ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

빅데이터 환경에서의 B-tree 구조 기반 링크정보 관리서버의 개발☆

A Study on the Link Server Development Using B-Tree Structure in the Big Data Environment

박 승 범[1] 황 종 성[2] 이 상 원[3*]
Sungbum Park Jong Sung Hwang Sangwon Lee

요 약

주요 기업들과 포털들은 사용자들에게 웹 기반 환경에서 보다 효율적인 콘텐츠 이용을 지원하기 위해 이른바 콘텐츠관리시스템(CMS, Contents Management Systems)과 콘텐츠의 데이터베이스 내 물리적 주소를 연결하여 관리하는 링크 서버를 적극적으로 도입하고 있다. 이를 통해 웹브라우저 화면에서 보여지는 콘텐츠의 URL과 실제 데이터베이스 안의 콘텐츠의 물리적 주소를 자동으로 연결해 주고, URL이나 데이터베이스의 물리적 주소의 변경시 두 주소를 재 연결하는 역할을 수행한다. 최근 빅데이터 환경의 도래에 따라 디지털 콘텐츠와 사용자 접속수가 폭발적으로 증가하고 있는 상황에서 CMS와 링크 서버에서 수행해야 하는 유효 링크 검사 횟수도 따라서 증가하고 있다. Peta-Byte 또는 Eta-Byte 환경 하에서 수행되는 유효 링크 검사를 기존 URL 기반의 순차적 방식으로 수행할 경우 속도저하에 따른 데이터 링크 식별률(identification rate)의 저하와 빈번한 링크 검사에 따른 데이터베이스에 부하를 주는 요인으로 작용될 수 있다. 따라서, 본 연구는 상기와 같은 종래의 문제점을 해결하기 위해 대량의 URL에 대해 B-Tree 기반의 정보 식별자의 구간별 개수 분석을 기반으로 URL 삭제 링크 및 추가 링크를 인식하고 효과적으로 관리하는 것이 가능하도록 해주는 링크 서버를 제공하는 데 있다. 본 연구를 통해 기존 방식보다 빠르고 낮은 부하를 주는 데드 링크 체크 처리가 가능해 질 것이다.

☞ 주제어 : 링크서버, B-tree 구조, 빅데이터, 콘텐츠 관리 서버

ABSTRACT

Major corporations and portals have implemented a link server that connects Content Management Systems (CMS) to the physical address of content in a database (DB) to support efficient content use in web-based environments. In particular, a link server automatically connects the physical address of content in a DB to the content URL shown through a web browser screen, and re-connects the URL and the physical address when either is modified. In recent years, the number of users of digital content over the web has increased significantly because of the advent of the Big Data environment, which has also increased the number of link validity checks that should be performed in a CMS and a link server. If the link validity check is performed through an existing URL-based sequential method instead of petabyte or even etabyte environments, the identification rate of dead links decreases because of the degradation of validity check performance; moreover, frequent link checks add a large amount of workload to the DB. Hence, this study is aimed at providing a link server that can recognize URL link deletion or addition through analysis on the B-tree-based Information Identifier count per interval based on a large amount of URLs in order to resolve the existing problems. Through this study, the dead link check that is faster and adds lower loads than the existing method can be performed.

☞ keyword : Link Server, B-Tree, Big Data, Content management Server

1. Introduction

Major corporations and portals operate link servers in order to manage digital content effectively in web-based environments. A link server, which was developed for linked information, was designed to store mapping information between content in physical storage devices, such as databases (DBs), and the URLs linked to the information in such DBs. In particular, a link server automatically connects the physical address of content in a DB to the content URL shown through the web browser screen, and re-connects the URL and the physical address when either is modified. Such linked

[1, 2] Big Data Strategy Center, National Information Society Agency Seoul 100-775, Korea (parksb@nia.or.kr, jshwang@nia.or.kr)
[3] Division of Information and Electronic Commerce, Wonkwang University Iksan 570-749, Korea
[*] Corresponding author(sangwonlee@wku.ac.kr)
[Received 17 November 2014, Reviewed 18 November 2014, Accepted 24 February 2015]
☆ This paper was supported by Wonkwang University in 2014.
☆ A preliminary version of this paper was presented at APIC-IST 2014 and was selected as an outstanding paper.

