(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년04월05일
(11) 등록번호 10-1723779
(24) 등록일자 2017년03월30일

(51) 국제특허분류(Int. Cl.)
G06Q 10/02 (2012.01) *G06F 17/30* (2006.01)
G06Q 10/10 (2012.01) *G06Q 50/12* (2012.01)
(52) CPC특허분류
G06Q 10/02 (2013.01)
G06F 17/30386 (2013.01)
(21) 출원번호 10-2016-7010770(분할)
(22) 출원일자(국제) 2011년03월03일
심사청구일자 2016년04월22일
(85) 번역문제출일자 2016년04월22일
(65) 공개번호 10-2016-0054017
(43) 공개일자 2016년05월13일
(62) 원출원 특허 10-2012-7026031
원출원일자(국제) 2011년03월03일
심사청구일자 2012년10월04일
(86) 국제출원번호 PCT/US2011/026966
(87) 국제공개번호 WO 2011/109583
국제공개일자 2011년09월09일
(30) 우선권주장
12/718,446 2010년03월05일 미국(US)
(56) 선행기술조사문헌
US20050033621 A1
US20050283389 A1

(73) 특허권자
식스 콘티넨츠 호텔스, 인코포레이티드
미합중국 조지아주 30346 애틀란타시 라비니아 드라이브 3
(72) 발명자
곤잘레스 다리오
미국 조지아주 30152 케네소 엑터 오버룩 2030
옹 타쿠안
미국 조지아주 30068 마리에타 브록턴 클로스 3045
(뒷면에 계속)
(74) 대리인
특허법인태평양

전체 청구항 수 : 총 16 항 심사관 : 복진요

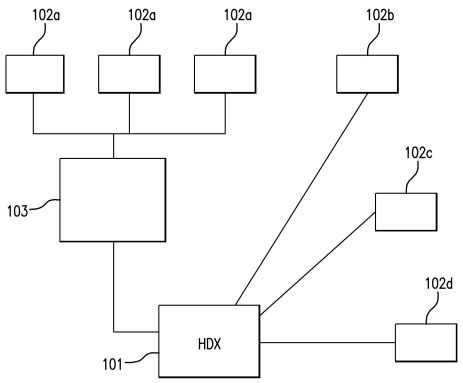
(54) 발명의 명칭 **상향식 방식의 최적화된 검색 시스템 및 방법**

(57) 요 약

호텔 객실을 탐색하기 위한 시스템 및 방법이 제공된다. 복수의 체류 기간의 호텔 객실에 연관된 실시간 가용성 속성들을 포함하는 사전-산출 데이터베이스가 생성되고, 날짜 범위(date range)에 대하여 적어도 하나의 호텔에 대한 쿼리가 수신된다. 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대한 가용성이 상기 쿼리에

(뒷면에 계속)

대 표 도 - 도1a



따른 비지니스 요구 사항들을 적용함으로써 계산된다. 최종 가용성이 상기 사전-산출 데이터베이스로부터의 가용성과 쿼리 속성들을 조합함으로써, 상기 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대해서 생성된다.호텔 객실들의 최종 가용성을 상기 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대하여 포맷될 수 있다.

(52) CPC특허분류
G06Q 10/10 (2013.01)
G06Q 50/12 (2013.01)
(72) 발명자
탄 대런 이순
미국 조지아주 30363 아틀란타 유닛 5011 17번 스트리트 390
맥클루어 조나단 피.
미국 조지아주 30005 알파레타 햄튼 패시지 2470
마테오 산티아고
미국 조지아주 30062 마리에타 딥 우즈 웨이 2172
라자라티남 라지엔데르쿠마르
미국 조지아주 30338 던우디 에이피티. 8315 애쉬포드 던우디 로드 4867

명 세 서

청구범위

청구항 1

호텔 객실(hotel room)의 가용성(availability)에 대한 데이터베이스를 검색하기 위한 컴퓨터에 의해 수행되는 방법으로서,

복수의 체류 기간(plurality of length of stays)에 대한 호텔 객실에 연관된 실시간 가용성 속성(real-time availability attributes)을 포함하는 사전-산출 가용성 상관 데이터베이스(relational database)를 컴퓨터 시스템에 의해 만드는 단계;

시작 날짜(start date)와 종료 날짜(end date)를 갖는 날짜 범위(date range)에 대하여 적어도 하나의 호텔에 대한 쿼리를 상기 컴퓨터 시스템에 의해 수신하는 단계;

상기 사전-산출 가용성 상관 데이터베이스로부터 상기 복수의 체류기간들 중 적어도 하나에 연관된 호텔 객실의 가용성을 상기 컴퓨터 시스템에 의해 결정하는 단계;

상기 호텔 객실의 가용성에 비즈니스 로직을 상기 컴퓨터 시스템의 가용성 규칙 계산기 엔진에 의해 적용하는 단계 -상기 비즈니스 로직을 적용하는 것은 가용성을 제어하기 위하여 호텔이 설정 가능한 속성을 포함하는 비즈니스 규칙을 적용하는 것을 포함하며, 여기서 비즈니스 규칙을 적용하는 순서는 처리 시간 사이클에 직접적인 영향을 가짐- ;

상기 가용성 규칙 계산기 엔진의 비즈니스 로직을 적용하는 것에 기초하여 상기 사전-산출 가용성 상관 데이터베이스로부터의 상기 가용성을 제한함으로써 상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 최종 가용성을 상기 컴퓨터 시스템에 의해 생성하는 단계;

상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대하여, 호텔 객실의 최종 가용성을 상기 컴퓨터 시스템에 의해 포맷하는 단계;

를 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 2

청구항 1에 있어서,

상기 체류 기간이 상기 체류 기간에 있는 날짜를 표현하는 각 비트를 갖는 비트 맵(bit map)으로 표현되는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 3

청구항 1에 있어서,

상기 쿼리는 여행 웹 사이트로부터 수신되는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 4

청구항 1에 있어서,

상기 쿼리는 클라이언트로부터 수신되는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 5

청구항 1에 있어서,

상기 적용하는 단계는 제한 규칙(restriction rule)으로 수행되는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 6

청구항 1에 있어서,

호텔 객실에 대한 상기 최종 가용성을 여행 웹 사이트에 전송하는 단계를 더 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 7

청구항 1에 있어서,

호텔 객실에 대한 상기 최종 가용성을 클라이언트에게 전송하는 단계를 더 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 방법.

청구항 8

호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템으로서,

복수의 체류 기간에 대한 호텔 객실에 연관된 실시간 가용성 속성을 포함하는 사전-산출 가용성 상관 데이터베이스를 만드는 수단;

시작 날짜와 종료 날짜를 갖는 날짜 범위에 대하여 적어도 하나의 호텔에 대한 쿼리를 수신하는 수단;

상기 사전-산출 가용성 상관 데이터베이스로부터 상기 복수의 체류기간들 중 적어도 하나에 연관된 호텔 객실의 가용성을 결정하는 수단;

상기 호텔 객실의 가용성에 비즈니스 로직을 적용하는 수단 -상기 비즈니스 로직을 적용하는 수단은 가용성을 제어하기 위하여 호텔이 설정 가능한 속성을 포함하는 비즈니스 규칙을 적용하는 수단을 포함하며, 여기서 비즈니스 규칙을 적용하는 순서는 처리 시간 사이클에 직접적인 영향을 가짐- ;

상기 비즈니스 로직을 적용하는 것에 기초하여 상기 사전-산출 가용성 상관 데이터베이스로부터의 상기 가용성을 제한함으로써 상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 최종 가용성을 생성하는 수단;

상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대하여, 호텔 객실의 최종 가용성을 포맷하는 수단을 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 9

청구항 8에 있어서,

상기 체류 기간이 상기 체류 기간에 있는 날짜를 표현하는 각 비트를 갖는 비트 맵으로 표현되는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 10

청구항 8에 있어서,

상기 쿼리를 수신하는 수단은 여행 웹 사이트로부터 상기 쿼리를 수신하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 11

청구항 8에 있어서,

상기 쿼리를 수신하는 수단은 클라이언트로부터 상기 쿼리를 수신하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 12

청구항 8에 있어서,

상기 적용하는 수단은 제한 규칙으로 상기 산출을 수행하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 13

청구항 8에 있어서,

호텔 객실에 대한 상기 최종 가용성을 여행 웹 사이트에 전송하는 수단을 더 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 14

청구항 8에 있어서,

호텔 객실에 대한 상기 최종 가용성을 클라이언트에게 전송하는 수단을 더 포함하는 호텔 객실의 가용성에 대한 데이터베이스를 검색하는 시스템.

청구항 15

그 내부에, 호텔 객실의 가용성에 대한 데이터베이스를 검색하기 위한 방법을 구현하기 위해 실행되도록 제작된 컴퓨터 판독가능 프로그램 코드를 저장한 컴퓨터로 판독 가능한 기록 매체로서, 상기 방법은

복수의 체류 기간에 대한 호텔 객실에 연관된 실시간 가용성 속성을 포함하는 사전-산출 가용성 상관 데이터베이스를 컴퓨터 시스템에 의해 만드는 단계;

시작 날짜와 종료 날짜를 갖는 날짜 범위에 대하여 적어도 하나의 호텔에 대한 쿼리를 상기 컴퓨터 시스템에 의해 수신하는 단계;

상기 사전-산출 가용성 상관 데이터베이스로부터 상기 복수의 체류기간들 중 적어도 하나에 연관된 호텔 객실의 가용성을 상기 컴퓨터 시스템에 의해 결정하는 단계;

상기 호텔 객실의 가용성에 비즈니스 로직을 상기 컴퓨터 시스템의 가용성 규칙 계산기 엔진에 의해 적용하는 단계 -상기 비즈니스 로직을 적용하는 것은 가용성을 제어하기 위하여 호텔이 설정 가능한 속성을 포함하는 비즈니스 규칙을 적용하는 것을 포함하며, 여기서 비즈니스 규칙을 적용하는 순서는 처리 시간 사이클에 직접적인 영향을 가짐- ;

상기 가용성 규칙 계산기 엔진의 비즈니스 로직을 적용하는 것에 기초하여 상기 사전-산출 가용성 상관 데이터베이스로부터의 상기 가용성을 제한함으로써 상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 최종 가용성을 상기 컴퓨터 시스템에 의해 생성하는 단계;

상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대하여, 호텔 객실의 최종 가용성을 상기 컴퓨터 시스템에 의해 포맷하는 단계를 포함하는 컴퓨터로 판독 가능한 기록 매체.

