated time.
- **Settlement of accounts processing:** based on the data of the income and the outlay, settlement of accounts processing was executed with the state of database as of the end time of every day, month and year.

4.2 Implementation of Correction Query

As shown in section 3.2, the correction query intends to the transaction time database. We used the commercial relational database and added the attributes of the addition time and the deletion time to each table to compose a transaction time database, depending on the necessity of the target business. Here, since transaction time is used as one of primary key attributes of the database, the unit of the transaction time had to be decided based on the frequency of data entry. In this system, data were entered from the terminals, and the data entry took several seconds at least. So, we made the unit of the transaction time 1 second. Incidentally, we made the attribute

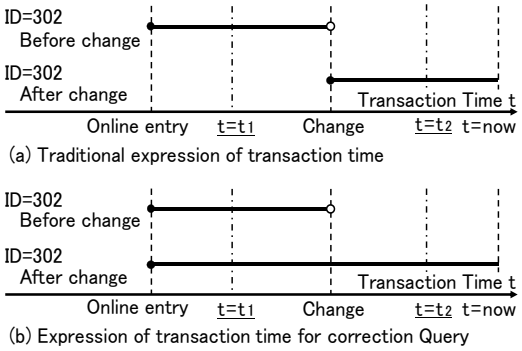

Figure 7: Implementation of transaction time

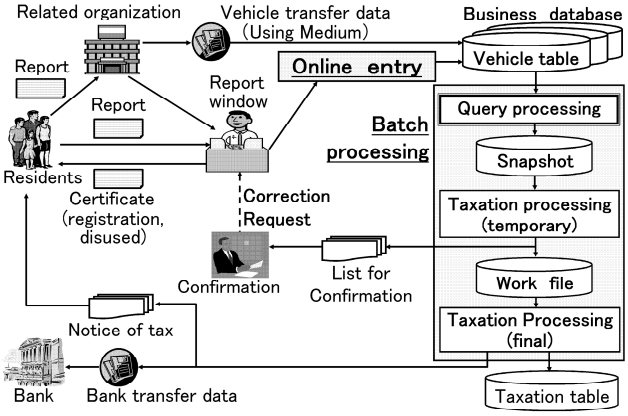

Figure 8: Dataflow of light vehicle tax business

of the transaction time the closed information in users including the records as for it, so users could query only the latest state at the query time.

As for the change records with the transaction time, the data after change was conventionally expressed in the form, of which addition time was the changed time as shown in (a) of Figure 7. In the implementing of a correction query, it was necessary to connect the data before and after correction. So, the query processing became complicated if the conventional expression was used. To solve this problem, we implemented the transaction time with the expression, in which the addition time of the data after change is the time that the data was added first, as shown in (b) of Figure 7. Incidentally, in this expression, the deletion time becomes the primary key attribute; though, in the conventional expression, the addition time is the primary key attribute.

4.3 Composition of Subsystem for Business

As the example of the business system, to which we applied the correction query, we show the light vehicle tax subsystem that is one of the subsystems about the local tax. The light vehicle is taxed on the light vehicles, which is owned by the residents as of April 1st that is the basic date. And, the taxation processing is executed based on the data reported by the residents.

Figure 8 shows the dataflow of the light vehicle tax busi-

(1) Data of vehicle table on 5/6

ID	Owner	Ta	Td	5/6
001	Keiji, T.	5/6	now	●
002	Jouto, J.	5/6	now	●
003	Haisha, S.	5/6	now	●

(2) Data of vehicle table on 5/7

ID	Owner	Ta	Td	5/6	5/7	S
001	Keiji, T.	5/6	now	●	●	●
002	Jouto, J.	5/6	5/7	●		
002	Jouto, J.	5/6	now		●	●
003	Haisha, S.	5/6	5/7	●		
004	Tuika, F.	5/7	now		●	

