Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.5, Oct. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.5.817

온라인 거래 장애 방지를 위한 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스 연구

김 정 환,† 고 무 성, 이 경 호‡
고려대학교 정보보호대학원

A Study on SQL Performance-Based IT Application Change Management Process to Prevent Failures of Online Transactions

Jeong-hwan Kim,† Moo-seong Ko, Kyung-ho Lee‡
Graduate School of Information Security, Korea University

요 약

금융기관 및 통신사 등 대량의 데이터를 다루는 회사에서 테스트환경은 구축 비용문제로 인한 스토리지 용량의 한계 및 중요정보 컬럼의 변환 등으로 운영환경과 항상 동일한 것은 아니다. 따라서, 이러한 테스트환경과 운영환경이 동일하지 않아 발생하는 SQL 성능저하는 예상치 못한 거래장애로까지 연결되어 기업의 금전적 손실 및 고객민원, 신뢰도 하락을 발생시키는 중요 원인이 되고 있다. 그동안 SQL과 관련한 성능연구는 DBMS Optimizer와 관련한 튜닝연구 중심으로 이루어져, 이러한 문제에 대해서는 다루어지지 않았다. 따라서, 본 논문에서는 테스트환경과 운영환경의 차이로 발생하는 SQL의 성능저하에 따른 온라인 거래장애 방지를 위해 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 제시하여 그 효과성을 검증한다.

ABSTRACT

Test environment on the company that handles a large amount of data such as telecommunications companies and financial institutions, may not always be the same as the production environment, which is caused by conversion of important columns about information and limitation of storage capacity due to the construction cost. Therefore, SQL performance degradation that occurs when the test and production environments are not the same, which is an important cause of connecting to the unexpected failures of online transactions, and it generates financial loss of business, customer complaints, a decrease in reliability. In studies related SQL performance, it has so far been conducted mainly studies of tuning associated with DBMS Optimizer, and it has not been addressed issues of this sector. Therefore, in this paper, I verify the validity about presentation of the advanced SQL Performance-based IT application change management process, in order to prevent failures of the online transactions associated with poor performance of SQL generated by differences in test and production environments.

Keywords: SQL Access Path, SQL Performance-based IT Application Change Management Process, Test and Production Environments, SQL Performance Tuning, Reputational Risk

I. 서 론

금융기관 및 통신사 등 대량의 데이터를 다루는 회사에서 테스트환경은 구축 비용문제로 인한 스토리

접수일(2014년 6월 16일), 수정일(2014년 9월 1일), 게재확정일(2014년 9월 10일)
† 주저자, jarnet2k@korea.ac.kr
‡ 교신저자, kevinlee@korea.ac.kr(Corresponding author)

지 용량의 한계 및 중요정보 컬럼의 변환 등으로 운영환경과 항상 동일한 것은 아니다. 따라서, 이러한 테스트환경과 운영환경의 차이는 비즈니스 운영에 문제를 일으킬 수 있는데, 그 중 온라인 SQL 프로그램의 예상하지 못한 성능저하는 거래장애를 일으켜 기업의 비즈니스 중단 및 고객민원, 신뢰도 하락을 발생시키는 중요 원인이 되고 있다. 이러한 장애는 테스트환경에서의 테스트가 충분히 이루어짐에도 IT 응용프로그램 변경관리 프로세스를 통해 운영환경에 적용하는 과정에서 발생하므로 테스트 과정에서 이를 찾아내기가 어려운 것이 현실이다. 특정 회사의 2011년, 2012년 거래장애 현황 자료를 분석해본 결과 약 18%의 온라인 거래장애가 이러한 원인인 것으로 나타났다.

따라서, 이러한 이유로 운영환경의 거래장애가 발생하는 경우 개발자는 테스트환경에서 테스트를 충실히 하여 온라인 어플리케이션에 탑재되는 SQL의 성능문제가 없음을 확인했음에도 운영환경에서의 갑작스런 성능저하에 따른 온라인 거래장애에 당황하는 사례가 발생하고, 거래장애 발생에도 DB관리자 등의 SQL 튜닝담당자 입장에서는 프로그램 적용시 운영환경에 대한 SQL 실행계획을 결정하는 예측할 수 없는 DBMS Optimizer의 문제로 해명하고 있는 실정이다. 2013년 한 보도에 따르면 포춘 500대 기업중 71%는 z/OS 기반의 메인프레임을 사용하고, 미국 25개 소매점중 23개사, 전세계 10대 보험사 중 9개사가 메인프레임을 사용하고 있다[1]. 2013년 12월 기준 Gartner의 자료에 따르면 전체 메인프레임 시장의 72%를 z/OS가 점유하고 있고, z/OS는 99.999%라는 강력한 가용성을 바탕으로 DB2를 탑재하여 세계시장에서 대용량 데이터를 다루는 많은 회사에서 사용되고 있다[2]. 따라서, 본 논문에서는 대용량 데이터를 다루는 회사에서 주로 사용되는 z/OS 기반의 DB2 환경에서, 운영환경에서 예상하지 못한 온라인 SQL 프로그램의 성능저하에 따른 거래장애를 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 통해 장애발생전 감지하도록 하여 온라인 거래장애가 유무형의 손실로 이어지는 IT운영리스크의 감소방안을 연구해본다.

II. 선행 연구

2.1 DB2 개요

DB2 데이터베이스 소프트웨어는 z/OS부터 Linux, UNIX 및 Windows 플랫폼에 이르기까지 다양한 환경에서 높은 수준의 성능, 규모 및 신뢰성을 제공한다. DB2의 z/OS 데이터베이스 소프트웨어는 높은 안정성, 가용성 및 확장성을 제공하며, SOA, CRM 및 데이터웨어하우스 등에 최적화되어 있다[3]. 이러한 z/OS 기반의 DB2는 크게 데이터 공유환경의 데이터 처리를 위한 Group Buffer 및 특정 DB내의 데이터 처리를 위한 Local Buffer로 구성된 메모리영역, 데이터 공유 환경에서 데이터의 Consistency를 관리하는 OS 레벨의 CF(Coupling Facility), 데이터 저장공간으로 구성되고 DBMS 기능, Lock 기능, 메모리 관리 기능을 처리하기 위한 Address Space를 할당한다.

2.2 SQL 성능을 결정하는 실행계획 연구

온라인 SQL 프로그램의 성능에서 가장 중요한 것은 SQL의 실행계획이다. 실행계획이란, DBMS 기반에서 프로그램에 탑재된 특정 SQL이 어떠한 경로로 데이터를 접근하는지에 대한 처리단계를 말하며, 이를 'Access Path'라고 한다.

아래 Fig. 1.은 SQL 튜닝을 통해 해당 SQL의 실행계획을 변경함으로써 성능을 개선시킨 사례이다.

Fig.1.은 서브쿼리(subquery)의 과도한 반복수행에 따른 성능저하를 SQL 튜닝을 통해 개선한 경우이다. 이 사례는 'DEPT_MAST' 테이블에 대해 'DEPT_CODE' 컬럼의 값이 중복된 로우가 없다는 조건을 전제로 한다. 개선 전의 경우 테이블에서 해당 조건을 만족하는 데이터를 조회하면서 조건에 만족하는 모든 검색대상 로우에 대해 서브쿼리를 처리

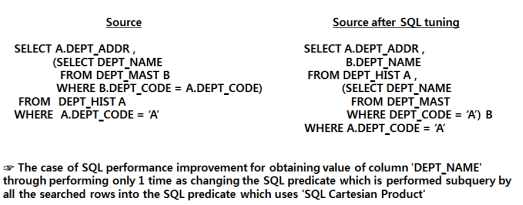


Fig. 1. Case of performance improvement by tuning the SQL using SQL Cartesian Product

함으로써 DBMS의 SQL 분석(parsing) 부하와 테이블의 직접접근(random access) 부하를 유발하여 수행 성능을 저하시킨다. 예를 들어, 해당 조건에 만족하는 데이터 로우 건수가 1,000건이라면 SQL문에서 사용한 서브쿼리를 모두 1,000번 실행하게 된다. 이 서브쿼리 처리를 Fig.1.의 개선후의 경우처럼 필요한 기능만을 1회 수행하도록 하는 SQL 카테시안 곱(Cartesian Product)의 형태로 변경하면 동일한 결과를 출력하면서, CPU 사용율 및 수행시간(elapsed time)을 획기적으로 감소시킬 수 있다. 이 경우처럼, 해당 SQL문이 최적화된 실행계획으로 데이터에 접근하도록 하여 동일한 결과를 출력하면서 SQL의 수행성능을 최적화하는 것이 SQL 튜닝이며, SQL의 실행계획은 위 사례에서와 같이 SQL의 수행성능에 가장 중요한 변수이다.

이는 DBMS 의 Optimizer가 인덱스의 클러스터율(Cluster Ratio)[1], 테이블 로우(row) 건수, 중요 인덱스 컬럼의 Cardinality[2] 등으로 구성된 DB 테이블의 상태 정보를 기반으로 DB2의 경우 Bind 시점에, 오라클의 경우 프로그램 최초 실행시 결정한다. 이를 결정하는 기준방식은 룰기준(RBO: Rule Base Optimizer)과 비용기준(CBO: Cost Base Optimizer)으로 나뉘는데, 룰기준의 경우 절대적 룰인 매칭율 등에 따라 실행계획을 결정하므로 비즈니스가 매우 다양하고 대용량 데이터를 다루는 규모가 큰 회사는 대부분 비용기준을 사용한다. 따라서, 본 논문에서는 DB 테이블의 상태정보를 기준으로 실행계획을 결정하는 비용기준 환경에서 연구를 진행하였다.

메인프레임 DB2 환경에서 SQL의 실행계획은 대부분 특정한 경우를 제외하고 SQL 프로그램 적용시점에 Bind를 수행하여 결정된다. Bind란, 메인프레임 DBMS 환경에서 소스프로그램을 pre-compile을 통해 분리한 SQL 부분에 대한 실행계획을 결정하는 작업을 의미한다.

2.3 테스트 및 운영환경 구성 사례 연구

대용량 데이터를 다루는 회사들의 테스트 및 운영환경 구성의 차이 확인을 위해 DB2 및 비용기준 Optimizer를 사용하는 국내 금융사들의 데이터 관리 방법에 대해 확인하였다. 확인 대상은 계정계 시스템의 데이터 용량이 10TB 이상인 3개 금융 관련 회사에 대해 테스트 DB 적재주기, 스테이징(staging) DB 구성기준, 프로그램 개발자의 테스트환경 테이블에 대한 사용권한을 조사하였다.

Table 1.에서 확인한 바와 같이, 테스트 DB의 적재주기가 약 3개월 단위이고, 프로그램 개발자에게 테스트 DB 테이블에 대한 Read, Insert, Update, Delete 및 Load 권한을 제공하는 경우, 테스트환경은 프로그램 개발자의 필요에 따라 운영환경의 동일 테이블의 환경과 매우 큰 차이가 발생할 수 있다는 것을 의미한다. 예를 들어, 운영환경의 A 테이블의 로우 건수가 1천만건이라고 가정할 때, 테스트환경의 데이터를 적재하는 담당자가 3개월 적재주기가 도래해 해당 운영환경의 전체 데이터(full data)를 중요 정보 컬럼을 변환하여 모두 테스트환경에 적재하였으나, 프로그램 개발자가 업무상 필요에 의해 테스트환경의 동일한 A테이블에서 900만건의 로우를 삭제하였다면 테스트환경의 A테이블은 100만건의 로우만 남게 되어 운영환경의 A테이블과는 큰 차이가 나게 된다. 이 경우, 테스트환경이 운영환경과 매우 달라진 상황에서 테스트환경에서 성능문제가 없었던 SQL 실행계획이 운영환경에서 역시 성능문제가 없다고 보장할 수는 없다. 왜냐하면, 동일한 DBMS 환경에서 동일한 룰의 Optimizer는 동일하거나 매우 유사한 테이블 상태의 통계정보에 대해서만 SQL 실행계획의 성능을 보장하기 때문이다. 이러한 데이터베이스 환경에 따른 SQL의 실행계획이 중요한 이유는 다음의 내용으로 확인할 수 있다.