청구항 16

사전-산출 데이터와 연산중인(on-the-go) 데이터 사이의 밸런싱에 의하여 데이터베이스에서의 호텔 객실의 가용성에 대한 검색을 개선하기 위한 컴퓨터에 의해 수행되는 방법으로서,

사전-산출 데이터베이스에 대한 쿼리를 수신하기 전에 연산된 데이터를 포함하는 상기 사전-산출 데이터베이스를 컴퓨터 시스템에 의해 만드는 단계 -상기 데이터는 복수의 체류 기간에 대한 호텔 객실에 연관된 실시간 가용성 속성을 포함함- ;

상기 사전-산출 데이터베이스에 대한 상기 쿼리를 상기 컴퓨터 시스템에 의해 수신하는 단계 -상기 쿼리는 시작 날짜와 종료 날짜를 갖는 날짜 범위에 대한 적어도 하나의 호텔에 대한 요청을 포함함- ;

상기 쿼리에 기초하여 상기 사전-산출 데이터베이스로부터 제1 검색 결과를 상기 컴퓨터 시스템에 의해 결정하는 단계 - 상기 제1 검색 결과는 상기 복수의 체류 기간들 중 적어도 하나에 연관된 제1 가용성을 포함함- ;

상기 쿼리 및 요구사항(requirement)에 기초하여 제2 검색 결과를 상기 컴퓨터 시스템에 의해 계산하는 단계 - 상기 제2 검색 결과는 상기 사전-산출 데이터베이스와는 다른 소스로부터 계산되며, 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜의 가용성을 포함하며, 상기 요구사항은 상기 제2 검색 결과를 계산하기 위한 처리 시간을 특정하는 요구사항을 포함함- ;

상기 제1 검색 결과와 상기 제2 검색 결과를 조합하여 상기 컴퓨터 시스템에 의해 최종 결과를 생성하는 단계 - 상기 최종 결과는 상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 최종 가용성을 포함함- ;

상기 최종 가용성을 상기 컴퓨터 시스템에 의해 포맷하는 단계 - 상기 포맷은 상기 복수의 체류 기간에 있어서의 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대한 상기 호텔 객실의 최종 가용성을 포맷하는 것을 포함하며, 상기 복수의 체류 기간의 각각은 그 체류 기간에 특유한 날짜의 체류를 포함하는,

호텔 객실의 가용성에 대한 검색을 개선하는 방법.

발명의 설명

기 술 분 야

[0001] 현존하는 호텔 및 여행 전산 예약 시스템은 검색 리퀘스트(search request)로부터 발견되는 단일의 가용성(공실 상황) 셀(single availability cell)들을 생성하는 하향식(top down) 검색 방식을 사용한다. 예를 들면, 그러한 시스템들은 사용자가 선택된 숙박수(즉, 단일 체류 기간(Length of Stay))에 대하여 이용가능한 호텔 객실들을 찾기 위해 호텔의 아이덴티티(identity), 체크-인 날짜, 및 체크-아웃 날짜와 같은 검색 기준(search criteria)을 입력하는 것을 허용할 수 있다. 제품 아이템(즉, 호텔 객실들)을 찾은 후에야 비로소 제품 아이템의 특정 세트에 대한 요금(rates)과 가용성을 찾아내기 위해 비지니스 로직(business logic)이 적용된다.

배 경 기 술

[0002] 이 선행기술 접근 방식은 몇 가지 단점이 있다. 이 접근 방식이 하향식 특성(top-down nature)을 갖기 때문에, 제품 아이템(예컨대, 대여 가능한 객실)들이 각 리퀘스트에 대해 검색되고, 발견되어야 할 필요가 있다. 이에 더하여, 검색 리퀘스트는 반드시 최소한으로 호텔 이름/코드, 체크-인 날짜 및 체크-아웃 날짜를 가져야 한다. 가용 리퀘스트(availability request)는 상술한 최소 요소들이 그 리퀘스트 내에 없을 경우, 거절될 것이다.

[0003] 또한, 선행기술 접근 방식인 코어 서비스(core service)는 한번에 오직 하나의 확인대상 호텔을 허용하고, 검색에 체크-인 날짜 및 체크-아웃 날짜가 사용된 것을 감안하면, 요금 및 가용성 응답이 오직 특정 숙박일수를 반영한다. 기본 검색은 대체 날짜들을 사용하는 것을 허용하지 않으므로, 대체 요금들도 발견되지 않는다.

[0004] 만일, 도시 검색이 수행되면(즉, 한 도시에 초점을 맞춘 검색), 이 리퀘스트는 단일 호텔들에 대한 다수의 검색들로 해체되고, 그러한 각각의 코어 호텔 검색(core hotel search)은 1회에 한번 수행된다. 이 코어 검색 서비스는 다수의 호텔들 사이의 관계를 인식하지 않는다. 각 코어 리퀘스트는 요금 및 가용성을 찾기 위해 수행되어야 할 복잡한 비지니스 규칙(business rule)들을 요구하는 각 리퀘스트로 수행되어 한번에 한 호텔을 검색하기 때문에, CPU 시간 및 리소스(resource)의 비용이 높다. 또한, 각 리퀘스트에 대해 CPU 및 리소스가 집중되는 특성을 고려할 때, 그 응답 시간(response time)이 상대적으로 늦어질 수 있다. 매(每) 리퀘스트에 대한 요금 및 가용성 규칙들을 찾고 산출하고자 하는 요구에 시간이 소요된다.

[0005] 또한, 선행기술 시스템에서, 이 리퀘스트는 매우 사용자 특정적(user-specific)이며, 트랜잭션 시스템(transaction system)이 제품의 대체 유형(예컨대, 다른 객실 유형 등)을 찾는 것을 허용하지 않는다.

발명의 내용

해결하려는 과제

[0006] 현존하는 기술의 가장 큰 단점은 단일 가용성 셀일 수 있다. 단일 가용성 셀은 검색 리퀘스트 내의 잘 정의된 체류 기간(LOS;Length of Stay)으로부터 유도된다. 이 LOS는 검색 리퀘스트 내의 체크-인 날짜와 체크-아웃 날짜로부터 유도된다. 검색 리퀘스트 내의 LOS로부터 유도될 수 있는 다중 후보 조합들(multiple possible combinations)을 볼 수 없다. 예를 들면, 만일 리퀘스터(requestor)가 또한 LOS의 관점에서 다른 옵션들을 평가하고자 한다면, 그것은 각 LOS 옵션에 대해 몇 가지 개별적인 검색 리퀘스트를 요구할 것이다. 선행기술은 대체 날짜 또는 제품 아이템의 유연성을 제공하지 않는다. 이 경우에는, 충분한 유연성을 갖는 응답을 생성하는데 요구되는 비용 및 시간이 비용 효율적(cost effective)이지 않다. 예를 들면, 만일 고객이 5월 첫째 주에 대해 가용성을 체크하고자 한다면, 49개의 개별적인 가용성 호출(call)이 요구될 것이다. 7개의 가용성이 그 주(week)의 각 요일에 대해 호출된다(즉, 7일×7 요일당 LOS값(LOS values per day)). 이 트랜잭션은 48배의 인자(즉,

1 호출과 49 호출의 차이)만큼 비용을 증가시킬 것이며, 이로 인해 해당 고객에게 매우 긴 응답시간(즉, 불쾌한 사용자 경험)을 줄 것이다.

[0007] 그러므로, 이하에서 보다 상세하게 설명하는 것처럼, 여행 및 호텔 트랜잭션 시스템을 개선하기 위한 기술의 필요성이 존재한다.

과제의 해결 수단

[0008] 본 발명에 따른 상향식 방식의 최적화된 검색 시스템 및 방법이 제공된다.

발명의 효과

[0009] 본 발명은 상향식 방식의 최적화된 검색 시스템 및 방법을 제공할 수 있다.

도면의 간단한 설명

[0010] 본 명세서에 통합되고, 명세서의 일부로서 구성된 첨부 도면들은 실시예들을 예시하며, 상세한 설명과 더불어 그 방법 및 시스템의 원리를 설명하는 것을 돕는다.

도 1a는 호텔 예약 시스템의 아키텍처를 도시한다.

도 1b는 본 발명에 따른 일 실시예에서의 기본 빌딩 블록들의 아키텍처를 도시한다.

도 2는 본 발명의 또 다른 실시예에서의 아키텍처를 도시한다.

도 3은 일 실시예에서, 본 발명을 도시하는 하이 레벨 시퀀스 다이어그램을 도시한다.

도 4는 본 발명에 따른 일 실시예에서, 본 발명과 함께 사용하기 위한 프레임워크 아키텍처 플로우 다이어그램(framework architecture flow diagram)이다.

도 5는 본 발명에 따른 일 실시예에서, 본 발명과 함께 사용될 수 있는 다양한 계산기를 도시하는 다이어그램이다.

도 6은 마스크 계산기(mask calculator)로 사용될 수도 있는 샘플 UML이다.

도 7은 본 발명에 따른 일 실시예에서, 본 발명에 의해 수행될 수 있는 단계들을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0011] 본 발명에 따른 방법 및 시스템이 개시되고 설명되기에 앞서, 여기에서 사용되는 전문용어는 단지 특별한 실시예들을 설명하기 위한 목적일 뿐이며, 제한될 수 있는 것으로 의도된 것이 아님은 이해되어야 한다.