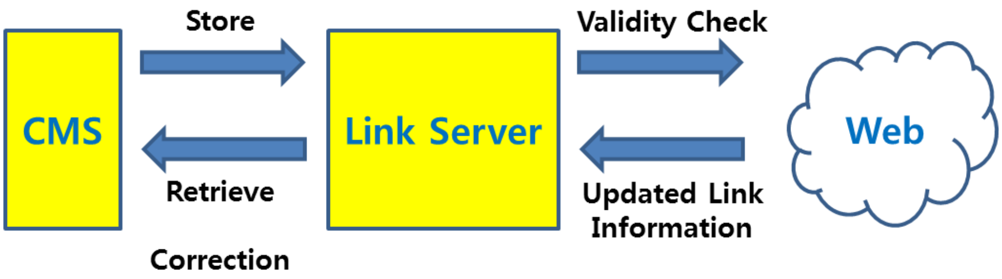
http://dx.doi.org/10.7472/jksii.2015.16.1.75

information is stored after being divided into nested and reference links. Currently, because a Content Management System (CMS) manages nested and reference links as a block unit, the DB contains information related to such blocks, as well.

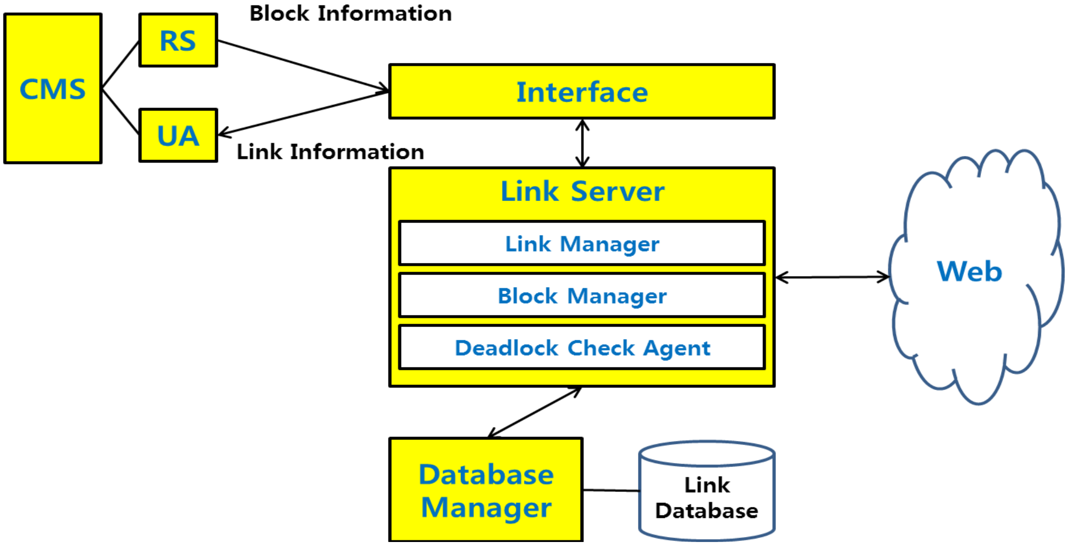
Link servers are used with CMS because of various advantages. Links can save the space required for large amounts of multimedia content. Links can represent many different versions, in addition to different media formats, for the same content. Because link servers reduce network traffic by transmitting only linked information instead of real content, performance improvement can be achieved. Thus, a link server can be utilized in many ways.

As such, link servers store and manage link-related information used for digital content independently. That is, once new digital content is stored, its related linked information is stored in a DB.