Table 1. Configurations of the test environment of companies with large volumes of data over 10TB

Financial Corp.	Test DB loading cycle	Staging DB	Authorities of developer
A Corp. & B Corp.	3 months	Use logically separated test DB	Read Insert Update Delete Load
C Corp.	N/A	N/A	Read Insert Update Delete Load

1) 데이터가 특정 컬럼기준으로 정렬된 정도를 비율로 표시한 것으로 데이터의 범위검색에 큰 영향을 미침.
2) 테이블 전체 로우에 대해 특정 컬럼이 갖는 값의 가짓수

Table 2. Major factors in system performance degradation[4]

Category	Detailed factor	Weight (%)
DB Config & Design	-	20
System Design	-	15
H/W Resources	-	5
Application & DB Design	CPU overhead	9
	Memory overhead	6
	etc.	3
	I/O disutility	42

Table 2.에서 보듯이, 시스템 성능 문제의 60%는 어플리케이션과 DB의 설계문제에서 발생하고, 그 중 70%(전체 중 42%)는 SQL의 수행시 성능문제에 따른 I/O 비효율에 기인한다. 따라서, 이러한 테스트환경과 운영환경의 차이는 테스트환경에서의 테스트 성능이 운영환경에서 동일하게 보장되지 않는 원인이 될 수 있으므로, 운영환경에서의 온라인 거래의 성능저하문제를 해결하기 위해서는 프로그램의 운영환경 적용 전에 테스트환경에서의 SQL 실행계획과의 점검이 필요하다.

2.4 IT 응용프로그램 변경관리 프로세스 사례 연구

A사는 국내 수위 수준 용량의 z/OS 기반 메인프레임 고객사로 세계적으로도 단일시스템 운영기준으로 매우 큰 규모의 DB2 환경에서 대용량 데이터베이스를 운영하는 금융사이며, 아래 Fig.2.에서 보듯이 테스트환경과 스테이징 환경에서 모두 거래성능을 검증하는 매우 보수적인 IT 응용프로그램 변경관리 프로세스를 적용하고 있다. A사는 온라인 SQL 프로그램을 운영환경에 적용하기 전에 해당 프로그램이 운영환경에서 성능이 저하되는 것을 방지하기 위해 테스트환경에서 온라인 SQL 프로그램 생성시 비용기준 Optimizer하에서의 SQL비용(SQL Cost) 한계 기준치를 초과하는 경우 프로그램 재작성을 요청하며, 테스트 거래시 과도한 CPU 자원 사용방지를 위한 CPU 처리시간(Path-Length)의 한계 기준치와 DB 테이블의 과도한 스캔 처리 방지를 위한 데이터 블록 액세스 수(Getpage)의 한계 기준치를 적용하여 이를 초과하는 경우 해당 온라인 SQL 프로그램을 재작성 요청 또는 튜닝을 진행하도록 하는 매우 보수적인 프로그램 성능관리 기준을 운영하고 있다. A사의 현재 IT 응용프로그램 변경관리 프로세스는 아래와 같다.

A사는 스테이징 환경 거래테스트의 경우 SQL 실행계획은 운영환경을 기준으로 생성하고, 거래테스트를 수행하는 물리적 DB모델은 테스트환경을 이용하는 구조를 가지고 있다. 따라서, 이러한 프로세스에서 스테이징 환경의 거래테스트로 운영환경에서의 성능문제가 없다는 것을 보장하기 위해서는 스테이징 환경을 운영환경과 동일하게 구성 및 운영하여야 하나, 이것은 운영환경의 OS, 미들웨어, DBMS, 스토리지 등의 인프라를 동일하게 물리적으로 구성해야 하는 막대한 비용이 투입되어야 하고, 데이터 적재 운영 등의 운영상의 문제들로 현실적으로 쉽지 않다.

Fig.2.의 스테이징 환경 거래 성능점검에서 성능문제가 없었음에도 거래테스트를 수행하는 물리적 DB는 테스트환경을 사용하게 되므로, 운영환경의 SQL 실행계획이 프로그램의 운영환경 적용시에 성능저하가 발생하지 않는다는 것을 완벽하게 보장하지는 못한다. 2.3에서와 같이 대용량 데이터를 다루는 많은 회사들은 운영환경의 모든 자원을 막대한 비용을 들여 스테이징 환경을 물리적으로 동일하게 구성하지 않으므로 대부분 이와 같은 문제를 가지고 있다.

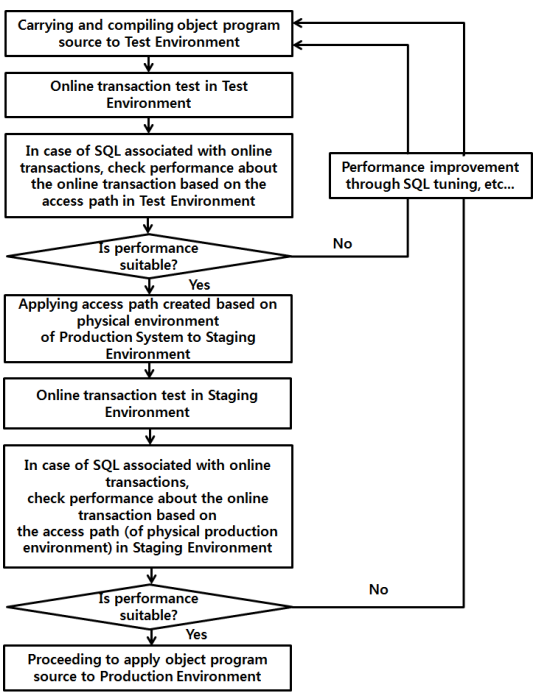


Fig. 2. Current IT Application Change Management Process of the financial institution 'A'

2.5 SQL 성능저하에 따른 장애 사례 및 개선점 조사

본 연구의 필요성을 확인하기 위해 수집한 B사의 거래장애 분석결과는 아래와 같다.

Table 3.에서 보는 바와 같이, B사의 장애사례를 분석해 본 결과 총 11건 중 2건이 테스트환경과 운영환경의 차이로 인해 발생하였고, 본 연구와의 관련성이 있음을 알 수 있다. 이와 같은 수준은 전체 건수 대비 약 18%에 해당하지만, 운영환경에서의 온라인 거래장애가 다른 장애원인처럼 비정형적으로 간헐적인 발생이 아닌 정규화된 프로세스 상에서 발생하는 문제로 본다면 매우 높은 수준의 발생빈도이다. 따라서, 위와 같은 문제를 해결하기 위해서는 프로그램 개발자가 단위 성능테스트를 수행한 테스트환경의 SQL 실행계획과 운영환경에서 수행될 SQL의 실행계획을 프로그램의 운영환경 적용 전에 비교하여 운영환경에서 성능저하가 예상되는 SQL 실행계획을 사전에 인지하여야 한다. 이를 통해 운영환경에서 성능저하로 인한 거래장애가 예상되는 경우 프로그램의 운영환경 적용 단계를 보류하고, SQL 튜닝담당자와 프로그램 개발자의 협업을 통해 이에 상응하는 조치 후에 프로그램의 운영환경 적용을 진행하여야 한다.

Table 3. Analysis of relevance with this study about the reasons for the failure of online transactions

Year	Causes of failures	Relevance	Count
2011, 2012	Copybook Change Management	No	2
2011, 2012	Omitted the program developer's test case	No	2
2011	Problem about to apply the program version	No	1
2011	Problem associated with system job	No	1
2011	Because row count of table in production environment is much more than that in test environment, problem is caused by difference of applied access path during big data range scan	Yes	1
2011, 2012	Problem associated with program source code	No	2
2012	Because the same tables of production and test environments have been maintained as different sort method, access path is changed when applying program	Yes	1
2012	Problem associated with interface	No	1

2.6 알려진 솔루션 조사 및 한계점 연구

2.5에서 확인한 문제를 해결하기 위해 DB2 환경에서 SQL 실행계획의 성능차이를 분석하는 알려진 솔루션을 조사 및 테스트한 결과는 다음과 같다. 조사결과, 확인된 리퍼런스는 APCOMPARE를 DB2 V10이상에서 Rebind[3] 또는 Bind시 'ERROR', 'WARN' 옵션으로 수행하면, 이전 실행계획을 비교하여 실행계획이 달라지는 경우 차이가 발생하는 내용이 무엇인지에 대한 정보를 PLAN_TABLE[4]의 'REMARKS' 컬럼에 표시한다는 내용이다[5]. 이내용에 대해 아래와 같이 테스트를 진행하였다.

테스트환경은 z/OS 기반의 DB2 V10이다. 검증해야 할 내용은 'APCOMPARE의 ERROR, WARN 옵션을 사용하여 SQL 패키지 Rebind시 실행계획 차이가 발생하는 경우 PLAN_TABLE의 REMARKS 컬럼에서 차이 발생 내용을 확인할 수 있다'고 설정하였고, Rebind시 SQL 실행계획 차이가 발생하는 경우와 차이가 없는 경우에 대해 APCOMPARE의 'ERROR', 'WARN' 옵션을 사용하여 Rebind를 수행하였다.

아래 Table 4.는 테스트 수행 결과이며, SQL의 실행계획 차이가 발생한 경우에도 PLAN_TABLE의 'REMARKS' 컬럼에 해당 내용이 표시되지 않아서 SQL 실행계획의 차이 내용을 알 수 없었고, 이 결과는 DB2의 리퍼런스 내용과는 일부 차이를 보이고 있다.

또한, 이러한 기능테스트 결과와는 별개로 2.5에서의 본 연구와의 관련성이 있는 장애원인을 해결하기 위해서는 SQL 프로그램이 운영환경에 적용되기

3) 동일한 프로그램에 대해 다시 Bind를 수행하는 것.
4) DB2에서 SQL의 실행계획 정보를 저장하는 테이블

Table 4. Functional test results of APCOMPARE

Option	Difference of access paths	Return code	Result value of PLAN_TABLE 'REMARKS'
ERROR	Yes	error	SPACE
	No	normal	SPACE
WARN	Yes	normal	SPACE
	No	normal	SPACE

전에 단위 성능테스트를 수행한 테스트환경과 운영환경에서 수행될 해당 SQL 실행계획의 차이를 비교하여 확인하여야 하나, DB2 개발사인 IBM이 해당 기능에 대한 내부처리 알고리즘 또는 로직을 제공하지 않고, APCOMPARE의 기능은 동일시스템 내에서만 사용이 가능하므로 이것만으로는 테스트환경과 운영환경의 실행계획 차이를 분석하는 프로세스를 구현할 수 없다.