[0012] 상세한 설명 및 첨부된 청구범위에서 사용된 바와 같은, 단수형 표현형식 "하나의(a, an)~" 및 "그(the)~"는 문맥상 명백하게 다르게 지시된 바가 없으면 복수의 지시대상물을 포함한다. 범위는 여기에서 하나의 특정 값에 대한 "약(about)~"에서부터, 및/또는 또다른 특정 값까지의 "약~"로 표현될 수 있다. 이러한 범위가 표현되는 경우, 단지 예를 든 것일 뿐이며, 실시예는 하나의 특정 값에서부터 및/또는 다른 특정 값까지를 포함할 수 있다. 마찬가지로, 값들이 근삿값으로 표현된 경우, 앞선 "약~"의 사용 선례에 의해, 그것은 또다른 실시예의 특정 값 형태로 이해될 것이다. 나아가 각 범위의 종단점들은 다른 종단점과 연관된 경우 및 다른 종단점과 독립된 경우 모두를 의미하는 것으로 이해될 수 있다.

[0013] "선택적(optional)" 또는 "선택적으로(optionally)"는 연이어 설명되는 이벤트(event) 또는 상황이 일어날 수도 혹은 일어나지 않을 수도 있음을 의미하며, 본 명세서는 상기 이벤트(event) 또는 상황이 일어난 사례 및 일어나지 않은 사례를 포함한다.

[0014] 본 명세서의 청구범위 및 상세한 설명의 전반에 걸쳐, 단어 "포함하다(comprise)" 및 이 단어의 변형예(예컨대, "포함하는" 및 "포함한다")는 "포함하지만 그에 한정되지 않는(including but not limited to)"를 의미하고, 예를 들어 다른 부가물들, 구성요소들, 정수(integer)들 또는 단계들을 배제하는 것을 의도하지 않는다. "예시적인(exemplary)"은 "하나의 예로서~(an example of)"를 의미하며, 선호함의 표시 또는 이상적인 실시예를 암시하는 것을 의도하지 않는다. "~와 같은(such as)"은 제한하는 의미로 사용된 것이 아니라, 설명을 위한 목적으로 사용된 것이다.

[0015] 개시된 것은 개시된 방법들 및 시스템들을 수행하기 위해 사용될 수 있는 구성요소들이다. 이들 및 다른 구성요소들은 여기에 개시되고, 이들 구성요소들의 조합들, 부분집합(subset)들, 상호작용(interaction)들, 그룹들 등이 개시된 것은 각각의 다양한 개별적 그리고 집합적 조합들 및 이들의 치환에 의한 특정 참조가 명시적으로 개시되어 있지 않더라도, 각각 모든 방법들 및 시스템들을 위해 특별히 계획되어 여기에 설명된 것으로 해석된다. 이것은 본 출원이 포함하고 있는 모든 관점에 적용되며, 개시된 방법들 내의 단계들에 한정되지 않는다. 그러므로, 만일 수행될 수 있는 다양한 추가적인 단계들이 있다면, 이들 각각의 추가적인 단계들은 어떤 특정한 실시예 또는 개시된 방법들의 실시예들의 조합과 더불어 수행될 수 있는 것으로 해석된다.

[0016] 본 발명에 따른 방법들 및 시스템들은 후술될 선호되는 실시예들 및 그안에 포함된 예시들의 상세한 설명, 그리고 도면과 이전 및 이후에 이어지는 설명을 참조함으로써 보다 쉽게 이해될 수 있다.

[0017] 본 발명은 그 기능이 향상된 호텔 객실 가용성 트랜잭션 시스템을 제공한다. 도 1a는 Six Continents Hotels, Inc.에 의해 그들의 전세계 체인들을 대상으로 운용되는 HOLIDEX(HDX) 시스템으로 불리우는 최첨단의 호텔 예약 시스템의 아키텍쳐를 도시한다. 본 발명은 이 시스템 또는 시장에서 이용될 수 있는 임의의 다른 적절한 호텔 예약 시스템과 결합되어 사용될 수 있다.

[0019] 도 1a의 예약 시스템에서, HDX 서버(101)는 날짜, 요금 뿐만 아니라 호텔 객실과 관련한 다른 상세한 정보에 대한 현재의 가용성을 저장한다. 다양한 클라이언트들(102a, 102b, 102c 및 102d)이 이 HDX 서버(101)에 리퀘스트를 보내올 수 있다. 예를 들면, 클라이언트(102a)는 중개 데이터 센터(intermediary data center)(103)를 통해 HDX 서버(101)와 연결되고, 예컨대 OTA 또는 XML 프로토콜을 사용하여 HDX 서버(101)와 통신하는 여행 웹 사이트를 포함하여 구성될 수 있다. 클라이언트(102b)는 예컨대, 페가수스 프로토콜(Pegasus Protocol)을 사용하여 HDX 서버와 통신하는 Sabre, Galileo, WorldSpan, Amadeus 또는 TravelWeb과 같은 글로벌 디스트리뷰션 시스템(GDS;Global Distribution Sytem)을 포함하여 구성될 수 있다. 호텔 브랜드의 기업 소유자는 예컨대, 3270 또는 XML 프로토콜로서, 클라이언트(102c)를 통해 HDX 서버와 통신할 수 있다. 결과적으로 개별 호텔들(102d)은 예컨대, XML 및/또는 HMI 프로토콜을 이용하여, HDX 서버(101)와 통신할 수 있다.

[0020] 클라이언트(102a, 102b, 102c, 및/또는 102d)에 의해 HDX 서버(101)로 제출된 가용성 검색 리퀘스트들은 최소한의 것으로서 호텔 코드, 체크-인 날짜 및 체크-아웃 날짜를 포함할 수 있다. 다수의 호텔들, 다수의 날짜 범위 등에 대한 검색은 통상적으로 HDX 서버(101) 상에서 다수의 검색들에 대한 수요를 초래하기 때문에, 이하에 덧붙여 설명된 바와 같이, 본 발명은 검색을 수행하는 개선된 방법을 제공한다. 주요 데이터 센터 마다에 있어서 오늘날 환경(즉, 하드웨어 및 네트워크)에서의 가장 큰 도전은 매년 가용성 트래픽이 60%씩 엄청나게 증가하고 있다는 것이다. 가장 효율적인 검색 프로세스는 사전-산출된(pre-compute) 데이터와 연산중인(on-the-go) 데이터 사이의 정밀한 균형이다. 매일 수백만의 검색 리퀘스트들이 예약 시스템을 히트(hit)하는 경우, 가용성 데이터의 일부를 미리 산출하여 갖는 것을 줄이는 것이 강력한 가용성 트랜잭션 시스템으로 이어진다. HDX 서버(101)상에서 가장 효과적으로 검색을 수행하는 방법을 결정하기 위해 가용성 프로세스는 다양한 기준(criteria)에 기초하여 각 단계를 분류화(categorization)하기 위해 토큰화(tokenized)된다. 분류화는 많은 형태를 가질 수 있으나, 일 실시예로서 다음의 [표 1]의 기준이 사용될 수 있다.

표 1

[0021]

기준	토큰화(tokenization)
발견 프로세스(Discovery process)	리퀘스트에 대한 제품 아이템을 찾기 위해 요구되는 로직(logic)과 데이터베이스 호출들의 양(量)에 기초하여 가용성의 요소 하나하나가 등급화(낮음에서 높음까지) 됨. 예컨대, 일 실시예에서, 그 등급은 다음과 같을 수 있음. - 호텔 = 낮음 - 요금 카테고리 = 낮음 - 제품 아이템= 높음
처리사이클(Processing cycles)	각각의 비지니스 로직 구성요소는 그것이 필요로 하는 처리 시간의 양(量)에 기초하여 등급화(낮음에서 높음까지) 될 수 있음. 예컨대, 일 실시예에서, 그 등급은 다음과 같을 수 있음. - 도착 근접도 = 낮음 - 체류 일수 = 중간 - 순이익 = 높음

연관 레벨(Association level)	각각의 비지니스 로직 구성요소는 특정 비지니스 로직을 실행하는 가용성의 요소와 관련될 수 있음. 예컨대, 일 실시예에서, - 순이익 = 제품 아이템 - 체류 일수 = 요금 카테고리

[0022] 도 1b에 도시된 바와 같이, 본 발명은 일 실시예로서, 가용성 규칙 계산기 엔진(Availability Rule Calculator Engine)(152) 뿐만 아니라 사전산출 가용성 데이터베이스(PreCompute Availability Database)(151)를 포함하며, 이 사전산출 가용성 데이터베이스(151)는 두 가지의 주요 특징을 갖는 상관 데이터베이스(relational database)를 포함하여 구성될 수 있다.

[0023] ● 각각의 행(row)은 하루 동안 팔릴 가능성이 가장 낮은 유니트(제품 아이템)를 나타낸다. 하루 단위로 제품 아이템의 세트를 찾기 위한 발견 프로세스는 요구되지 않는다. 하루 단위의 제품 아이템 하나하나는 등급과 날짜 범위 가용성에 의해 연관지어진 객실 유형에 따른 요금 코드 정의(rate code definition)를 사용하는 하이 레벨 요금 카테고리로부터 도출될 수 있다.

[0024] ● 테이블(table)의 크기는 관리될 수 있어야 한다. 이 테이블은 업데이트와 독출이 빠르다. 각 제품 아이템에 대한 구성요소의 수는 각각의 행(row)에서 사전산출을 위한 구성요소의 수를 최소화로 유지할 수 있도록 하기 위해 매우 선택적이다.

[0025] 데이터베이스 프라이머리 키(database primary key)는 호텔 코드, 날짜, 객실 유형, 및 요금 카테고리에 의해 형성될 수 있다. 일 실시예로서, 각 행(row)의 다른 키 데이터(key data)는 다음과 같을 수 있다.

[0026] ● **물리적 체류기간(Physical LOS)** : 이 스트링 값(string value)은 최대, 예를 들면, 14일의 체류(물론 어떤 다른 최대 LOS가 선택될 수도 있음)의 각각의 밤에 대한 비트 마스크(bit mask)를 나타낸다. 객실 유형의 물리적 가용성은 배정(allocation), 객실의 총수(total number of rooms) 및 단체 할인 요금(run of the house) 객실 유형(객실 지정 없는 객실 유형)의 개념(concepts)을 사용할 수 있다. 이들 규칙들은 매일 밤 LOS 마스크에 반복적으로 적용될 수 있다.