Figure 9: Query result of vehicle table with correction query

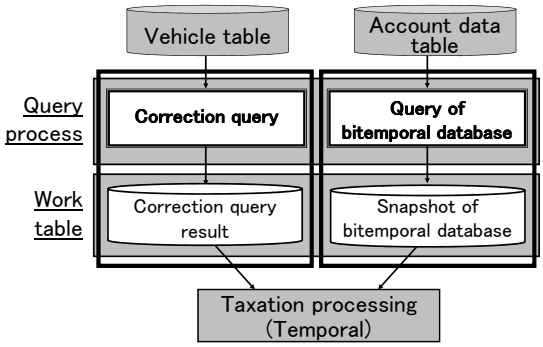

Figure 10: Combination with conventional query method

ness. The acquisition reports of the light vehicles should be reported within 15 days; the disused and transfer reports should be reported within 30 days. However, these reports are accepted in the related organizations such as the Light Motor Vehicle Inspection Organization, the Land Transport Bureau and the light vehicle stores in addition to the report windows of the local government. The data accepted at the related organizations were delivered to the local government with paper reports for online entry or with mediums for lump-sum entry. For such operation, it often takes time to reflect the transfer data of the real world into the vehicle table of the system. Therefore, the taxation processing was executed for the data entered by the deadline time, and thereafter, tax correction processing was executed monthly for the data newly entered by the corresponding deadline time.

Online entry at the report windows could not be suspended during business hours, because the light vehicles license plate issue certificates or the disuse report receipt certificates had to be published immediately reflecting the reported data. On the other hand, the taxation processing and the tax correction processing were executed by the batch processing to make the tax payment notices to the residents and the account transfer requests to the financial institutions. So, to prevent the taxation error, the checklist and the statistics documents for the confirmation were made by the temporary processing first. And, when the data error was detected, it was corrected by the online entry. After this confirmation and correction, the final processing was executed.

So, the target data for the final processing was the state of database as of the deadline time, in which only the corrections after the time were reflected. Figure 9 shows the taxation processing case, of which the deadline time was May 6th and the execution time was May 7th. We show the state of database as of May 6th in (1) of Figure 9, and the data entered by this time was the target for the processing. We show the state as of May 7th in (2) of Figure 9, in which the change of the vehicle $ID = 002$, deletion of $ID = 003$ and addition of $ID = 004$ were reflected. Here, the transaction time of $ID = 002$ was implemented with the expression shown in (b) of Figure 7. In Figure 9, "●" of column "$5/6$" shows the snapshot data of May 6th; column "$5/7$" shows the snapshot data of May 7th; column "S" shows the correction query result, of which the query time was May 6th and the correction query time was May 7th.

As shown in the column "S", the correction query result was the snapshot at May 6th, in which only the correction entered by May 7th were reflected. So, the addition data $ID = 004$ was not included.

4.4 Combination with Other Query Methods

In the actual mission-critical systems, it is necessary to query the database in a wide range of conditions. For example, the light vehicle tax was paid by the tax notice or the bank transfer. Here, as for the bank transfer, it was requested by the resident with its transfer period. So, the data for the bank transfer needed to have the valid time attribute, and we had to implement it as a table of bitemporal database. On the other hand, we queried the master table by the multiversion concurrency control, because it was the table of the snapshot database without managing the transactiontime. In this way, as the constitution of the table was different with the condition of the target business, it was necessary to combine various kinds of query results to make the final outputs such as the financial statements and so on.

In the application system, to solve this problem, we composed temporary files of the batch processing by the work tables, which are usually composed by the sequential access method (SAM) file. And, in the whole batch processing, we processed data by the query function of the database, to simplify each individual query procedure and maintain its performance. For example, as for the above-mentioned bank transfer, we queried the vehicle table by the correction query and queried the account data table by the snapshot of the bitemporal database on the other hand. Afterward, as shown in Figure 10, we combined these results by utilizing the query function of the database in the temporary processing executed next.

5 EVALUATION

5.1 Evaluation about Systems Operation

In the application system, online entry could not be suspended during business hours, because the certificates reflecting the entry data had to be published immediately as shown in Figure 8. On the other hand, conventionally, the batch processing using the data that took time until its entry or was not including the valid time data, could not be executed concurrently with the online entry. So, it had to be executed at the overtime like "batch processing 3" or "4" of (1) of Figure 11.