III. SQL 성능비교 알고리즘 구현 및 테스트

3.1 SQL의 성능결정 요인 분석

2.2에서 확인한 바와 같이, 온라인 어플리케이션에서의 SQL 성능은 실행계획이 가장 중요하다. 이러한 SQL 실행계획 정보를 DB2 환경에서는 PLAN_TABLE에 각 정보를 컬럼형태로 보관하고 있다. 어플리케이션의 성능과 관련한 SQL 실행계획에 대한 중요 성능결정 영향변수를 PLAN_TABLE 컬럼에서 찾아보기 위해 A금융사가 2013년에 온라인 거래의 성능개선을 위해 튜닝을 수행한 SQL 튜닝가이드에 대한 튜닝요인(튜닝포인트)을 분석하였다.

A금융사의 z/OS 기반의 DB2 V9, 비용기준 Optimizer 사용 환경에서 온라인 어플리케이션에 적용되는 SQL의 성능개선을 위해 수행했던 총 230본의 SQL 튜닝가이드에 대한 튜닝요인을 분석한 결과 Table 5.와 같은 결론을 얻었다. 분석결과, 성능저하가 없어 튜닝할 내용이 없던 44개의 SQL을 제외하면 186본의 SQL 튜닝가이드에서 11개의 튜닝

Table 5. Contents of analysis for Online SQL Tuning of financial institution 'A' in 2013

SQL Tuning Point	Tuning method	Count
Enhance the ability to discriminate of index (increase count of matching columns, change matching index, etc...)	Change SQL	96
	Add Index	49
	Change Index	86
Delete unnecessary SQL function	Change SQL	15
Delete unnecessary Join[5]	Change SQL	11
Modify business logic to call SQL program	Modify non-SQL Business Logic	4
Request examination about extraction range limit in task requisite	Request for conference to relevant department	3
Adjust SQL syntax that does not match transaction pattern and data status	Change access path by changing SQL	1
Delete unnecessary Inline View[6] scan	Change SQL	4
Decrease execution calls of the same pattern table scan	Change SQL	6
Change into optimized access path through optimization of table datas (for example, increase cluster ratio)	Rebind after Reorg[7]	2
	Rebind and Change Process	2
Adjust access path according to changing status of data in table	Rebind	2
Prevent performance degradation by unexpected access path change through fixing to the access path defined at current time	Change SQL	1
No tuning point with no performance degradation	-	44
Total(Tuning point in respect of whole tuning guide)		282

5) 두 개 이상의 테이블을 동시에 쿼리하는 것으로, 최초 검색한 테이블(드라이빙 테이블)의 컬럼을 다른 테이블을 검색하기 위한 외래 키(foreign key)로 사용함.
6) 쿼리(query) 내에 뷰를 생성하여 사용하는 것.
7) 테이블의 상태를 최적화하는 데이터모음 및 정렬 작업

요인를 도출하였고, 튜닝방식을 매핑하여 총 14개의 튜닝기준을 설정하였다. 186본의 튜닝가이드에서 적용한 튜닝요인은 총 282개였으며, 이중 인덱스 변별수준을 높이는 튜닝요인이 약 71%로 매우 높은 비중으로 도출되었다. 이는 인덱스 변별수준을 높이기 위해 매칭인덱스[9] 변경, 매칭컬럼수[10] 증가, 인덱스만으로 데이터 검색, 인덱스 스크리닝[11](Index Screening) 처리추가, 드라이빙 테이블 변경, 다중 인덱스 스캔(Multiple Index Scan) 회피 등을 처리하기 위한 SQL 변경, 인덱스 추가, 인덱스 변경 등의 튜닝을 모두 포함한다. 위의 분석 내용을 정리하여 프로그램 개발자가 SQL 프로그램 작성시에 아래 내용을 반영하여 성능을 최적화하도록 하는 가이드를 도출할 수 있었다.

- 인덱스 변별수준을 높임.
- 불필요한 함수처리 횟수를 줄임.
- 불필요한 조인을 제거함.
- 동일 패턴의 테이블 스캔이 반복되는 경우, 인라인-뷰를 생성하여 스캔 횟수를 줄임.
- 과도한 범위검색이 발생하는 경우, 업무부서와의 협의를 통해 업무상 필요한 검색범위로 조회 조건을 제한함.
- 조회 조건에 맞게 테이블 상태(클러스터율 등)를 최적화함.

위의 분석결과를 기준으로 PLAN_TABLE의 컬럼에 대하여 성능 결정 영향 수준을 정의하는 작업을 아래와 같이 수행하였다.

3.2 성능결정 요인을 DB2의 PLAN 테이블에 적용

위의 SQL 튜닝요인에 대한 통계분석 결과를 DB2 환경에서 온라인 어플리케이션 SQL의 실행계획 정보를 담고 있는 PLAN_TABLE에 적용하여 각 속성의 성능 영향 수준을 성능결정영향도라 명명하고, 아래 기준으로 분류되었다.

- 상: 성능차이를 매우 크게 발생시키는 항목
- 중: 성능에 일부 영향을 주는 항목

Table 6. Determining the level of impact for SQL performance of DB2 PLAN_TABLE columns

Column	Content	Level of impact
QUERYNO	query number	None
QBLOCKNO	query block number	None
PROGNAME	program name	None
PLANNO	PLAN number	None
METHOD	Join method	Middle
CREATOR	package creator	None
TNAME	table name	High
ACCESSTYPE	access type for resource	High
MATCHCOLS	count of matching columns	High
ACCESSCREATOR	constructor	None
ACCESSNAME	accessed index	High
INDEXONLY	whether query is handled using just indexes	High
SORTN_UNIQ	new table unique sort	Middle
SORTN_JOIN	new table join sort	Middle
SORTN_ORDERBY	new table orderby sort	Middle
SORTN_GROUPBY	new table groupby sort	Middle
SORTC_UNIQ	composite table unique sort	Low
SORTC_JOIN	composite table join sort	Low
SORTC_ORDERBY	composite table orderby sort	Low
SORTC_GROUPBY	composite table groupby sort	Low
PREFETCH	prefetch type	Middle
MIXOPSEQ	order of sentences in multiple index scan	None
VERSION	program version	None
COLLID	Collection ID of package	None
JOIN_TYPE	Join type	High
QBLOCK_TYPE	CRUD[8] type	None
BIND_TIME	Bind time	None
PARENT_QBLOCKNO	parent query block number	None
PARENT_PLANNO	parent query PLAN number	None

8) 생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)의 데이터 처리 기능을 일컫는 용어
9) 쿼리가 수행될 때 참조하는 인덱스
10) 쿼리가 수행될 때 참조하는 인덱스의 직접 액세스(access)하는 컬럼수로 인덱스 컬럼의 순서가 중요함.
11) 쿼리가 수행될 때 직접 액세스하는 컬럼은 아니나, 해당 컬럼을 간접적으로 인덱스 스캔 처리하는 것.

- 하: 성능에 미미한 영향을 주는 항목
- 없음: 부가정보(프로그램명 등)

Table 6.은 PLAN_TABLE 컬럼 50개 중 SQL 실행계획 성능차이 분석에 관여하는 항목 29개를 선별하여 성능결정영향도를 컬럼별로 분석한 결과이다.

Table 6.에서 보듯이, 2013년 A금융사의 온라인 어플리케이션 SQL의 성능개선을 위해 SQL 튜닝담당자를 통해 프로그램 개발자에 제공되었던 프로그램 튜닝가이드의 튜닝요인 통계 분석결과를 기준으로 DB2에서 제공하는 PLAN_TABLE 컬럼에 대해 성능결정영향도를 결정한 위의 결과에 따라, 테스트환경에서 성능을 테스트한 SQL 실행계획이 해당 프로그램을 운영환경에 적용시 수행될 SQL 실행계획 정보와 어떤 항목의 차이를 비교하여 성능저하 문제발생을 미리 인식할 것인지에 대한 기준을 제시한다. 그 이유는 SQL 실행계획을 보여주는 PLAN_TABLE에 대해 영향도가 낮은 컬럼을 모두 비교하는 경우, 운영환경에 적용시 성능영향도가 미미하거나 관계없는 컬럼임에도 차이가 발생하였다고 인식하여 무분별한 오탐 및 어플리케이션 개발자와 SQL 튜닝담당자의 불필요한 튜닝 공수로 인해 업무 피로도를 증가시킬 수 있기 때문이다. 따라서, PLAN_TABLE 정보에 대한 성능결정영향도를 분류하는 기준을 설정하는 것은 매우 중요하다.

3.3 SQL 실행계획의 성능비교 알고리즘 구현

3.1과 3.2에서 분석된 온라인 어플리케이션 SQL 성능에 영향도가 높은 PLAN_TABLE 컬럼의 성능결정영향도 결과에 따라 SQL의 실행계획 성능차이를 비교하는 알고리즘을 정의하고 해당 알고리즘을 수행하는 프로그램을 구현하였다. SQL의 실행계획 비교는 동일한 SQL 프로그램에 대해 Rebind에 따른 최종기준의 실행계획 정보와 직전기준의 실행계획 정보를 성능결정영향도를 기준으로 선정한 비교대상 컬럼의 동일여부를 체크하여 차이가 있는지를 비교하는 것이며, Fig. 3.은 해당 알고리즘을 간략히 도식화한 것이다.

다음 장의 Table 7.은 위 Fig. 3.에서 도식화한 알고리즘에 대한 SQL 실행계획의 차이를 비교하는 프로그램 구현을 위해 슈도코드(pseudocode)를 이용한 세부적인 알고리즘을 제시한 것이다.

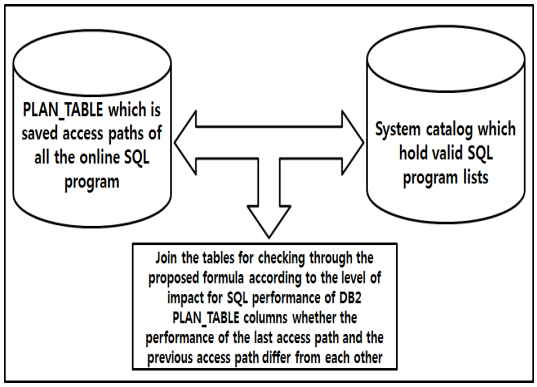


Fig. 3. Methodology for comparing the SQL performance based on SQL access paths using System Catalog Tables with PLAN_TABLE

Table 7.에서 제시한 SQL 실행계획의 성능을 비교하는 알고리즘의 단계 '1-a'에서 사용자로부터 입력받는 'PLAN OWNER', 'PROGRAM NAME', 'COLLECTION ID'는 PLAN Owner[12], 프로그램명, 패키지 Collection ID[13]를 나타낸다. 단계 '3'에서 G.column과 H.column은 3.2에서 SQL 성능결정영향도에 근거해 결정된 비교대상 항목이 되며, 최종기준의 실행계획 정보 G와 직전기준의 실행계획 정보 H를 추출하여 해당 항목을 비교함으로써 성능차이 발생여부를 감지한다. 이 알고리즘은 DB2를 사용하는 모든 고객사가 가진 시스템 정보(System Catalog) 테이블인 SYSIBM.SYSPACKAGE와 PLAN_TABLE만으로 동일한 SQL 프로그램의 동일버전에 대해 성능변화를 각 고객사가 비즈니스 상황에 맞게 PLAN_TABLE 비교대상 컬럼을 조정하여 비교함으로써 SQL 프로그램의 실행계획 결정 프로세스를 수행시 유연하고 효율적으로 실행계획의 성능 변화가 있는지를 체크해낼 수 있다. DB2에서 PLAN_TABLE의 'VERSION' 컬럼값이 동일하다는 의미는 SQL에 변경이 가해지지 않았다는 것으로 이는 동일한 SQL 프로그램에 대한 실행계획의 성능차이를 체크한다는 의미이다. 따라서, Table 7.의 SQL 실행계획의 성능비교 알고리즘은 DB2를 사용하는 모든 고객사에서 사용이 가능하며, 실행계획의 성능차이는 단계 '4'에서의 최종결과값

12) SQL 프로그램의 실행계획 담당 Owner
13) 동일 SQL 프로그램을 여러 시스템 또는 데이터베이스에서 사용하기 위해 생성된 패키지셋(package set)의 복사본을 구분한 단위를 정의한 ID

Table 7. Set algorithm to compare the SQL performance based on SQL access paths using pseudocode

Phase	Contents
1	Create the basic result sets for obtaining the information of the access paths by implementing and run the program (for example, query applied the following pseudocode).