[0027] ● **요금 총액(Rate Amount)** : 요금 총액은 제품 아이템의 요금 카테고리와 연관된 요금 코드로부터 도출될 수 있다. 총액 값은 일률적인 금액(flat amount) 또는 백분율 값(percent value)일 수 있다. 이와 동시에 총액은 요금 코드 레벨 또는 또다른 요금 코드 기준으로부터 발견될 수 있다.

[0028] ● **수익 LOS(Revenue LOS)** : 이 스트링 값은 최대, 예를 들면, 14일 체류(또는 그밖의 다른 최대 LOS)의 각각의 밤에 대한 비트 마스크를 나타낼 수 있다. 수익 관리 구성요소(yield management component)는 LOS에 기초한 순이익 계산을 사용할 수 있다. 플러스의 이익은 열림(open)(1)으로 표현되고, 마이너스의 수익은 닫힘(closed)(0)으로 표현될 수 있다. 다른 요금 또는 오버라이드(override)들에 기초하여, 다른 값을 사용하도록 지시할 수 있는 몇 가지의 이익 옵션이 있을 수 있다. 이들 규칙들은 매일 밤 LOS 마스크에 반복적으로 적용될 수 있다.

[0029] ● **요금 코드(Rate Code)** : 패리티 컨셉(parity concept)은 모든 요금 코드들 사이의 관계의 레벨을 유지하기 위한 요금 코드를 사용할 수 있다.

[0031] 이 상관 데이터베이스의 레이아웃은 제품 아이템을 도출해내는 매우 빠른 방법을 만들어낸다. 그 초기 단계에서 필터들을 구현해내기 위하여 사전산출 가용성 데이터베이스(151)가 속성 테이블들(attribute tables)과 연결되는 것은 가능하다. 일 실시예에서, 침대 유형, 침대의 개수, 및/또는 최대 수용가능 인원과 같은 몇몇 속성들은 제품 아이템의 사전 선택을 돕는다. 이것은 비자격(non-qualifying) 제품 아이템 상에서 추가적인 프로세싱 사이클이 수반되는 가용성 프로세스의 끝부분에서 필터들을 제공하도록 하는 현존 기술과는 다른 것이다.

[0032] LOS 도출 값들은 비트 마스크 레프리젠테이션(bit mask representation)을 사용할 수 있다. 각각의 비트(bit)는 하룻밤(a night)을 나타내며, 그 값들은 열림(1) 또는 닫힘(0)이 될 수 있다. LOS가 얼마나 길게 표현될 수 있을지에 대한 제한은 없다. 적정 값(proper value)은 적정 청중(proper audience) 또는 호텔 브랜드에 상응할 수 있다. 예를 들면, 일 실시예에서, 장기 체류형(extended stay) 브랜드들은 최장 28일 또는 그 보다 긴 LOS가 산출될 수 있다.

[0033] 도 2에 도시된 바와 같이, 일 실시예에서, 가용성 규칙 계산기 엔진(151)과 사전산출 가용성 데이터베이스(152)는 웹스피어 MQ(202)로부터 메시지를 가로채어, 그 메시지들을 처리하고, 요구된 데이터로 오라클 데이터베이

스(203)를 업데이트하는 J2EE 애플리케이션(201)을 포함하는 사전산출 프레임워크(PreCompute framework)를 포함할 수 있다. 프레임워크는 코어 자바 애플리케이션(204)을 포함할 수 있는 J2EE 애플리케이션(201)을 포함할 수 있다. 도 2는 본 발명의 일 실시예에 따른, 각 애플리케이션 내에서 주요 구성요소들을 구비하는 전체 프레임워크(framework)를 도시한다. 물론 다른 적절한 프레임워크들도 사용될 수 있다.

[0034] 도 3은 도 1b 및 도 2의 사전산출 엔진에서의 전반적인 메시지 처리 흐름을 도시한 하이 레벨 시퀀스 다이어그램이다. 이 시퀀스 다이어그램은 메시지들이 시퀀스에 도착하여 즉시 처리될 때의 시나리오를 나타낸다. 도 4는 일 실시예에 있어서, 본 발명의 전반적인 플로우를 도시한 프레임워크 아키텍쳐 플로우 다이어그램(framework architecture flow diagram)이다.

[0036] 도 2 내지 도 4의 구조적인 구성요소들은 이하에서 보다 상세하게 정의될 것이다.

[0037] ● **사전산출 MD 빈(PreComputeMDBean)(211)** : 사전산출 MD 빈(211)은 HOLIDEX(101)로부터 메시지를 수신하는 MQ 시리즈(Series)(202) 큐(Queue)(411)를 청취하는 메시지 구동 빈(Message Driven Bean)이다. 이것은 HOLIDEX(101)로부터 수신된 XML 메시지로 메시지 관리자(MessageManager)(214)를 호출한다.

[0038] ● **무상태 세션 빈(Stateless Session Bean)(212)** : 이어지는 무상태 세션 빈(212)이 사용될 수 있다.

[0039] ● **부트스트랩 빈(BootStrapBean)** : 애플리케이션을 초기화하는데 사용됨. 로그4j(Log4j) 및 다른 타이머 태스크 오리엔티드 빈(Timer Task oriented bean)들이 애플리케이션이 시동되었을 때 이곳에 초기화된다.

[0040] ● **메시지관리자 빈(MessageManagerBean)** : 이 EJB는 모든 소스들(sources) 및 클라이언트들(clients)에 대한 메시지를 처리하는데 사용된다. 기본적으로 이것은 메시지 관리자(MessageManager)(214)를 호출하거나 또는 그것을 확장할 수 있다. 또한 이 빈(bean)은 MDB(211)에 의해 호출될 수 있다.

[0041] ● **타이머 빈들(Timer Beans)(231)** : 이 EJB는 타이머 서비스(Timer Service)를 이용한다. 이 EJB는 펜딩 메시지 테이블(pending message table) 안에 펜딩되는 메시지들을 처리하는데 사용될 수 있다. 이 EJB는 사전 설정된 시간에 동작하여, 메시지관리자(MessageManager)를 호출하며, 펜딩 메시지를 가지고 있는 임의의 호텔을 잠금(lock)하고, 메시지를 처리한다. 홀리덱스(Holidex)로부터의 이벤트(events)들을 순차적으로 처리하는 것은 중요하다. 클러스터 환경의 구현(즉, 다른 노드들은 동일 호텔에 대한 업데이트를 동시에 획득할 수 있음)은 이 업데이트 처리 방법을 요구한다.

[0042] ● **메트릭 관리자(Metric Manager)(232)** : 이 무상태 세션 빈(stateless session bean)은 타이머 서비스(Timer Service)를 이용하여 사전 정의된 시간(매시간을 말함)에 메트릭스(metrics)를 수집하는데 사용된다. 그것이 타임드오브젝트(TimedObject)를 구현한다. 이 EJB는 부트스트랩 빈(BootStrapBean)에 의해 초기화되게 되고, 타이머가 생성되게 된다.

[0043] ● **데이터 클라이언트들(Data Clients)(233)** : 데이터 클라이언트 구성요소는 사전산출 엔진(PreCompute engine) 내부에 존속하는 단일 구성요소이다. 이것의 역할은 두 가지이다. 첫째, 이것은 푸시 클라이언트(Push client)를 대리하여 제품 아이템을 어떻게 구축할지에 관련된 특정 정보를 제공한다. 또한, 둘째, 이것은 그들 푸시 클라이언트에 변화가 생성되었을 때, 이를 알리는 수단을 제공한다.

[0044] ● **로깅(Logging)(234)** : Log4J는 애플리케이션에 대한 로깅 메카니즘으로서 사용되게 된다. 로그된 메시지들은 애플리케이션 서버가 운용되고 있는 서버 상의 파일로 보내지게 된다. 로그4j에 대한 구성은 부트 스트랩 EJB를 이용하거나 또는 애플리케이션 서버 내의 시동 클래스(start up class)를 이용하여 로드된다. 다른 로깅 레벨들, 즉 디버그(Debug), 인포(Info), 경고(Warn), 및 에러(Error)가 있다.

[0045] ● **메시지 관리자(MessageManager)(214)** : 메시지 관리자(214)는 주로 두 가지 오퍼레이션을 수행할 수 있다. 첫째, XML 메시지를 자바 오브젝트(Java object)로 변환하고, 이어 메시지 유형에 기초하여 특정한 사전산출 관리자(PreCompute Manager)(216)를 호출할 수 있다. 또한, 호텔 코드 및 메시지 시퀀스 넘버(message sequence number)에 기초하여 메시지들의 순차적 처리를 관리한다. 메시지들이 순차적이지 않으면, 이 메시지들은 이후 처리를 위해 분리된 테이블 내에 저장될 것이다.

[0046] ● **관리자 팩토리(ManagerFactory)(218)** : 관리자 팩토리(218) 클래스는 메시지 유형에 기초하여 사전산출 관리자(PreComputeManager)(216)를 동적으로 생성한다. 관리자를 생성하게 되면, 이것은 이후 사용을 위해 저장(cache)될 수 있다.

[0047] ● **XML스키마 자바 클래스들(XML Schema Java Classes)** : 이들 XML스키마 자바 클래스들은 메시지 관리자(21

4)의 일부분일 수 있으며, HOLIDEX(101)로부터 수신된 XML 메시지들에 기초하여 생성된다. 이들 클래스들은 그들의 메소드(method) 내에서의 속성들 및 XML 요소들의 값을 유지한다. 이들 XML 빈 클래스들은 데이터 캐리어(data carrier)들로서 사용되기 때문에 각 메시지에 대한 개별적인 전송 오브젝트를 피하게 된다.