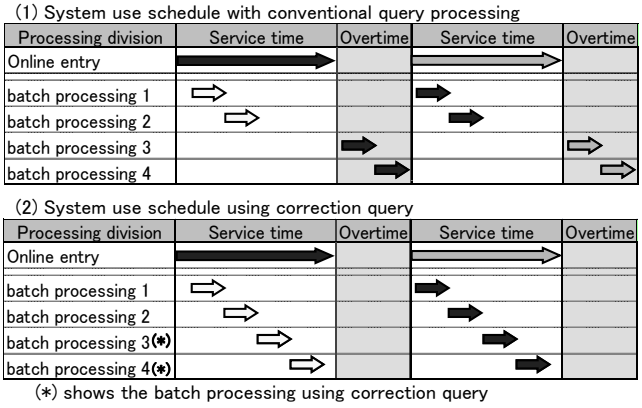

Figure 11: Reduction of overtime batch processing

Table 1: Application rate of correction query.

No	business	total	T_a	T_d
(a)	resident	36	32(89%)	18(50%)
(b)	tax	72	58(81%)	31(43%)
(c)	welfare	40	37(93%)	24(60%)
(d)	office	63	42(67%)	8(13%)
	sum	211	169(80%)	81(38%)

In contrast, as the batch processing like this became able to be executed concurrently with the online entry by utilizing the correction query in the application system, it could be executed during the business hours on the next day as shown in (2) of figure 11.

As a result, all the batch processing to query database were executed during the business hours, and the overtime work could be reduced. Incidentally, the confirmation and the correction entry were also executed at the same time.

5.2 Evaluation about Coverage

Table 1 shows the application table number and rate of the correction query in the application system. We added the addition time T_a to the tables to manage the records with the transaction time; and we added the deletion time T_d to the tables for the correction in addition to T_a. Therefore, the rate of the column T_d of Table 1 is the application rate of the correction query. Here, the row number is the same as the subsystem classification number shown in section 4.1. And, it targets only the transaction table, so it excludes the following tables: the master tables such as the parameter table and the code table; the temporary data tables such as the work table; the derivation datas table such as the total sum.

Here, the table rate to have the addition time is 80%; the table rate to have the deletion time is 38%. That is, the correction query was applied to about 50% of the tables that manage the records. Here, the application rate depended on the subsystem. It was applied to only the 13% tables in the subsystems about the city office; on the other hand, it was applied to from the 43% to 60% tables in the other subsystems.

As shown in section 4.4, the queries with a wide range of conditions were necessary in the actual system operation. Table 2 shows the evaluation of the query method for these query condition. In addition, it shows the kind of the database corresponding to the query method, too. In table 2, "◯" shows that batch query processing can be executed during the online entry; "×" shows that there is the problem to execute the processing. The conventional query methods, i.e. the multiversion concurrency control and the snapshot of bitemporal database, have the problem for the query condition as of the designated transaction time with correction. By the correction query, we could execute the batch query processing even in the above-mentioned condition.

On the one hand, the multiversion concurrency control is necessary to query the tables of the snapshot database; the snapshot of the bitemporal database is necessary to query as of the designated valid time reflecting correction entry. Therefore, it is necessary to make the batch processing such a structure that can combine these query results for making the final output as shown in Figure 10.

5.3 Evaluation about Implementation

For the correction query was implemented in the query processing as shown in Figure 4, the online entry processing was same as before. And, as for the database table, we could implement the correction query easily, because we implemented the transaction time using the expression shown in (b) of Figure 7. For example, the correction query shown in Figure 9 could be executed by the following simple SQL.

$$select\ ID,\ Owner,\ T_a,\ T_d\ from\ Vehicle\ Table \\ where\ T_a \leq 5/6\ and\ T_d = now \quad (6)$$

In addition, there is the thing that plural history data are queried if T_d is designated as the past, not now. In this case, the history data that has earliest T_d becomes the query target. However, in the actual system operation, T_d was usually designated at "now", that is the time when the batch processing was executed. Therefore, such operation was unnecessary.