	Phase 1-a) Accept input values about 'COLLECTION ID', 'PROGRAM NAME', 'PLAN OWNER' from external variables
	
	//Perform the following phases for obtaining the last access path and the previous access path about SQL program for comparing the access paths.
	
	//Select rows with the equal values to 'COLLECTION ID', 'PROGRAM NAME' values (were accepted in phase '1-a') from 'PLAN OWNER'.PLAN_TABLE, and then select the valid rows by joining table 'SYSIBM.SYSTABLES' using the predicate 'INNER JOIN' like following phases.
	
	Phase 1-b) Result set A: Select rows from 'PLAN OWNER'.PLAN_TABLE in condition when COLLID = 'COLLECTION ID' AND PROGNAME = 'PROGRAM NAME', and then join table 'SYSIBM.SYSTABLES' using the predicate 'INNER JOIN' with the same values 'COLLID', 'PROGNAME' among the selected columns of 'PLAN OWNER'.PLAN_TABLE
	Phase 1-c) Result set B: Select rows except for rows matched in condition when column 'VALID' (of table 'SYSIBM.SYSTABLES') = 'N' among A
	
	//Perform the following phases continuously.
	Phase 1-d) Result set C: Obtain the set of unique values of 'COLLID' and 'PROGNAME', 'VERSION', 'BIND_TIME' columns of B by performing the predicate 'GROUP BY'
	Phase 1-e) Result set D: Sort C by 'COLLID' ASC, 'PROGNAME' ASC, 'VERSION' DESC, 'BIND_TIME' DESC (meaning: ASC = accending, DESC = descending)
	Phase 1-f) Result set E: Select result rows composed of five columns 'COLLID', 'PROGNAME', 'VERSION', 'BIND_TIME' and 'Row Numbering Value' (Row Numbering Value is sequentially assigned as method of partitioning by columns 'COLLID', 'PROGNAME', 'VERSION' and sorting by columns 'COLLID' ASC, 'PROGNAME' ASC, 'VERSION' DESC, 'BIND_TIME' DESC) from D
	Phase 1-g) Result set F: Select result rows in condition when 'Row Numbering Value' 〈= 2 from E (for obtaining two values of the 'BIND TIME' about the last access path and the previous access path for the same 'COLLECTION ID', 'PROGRAM NAME', 'VERSION')

2	Obtain the last access path and the previous access path about the SQL program by running the program applied the following pseudocode for result set (was obtained in phase '1').
	Phase 2-a) Result set G: Select result rows (for obtaining the last access path about the SQL program) in condition when 'Row Numbering Value' = 1 from F, and then join table 'PLAN OWNER'.PLAN_TABLE using the predicate 'INNER JOIN' with the same values 'COLLID', 'PROGNAME', 'VERSION', 'BIND_TIME' among selected columns of F Phase 2-b) Result set H: Select result rows (for obtaining the previous access path about the SQL program) in condition when 'Row Numbering Value' = 2 from F, and then join table 'PLAN OWNER'.PLAN_TABLE using the predicate 'INNER JOIN' with the same values 'COLLID', 'PROGNAME', 'VERSION', 'BIND_TIME' among selected columns of F

Phase	Contents
3	Obtain the final result values to count rows by comparing result set G with H (were obtained in phase '2') using the predicate 'FULL OUTER JOIN' for determining whether the performances of the last access path and the previous access path differ from each other.
	Phase 3) Result value I : Select (result value columns) count-1, count-2, count-3 from G FULL OUTER JOIN H in condition when G.column = H.column … 〈explanation about component items in SQL of phase '3'〉 · count-1 = total count of the joined rows · count-2 = row count of the last access path information · count-3 = row count of the previous access path information · G.column = always fixed columns 'QBLOCKNO', 'PROGNAME', 'PLANNO', 'VERSION' and the other flexibly-assigned columns by the level of impact for SQL performance in business situations of each company · H.column = the same columns as G.column
4	Determine whether the performances of the last access path and the previous access path differ from each other, which is based on the following formula by using values of result set I (was obtained in phase '3').
	Phase 4) There is no performance difference of the last access path and the previous access path only if it is the case in condition when count-1=count-2=count3.

'count-1', 'count-2', 'count-3' 를 이용하여 아래과 같은 간단한 방식으로 체크한다. 이 최종결과값 3개는 3.2에서처럼 SQL 성능결정영향도에 근거해 정한 성능차이 검증기준에 따라 결정된 PLAN_TABLE의 성능비교 대상 컬럼이 SQL 실행계획의 성능비교 알고리즘을 통해 모두 검증될 수 있도록 반영하였다.

count-1: 최종 실행계획 정보와 직전 실행계획 정보의 비교결과가 동일한 로우 건수
count-2: 최종 실행계획 정보의 로우 건수
count-3: 직전 실행계획 정보의 로우 건수

공식: *count-1=count-2=count-3* 인 경우만 동일한 SQL의 실행계획 재생성에 따른 전후 실행계획의 성능 차이가 없다.

3.4 알고리즘 적용 및 테스트

3.3에서 제시한 알고리즘을 이용하여 SQL 실행계획의 성능을 비교하는 로직을 아래 Fig.4.에서처럼 실제 소스에 적용하였다.

위 Fig. 4.는 IBM의 REXX 언어(language)를 이용해 실환경에서 사용하기 위해 프로그래밍한 소스이다. 참고로 REXX(REstructured eXtended eXecutor) 언어는 IBM이 제공하는 많은 컴퓨터 운영체제를 지원하는 구조화 프로그래밍 언어로 상용버전과 오픈 소스 인터프리터가 모두 존재한다[6].

다음은 구현한 프로그램을 이용해 실행계획의 성능차이를 정상적으로 체크하는지 기능테스트를 진행한 내용이다.

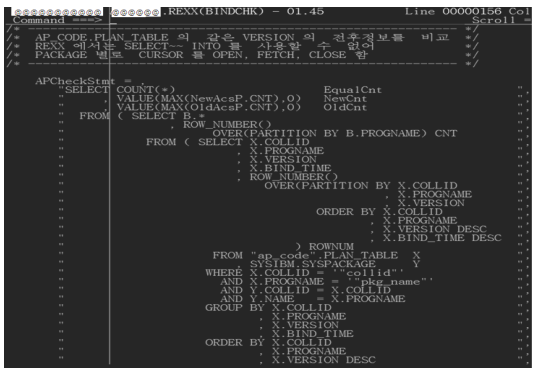


Fig. 4. Applied source-code of algorithm to compare the SQL performance based on SQL access paths

테스트환경은 z/OS 기반의 DB2 V10이다. 검증해야 할 내용은 'DB2에서 SQL 프로그램 실행계획의 성능 비교 알고리즘을 적용한 로직을 탑재하여 프로그램을 구현하면 각 회사의 비즈니스 환경에 따라 결정한 성능결정영향도 기준을 반영한 실행계획의 성능 변화를 체크할 수 있다'고 설정하였고, 수행시나리오 및 테스트결과는 아래와 같다.

- 수행시나리오: 동일 테이블을 사용하는 유효한 모든 SQL 프로그램 패키지[14]의 Rebind를 2회 연속 수행하여 최초 수행시 SQL 실행계획 의 성능차이 발생 프로그램 수를 체크하고, 2회차 수행시 실행계획의 성능차이 발생 프로그램이 없음을 확인한다. 이는 2회차 Rebind 수행시에는 최초 Rebind 수행시 결정한 실행계획을 2회차 Rebind시 결정되는 SQL 실행계획의 직전 실행계획으로 비교하여 체크하게 되므로, 실행계획 성능차이가 있는 프로그램이 발생되면 안된다는 것을 의미한다.

Table 8.의 테스트 결과에서 보듯이, A테이블을 사용하는 모든 SQL 프로그램을 찾아 Rebind를 수행한 결과, 수행시점에 A테이블을 사용하는 유효한 패키지 12개에 대해 실행계획의 성능을 비교하는 알고리즘을 탑재한 프로그램을 반영하여 최초 Rebind 수행시 실행계획의 성능이 변경된 것으로 체크된 패키지가 10개 발생하였으나, 연속하여 Rebind를 재수행시 동일한 12개의 패키지를 Rebind하여 실행계획의 성능이 변경된 것으로 체크된 패키지는 발생하지 않았다. 그 이유는 동일한 환경에서 1회차에 Rebind 하여 결정된 SQL 프로그램의 실행계획이 2회차 Rebind 수행시의 직전기준 실행계획으로 비교되기 때문이다. 자세히 말하자면, 최초 Rebind 시에 결정된 실행계획과 비교대상이 되는 직전 실행계획 정보는 각 프로그램별로 과거 실행계획을 생성했던 시점이 달라지므로 해당 실행계획을 생성했던 과거시점의 테이블의 상태(클러스터율, 테이블 전체 로우 건수, 중요 인덱스 컬럼의 Cardinality 등)가 Rebind 시점과 차이가 큰 프로그램인 경우 실행계획의 성능차이 발생 확률이 높아지지만, 2회차 Rebind 수행시에는 생성된 실행계획 정보가 1회차 Rebind 수행시 생성되었던 실행계획 정보로 비교대상이 동적으로 변경되도록 SQL 프로그램의 실행계획 성능을 비교하는 로직에 구현되어 있고, 동일한 프로그램이 수분차이로 실행계획 정보를 생성시 해당 테이블의 상태 정보가 변경되지 않은 상태이므로 2회차 Rebind 수행시에는 실행계획의 성능차이가 발생한 프로그램이 없다. 따라서, 본 테스트 결과로 검증대상 명제는 입증된다. Fig.5.와 Fig.6.은 1회차, 2회차 Rebind 수행시 실행계획을 비교하여 실행계획의 성능 변경대상 패키지수를 체크한 테스트 수행결과를 보여준다.

Fig.5.에서 표시된 Join Count, Old Count, New Count는 각각 동일한 SQL에 대하여 '최종 실행계획 정보와 직전 실행계획 정보의 비교결과가 동일한 로우 건수', '직전 실행계획 정보의 로우 건수', '최종 실행계획 정보의 로우 건수'를 나타내며, 3.3에서 제시한 알고리즘 체크방식에 근거하여 3개의 값이 모두 동일한 경우를 Rebind 후의 SQL 실

Table 8. Test results of the program for analyzing the differences in the access paths of the SQL

Rebind round	Target table	Total package	Change access path	No change access path
1	A	12	10	2
2	A	12	0	12

14) DB2 환경에서 어플리케이션의 SQL 프로그램 부분을 패키징하여 실행되는 단위를 패키지라 말하며 Bind 및 Rebind의 단위이다.