[0048] ● **사전산출 관리자(PreComputeManager)(216)** : 일 실시예에서, 각 메시지 유형 또는 메시지 유형들의 조합에 대한 개별 사전산출 관리자(216)가 존재할 것이다. 이들의 기본적인 기능은 수신된 자바 오브젝트를 몇가지 비즈니스 로직(business logic) 및 사전산출 유틸리티 클래스들(PreCompute Utility classes)을 이용하여 희망하는 데이터베이스 관련 필드들(fields)로 변환하는 것이다. 대부분의 비즈니스 프로세싱은 이곳과 유틸리티 클래스들(Utility Classes) 내에서 일어난다.

[0049] ● **호텔락 유틸(HotelLockUtil, 220)** - 이 클래스는 하우스 메시지 테이블에 있는 호텔 데이터를 잠금(lock)하는데 사용될 수 있다. 레코드가 잠금되게 되면, 해당 호텔에 대한 메시지들이 처리된다. 순서대로 메시지들을 처리하고 동일한 호텔에 대해 다른 메시지들을 갖는 경합 조건(race condition)을 방지하기 위해 호텔을 잠금할 필요가 있다. 메시지들이 순차적으로 기록되지 않으면, 전체 데이터베이스가 쓸모없게 될 것이다.

[0050] ● **펜딩 메시지 유틸(PendingMessageUtil)** - 이 클래스는 펜딩 메시지 테이블(pending message table)에 있는 메시지들을 검색(retrieve) 및 삭제, 저장하기 위해 사용된다. 메시지들이 클러스터 환경을 사용하여 MQ 큐로부터 독출되는 동안, 순차적으로 그 메시지들을 처리할 필요가 있다. 각 호텔 메시지는 처리하는 시퀀스 순서를 유지하기 위해 시퀀스 넘버를 포함한다.

[0051] ● **유효화 검증 클래스(Validation classes, 222)** - 일 실시예에서, 각 메시지에 대해서 별도의 유효화 검증 클래스들이 존재한다. 이들 유효화 검증 클래스들은 메시지들을 유효화 검증하여, 만약 에러가 발견되면, 에러가 있는 메시지를 사전산출예외(PreComputeException)로 던진다.

[0052] ● **데이터 액세스 오브젝트들(DAOs: Data Access Objects, 226)** - 데이터 액세스 오브젝트들(226)은 오라클 데이터베이스(203)와 관련된 통신 및 클러드(CRUD) 오퍼레이션(operation)들을 추출한다. 사전산출(PreCompute) 관리자(216)는 테이블들을 업데이트하기 위해 적절한 DAO(226)를 호출한다. DAO(226)는 단일 테이블에 기초하거나, 관련된 테이블들에 기초할 수 있다. 기본적으로, DAO(226)는 DB커넥터(Connector) 클래스(228)로부터 데이터베이스 연결을 획득하고, 데이터베이스에 대한 SQL 호출을 준비하고, 클러드 오퍼레이션(operation)들을 수행한다. 이러한 오퍼레이션(operation)들이 완료되면, 데이터베이스(203)에 대한 연결을 해제(release)한다.

[0053] ● **DB커넥터(228)** - DB커넥터(228)는 데이터베이스(203)를 위한 연결 메커니즘을 획득하는 자바 클래스이다. 일 실시예에서는, 데이터소스 오브젝트를 호출함으로써 데이터베이스(203)에 대한 연결을 얻을 수 있다. 이 클래스는 연결 해제, 트랜잭션 롤링 백 등과 같은 다양한 메소드들을 가질 것이다.

[0054] ● **사전산출 유틸리티 구성요소들(PreCompute Utility Components)** - 애플리케이션을 위해 필요한 모든 공통 기능은 날짜 유효화 검증 또는 애플리케이션의 필요에 따른 특정 데이터의 변환과 같은 유틸리티 클래스들로서 개발될 수 있다. 메시지 유형들에 특정하는 여러 유틸리티 클래스들이 존재할 수 있다.

[0056] 가용성 규칙 계산기 엔진(Availability Rule Calculator Engine, 152)은 두 개의 카테고리로 그룹화되는 로직을 포함할 수 있다.

[0057] ● 검색 결과의 체크-인 날짜 값에 기초하며, 따라서 사전-산출될 수 없는 비지니스 로직. 이러한 로직의 예시에는 사전 구매 제한(advance purchase restriction)들이 있을 수 있다.

[0058] ● 처리 시간([표 1]과 관련하여 상술되어진 기준에 기초함)면에서 낮게 등급화되며, 가용성의 명확하게 정의된 요소들과 연관된 비지니스 로직.

[0059] 전반적인 목표는 사전산출 가용성 데이터베이스(151)와 가용성 규칙 계산기 엔진(152) 로직간의 양호한 균형을 발전시키는 것이다. 이 균형은 높은 성능을 유지하기 위해 동적으로 조정될 수 있다. 새로운 로직이 가용성 모델에 추가되는 경우, 사전산출 가용성 데이터베이스(151)와 가용성 규칙 계산기 엔진(152) 로직 간의 새로운 균형(즉, 스위트 스폿(sweet spot))을 생성하는, 일부 가용성 요소들 내에서의 점수를 수정할 수 있다.

[0060] 가용성 규칙 계산기(Availability Rule Calculator engine, 152)는 가용성 프로세스의 "온-더-고(on-the-go)" 로직을 처리한다. 비지니스 규칙은 다른 구성요소(즉, 가용성 계산기)들로 분리될 수 있기 때문에, 그것들은 독립적으로 구현될 수 있다. 각 리퀘스트에 실시간 적용될 수 있는 여러 펜스(fence) 및 제한 사항(restriction)이 있을 수 있다. 예를 들어, 이것들은 호텔들이 가용성을 제어하기 위해 설정할 수 있는 속성(attribute)들을 나타낼 수 있다(예를 들어, 사전 구매, 체류 밤 수, 최소/최대 체류, 특별 요구 사항들). 다른 펜스들 및 제한

사항들은 오직 호텔 레벨, 요금 카테고리 레벨, 및 요금 코드 레벨에서 확인될 수도 있다. 비지니스 규칙들이 어떻게 적용되는지에 대한 순서는 처리 시간 사이클(processing cycle)에 직접적인 영향을 미칠 수 있다.

[0062] 가용성 규칙 계산기 엔진(152)은 두 가지 주요 구성 요소를 가질 수 있다:

[0063] ● **제품 항목 콜렉터(Product item collector)**. 가용성 프로세스의 조정자(orchestrator). 이것은 규칙 및 그 규칙이 적용될 순서를 특정하기 위해 콜렉션옵션들(CollectionOptions)을 사용한다.

[0064] ● **가용성 규칙 계산기들**. 각 계산기 (도 5에 관련하여 아래에서 보다 상세하게 설명됨)는 마스크계산기(MaskCalculator,(507)) 인터페이스를 구현하고, 콜렉터가 호출하게 하는 getMask() 메소드를 제공해야 한다. 반환되는 마스크(mask)는 규칙을 적용하기 위해 현재 ProductItems LOS 마스크에 대해, 다른 마스크들과 AND된다. 또한, 그것들은 getReasonCode()를 구현함으로써, 이유 코드(reson code)를 공급해야 한다.

[0065] 일 실시예에서는, 아래에서 보다 상세하게 설명되는 바와 같이, 가용성 규칙 계산기 엔진(152)이 구축되고 운용될 수 있다.

[0067] **클래스 모델(Class Model)**

[0068] 내부적으로, 콜렉터(502)는 각각이 체류 기간(LOS) 규칙의 특정 클래스에 대해 전용된, 여러 "계산기" 구성 요소(아래에서 보다 상세하게 설명되는, 도 6의 507 및 601-606)와 협력(collaborate)한다. 일반적으로, 각 계산기는 전체(overall) LOS를 도출하기 위해 ProductItem LOS 마스크와 함께 AND되는 비트 마스크(bit mask)를 반환한다.

[0070] **콜렉션 콘텍스트(Collection Context)**

[0071] CollectionContext 오브젝트(503)가 수집 주기(collection cycle)에 대하여 글로벌 상태(global state)를 유지하기 위해 사용된다. 호출자가 콜렉터(502)상에서 collect()를 인보크(invoke)하면(ProductItemCollector 인터페이스(504)를 통해), 콜렉터(502)는 CollectionContext 오브젝트(503)를 생성할 것이다. CollectionContext(503)은 후술하는 특성들을 가질 수 있다.

[0072] · 호텔 코드

[0073] · 요금 카테고리 코드(Rate Category Codes)

[0074] · 날짜

[0075] · 데이터베이스 오브젝트

[0076] · 수집옵션(CollectionOptions)

[0077] · 캐시된 팩트(Cached Facts)

[0078] CollectionContext(503)는 각 계산기의 생성자(constructor)에게 전달된다. 이 생성자는 각 계산기의 환경을 초기화하는, 클래스의 소프트웨어 구성요소이다. 계산기들은 데이터베이스(203)를 얻어 CRS 테이블을 쿼리하기 위해 CollectionContext(503)을 사용한다. 또한 이 계산기들은 콘텍스트 오브젝트를 통해 서로 팩트들(facts)을 공유하기 위해 그 데이터베이스를 사용한다. 예를 들어, 호텔이 HIRO 호텔로 찾아지면, 그 팩트는 하나 이상의 계산기에 대하여 중요할 수 있다.

[0080] **계산기(Calculator)들**

[0081] 계산기들(601-606)은 PACE 엔진을 위한 키 비지니스 로직 구성요소들이다. CollectionStrategy(505)는 CollectionOptions(506)에서 특정된 내용에 기초하여 어떤 계산기가 사용될지를 결정한다. 각 계산기는 MaskCalculator(507) 인터페이스를 구현하고, 콜렉터(502)가 호출할 getMask() 메소드를 제공할 수 있다. 반환된 마스크는 규칙을 적용하기 위해 현재 ProductItems LOS 마스크에 대해, 다른 마스크들과 AND된다. 또한, 이유 코드도 예를 들어, getReasonCode()를 구현함으로써 공급될 수 있다.

[0082] MaskCalculator(507)를 위한 예시적 인터페이스는 아래와 같을 수 있다.