Then, upon an information request from search servers or user agents for such linked information, the link server actively transmits the request results individually by structuring these results. Because of the recent advent of Big Data environments, the amount of digital content and user connections have increased significantly. In such a context, link servers, which show more efficient storage performance and facilitate rapid extraction of desired information by storing linked information in the DB instead of files, should be developed. In addition, a faster and more efficient check method, compared to the existing dead link check, is required to maintain consistently



(Figure 1) Link server function



(Figure 2) Link server structure

reliable CMS services by checking the validity of links periodically through the dead link check agent, which can be smoothly applied even in petabyte or etabyte storage spaces.

This paper is organized as follows. Chapter 2 introduces previous studies on link servers and dead link management via link agents. Chapter 3 explains a B-tree-based data link check agent. Chapter 4 describes a representative case of the B-tree-based data link check. Finally, Chapter 5 presents the conclusions of this paper.

2. Related Works: Dead Link Management through Link Server and Link Agent

2.1 Link Server Function

The digital content provided in a CMS does not provide practical content to a document, but consists of formats that infer or refer to an entire or partial document using a link. A link server separates, stores, and manages the linked information of virtual documents.

Fig. 1 shows the relationship between a link server and a CMS in which the linked information of the web document pointed by the link server is managed to be valid. A CMS transmits a request for storing the linked information to a link server; subsequently, the link server extracts the required linked information and transmits it to the CMS. In addition, the link server indicates error details to the CMS when the linked information is invalid. Because the linked information refers to a certain portion of documents accessed over the web, invalid linked information might occur occasionally. A link server checks the validity of the linked information periodically and stores the updated information in a DB. A dead link check agent maintains and checks the validity of the linked information.

2.2 Structure of Link Server and Dead Link Check Agent

Linked information stored in a link server is transmitted from a user agent. That is, a user agent processes newly created virtual documents and extracts linked information accordingly. Once a link server is requested to store linked information extracted by a user agent, a link server accepts the request and stores the linked information. Once a search server requests the linked information, the stored information is transmitted to the search server by extracting the link in a DB via the link server. The boxes with lines in Fig. 2 are operated through an interface for the interoperation with an existing CMS. A link server broadly consists of a link manager, a block manager, a DB manager, and a data link check agent. The link manager stores links in a DB by classifying them into nested or reference links while storing, modifying, or deleting newly created links. The block manager does not manage individual linked information, but manages nested and reference links through block IDs. The DB manager was designed to process all the DB requirements. The data link check agent confirms the link validity periodically, records errors once problems are found, and notifies subsequent executions about the errors. The data link check agent also actively searches and manages broken URLs.

2.3 Algorithm Design for Performance Improvement of Data Link Check Agent in Big Data Environment

The existing technologies are used frequently to check the URLs of files located in servers that provide various services regarding web documents. Such technologies are generally related to Hypertext Transfer Protocol (HTTP) request technologies for corresponding URLs.. A URL in a web server, which needs to be checked, is called through programs that support the HTTP protocol. Then, the return code is analyzed; if it is 200, the URL is recognized as a valid URL. Because the current technologies employ sequential search methods when searching for corresponding URLs, they require a substantial amount of time for the search. In addition, while searching corresponding URLs, a considerable workload is applied to the DB because an entire set of data is scanned.

These existing URL check methods can be used smoothly when the number of URLs to be checked is small. However, if the number of URLs is large, the checking process requires a substantial amount of time. Moreover, because of firewall systems in the case of web servers located in external institutions that monitor network connections from the outside and block illegal or too many connections, a normal URL check cannot be performed often.

In order to resolve the aforementioned problems, this study is aimed at providing a deadlock check agent that can recognize URL link deletions or additions through analysis on the B-tree-based Information Identifier count per interval based on a large amount of URLs.

The study also intends to provide a deadlock check agent that can process high speed validity checks through URL management via a DB that stores Information Identifiers in an internal network, not an external network, which is a limitation of the existing HTTP request mode experienced for a large amount of URLs

3. Data Link Check Agent Based on B-Tree

This study designs a deadlock check agent that performs rule-based URL analysis and HTTP communication with a data link check agent that analyzes and records B-tree-based Information Identifier count per interval. This agent also includes a link server that provides the communication frequency, the URL management rule, and the date of the URL management rule update requested from the data link check agent according to the communication frequency set. In addition, the data link check agent checks the errors of the URL management rule and deletes/adds URL information to a CMS according to the set frequency, thereby transmitting the results to the link server. The link server manages the URL information. That is, it performs updates and records errors in the URL management rule and the URL information that was deleted/added.