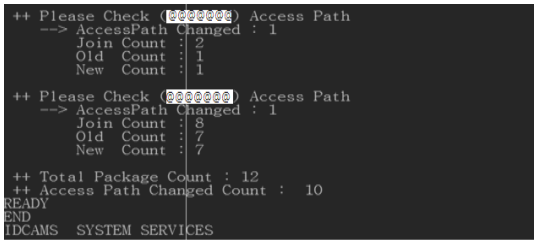


Fig. 5. Confirming the occurrence of some differences in the access paths at the run time of first Rebind

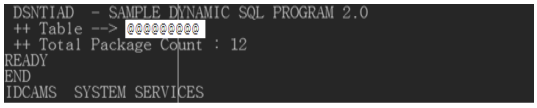


Fig. 6. Confirming the occurrence of the same in the access paths at the run time of second Rebind

행계획과 Rebind 전의 SQL 실행계획이 성능차이가 없는 것으로 처리하였다.

IV. SQL 성능 기반의 개선된 프로세스와 검증

4.1 A금융사의 테스트 및 운영환경 구성현황

본 논문에서 사례로 제시한 z/OS 기반의 DB2 환경에서, 운영환경에서 예상하지 못한 온라인 SQL 성능저하에 따른 거래장애를 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 적용하여 장애발생전 감지해 보는 테스트를 진행한 A금융사의 테스트환경과 운영환경의 구성은 다음과 같다.

Table 9.에서처럼, A금융사는 테스트 DB의 적재방식이 중요정보를 변환한 일부 테이블을 약 3개월 단위로 적재하고, 프로그램 개발자에게 테스트환경의 테이블에 대한 Read, Insert, Update, Delete 및 Load 권한을 제공하고 있으며, 스테이징 DB의 별도 물리적 구성이 없고, 테스트환경에서의 프로그램 개발자의 데이터 변경작업에 대한 통제가 없으므로 II에서 설명한 바와 같이 테스트환경은 프로그램 개발자의 대량 데이터 변경 등의 업무처리 상황에 따라 운영환경의 동일 테이블의 상태와 매우 큰 차이가 발생할 수 있다. 따라서, 이것은 동일한 SQL 프로그램이 테스트환경과 운영환경에서 다른 실행경로로 실행계획이 결정되어 SQL이 수행될 수도 있다는 것을 의미한다. 물론, 테스트환경에서의 실행계획이 운영환경과 다른 경우에 운영환경의 실행계획이 더 좋은 수행성능을 보여줄 수도 있으나, 그와 반대의 경우는 SQL 프로그램의 성능저하로 인한 온라인 거래 지연 또는 장애발생으로 이어질 수 있다.

Table 9. Configurations of the test and production environments of financial institution 'A'

Category	Content
Production DB volume	about 34TB
Test DB volume	about 27TB
Test DB loading cycle	Loading data which is converted important columns (for example, PII) at the beginning of each quarter and loading cycle is 3months
Staging DB configuration	Use logically separated test DB
Developer's table permissions in test environment	Read, Insert, Update, Delete, Load (No permission of 'table Create and Drop')
Developer's table permissions in production environment	Read, Insert, Update, Delete, Load (No permission of 'table Create and Drop')
Job execution control for developers in test environment	It is controlled minimally(for example, job associated with system)
Job execution control for developers in production environment	Normalized approval process through IT job management system

4.2 개선된 SQL 성능 기반의 프로세스 제안

A금융사의 시스템 구성환경에서 테스트환경에서의 거래테스트시 성능저하가 없었으나, 운영환경에 해당 SQL 프로그램 적용시 다르게 결정된 SQL 실행계획에 따른 SQL 성능저하를 사전감지하여 거래장애로 연결되는 것을 방지하기 위해 2.4에서 조사했던 현재의 IT 응용프로그램 변경관리 프로세스의

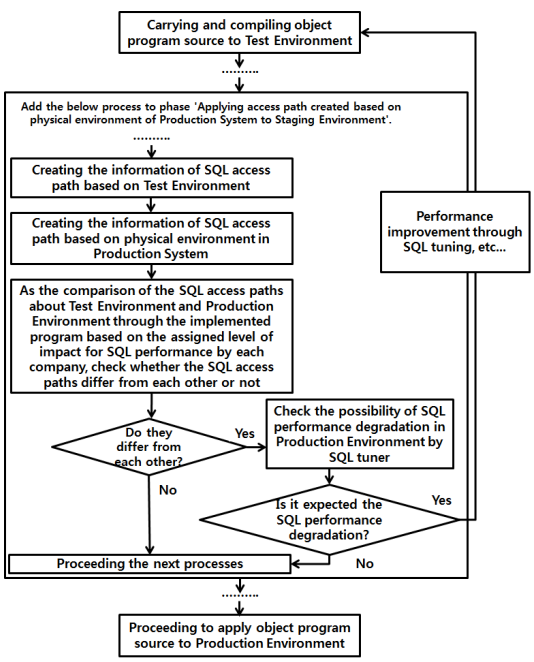


Fig. 7. The SQL Performance-based IT Application Change Management Process

'물리적 운영환경 기반으로 생성된 실행계획을 스테이징 환경에 적용' 단계 내부에 다음 처리를 추가하는 방안을 Fig. 7.과 같이 제시한다.

A금융사는 스테이징 DB를 물리적으로 별도 구성하지 않고, 거래테스트시 테스트환경의 DB를 물리적으로 같이 사용하므로, 스테이징 환경에 프로그램 적용시 운영환경의 테이블 물리모델 상태 정보를 기준으로 운영환경에 적용할 SQL 프로그램의 실행계획을 생성하여 테스트환경에서 생성한 SQL 실행계획과 비교를 수행한다. 수행 결과, 비교 알고리즘에 따라 SQL 실행계획의 변경이 발생하는 경우 SQL 튜닝담당자는 운영환경 테이블의 물리적 상태정보를 기준으로 생성한 SQL 실행계획을 분석하여 성능저하 발생이 예상되면 다음 단계의 프로그램 변경관리 프로세스를 중단하고, 프로그램 개발자에 튜닝가이드를 제공하여 성능문제를 해결한 후에 IT 응용프로그램 변경관리 프로세스를 처음부터 다시 진행한다.

4.3 SQL 성능 기반의 개선된 프로세스 구현

4.3.1 SQL 성능 기반 프로세스 세부요구사항 정의

4.2에서 제시한 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 구현하기 위해 A금융사의 구성환경에서 아래과 같이 세부요구사항을 정의한다.

4.3.2 논리적 테이블 및 프로그램의 구성

4.3.1의 세부요구사항 구현을 위해 Table 11.과 같이 논리 테이블 및 프로그램을 구성하였다. 단, 테스트환경에서 테스트환경과 운영환경의 SQL 실행계획 성능을 비교하게 되므로, 테스트환경의 SQL 실행계획은 IT 응용프로그램 변경관리 프로세스에서 테스트환경의 Bind시 생성되는 PLAN_TABLE을 사용하도록 정의하여, 테스트환경의 SQL 실행계획 정보를 저장하는 테이블은 별도 구성하지 않았다. 온라인 SQL 프로그램을 스테이징 환경 적용단계에서 개발자가 기테스트했던 테스트환경의 실행계획과 성능비교를 위해 추출이 필요한 운영환경의 물리적 테이블 상태 기준으로 생성한 SQL 프로그램의 실행계획을 생성 및 전송하는 프로세스 부분만 운영환경에 적용되고, 이 정보를 테스트환경에서 전송받아 SQL 프로그램의 실행계획을 비교하는 알고리즘 처리 등의 프로세스는 테스트환경에서 처리된다. 이는 테스트환경의 SQL 실행계획을 운영환경에서 전송받아 운영환경에서의 SQL 실행계획과 비교를 수행할 수도 있으나, 제안한 프로세스가 운영환경의 물리적인 환경과 동일하지 않아도 처리가 가능한 알고리즘의 단순로직 처리 부분들은 운영환경의 기존 자원을 최소로 사용함으로써 프로세스 적용에 따른 운영환경의 영향을 최소화하기 위해 테스트환경에서 처리되도록 고안하였다.

Table 10. Defining the requirements about the SQL Performance-based IT Application Change Management Process

Category	Detailed requirement	Applied system
Data Creation	Create information of SQL access path in test environment	Test
	Create information of SQL access path in production environment	Production
Transmit	Transmit SQL access path data file in production environment to test environment	Production
Apply to table	Apply SQL access path data file in production environment to table named 'SQL access path data in production environment' in test environment	Test
Comparison	Compare between the test and production SQL access paths about the same SQL program	Test
Processing	Call SQL tuner after stop the process when SQL access paths differ from each other	Test
	When SQL tuner compare performances of SQL access paths, stop next step of change management process if there were any difference in performance. If not, proceed next step of it.	Test
Record Managing	Manage changed SQL access paths list and historical records of determining whether if performance degradation occur	Test

Table 11. Composition of programs and tables for the implementation of the process

Category	Resource content	Target system
Table	SQL access path data in production environment	Test
	Master information for the SQL program to be determined by SQL tuner whether there will be any performance degradation	Test
	Historical record determined the performance degradation by SQL tuner	Test
Program	Creation of SQL access path data file in production environment	Production
	Apply SQL access path data file in production environment to table named 'SQL access path data in production environment'	Test
	After comparing the two access paths, stop next step of change management process if there were any difference in performance.	Test
	When SQL tuner compare performance of SQL access paths, proceed next step of change management process if there were not any difference in performance.	Test

4.3.3 SQL 실행계획의 성능비교 알고리즘 적용

테스트환경에서의 SQL 실행계획이 운영환경에 적용될 SQL 실행계획과 차이가 발생하는지를 미리 체크하기 위해 3.4에서 구현한 SQL 실행계획 성능비교 알고리즘을 응용하여 적용하였다. 아래 Fig.8.은 해당 알고리즘을 REXX 프로그램에 적용한 화면이다.

Fig.8.의 내용은 Table 7.에서 제시한 SQL 실행계획 변경여부를 체크하는 알고리즘을 응용하여 Result Set 'G' 부분에 테스트환경의 SQL 실행계획 정보를 추출하고, Result Set 'H' 부분에 운영환경의 SQL 실행계획 정보를 추출하여 적용함으로써, 두 실행계획을 Table 7.에서의 성능영향결정도 분석에 따른 영향변수를 체크하는 동일한 방법으로 구현되었다.

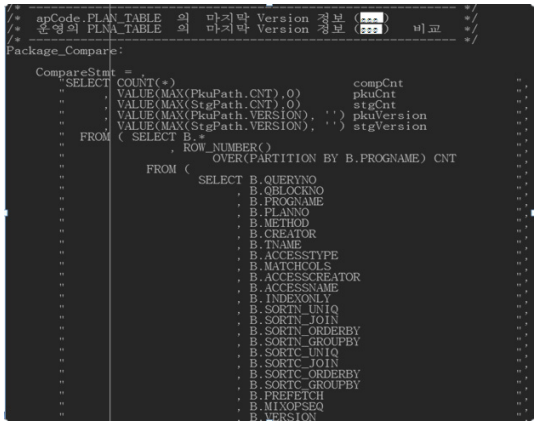


Fig. 8. Application of the algorithm to analyze the differences in the access paths of the same SQL between test and production environments

4.4 SQL 성능 기반의 개선된 프로세스 테스트

4.3에서 정의한 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스 및 응용알고리즘을 구현하여 아래와 같이 테스트를 진행하였다.