[0083]
```
public interface MaskCalculator {
```

[0084]
```
        int getMask(ProductItem pi) throws ServiceException;
```

[0085]
```
        int getReasonCode();
```

[0086] }

[0087] 임의 개수의 MaskCalculators(507)가 있을 수 있다. 도 6은 현재 설정(current set)을 위한 UML를 도시한다. 일 실시예에서, 계산기들은 IHG 가용성 모델의 매우 구체적인 펜스(fence) 또는 제한 규칙들이 있을 수 있다. 예를 들어:

[0088] ● 도착 계산기(Arrival Calculator, 601) - 이것은 블랙아웃 일자(blackout date) 또는 입실 금지(closed to arrival)가 호텔 또는 요금 카테고리 레벨에 설정되었지에 대한 유효성 검사를 위해 리퀘스트의 체크-인 날짜를 사용한다.

[0089] ● 최소최대 계산기(MinMax Calculator, 602) - 이것은 LOS를 유도하기 위해 체크-인 날짜와 체크-아웃 날짜를 사용한다. LOS 값은 만약 체류 제한이 존재한다면, 체류 제한의 최소 및 최대 일자를 충족해야만 한다.

[0090] ● 수익 계산기(Revenue Calculator, 603) - 이것은 특정 요금 총액에 대한 LOS가 흑자 수익(positive revenue)을 냈는지를 검증한다.

[0091] ● 객실SS 계산기(RoomSS Calculator, 604) - 이것은 특정 객실 유형이 호텔의 객실 판매 전략(room sell strategy)과 연관되는지를 검증한다. 호텔 객실 판매 전략은 판매용으로 확인가능한 객실을 식별한다.

[0092] ● 체류일자 계산기(StayoverDays Calculator, 605) - 이것은 LOS를 유도하기 위해 체크-인 날짜와 체크-아웃 날짜를 사용한다. LOS 값은 체류가 요구되는 요일(days of the week)을 충족해야만 한다.

[0093] ● 액티브 일수 계산기(Active Days Calculator, 606) - 이것은 LOS를 유도하기 위해 체크-인 날짜와 체크-아웃 날짜를 사용한다. LOS 값은 액티브가 플래그되는 요일을 충족해야만 한다.

[0095] **호텔 규칙 캐시(Hotel Rule Cache)**

[0096] 가능한 최대한 효율적인 PACE 엔진을 갖는 것은 장점이 있다. 도 5 및 도 6을 참조하면, CRS 데이터베이스에 인코딩된 호텔 규칙들을 포함하기 때문에, 이것의 상당 부분은 MaskCalculators(507)의 구현예에 속할 수 있다. 매번, PACE 엔진은 새로운 MaskCalculator(507)를 생성하고, 계산기(507)는 데이터베이스(203)로부터 필요한 데이터를 한 번만 사전 패치(prefectch)한 후, 자신의 getMask() 메소드의 각 호출(invocation)에 대해 사전 패치한 데이터를 반복적으로 사용함으로써 스스로 최적화할 수 있다. 그럼에도 불구하고, 거의 변화하지 않는 규칙들에 대하여 모든 계산기들이 데이터베이스에 쿼리하도록 하는 것은 성능상 상당한 타격이기 때문에, 회피되어야만 한다.

[0097] 이를 해결하기 위하여, CollectionContext 오브젝트(503)(도 5)는 MaskCalculators(507)를 대신해서 규칙들을 캐시하기 위한 지원을 포함할 수 있다. MaskCalculator(507)이 데이터베이스(203)로부터 자신의 규칙들을 구축하였으면, 그 규칙들을 CollectionContext(503)상에 캐싱할 수 있다. 결과적으로, 이것은 모든 MaskCalculators(507)가 데이터베이스를 히트하여 규칙들을 재구축하기 전에, 그들이 필요한 규칙들이 여전히 존재하는지를 확인하기 위해 체크할 수 있다는 것을 의미한다. 이 모델은 성능상의 증가에 기여하고, 그로 인해 가능할 때마다 사용될 수 있다.

[0098] 여기에, 규칙들을 캐싱하기 위한 것으로 보일 수 있는 메소드들이 개시된다.

[0099]
```
public class CollectionContext {
```

[0100]
```
        public Object getHotelRules(MaskCalculator c);
```

[0101]
```
        public void putHotelRules(MaskCalculator c, Object rules);
```

[0102]
```
}
```

[0103] MaskCalculator(507)가 규칙들을 캐시하면, 그 규칙들은 후술하는 이벤트 중 하나가 발생할 때까지 메모리에 남아있을 수 있다.

[0104] ● 호텔 규칙들이 데이터베이스(203)에서 변경하는 것

[0105] ● LRU 캐시가 리소스 관리를 위한 호텔 규칙들을 제거(purge)하는 것

[0106] ● 애플리케이션이 다시 시작하는 것

[0107] 일 실시예에서, MaskCalculators(507)는 캐시 메소드를 사용하는데 최선을 다할 수 있고, 그것들은 신뢰할 수

있다고 가정한다. 만약 캐시-변경 관리를 위한 적절한 솔루션이 사용가능하지 않다면, PACE 엔진은 그 캐시가 활성화되지 않은 것을 보증할 수 있다. 그럼에도 불구하고, MaskCalculators(507)는 항상 캐시가 활성화 상태라고 가정할 수 있다.

[0109] ● **콜렉션 전략들(Collection Strategies)**

[0110] PACE 엔진은 다른 목적으로 이용될 수 있기 때문에, 모든 경우를 위한 단일 검색 알고리즘에 적합하지 않다. 예를 들어, 풀 애플리케이션(Pull application)으로부터의 가용성 리퀘스트들은 먼저 상위 레벨 제한들을 살펴보고, 마지막으로 제품 아이템들을 살펴볼 수 있는 반면, 푸쉬(Push) 애플리케이션은 먼저 제품 아이템들을 살펴보고, 가능한 가장 효율적인 순서로 규칙들을 적용하기를 원할 수 있다.

[0111] 이를 해결하기 위해, PACE 엔진은 콜렉션을 실제로 어떻게 수행할지를 나타내기 위해 CollectionStrategy(505) 인터페이스를 사용할 수 있다. 이것은 기본적으로 이 기술분야에 종사하는 사람들에게 알려진 추출 알고리즘(abstract algorithm) 디자인 패턴의 구현예이다. 전략의 선택은 CollectionOptions(506)에 있는 세팅(setting)들에 기초할 수 있다.

[0112] CollectionStrategy(505)를 위한 예시적 인터페이스가 아래에 제공된다.

[0113]
```
public interface CollectionStratege {
```

[0114]
```
        void collect(CollectionContext ctx,
```

[0115]
```
                ProductItemConsumer consumer) throws ServiceException;
```

[0116]
```
}
```

[0117] 일 실시예에서, 지원된 CollectionStrategies(505)은 아래 요소들을 포함할 수 있다:

표 2

[0118]

클래스	설명
BaseStrategy	기본 산출 템플릿 메소드로 클래스를 추출
DefaultStrategy	요금-카테고리 코드들에 대한 SQL 기준으로 BaseStrategy를 확장
ProductItemIDStrategy	제품 아이템 ID에 대한 SQL 기준으로 BaseStrategy를 확장

[0119] ● **쿼리들**

[0120] PACE 엔진이 인보크(invoke)되었을 때, PACE 엔진은 궁극적으로 제품 아이템 테이블로부터 제품 아이템들을 독출하여, 최종 LOS를 도출하기 위해 MaskCalculators(507)를 적용할 수 있다. 제품 아이템들을 선택하기 위해 사용되는 쿼리는 복수의 인자들을 고려할 수 있다. 그 인자들은 아래 요소들을 포함한다.

[0121] · 선택 기준(Selection Criteria)

[0122] · LOS 히스토리(History)

[0123] · 판매량 제외분(sell-through exclusions)

[0125] 상기 선택들이 쿼리에 대한 이러한 양태들을 보다 자세하게 후술한다.

[0126] · 기본 쿼리(Basic Query)

[0127] · LOS 히스토리

[0128] · 판매량 제외분(sell-through exclusions)

[0129] (1) 기본 쿼리

[0130] 기본 쿼리는 제품 아이템들을 선택하는 SQL의 가장 간단한 형식을 나타낸다. 실제로는, 이것이 사용된 실제 SQL이 아닐 수도 있다. 그러나, 이것이 다른 완전하게 형성된 쿼리들에 대한 베이스라인으로서 쓰일 수 있기 때문에, 이것이 개시되어 있다. 다시 말하면, 모든 쿼리들을 이해하기 위해, 기본 쿼리에 대한 이해가 중요하다.

[0131] 기본 쿼리를 위한 기준은 후술하는 것을 포함한다:

[0132] · 호텔 코드

[0133] · 날짜 범위

[0134] · 요금 카테고리 코드들

[0135] · 유효한 상태(Valid Status)

[0136] · 변경 토큰(선택 사항)

[0137] · 제외된 객실 코드(선택 사항)

[0138] · 제품 아이템 ID들(선택 사항)

[0139] "유효한 상태"란 단지 무효한(invalid) 제품 아이템 행(row)들을 건너뛰는 것을 의미한다. 변경 토큰은 특정된 값 이상의 토큰 값을 갖는 행들을 선택하기 위해 사용될 수 있다. 제외된 객실 유형들은 제외된 객실들을 걸러내기 위해 사용된다. 마지막으로, 제품 아이템 ID들의 목록은 제품 아이템들의 보다 정확한 선택을 위하여 변경 토큰을 대신하여 포함될 수 있다.

[0140] 토큰 값과 제품 아이템 ID들은 사전산출(PreComputer)에 의해 오직 실제로 수정된 이러한 제품 아이템들만을 필터링해내는데 도움을 주기 위해 사용될 수 있다. 사전산출을 위해 언제 및 왜 제품 아이템이 변경되었는지를 최적으로 표시하기 위해 사용될 수 있는 다양한 방법들이 있으나, 일 실시예에서는 토큰 번호에 제품 아이템 ID를 부가하는 것이 사용될 수도 있다.