A data link check agent is installed in a CMS or a specific server that can be connected internally to the CMS. That is, once a communication error occurs with the link server, the information that was deleted/added (ID) is recorded in a log and transmitted to the link server. The link server manages the URL information. In particular, the Domain information from the URL information is managed uniquely, whereas the Prefix, ID, and Suffix are managed by regular expression.

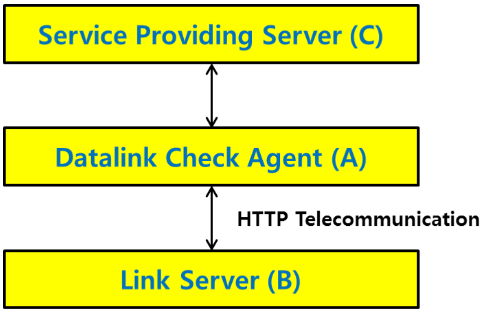
The data link check agent performs a rule-based check analysis on the four parts of the URL, which consists of "URL = Domain + Prefix + ID + Suffix". Here, the data link agent assumes that no change occurred in the web program. That is, only the ID mapping field in the DB that corresponds to said ID is checked to replace the entire URL check, whereas the Information Identifier is defined as a value of the ID mapping field.

The data link check agent is utilized as a pre-condition of the Information Identifier when the B-tree-based Information Identifier count per interval is analyzed with regard to URL information. The Information Identifier is any arbitrary character string that has the form of sequential or constantly increasing or decreasing values, and it is represented by character or numerical data types, including char, varchar, string, integer, double, and float. The Information Identifier can be represented by literal or numeric values that can be compared, such as ">, =, <"; however, the preference is for a combination of multiple fields.

The data link check agent analyzes and records the Information Identifier count per interval using a *node* that represents the serial number that follows the B-tree node navigation; *min*, which represents the minimum value in an interval; *max*, which represents the maximum value in an interval; *count (min..max)*, which represents the Information Identifier count between min and max; *min (F)*, which represents the minimum value of the Information Identifier; *max (L)*, which represents the maximum value of the Information Identifier; and *min(i)* and *max(i)*, which represent the next node value (one of the min, max, and mid values) that will be referred over the B-tree node navigation.

The data link check agent recognizes the deleted/added information through n + 1 queries at maximum when the information count is 2^n based on the B-tree structure, and determines whether the information is deleted/added by comparing the analyzed information of count per interval recorded initially and count (ID).

Fig. 3 shows the schematic diagram that describes the deadlock check agent according to one preferred example of this study, whereas Fig. 4 shows a process flow diagram that illustrates the operation of the deadlock check agent according to the preferred example. As expressed in the figures, the deadlock check agent broadly consists of the data link check agent (A) and the link server (B) that perform the HTTP communication by default.



(Figure 3) Deadlock check agent structure

4. Example of B-Tree-Based Data Link Check

The B-tree-based data link check performs a rule-based URL analysis and analysis on the B-tree-based Information Identifier count per interval; then, the B-tree-based data link check records the analysis results. Subsequently, it checks errors of the URL management rule in the CMS (C) and deleted/added URL information according to the set frequency, thereby transmitting the check results to the link server (B). First, the URL is divided into four parts as shown below. Here, the analysis process of the rule-based URL management is explained as follows:

(1) Domain is a character string registered in the Internet domain name server, which cannot be modified; otherwise, the corresponding sites are closed, integrated, or separated. (2) Prefix is a character string that appears before ID. This is created by a web program and helps map the ID and a specific field (hereafter referred to as the ID mapping field) in a DB. (3) ID is a character string that identifies information that corresponds to a value of the ID mapping field and can also correspond to multiple fields. (4) Suffix is a character string that appears after ID. This is created by a web program and has no relationship to the ID mapping field.