테스트환경은 z/OS 기반의 DB2 V10이다. 검증해야 할 내용은 'DB2에서 SQL 프로그램 실행계획의 성능비교 알고리즘을 적용한 로직을 탑재하여 프로그램을 구현하면 각 회사의 비즈니스 환경에 따라 결정한 성능결정영향도 기준을 반영하여 테스트환경과 운영환경의 실행계획 성능 변화를 체크할 수 있다'고 설정하였고, 수행시나리오 및 테스트결과는 아래와 같다.

- 수행시나리오: 특정 CM패키지번호[15]에 대한 온라인 SQL 프로그램의 테스트환경과 운영환경의 실행계획을 비교하여 성능차이가 발생하는 경우 해당 CM패키지번호, 온라인 SQL 프로그램 정보를 'SQL 튜닝담당자에 의해 성능저하가 발생할지 여부를 판단해야 할 SQL 프로그램 마스터 정보' 테이블에 반영하여 DB관리자 등의 SQL 튜닝담당자가 성능저하문제를 판단할 수 있도록 한다.

Table 12.의 테스트 결과에서 보듯이, 테스트대상 SQL 프로그램인 QEAxxxx와 QFAxxxx에 대

15) IT 응용프로그램 변경관리 프로세스에서 프로그램을 변경적용하기 위한 적용단위를 뜻함.

Table 12. Test results of the SQL Performance-based IT Application Change Management Process for analyzing the differences in the access paths of same SQL between test and production environments

CM package number	Program name	Access path	Changed content
KEAxx	QEAxxxx	Not chang-ed	-
KFAxx	QFAxxxx	Chan-ged	PREFETCH

하여 테스트환경과 운영환경의 실행계획을 생성하고 이를 비교하였다. QEAxxxx는 성능결정영향도에 근거하여 테스트환경과 운영환경에서의 실행계획의 성능이 동일했지만, QFAxxxx은 운영환경에서의 실행계획이 변경된 것으로 결과가 도출되었다. 따라서, 테스트환경과 운영환경의 SQL 실행계획에 대해 성능결정에 영향을 미치는 어떤 영향변수가 변경되었는지를 확인해 본 결과 PLAN_TABLE의 PREFETCH[16] 정보가 달라졌다는 것을 아래 그림과 같이 확인할 수 있었다.

Fig. 9.과 Fig. 10.의 그림에서처럼, 동일한 QFAxxxx 프로그램에 대해 테스트환경에서는 PLAN_TABLE의 PREFETCH 컬럼 정보가 'L','S' 로 되어있어 최초 List Prefetch[17]를 수행한 후 Sequential Prefetch[18]를 수행하나, 운영환경에서는 PREFETCH 컬럼정보가 'L','D' 이므로 최초 List Prefetch는 테스트환경에서처럼 동일하게 수행하고 테스트환경에서 Sequential Prefetch를 수행하던 부분은 Dynamic Prefetch[19]를 수행

PREFETCH	COLUMN_FN_EVAL	MIXOPSEQ
L		0
S		1
		2
		3
		0

Fig. 9. Access path of QFAxxxx SQL program in the test environment

SORTC_GROUPBY	PREFETCH	MIXOPSEQ
N	L	0
N	D	1
N		2
N		3
N		0

Fig. 10. Access path of QFAxxxx SQL program in the production environment

하는 것으로 변경되었음을 알 수 있다. 이 경우 운영환경에서 변경된 SQL 실행계획으로 수행되는 프로그램이 운영환경에 그대로 적용시 온라인 거래의 성능저하를 일으키는지에 대해서 확인하는 절차를 수행하기 위해 '물리적 운영환경 기반으로 생성된 실행계획을 스테이징 환경에 적용' 이후 단계의 변경관리 절차를 중단후, SQL 튜닝담당자의 검증에 따라 성능저하가 없다고 판단하여 이후 단계의 변경관리 절차를 진행할 것인지, 성능저하가 거래장애를 일으키거나 서비스를 지연시킬 수 있는 수준으로 예상되어 변경관리 절차를 취소하고 SQL 프로그램의 튜닝을 진행해야 할 것인지를 결정하도록 해야 한다. 이러한 새로운 절차를 통해 성능저하에 따른 운영환경의 예상하지 못한 온라인 거래장애를 차단하여 IT운영리스크를 제거한다.

V. SQL 성능 기반 프로세스의 효과성 분석

5.1 개선된 프로세스의 정량적 효과분석 방향

V에서는 거래장애방지를 위해 온라인 어플리케이션 SQL의 실행계획 성능차이 사전감지 절차를 적용한 개선된 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스의 정량적 가치 산출에 기준이 되는 요인들을 제시하고, A금융사의 조사된 자료들을 토대로 이를 산출해본 후 산출기준을 정리해보고, 실무에서의 사례를 통해 성능적 가치를 확인해본다.

16) 대량의 데이터를 처리하는 범위검색을 수행하는 경우, 많은 입출력 횟수에 따른 성능저하를 방지하기 위해 DBMS의 기능적 판단에 의해 데이터의 블록을 미리 메모리에 상주시키는 방식으로 Sequential, Dynamic, List Prefetch 방식으로 나뉜다.

17) 테이블 데이터에 대해 매칭인덱스의 클러스터율이 높지 않은 수준인 경우 참조한 인덱스의 RowID(RID)를 메모리에서 데이터 블록 번호 순서로 미리 정렬하여 데이터 블록을 찾아가는 Prefetch 방식

18) 클러스터율이 높은 데이터의 검색조건에 대한 범위검색 등의 처리시 단일 입출력으로 여러 개의 블록을 메모리에 읽어들이는 Prefetch 방식

19) 여러 개의 블록을 단일 입출력으로 메모리에 읽어들이는 것이 성능에 효율적이라고 DBMS가 SQL의 실행 시점에 판단하여 처리하는 Prefetch 방식

5.2 정량적 효과분석 자료 및 기준설정

5.2.1 정량적 효과분석을 위한 기초 자료수집

SQL 성능 기반 IT 응용프로그램 변경관리 프로세스의 정량적 가치분석을 위한 기초 자료는 2013년 기준의 A금융사에 대한 관련 자료를 수집하여 다음과 같이 정리하였다.

- 2013년 영업일 평균 온라인거래수: 6,100만건
- 2013년 온라인거래 중 단말거래 비율: 17.6% (온라인 거래 중 영업점 단말의 발생비중 분석)
- 2013년말 기준 온라인거래수: 16,152
- 2013년말 기준 온라인 SQL 프로그램수: 19,154(전체 SQL 중 온라인 거래에 영향을 주는 성능체크 대상만 산출)
- 2013년 영업일수: 248일
- 2013년 제1금융권 평균연급여: 7,538만원(한국경제신문기고 제1금융권 11개 기업 산술평균)[7]
- 2013년 영업점 창구업무 수행직원: 약 8,200명 (콜센터 직원은 산출에서 제외)
- 거래장애시 장애분석 및 처리를 위한 실투입인력은 아래 Table.13.과 같이 산출하였다.

Table 13.은 실투입인력의 정량적 가치 산출을 예로 보여주기 위해 A금융사가 장애처리를 위해 IT 및 비IT부문에 투입하는 인력산출 기준 및 장애처리 예상 투입시간을 보수적인 기준에 근거해 정리하였다. 장애 처리시간 중 전담투입되지 않는 IT지원 인력의 투입비중은 시스템과 프로그램 개발 지원인력으로 분류하고, A금융사의 해당 업무담당자 각 10명씩 총 20명에 투입비중수준을 설문하여 시스템 지원인력의 장애처리시간 중 투입비중을 전체 인력의 50%, 프로그램 개발 지원인력은 전체 인력의 30%로 설정하였다.

5.2.2 온라인 거래 영향 계수 산출

특정 온라인 SQL 프로그램의 성능저하가 발생했을 때의 온라인 거래 영향도를 파악하기 위해, 특정 온라인 SQL 프로그램이 장애발생시 영향을 미치는 온라인 거래수를 온라인 거래 영향 계수로 정의하고, 다음과 같은 공식에 의해 산출하도록 한다.

온라인 거래 영향 계수 = 온라인거래수/온라인 SQL 프로그램수(단, 온라인 거래 영향 계수 < 1 인 경우는 1로 보정함.)

온라인 거래 영향 계수가 1보다 작은 경우는 전체 온라인 SQL 프로그램 개수가 온라인 거래보다 많은 경우이나, 특정 온라인 거래는 다수의 온라인 SQL 프로그램을 사용할 수 있다. 또한, 유효한 1개의 온라인 SQL 프로그램에서 성능저하가 발생하는 경우 해당 SQL 프로그램을 사용하는 다수 개의 온라인 어플리케이션과 관련된 온라인 거래에 장애를 일으키고, 최소 1개이상의 거래에 영향을 미치므로 특정 회사가 위 공식에 의해 산출한 온라인 거래 영향 계수가 1보다 작은 경우는 1로 보정하여 적용한다.

Table 13. Calculating the human resources for the trouble shooting about the failure of transactions caused by degradation of SQL performance

Category	Input staff	Required time (hours)	Staff category
Problem Analysis and solving	✓ each OS, DB, Middleware Staff ✓ two of program developers ✓ two of reporters(system engineer and developer)	1	IT
Program Change Management	✓ two of program developers ✓ one of change managers ✓ one of reporters(developer)	0.5	IT
Business Support on Fault Term	✓ bankers ✓ 50% of the staffs excepted dedicated processing staffs among fifteen of system support staffs ✓ 30% of the staffs excepted dedicated processing staffs among twelve of developed program supporters	1.5	IT & non-IT

5.3 개선된 프로세스의 정량적 효과분석 결과

5.3.1 유형가치산출

5.3.1.1 IT부문의 효과(투입인력기준)

장애발생시 장애처리를 위한 투입인력의 정량적 가치 산출방식을 다음과 같이 제시한다.

> 인력 투입가치 = (연평균임금/영업일) × (투입시간/일영업시간) × 투입인력 × 투입비중 × 온라인 거래 영향 계수

Table 13.에서 정리한 A금융사의 사례로 IT부문의 효과를 투입인력의 정량적 가치로 산출하면 다음과 같다.

- A금융사의 2013년 시스템 인력 투입가치 = (7,538만원/248일) × {(1시간/8시간) × 4명 × 1 + (1.5시간/8시간) × (15명 - 4명) × 0.5)} × 1 = 465,426원

위 산출내용은 A금융사의 2013년 기준 1개의 온라인 SQL 프로그램 성능저하에 따른 거래장애시의 시스템 인력 투입 가치를 산출한 것이다. 따라서, 위와 같은 방식으로 A금융사의 2013년 기준 1개의 온라인 SQL 프로그램의 성능저하에 따른 1시간 30분 거래장애발생시 IT부문의 정량적 효과를 모두 산출하면, 개발 인력 투입가치 324,848원, 변경관리 인력 투입가치 18,997원으로 시스템 인력 투입가치와 합산하여 총 809,271원의 IT부문의 인력 투입에 따른 유형손실이 발생한다. 이는 장애처리가 즉시 복구되는 최소 처리시간인 1시간 30분으로 설정하여 매우 보수적으로 산출한 것으로, 장애처리시간이 길어지면 프로그램 변경관리 처리시간 30분을 제외한 나머지 시간에 시스템 전담인력과 개발 전담인력의 투입가치가 기하급수적으로 증가하게 된다. 또한, 온라인 거래 영향 계수가 커질수록 유형손실 비용은 그 배수만큼 증가하게 된다. 예를 들면, 2013년 A금융사의 장애처리시간이 6시간이고, 이중 프로그램 변경관리 처리시간이 동일하게 30분이며, 온라인 거래 영향 계수가 1.2라고 가정하여 시스템 인력 투입 가치를 위 기준으로 산출하면 다음과 같다.