As shown in section 5.2, it is necessary to query the database in a wide range of conditions corresponding with the business needs and to combine these results to make the final output. For this problem, in the application system, we took the constitution of batch processing, in which we used the database work table instead of the SAM file as shown in Figure 10. As a result, we could combine them easier by using SQL function. By adopting the above-mentioned constitutions, in the application case to the local government of a population of about 40 thousand, the performance deterioration of query and online entry didn't occur comparing with the conventional method.

6 CONSIDERLATION

By the correction query, the problem of conventional query method, that is the query condition as of the designated transaction time with correction during online entry, could be solved. As the result of having applied it to an actual mission-critical

Table 2: Evaluation of query method with query time condition.

Target database	Query method	As of query start time	As of designated valid time with correction	As of designated transaction time with correction
Multiversion concurrency control	Snapshot database	○	×	×
Snapshot	Bitemporal database	○	○	×
Correction Query	Transaction time database	○	×	◯

system, we confirmed the effect that the overtime batch processing to query the database became unnecessary. In recent years, such the operation of mission-critical systems is increasing because of the rapid development of the internet business such as the electronic commerce, the electronic government and so on, in which users directly enter their data to the systems and the online entry cannot be suspended. So, the batch processing has to be executed in the online entry service time. Therefore, we consider that the correction query is effective, by which we can execute the batch query processing without suspending the online entry.

In the actual mission-critical systems, a wide range of data management and data query are necessary based on the business needs. So, it is necessary that the database can be queried by plural methods, and the final output has to be made by combining these query results. In particular, querying the database containing records is complicated. So, the method to meintain query performance is important. Therefore, we consider that our proposal method is effective: the implementation of the transaction time by the proposed expression; the method using the work table to process the data step by step by utilizing the database function to simplify each query and combine their results to make final output.

The application rate of the correction query deeply depends on the subsystems as shown Table 1. Excepting the subsystems about the city office, because the subsystems deal the data based on the reports of real world, the wrong entry data has to be corrected as shown by the light vehicle business in section 4.3. On the other hand, as for the subsystems about the city office, the reports were often omitted in the business. For example, the slips of the financial accounting subsystem were managed in the database. So, when an approval slip was wrong, the new split was published for its adjustment. Therefore, we consider that the correction query is effective for the system that needs the internal correction to consistent its data with the state of the real world.

7 CONCLUSION

As for the system that takes time until the state of real world is reflected into its database, the batch processing is often executed using the data entered by the designated time. However, in this case, when the entry data is corrected, the integrity of the query result of the batch processing cannot be maintained during the online entry by the conventional query method. In this paper, we propose the correction query to query the data entered by the designated time with reflecting the corrections entered after the time. Moreover, we applied this to the mission-critical system and confirmed the effect to reduce the overtime batch processing in the actual systems operation.

Future study will focus on the development of the method, by which database can be updated with a large quantity of data in a lump during the online entry.

REFERENCES

[1] L. Bækgaard and L. Mark, “Incremental Computation of Time-Varying Query Expressions,” IEEE Trans. knowledge and Data Eng., Vol. 7, No. 4, pp. 583–590 (1995).

[2] P. A. Bernstein and N. Goodman, “Multiversion Concurrency Control-Theory and Algorithms,” ACM Trans. on Database Sys., Vol. 8, No. 4, pp. 465–483 (1983).

[3] E. F. Codd, “Extending the database relational model to capture more meaning,” ACM Trans. on Database Sys., Vol. 4, No. 4, pp. 397–434 (1979).

[4] N. Edelweiss, P. N. Hübler, M. M. Moro and G. Demartini, “A Temporal Database Management System Implemented on top of a Conventional Database,” Proc. International Conference of the Chilean Computer Science Society, pp. 58–67 (2000).

[5] J. Gray and A. Reuter, “Transaction Processing: Concept and Techniques,” Morgan Kaufmann, San Francisco (1992).

[6] C. S. Jensen, L. Mark and N. Roussopoulos, “Incremental Implementation Model for Relational Database with Transaction Time,” IEEE Trans. knowledge and Data Eng., Vol. 3, No. 4, pp. 461–473 (1991).

[7] C. S. Jensen, C. E. Dyreson and et al., “The Consensus Glossary of Temporal Database Concept – February 1998 Version, Temporal Database: Research and Practice.” (the book grow out of a Dagstuhl Seminar, June 23–27, 1997), Lecture Notes in Computer Science 1399, Springer–Verlag, pp. 367–405 (1998).