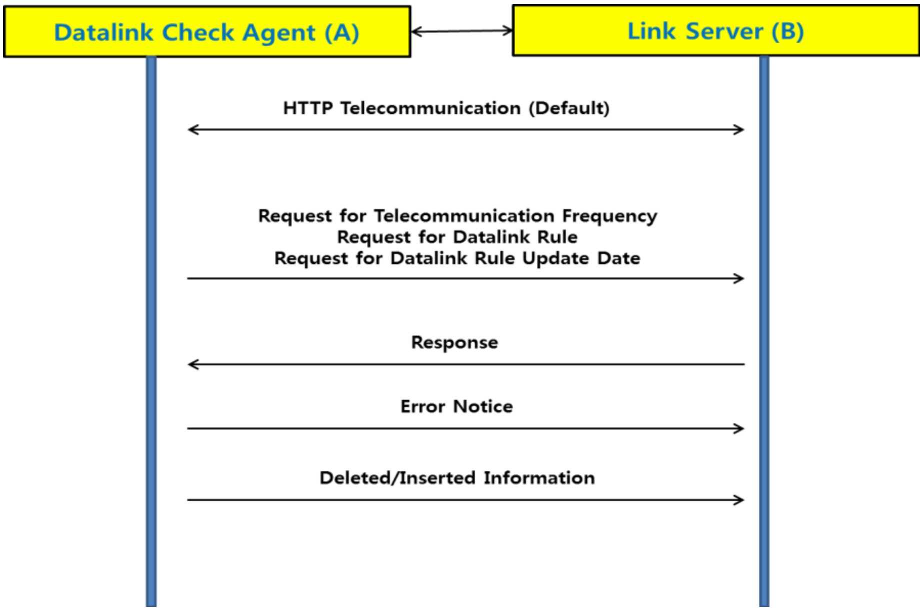
For example, in the URL www.knowledge.go.kr/SerarchSF1/search_view.jsp?mdno=15426388, Domain is www.knowledge.go.kr, Prefix is "/SearchSF1/search_view.jsp?mdno=", ID is "15426388", and there is no Suffix.

Checks through the analysis of the four URL parts can be replaced by the checks described below if the condition where the corresponding URL is neither redirected nor forwarded is satisfied. (1) In the case where only the ID modification is found, only the ID mapping field is checked. (2) In the case of Prefix modification (including ID modification), which is a case where another web program is used or the parameter (the character string that is within the Prefix and is delivered to a program) must be checked for ID mapping in a DB, it is essential to check the ID mapping field in consideration of such a case so that this modification can be transmitted to the link server (B), thereby updating the required items. (3) In the case of Suffix modification (including ID modification), which refers to the case where the same web program is used, only the ID mapping field is checked. As explained above, the rule-based URL management check in this study checks only the ID mapping field that corresponds to the ID if a modification is not made to the web program, which removes the necessity for an entire URL check. The Information Identifier is the ID mapping field value.

URL = Domain + Prefix + ID + Suffix

Moreover, the analysis procedure for the B-tree-based Information Identifier count per interval can be described as follows. First, the pre-condition of the Information Identifier for analysis of the B-tree-based Information Identifier count per interval is as follows: (1) The Information Identifier is any arbitrary character string that has the form of sequential or constantly increasing or decreasing values. (2) The Information Identifier is represented as character or numerical data types, including char, varchar, string, integer, double, and float. (3) The Information Identifier can be represented by literal or numeric values that can be compared, such as ">, =, <". (4) The Information Identifier can be a combination of multiple fields.