[0141] 제품 아이템들이 ProductItemConsumer에 배달되는 순서가 두 가지 이유에서 중요하다. 첫째, 계산기들(507)이 콘텍스트 스위칭에 대하여 효율적이게 한다. 둘째, ProductItemConsumer들이 그들이 구축하는 결과를 쉽게 체계화(organize)할 수 있게 한다.

[0142] 이를 위하여, 결과가 아래와 같이 순서화될 수 있다.

[0143] · 날짜

[0144] · 요금 카테고리 코드

[0145] · 객실 코드

[0147] 상술되어진 기술에 기초하여, 도 7을 참조하면, 일 실시예에서, 본 발명은

[0148] ● 복수의 체류 기간(plurality of length of stays)에 대한 호텔 객실에 연관된 실시간 가용성 속성(real-time availability attribute)들을 포함하는 사전-산출 데이터베이스(pre-compute database)를 만드는 단계(단계 701);

[0149] ● 시작 날짜와 종료 날짜를 갖는 날짜 범위(date range)에 대하여 적어도 하나의 호텔에 대한 쿼리를 수신하는 단계(단계 702);

[0150] ● 상기 쿼리에 따른 비지니스 요구 사항(business requirement)들을 적용함으로써, 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대한 가용성을 계산하는 단계(단계 703);

[0151] ● 상기 사전-산출 데이터베이스로부터의 가용성과 쿼리 속성들을 조합함으로써, 상기 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대해서 최종 가용성을 생성하는 단계; 및

[0152] ● 상기 복수의 체류 기간에 해당하는 상기 날짜 범위 내의 각 날짜에 대한 상기 적어도 하나의 호텔에 대하여, 호텔 객실들의 최종 가용성을 포맷하는 단계(단계 705)에 해당하는 단계들을 수행할 수 있다.

[0154] 방법 및 시스템들이 바람직한 실시예들 및 특정 예시들과 연결되어 설명되었으나, 본 명세서의 실시예들은 제한적이라기 보다는 오히려 모든 관점에서 예시적이 되도록 의도한 것이기 때문에, 이는 본 발명의 범위(scope)가 개시된 특정 실시예에 한정되는 것을 의도한 것은 아니다.

[0155] 별도로 분명하게 명시된 것이 없으면, 임의의 방법이 본 명세서에서 그것의 단계들이 특정 순서로 수행되는 것을 요구하는 것처럼 구성되는 것을 개시하는 것으로 의도되지 않는다. 따라서, 방법 청구항은 자신의 단계 이후에 행해져야 하는 순서를 실제로 나열하지 않으며, 특정 순서들로 제한되어야 하는 단계들에 대한 설명들 또는 청구항들에서 특별히 다르게 언급되지 않는 경우, 어떤 면에서도, 추론되는 순서를 의도하지 않는다. 이는 해석에 있어서, 단계들의 나열 또는 동작의 흐름과 관련한 논리의 문제들; 문법적 체계(grammatical organization)

또는 구두법(punctuation)으로부터 도출되는 자명한 의미(plain meaning); 본 명세서에서 설명된 실시예들의 수(數) 또는 유형을 포함하는 임의의 가능한 비-표현 근거(non-express basis)를 수용한다.

[0156] 본 발명의 범위 또는 정신(spirit)으로부터 벗어나지 않고도, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자들에게, 본 발명의 다양한 변경들 및 변형들이 만들어질 수 있다는 것은 명백하다 할 것이다. 이 분야에서 숙달된 기술을 가진 이들에게는 여기에서 개시된 상세한 설명의 고찰 및 실행으로부터의 다른 실시예들이 명백할 것이다. 상세한 설명 및 예시들은 단지 예시로서 고려될 수 있을 뿐이며, 본 발명의 진정한 범위 및 정신은 후술될 청구범위에 의해 지시되는 것에 있음을 의도한 것이다.