- 시스템 인력 투입가치 = (7,538만원/248일) × {(5.5시간/8시간) × 4명 × 1 + (6시간/8시간) × (15명 - 4명) × 0.5)} × 1.2 = 2,510,927원

즉, 장애처리시간은 1시간 30분에서 6시간으로 총 4배 증가했지만, 시스템 인력의 투입가치는 전담 지원인력 4명의 투입시간이 1시간에서 5.5시간으로 약 5.5배 증가하고, 온라인 거래 영향 계수가 1에서 1.2로 증가함에 따라, 약 5.4배 증가했음을 알 수 있다. 이는 단위 SQL 프로그램의 성능문제가 거래장애 또는 서비스 지연을 발생시키는 경우 IT 투입인력의 유형손실은 전담 지원인력수가 증가할수록, 온라인 거래 영향 계수가 커질수록 장애처리시간의 증가에 비해 더욱 급격히 증가한다는 것을 보여준다.

5.3.1.2 비즈니스 부문의 효과(투입인력기준)

A금융사의 사례로 비즈니스 부문의 효과를 투입인력의 정량적 가치로 산출하면 다음과 같다. 비즈니스 부문의 투입인력 가치 산출 방법은 IT부문의 투입인력 가치 산출 방식과 동일하나, 투입비중은 영업지원 인력 투입비중의 현실적 적용을 위해, A금융사의 2013년 전체 온라인 거래 중 단말채널을 통한 거래 발생비율인 17.6%를 적용하였다. 이는 장애가 처리되는 시간에 전체 온라인 거래 중 영업지원 인력을 통한 거래는 단말채널을 통해 발생하고, 거래장애는 영업지원 인력이 수행하는 거래처리에 영향을 주기 때문이며, 영업지원 인력이 일부 직접 처리하는 콜센터 채널 등의 기타 발생거래는 매우 미미하여 무시하기로 한다.

- A금융사의 2013년 비즈니스 인력 투입가치 = (7,538만원/248일) × (1.5시간/8시간) × (8,200명 × 0.176) × 1 = 약 82,249,306원

위와 같은 방식으로 A금융사의 2013년 기준 1개의 온라인 SQL 프로그램의 성능저하에 따른 1시간 30분 동안 거래장애 또는 문제가 되는 수준의 서비스 지연이 발생한다면, 장애처리시간 동안 비즈니스 인력 투입에 따라 총 82,249,306원의 유형손실이 발생한다. 이는 IT부문과 동일하게 온라인 거래 영향 계수가 커질수록, 장애처리시간이 길어질수록 유형손실 비용은 그 배수만큼 증가하나, 전담인력의 투입이 없고 장애처리시간 동안 일정한 수준의 영향을

지속적으로 받기 때문에, 비즈니스 인력 투입가치는 장애처리시간 및 온라인 거래 영향 계수에 절대적으로 정비례한다. 또한, 투입비중을 총 온라인 거래 중 단말채널의 거래 발생비율로 결정한다면, 회사의 단말거래 발생비율의 증가 또는 감소에 비즈니스 인력의 투입가치는 정비례하게 된다. 예를 들어, A금융사의 단위 온라인 SQL 프로그램 성능저하에 따른 6시간 동안의 거래장애시 2013년의 비즈니스 인력 투입가치는 다른 영향변수가 모두 동일하다고 하면, 앞서 산출한 1시간 30분 동안의 장애발생에 따른 유형손실의 4배인 328,997,224원이 산출된다.

5.3.2 무형가치산출

기업이 IT에 대한 투자의사를 결정함에 있어 크게 리스크, 유형효과, 무형효과로 구성된 ROI 대시보드를 참조하여 IT ROI를 산출하게 되는데, 그 중 가장 흔히 간과하는 무형 효과 가운데 하나는 솔루션을 구축하지 않았을 때 발생할 수 있는 리스크를 회피하는 효과다[8]. 즉, 경쟁사가 해당 솔루션을 구축했을 때 그 경쟁사가 모든 면에서 앞서 갈 수 있는 유리한 위치를 차지할 가능성이 높다. 이는 브랜드 가치 하락, 시장 점유율 하락, 주요 고객 이탈, 직원 충성도 저하, 투자자의 신뢰 상실 등과 같이 중요하지만 눈에 보이지 않는 리스크를 말한다[9]. 이러한 중요한 무형가치에 대한 리스크는 기업의 평판리스크(reputational risk)에 기인한다. 평판리스크란, 경영부진, 금융사고, 사회적 물의 야기 등으로 고객, 주주 등 외부여론이 악화됨에 따라 손실이 발생할 가능성이라고 정의할 수 있다. 이러한 평판리스크로 발생하는 손실은 재무제표상의 재무적 변동으로 파악하기 어렵다. 평판리스크는 그 발생원인이 매우 다양하고 예측하기 어렵지만, 짧은 시간 내에 수익성, 주가, 유동성 등을 통해 기업에 심각한 영향을 미칠 가능성이 있다. 2004년 6월 EIU(Economic Intelligence Unit)와 PWC(Price Waterhouse and Cooper's)가 글로벌 대형 금융기관을 대상으로 조사한 바 있는 기업의 시장가치에 대한 리스크 종류별 잠재적 영향 조사에 따르면, 글로벌 금융기관들은 신용, 시장, 운영리스크 보다 평판리스크를 기업가치에 대한 가장 큰 잠재적 위협으로 지목하였다[10]. 따라서, 온라인 SQL 프로그램의 성능저하에 따른 거래장애는 비지니스의 연속성(Business Continuity)을 저해하여 고객민원 증가, 신뢰도 하락 및 기업 이미지 실추 등 기업의 평판리스크를 증가시키는 무형가치의 손실을 발생시키게 된다. 2013년 IBM이 발표한 '글로벌 평판리스크와 IT 연구(Global Reputational Risk and IT Study)' 결과에 따르면, 23개 업종의 시니어급 임원, IT관리자 등 총 602명에게 IT리스크에 영향을 받는 요인을 설문한 결과 브랜드 평판 74%, 고객 만족 73%, 수익성 60%가 영향을 받으며, 2012년 동일한 23개 업종의 시니어급 임원 423명에게 설문한 결과로 이러한 평판리스크에 가장 중요한 핵심임무는 보안(84%), 비즈니스 영속성(77%), 기술 지원(68%)으로 조사되었다. 또한 동자료에서 제시한 Aberdeen Group의 보고서에 따르면 시스템 가동중단에 따른 평판리스크 관련 손실에 대해 데이터 센터 가동 중단의 시간당 경제적 손실은 181,770달러, 업무 차질에 따른 사건당 경제적 손실은 418,017달러에 달하며, 이와 같은 사건이 연평균 2,3회 발생한다면 총 비용은 1백만달러에 육박하게 된다고 한다[11][12]. 온라인 SQL의 성능저하에 따른 거래장애는 이러한 시스템 가동 중단시의 업무 차질로 인한 평판손상에 따른 경제적 손실과 그 맥락이 같다. 그리고 이렇게 손상된 평판을 회복하는데 걸리는 시간인 목표 복구 시간(Recovery Time Objective: RTO)은 사안에 따라 차이는 있으나 평균 6개월에서 12개월이 걸리는 것으로 조사되었다[13]. 따라서, 온라인 SQL 프로그램과 연관된 거래장애로 인한 서비스 레벨의 침해수준은 해당 온라인 SQL 프로그램의 비즈니스 영향도에 따라 차이는 있을 수 있으나, 시스템 가동 중단 수준에 근접하는 중대한 문제로 귀결되는 경우 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스의 무형가치는 위와 같은 평판리스크에 대한 사건당 경제적 손실과 손상된 평판을 회복하는데 걸리는 시간과 비용의 무형손실의 합이 된다. 이를 정리하면 다음과 같다.

> 무형가치 = 평판리스크의 경제적 손실가치 + 손상된 평판 회복비용

따라서, 지금까지 도출한 결과로 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스(SQL Performance-based IT Application Change Management Process: PITACMP)의 총가치를 산출한 내용은 아래와 같다.

5.3.3 총가치산출

개선된 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스(PITACMP)의 총가치는 유형가치인 장애처리를 위한 투입인력 가치와 무형가치인 평판손상 방지의 가치를 합산한 것으로 아래와 같다.

> PITACMP의 총가치 = 장애처리를 위한 IT 인력 및 비즈니스 인력 투입 가치 + 평판리스크의 경제적 손실가치 + 손상된 평판 회복비용

정리하자면, IT인력의 투입가치는 전담 지원인력 수가 증가할수록, 온라인 거래 영향 계수가 커질수록 장애처리시간의 증가에 비해 더욱 급격히 증가하며, 비즈니스 인력의 투입가치는 장애처리시간의 증가에 대해 온라인 거래 영향 계수에 정비례한다. 또한, 평판리스크의 경제적 손실가치는 온라인 SQL의 성능저하에 따라 발생한 거래장애의 서비스 레벨 침해수준이 시스템 가동 중단 수준에 근접할수록 Aberdeen Group이 조사한 시스템 가동 중단시 업무 차질에 따른 사건당 평판리스크의 경제적 손실액인 418,017달러에 근접한다고 볼 수 있으며, 이에 근접할수록 손상된 평판을 회복하기 위한 비용은 커지게 된다. 앞서 사례로 제시한 A금융사의 경우 시스템 가동 중단에 대해 소프트웨어 장애, 하드웨어 장애, 네트워크 중단, 프로그램 에러, 바이러스 침입 등에 따른 손실과 같이 문제의 원인을 구체적으로 정의한 후, 시스템 가동 중단에 따른 경제적 손실액의 정량적 수치를 산출하기 위한 최빈손실금액, 최대손실금액을 설정하여 산출범위의 평가 근거로 사용한다. 최빈손실금액은 과거 발생사건들의 유형손실의 평균금액에 시스템 가동 중단에 따른 복구처리 비용을 가산하여 산출하는데, 이 복구처리 비용에 대한 수치를 약 4~5억원 수준으로 적용한다. 최대손실금액은 2011년도 D금융사의 시스템 가동 중단에 따른 피해보상액을 추정 적용하는데, 이는 단일사건으로는 가장 큰 피해 규모로 단위 사건에 대한 손실예상 한계치라고 기준을 설정하였다. 여기서 의미하는 바는 최빈손실금액 산출시 무형가치손실에 해당하는 시스템 가동 중단시의 복구처리비용을 약 4~5억원으로 적용하고 있다는 것에서 Aberdeen Group이 평판리스크와 관련하여 제시한 업무차질에 따른 사건당 경제적 손실액인 418,017달러와 유사하다. 이는 앞에서 산출해 본 장애처리를 위한 인력 투입가치인 유형가치와는 다르게 평판손상의 절대적 가치를 산출하는 것이 쉽지는 않지만, 거래장애의 서비스 레벨 침해 수준에 따라 사건당 최대 손실가치가 일반적으로 418,017달러에 달할 수 있고, 손상된 평판에 대한 회복비용이 이에 비례하여 증가한다면 개선된 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스의 총가치는 장애상황에 따른 서비스 레벨 침해수준이 높으면 매우 커질 수 있음을 보여주고 있으며, A금융사처럼 자체적인 기준을 설정하여 적용하는 경우에는 각 회사의 설정기준에 따라 평판손상에 따른 손실이 발생할 수 있음을 의미한다. 이렇게 발생가능한 유무형 손실을 방지하는 가치의 합이 개선된 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스의 총체적 가치이다.