In the analysis and recording method of the Information Identifier count per interval, a node represents the serial number that follows a B-tree node navigation, whereas min represents the minimum value in an interval and max represents the maximum value in an interval. In addition, count (min..max) represents a median value calculated by dividing count by two and rounded, whereas min (F) represents the minimum value of the Information Identifier and max (L) represents the maximum value of the Information Identifier. Further, min(i) and max(i) represent the next node value (one of min, max,


Datalink Check Agent (A)
Link Server (B)
HTTP Telecommunication (Default)
Request for Telecommunication Frequency
Request for Datalink Rule
Request for Datalink Rule Update Date
Response
Error Notice
Deleted/Inserted Information

(Figure 4) Deadlock check agent procedure

(Table 1) Information identification notation

node	min	max	count	mid
1	min(F)	max(L)	count(min..max)	mid(min..max)
⋯	Subsequent operations follow the B-Tree Node Navigation algorithm.			
n	min(i)	max(i)	count(min..max)	mid(min..max)

and mid values) that will be referred over the B-tree node navigation. Table 1 lists the analysis and recording method of the Information Identifier count per interval. records. Because the SQL statement is a general one, it can be performed periodically without affecting existing service operations.

A data link check agent (A) performs the aforementioned rule-based URL analysis and the analysis on the information identifier count per interval based on the B-Tree algorithm followed by recording. This data link check agent (A) is installed in the CMS (C) that provides the URL information, or in a specific server that can be connected internally with the CMS (C). When the data link check agent (A) encounters errors communicating with the link server (B), it records a log concerning the deleted/added information (ID) and retransmits the update to the link server (B). The link server (B) performs an HTTP communication with the data link check agent (A) and provides the communication frequency, the URL management rule, and the date of the URL management rule update requested from the data link check agent (A).

In addition, the link server (B) manages the URL information. That is, it performs updates and records the errors in the URL management rule and the deleted/added URL information. As mentioned above, the link server (B) manages the URL information; in particular, the Domain information from the URL information is managed uniquely, whereas the Prefix, ID, and Suffix are managed by regular expression.

Therefore, this study provides an effective URL check method using a DB installed in an internal network, not in an external network, to store the Information Identifiers of a large amount of URLs. Such external storage is considered a vulnerability of the existing HTTP request method. This way, accurate management and maintenance over internal and external URLs of CMS (C), as well as rapid checks over a large amount URLs, is provided. Moreover, the scope of this study is limited to URLs generated by web programming languages such as JSP, ASP, PHP, and CGI Script.

5. Conclusions

In recent years, the amount of digital content and the number of users that connect over the web to use such digital content have increased significantly because of the advent of Big Data environments; such increase leads to a growth in the number of link validity checks that should be performed in CMS and link servers. This study aimed to provide a link server that can recognize URL link deletion/addition through the analysis of the B-tree-based Information Identifier count per interval based on a large amount of URLs in order to resolve the aforementioned problems. Through this study, a dead link check that is faster and applies lower loads than existing methods was achieved. In order to explain the main idea of this study, some representative cases were described in this study. However, this study is not limited to such examples, and without departing from the spirit of the technical concept, various modifications, additions, and substitutions are possible. Therefore, the examples disclosed in this study do not aim to limit the technical concept of this study, but to explain it more clearly. Accordingly, the technical scope of this study cannot be limited to such examples. In order to understand the infrastructure of B-tree structure in the environment of Big Data, we should research on performance of Link Information Management Servers. And, further studies on B-Tree-Based Data Link Check should be performed. Performance evaluation of deadlock check agent structure would be certainly helpful.

Reference

[1] R. Ramakrishnan and J. Gehrke, "Database Management Systems," McGraw-Hill, 2014. doi:10.1109/2.869369

[2] T. Connolly and C. Begg, "Database Systems: A Practical Approach to Design, Implementation, and Management," Addison-Wesley, 2014. doi:10.1287/isre.6.2.118

[3] C. J. Date, "An Introduction to Database Systems," Addison-Wesley, 2013.