부호의 설명

[0157] 101: HDX 서버

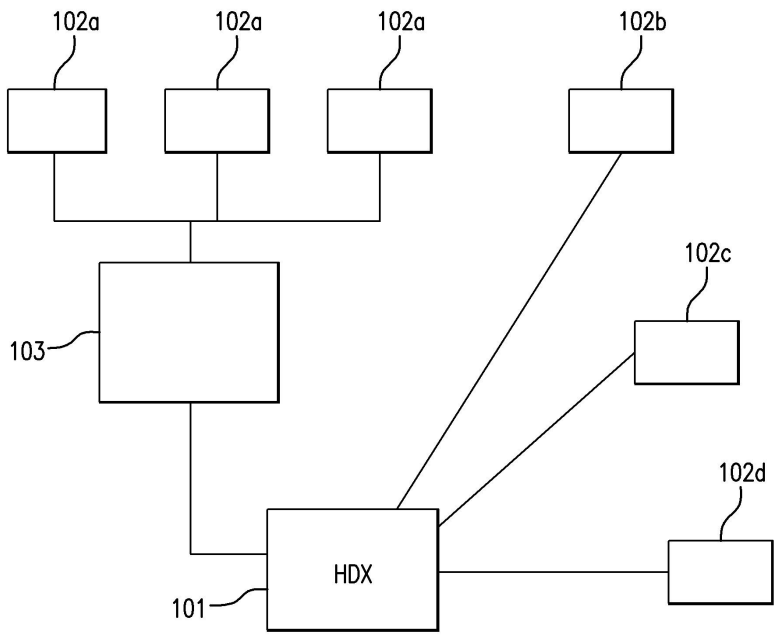
102a, 120b, 120c: 클라이언트

102d: 호텔

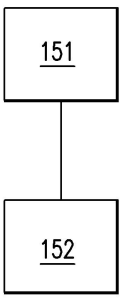
103: 중개 데이터 센터

도면

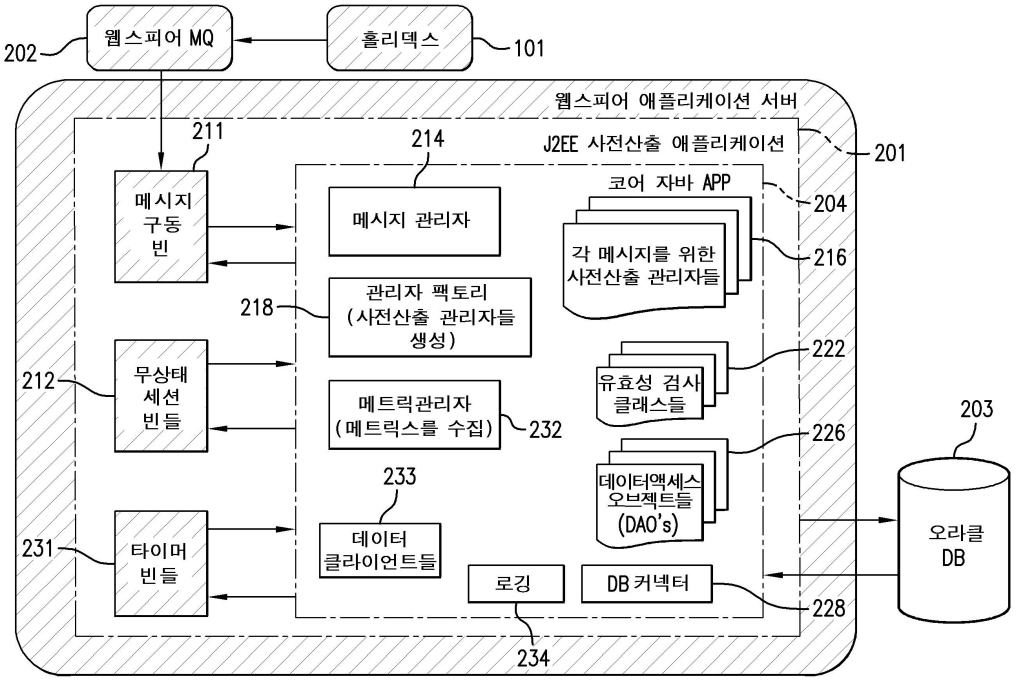
도면1a



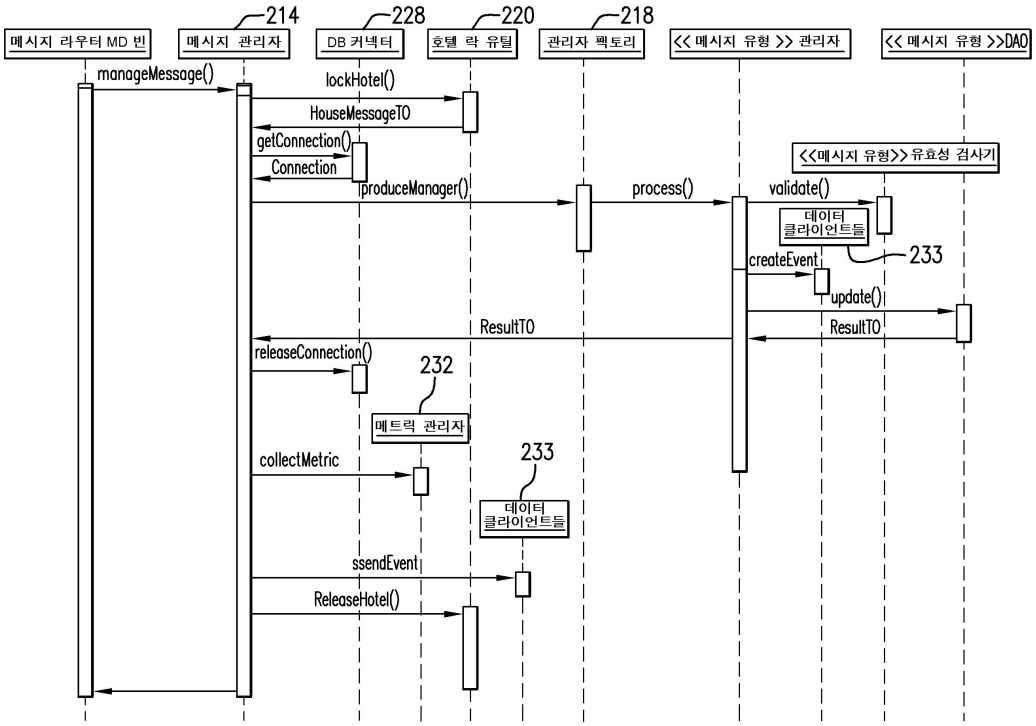
도면1b



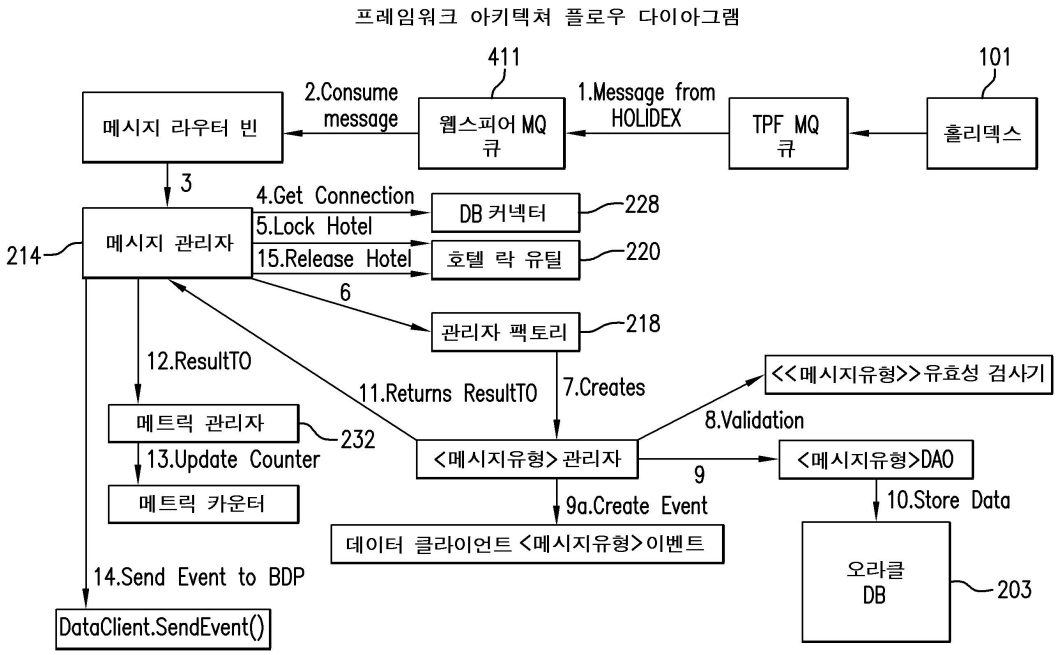
도면2


웹스피어 MQ
홀리덱스
202
101
웹스피어 애플리케이션 서버
J2EE 사전산출 애플리케이션
201
211
214
메시지 구동 빈
메시지 관리자
코어 자바 APP
204
각 메시지를 위한 사전산출 관리자들
216
218
관리자 팩토리 (사전산출 관리자들 생성)
212
무상태 세션 빈들
메트릭관리자 (메트릭스를 수집)
232
222
유효성 검사 클래스들
226
데이터액세스 오브젝트들 (DAO's)
203
233
데이터 클라이언트들
231
타이머 빈들
로깅
DB 커넥터
228
오라클 DB
234

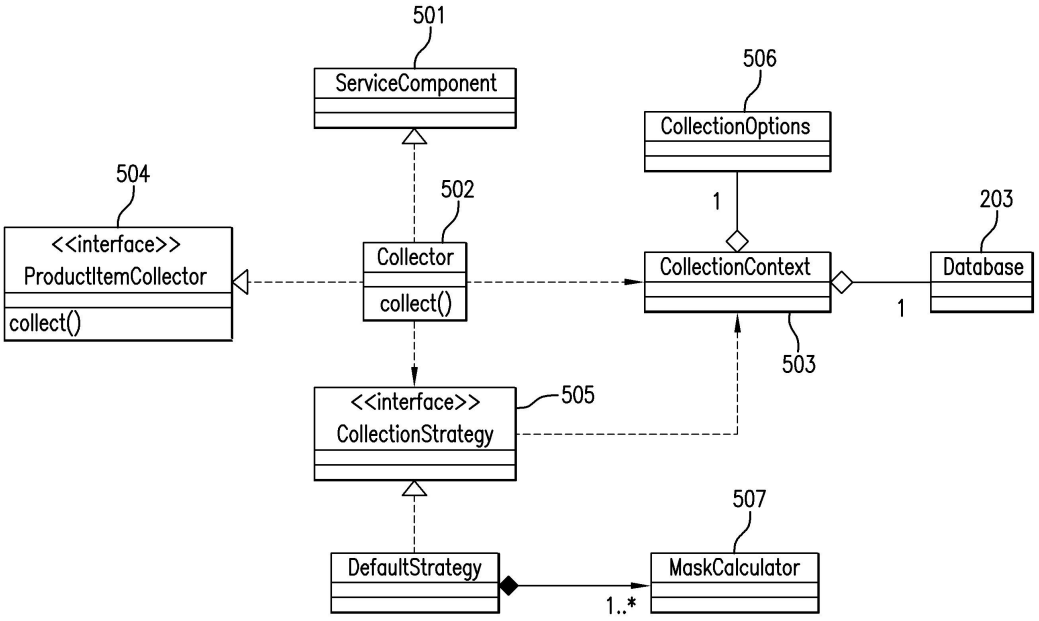
도면3


메시지 라우터 MD 빈
메시지 관리자
214
DB 커넥터
228
호텔 락 유틸
220
관리자 팩토리
218
<< 메시지 유형 >> 관리자
<< 메시지 유형 >>DAO
manageMessage()
lockHotel()
HouseMessageTO
getConnection()
Connection
produceManager()
process()
<<메시지 유형>> 유효성 검사기
validate()
데이터 클라이언트들
233
createEvent
update()
ResultTO
ResultTO
releaseConnection()
232
메트릭 관리자
collectMetric
233
데이터 클라이언트들
ssendEvent
ReleaseHotel()

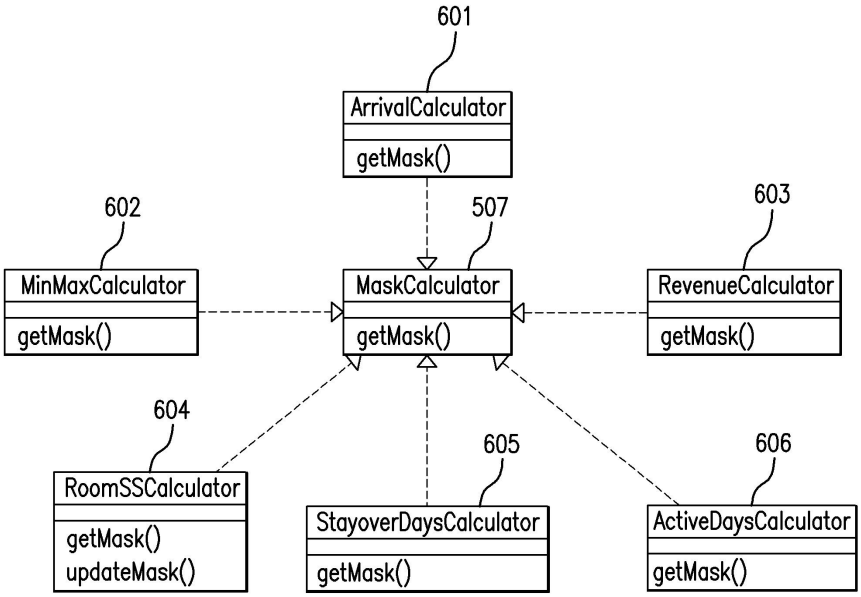
도면4


프레임워크 아키텍쳐 플로우 다이아그램
411
101
메시지 라우터 빈
2.Consume message
웹스피어 MQ 큐
1.Message from HOLIDEX
TPF MQ 큐
홀리덱스
3
4.Get Connection
5.Lock Hotel
15.Release Hotel
DB 커넥터
228
214
메시지 관리자
호텔 락 유틸
220
6
관리자 팩토리
218
12.ResultTO
11.Returns ResultTO
7.Creates
<<메시지유형>>유효성 검사기
8.Validation
메트릭 관리자
232
13.Update Counter
<메시지유형> 관리자
9
<메시지유형>DAO
메트릭 카운터
9a.Create Event
10.Store Data
데이터 클라이언트 <메시지유형>이벤트
오라클 DB
203
14.Send Event to BDP
DataClient.SendEvent()

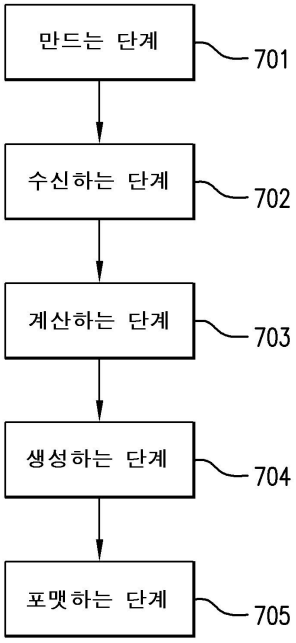
도면5


501
ServiceComponent
506
CollectionOptions
504
502
1
203
<<interface>>
ProductItemCollector
collect()
Collector
collect()
CollectionContext
Database
1
503
<<interface>>
CollectionStrategy
505
507
DefaultStrategy
MaskCalculator
1..*

도면6


601
ArrivalCalculator
getMask()
602
MinMaxCalculator
getMask()
507
MaskCalculator
getMask()
603
RevenueCalculator
getMask()
604
RoomSSCalculator
getMask()
updateMask()
605
StayoverDaysCalculator
getMask()
606
ActiveDaysCalculator
getMask()

도면7


만드는 단계
701
수신하는 단계
702
계산하는 단계
703
생성하는 단계
704
포맷하는 단계
705