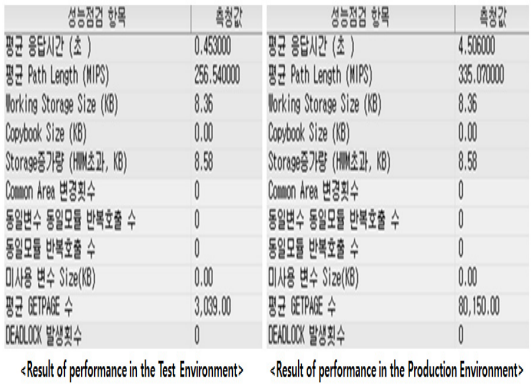
5.4 개선된 프로세스의 성능적 효과분석 결과

제안한 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스가 프로그램 적용단계에서 실무에서 발생했던 아래와 같은 사례를 사전감지하여 성능저하에 따른 온라인 장애를 차단하는 성능적 효과를 다음과 같이 분석하였다.

- 사례: 테스트환경과 운영환경의 검색대상 테이블의 로우 건수는 유사하나, 범위검색 대상 쿼리조건 컬럼값에 대한 해당 테이블의 클러스터율의 차이로 동일한 SQL 프로그램이 테스트환경과 운영환경의 실행계획이 달라져 운영환경에 그대로 적용시 성능저하가 발생하는 경우

위 사례와 같이, 동일 검색대상 테이블에 대해 클러스터율이 다른 테스트환경과 운영환경에서 거래성능을 측정한 결과는 아래 Fig.11.과 같다.

이 성능측정 결과는 다른 조건이 같고 검색대상 컬럼에 대한 클러스터율이 다른 경우 동일한 검색조건의 SQL 프로그램이 Fig.11에서와 같이 큰 성능차이가 발생했음을 보여준다. 성능측정시 테스트환경은 해당 테이블에 대한 SQL의 중요 범위검색 조건 컬럼의 클러스터율이 100%였으며, 운영환경은 67%였다. 즉, 테스트환경은 SQL에서 범위검색을 위한 컬럼에 대해 물리적으로 테이블 로우의 정렬수준이 100%이므로, 동일한 결과값을 획득하기 위한 검색대상 데이터 블록의 수가 67%수준인 운영환경보다 매우 적어 CPU사용율과 거래응답시간의 성능이 매



성능점검 항목	측정값
평균 응답시간 (초)	0.453000
평균 Path Length (MIPS)	256.540000
Working Storage Size (KB)	8.36
Copybook Size (KB)	0.00
Storage증가량 (HWM초과, KB)	8.58
Common Area 변경횟수	0
동일변수 동일모듈 반복호출 수	0
동일모듈 반복호출 수	0
DI사용 변수 Size(KB)	0.00
평균 GETPAGE 수	3,039.00
DEADLOCK 발생횟수	0

<Result of performance in the Test Environment>

성능점검 항목	측정값
평균 응답시간 (초)	4.506000
평균 Path Length (MIPS)	335.070000
Working Storage Size (KB)	8.36
Copybook Size (KB)	0.00
Storage증가량 (HWM초과, KB)	8.58
Common Area 변경횟수	0
동일변수 동일모듈 반복호출 수	0
동일모듈 반복호출 수	0
DI사용 변수 Size(KB)	0.00
평균 GETPAGE 수	80,150.00
DEADLOCK 발생횟수	0

<Result of performance in the Production Environment>

Fig. 11. Result of the transaction performance depending on the change of 'Cluster Ratio' based on the searching column of SQL

Table 14. Comparison of the transaction performance depending on 'Cluster Ratio' based on the searching column of SQL

Environment	Cluster Ratio	Getpage	CPU usage (MIPS)	Response time
Test	100%	3,039	256.54	0.453
Production	67%	80,150	335.07	4.506

우 좋음을 알 수 있다.

Table 14.는 클러스터율 변화에 따라 성능차이가 발생하는 Fig.11.의 테스트 결과에 대한 성능 비교 결과를 보여주고 있다.

이 사례의 경우, 테스트환경과 운영환경의 SQL 실행계획은 클러스터율 차이에 따라 매칭인덱스 컬럼이 달라져 있었다. 따라서, 앞서 4.4에서 실증해본 바와 같이 고안한 알고리즘을 탑재한 테스트환경과 운영환경에서의 SQL 실행계획 성능비교를 통해 차이를 인지함으로써 운영환경 적용전에 미리 성능저하 문제를 확인하여 운영환경에서의 온라인 거래 장애를 사전 차단할 수 있다.

VI. 결론 및 향후 연구

6.1 결론

대용량 데이터를 다루는 은행, 증권사 등의 금융기관과 통신사 등의 대형회사들은 보수적이고 정형화된 어플리케이션 테스트 및 변경관리 프로세스를 적용하고 있음에도, 테스트시스템과 운영시스템의 환경차이로 발생하는 운영시스템에서의 온라인 SQL 프로그램의 성능저하 문제는 거래장애가 발생한 후 문제분석 및 해결방안을 찾는 등 사전감지 프로세스가 없는 실정이었다. 따라서, 본 논문에서는 대용량 데이터베이스에서 많이 사용되는 DB2 기반에서 테스트환경과 운영환경의 온라인 어플리케이션에 탑재된 SQL 프로그램의 실행계획 성능을 비교하는 알고리즘 및 프로세스를 고안하고, 이를 SQL 튜닝담당자와 어플리케이션 개발자가 운영환경에서의 성능문제 발생 전에 협업할 수 있도록 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 제시, 적용하고 구현해 본 사례를 통해 온라인 거래장애를 예방하는 정량적 효과를 도출하는 기준을 제시하고 성능적 효과를 확인하였다. 대용량 데이터를 다루는 많은 회사들은 본 논문에서 제시한 알고리즘 및 개선된 SQL 성능 기반 IT 응용프로그램 변경관리 프로세스를 각 회사의 비즈니스 상황에 맞게 적용하고 유지, 보수해 나간다면, 테스트환경과 운영환경의 차이에 따라 SQL 실행계획이 운영환경에서 달라져 발생하는 예상하지 못한 온라인 거래장애 및 서비스 지연을 미리 감지하여 예방함으로써, 온라인 거래 장애로 인한 장애처리인력의 투입비용, 평판손상 및 회복에 따른 비용 등의 유무형 손실을 방지할 수 있을 것이다.

6.2 향후 발전방향

본 연구는 전세계적으로 대용량 데이터를 다루는 많은 회사에서 Core-Business IT에 사용하고 있는 메인프레임 z/OS 기반의 DB2 환경에서 이루어져, 오라클, Sybase 등 오픈시스템 기반 DBMS 환경에서의 테스트가 이루어지지 않았다. 그러나, DB2가 전세계적으로 대용량 데이터베이스를 처리하는 비즈니스 환경에서의 높은 시장점유율 및 영향력를 생각해볼 때, 본 논문에서 고안한 SQL 실행계획의 성능차이 분석 알고리즘 및 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스가 대용량 데이터를 다루는 회사에서 테스트환경과 운영환경의 차이로 인한 예상하지 못한 운영환경에서의 SQL 성능저하 문제를 사전에 감지하여 온라인 거래장애를 해결하는 방안을 제시한 것은 명백한 사실이다. 또한, SQL 실행계획의 성능차이를 분석하는 기반이 되는 SQL PLAN 테이블은 DB2 외의 DBMS도 그 구조는 다르지만 제공되고 있고, 그 속성 또한 매우 유사하므로 본 논문에서 제시한 SQL 성능결정영향도를 반영한 SQL 실

행계획 성능차이 분석 알고리즘 및 개선된 SQL 성능 기반의 IT 응용프로그램 변경관리 프로세스를 참조하여 구현함으로써 확대 적용해 나가는 것이 필요하다. 이는 IT장애발생으로 인한 유형적 손실과 더불어 평판리스크로 귀결되어 영속적 기업활동에 더욱 중요시되고 있는 고객신뢰도 및 브랜드 가치 등의 무형가치의 훼손을 방지함으로써 기업가치를 유지하고 경쟁력을 높이기 위해 매우 중요한 부분이라 하겠다.

References

[1] Overseas Mainstream System 'Mainframe' is 50 years-old,http://www.zdnet.co.kr/news/news_view.asp?artice_id=20130707115153&type=det

[2] The Server OS market-share in 2013, http://zlinuxtdy.wordpress.com/2013/12/09/2013%EB%85%84-%EC%84%9C%EB%B2%84-os-%EC%8B%9C%EC%9E%A5-%EC%A0%90%EC%9C%A0%EC%9C%A8/

[3] IBM Corporation, "DB2 10 for LUW: Basic Administration for Linux and Windows," IBM Korea, August. 2012

[4] Eung-su Han, "The Guide for Data Architecture Professional," Korea Database Agency, pp.532, Mar. 2010

[5] DB2 Access path Enhancements with DB 2, http://mainframedba.blogspot.kr/2013/01/db2-access-path-enhancements-availabe.html

[6] Definition of REXX, http://ko.wikipedia.org/wiki/REXX

[7] The average salary of bank employees in 2013, http://www.hankyung.com/news/app/newsview.php?aid=2014033147361

[8] Tom Picello, "IT ROI, Analysis the values of IT Investment," Daechung Media, pp.202, May. 2004

[9] Tom Picello, "IT ROI, Analysis the values of IT Investment," Daechung Media, pp.181, May. 2004

[10] Ji-hun Jung, "The importance of strategic and reputational risks, and management methods," Financial Supervisory Service, pp.95, 2006.

[11] IBM Global Technology Service, Reputational risk and IT: Six keys to effective reputational and IT risk management, IBM Corporation, Mar. 2013

[12] IBM Global Technology Service, Reputational risk and IT: How security and business continuity and shape the reputation and value of your company, IBM Corporation, Sep. 2012

[13] Yong-deuk Kim, "Requires a new approach to reputational risk in the social media age," SW Insight 2013-11, National IT Industry Promotion Agency, Nov. 2013

〈저자소개〉

김 정 환 (Jeong-hwan Kim) 정회원
2007년 2월: 국민대학교 경영학부(재무.금융전공) 학사
2013년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 금융정보보안, 위험관리, 데이터베이스, 어플리케이션 성능튜닝

고 무 성 (Moo-seong Ko) 학생회원
2014년 2월: 한국산업기술대학교 컴퓨터공학과 학사
2014년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 정보보호, 위험관리, 개인정보보호정책

이 경 호 (Kyung-Ho Lee) 종신회원
1989년 8월: 서강대학교 수학과 학사
1997년 8월: 서강대학교 정보통신대학원 석사
2009년 8월: 고려대학교 정보보호대학원 박사
1994년 2월~현재: 삼성그룹, nhn, 시큐베이스 등 근무
2011년 9월~현재: 고려대학교 정보보호대학원 조교수
〈관심분야〉 위험관리, 정보보호컨설팅, 정보보호 및 개인정보보호정책