[8] C. S. Jensen and R. T. Snodgass, “Temporal Data Management,” IEEE Trans. knowledge and Data Eng., Vol. 11, No. 1, pp. 36–44 (1999).

[9] T. Kudou, M. Ishino, K. Saotome, N. Kataoka and T. Mizuno, “Implementation of Integrity Maintenance

Method of Query Result by Bitemporal Database," International Journal of Infomatics Society, Vol. 1, No. 1, pp. 16–26 (2009).
[10] A. Motro, "Integrity = validity + completeness," ACM Trans. on Database Sys., Vol. 14, No. 4, pp. 480–502 (1989).
[11] G. Özsoyoğlu and R. T. Snodgrass, "Temporal and Real-Time Databases, A survey," IEEE Trans. knowledge and Data Eng., Vol. 7, No. 4, pp. 513–532 (1995).
[12] L. Shrira and H. Xu, "SNAP: Efficient Snapshots for Back-in-Time Execution," Proc. 21st International Conference on Data Engineering., pp. 434–445 (2005).
[13] R. Snodgrass and I. Ahn, "Temporal Databases," IEEE COMPUTER, Vol. 19, No. 9, pp. 35–42 (1986).
[14] B. Stantic, J. Thornton and A. Sattar, "A Novel Approach to Model NOW in Temporal Databases," Proc. 10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic, pp. 174–180 (2003).

(Received August 24, 2010)
(Revised April 24, 2011)

Tsukasa Kudo received the M. Eng. from Hokkaido University in 1980 and the Dr. Eng. in industrial science and engineering from Shizuoka University, Japan, in 2008. In 1980, he joined Mitsubishi Electric Corp. He was a researcher of parallel computer architecture, an engineer of application packaged software and business information systems. Since 2010 he is a Professor of Shizuoka Institute of Science and Technology. Now, his research interests include database application and software engineering. He is a member of IEIEC, Information Processing Society of Japan and The Society of Project Management.

Yui Takeda received the B.E. from Keio University, Japan in 1987. In 1987, she joined Mitsubishi Electric Corp. She was an engineer of artificial intelligence and application software. Since 2001, she joined Mitsubishi Electric Information Systems Corp. Now, she manages intellectual property rights.

Masahiko Ishino received the master's degree in science and technology from Keio University in 1979 and received the Ph.D. degree in industrial science and engineering from graduate school of Science and technology of Shizuoka University, Japan, in 2007. In 1979, he joined Mitsubishi Electric Corp. Since 2009, he is Professor of Fukui University of Technology. Now, His research interests include Management Information Systems, Ubiquitous Systems, Application Systems of Datamining, and Information Security Systems. He is a member of Information Processing Society of Japan, Japan Industrial Management Association, Japan Society for Management Information.

Kenji Saotome received the B.E. from the Osaka University, Japan in 1979, and the Dr.Eng in Information Engineering from the Shizuoka University, Japan in 2008. From 1979 to 2007, he was with Mitsubishi Electric, Japan. Since 2004, he has been a professor of Hosei business school of innovation management. His current research areas include LDAP directory applications and single sign-on system. He is a member of the Information Processing Society of Japan.

Kazuo Mutou received the master's degree in precision engineering from Yamanashi University in 1982 and received the Ph.D. degree in mechanical system engineering from graduate school of Science and technology of Tokyo University of Agriculture and Technology in 1993, in 1982, he was with Polytechnic University. Since 2008, he is Assistant Professor of Shizuoka Institute of Science and Technology. Now, His research interests include CAD/CAE/CAM/CAT Systems, MES Systems, and Digital Manufacturing Systems, etc. He is a fellow of Society of Automotive Engineers of Japan and a member of the Japan Society for Precision Engineering, etc.

Nobuhiro Kataoka received the Ph.D. in information science from Tohoku University. Since he jointed Mitsubishi Electric Corporation he has been engaged development of software engineering, and computer system design. He is currently a professor at School of Information Technology and Electronics Tokai University in Japan. His research interests is modeling for Information system development.