[4] C. S. Jensen, D. Lin and B. C. Ooi, "Query and update efficient B+-tree based indexing of moving objects," VLDB '04 Proceedings of the Thirtieth International Conference, Vol. 30, pp. 768-779, 2004. doi:10.1145/342009.335427

[5] H. Berliner, "The B* Tree Search Algorithm: A Best-First Proof Procedure," Artificial Intelligence, Vol. 12, Iss. 1, pp. 23-40, 1979. doi:10.1016/0004-3702(79)90003-1

[6] R. Bayer, "The universal B-tree for multidimensional indexing: General concepts," Worldwide Computing and Its Applications (Lecture Notes in Computer Science), Vol. 1274, pp. 198-209, 1997. doi: 10.1007/3-540-63343-X_48

[7] S Wu, D Jiang, B. C. Ooi and K. L. Wu, "Efficient b-tree based indexing for cloud data processing," Vol. 3, Iss. 1-2, pp. 1207-1218, 2010. Doi:10.14778/1920841.1920991

[8] K Kousha and M Thelwall,"Google Scholar Citations and Google Web/URL Citations: A Multi-Discipline Exploratory Analysis," Journal of the American Society for Information Science and Technology, Vol. 58, Iss. 7, pp 1055?1065, 2007. doi:10.1002/asi.v58:7

[9] V. Mayer-Schonberger K. Cukie, "Big Data: A Revolution that Will Transform how We Live, Work, and Think," Houghton Mifflin Harcourt, 2014. doi: 10.2501/IJA-33-1-181-183

[10] H. Chen, R. H. L. Chiang and V. C. Storey, Business Intelligence and Analytics: From Big Data to Big Impact," MIS Quarterly, Vol. 36, No.4, pp. 1165-1188, 2012. http://dl.acm.org/citation.cfm?id=2481683

[11] J. Bughin, M. Chui and J. Manyika, "Clouds, Big Data, and Smart Assets: Ten Tech-Enabled Business Trends to Watch," McKinsey Quarterly, Vol. August, 2010. http://www.mckinsey.com/insights/high_tech_telecoms_internet/clouds_big_data_and_smart_assets_ten_tech-enabled_business_trends_to_watch

[12] E Turban, L. Volonino and G. R. Wood, "Information Technology for Management: Advancing Sustainable, Profitable Business Growth," Wiley, 2014. http://as.wiley.com/WileyCDA/WileyTitle/productCd-EHEP002524.html

[13] E. Turban, J. E. Aronson and T. P. Liang, "Decision Support Systems and Intelligent Systems," Prentice Hall, 2014. doi:10.2307/249703

[14] L. C. Zhong and J. M. Rabaey, "An Integrated Data-Link Energy Model for Wireless Sensor Networks," 2004 IEEE International Conference on Communications, Vol. 7, pp. 3777-3783, 2004. Available at your request. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1313260&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1313260

◐ 저자소개 ◑

박 승 범 (Sung Bum Park)

received his MS degree in management information and his PhD degree in management science from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 2002 and 2011, respectively. Since 2002, he has been working as an executive principal researcher for the National Information Society Agency, Seoul, Rep. of Korea. His current research interests include new media, performance evaluation of information systems and digital content distribution.

황 종 성 (Jongsung Hwang)

is the Head of Gov 3.0 center of National Information Society Agency and a member of presidential Gov 3.0 committee in Republic of Korea. He has worked for IT policy development and project management for the Korean government since he joined NIA in 1995. His experiences includes e-government, smart city, and geospatial information infrastructure. In particular, he served as a CIO, assistant mayor, of Seoul metropolitan government during 2011 to 2013, launching open Gov 2.0 initiatives of Seoul. He has received his master (1987) and doctoral degree (1994) in political science from Yonsei University in Seoul.

이 상 원 (Sangwon Lee)

received his PhD degree in management engineering from the Korea Advanced Institute of Science and Technology, Seoul, Rep. of Korea, in 2002 and 2009, respectively. From January of 1995 to January of 2000, he worked for Daewoo Information Systems, Co., Ltd., Seoul, Rep. of Korea. From March of 2010 to February of 2011, he worked as a research professor for Ewha Womans University. Since March of 2011, he has been an assistant professor with the Department of Information and Electronic Commerce, Wonkwang University, Iksan, Rep. of Korea. His research interests are data design and data analysis.