(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109063153 A
(43)申请公布日 2018.12.21

(21)申请号 201810900713.1

(22)申请日 2011.03.03

(30)优先权数据
12/718,446 2010.03.05 US

(62)分案原申请数据
201180022331.X 2011.03.03

(71)申请人 六洲饭店有限公司
地址 美国乔治亚州

(72)发明人 D·冈萨雷斯 T·昂 D·Y·谭
J·P·麦克卢尔 S·马特奥
R·拉贾拉蒂楠

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 周全 钱慰民

(51)Int.Cl.
G06F 17/30(2006.01)
G06Q 10/02(2012.01)
G06Q 50/12(2012.01)

权利要求书5页 说明书11页 附图7页

(54)发明名称

自底向上优化的搜索系统和方法

(57)摘要

提供了自底向上优化的搜索系统和方法。可生成预计算数据库，所述预计算数据库包括与在多个入住时长上的酒店房间相关联的实时可用属性，并且可接收针对一日期范围对至少一家酒店的查询。可通过根据该查询应用商业需求来计算在多个入住时长上的日期范围内的每一天的可用性。针对在多个入住时长上的日期范围内的每一天，可通过组合来自预计算数据库的可用性和查询属性来创建最终可用性。针对在多个入住时长上的日期范围内的每一天，可格式化至少一家酒店的酒店房间的最终可用性。

CN 109063153 A

1.一种用于通过在预计算数据和实时活动数据之间进行平衡来改进数据源的搜索的计算机实现的方法，所述方法包括以下步骤：

由计算机系统将可用性查询过程的多个步骤分类为多个类别，其中所述可用性查询过程处理查询以便搜索所述数据源并输出指示酒店房间的可用性的最终结果；

由所述计算机系统将多个得分关联到所述多个类别，其中每个得分基于数据库调用量、发现逻辑和处理时间；

在接收到所述查询之前，由所述计算机系统基于所述多个步骤的一个步骤的相应得分执行所述步骤，以便生成用于预计算数据库的数据；

由所述计算机系统基于执行所述步骤来更新所述预计算数据库，所述预计算数据库的数据包括与多个入住时长上的酒店房间相关联的实时可用性属性；

由所述计算机系统接收对所述预计算数据库的所述查询，所述查询包括针对一日期范围对至少一家酒店的请求，所述日期范围具有起始日期和结束日期；

由所述计算机系统基于所述查询确定来自所述预计算数据库的第一搜索结果，所述第一搜索结果包括与所述多个入住时长中至少一个入住时长相关联的第一可用性；

由所述计算机系统将逻辑添加到所述可用性查询过程；

由所述计算机系统基于所添加的逻辑更新与所述多个类别相关联的得分；

由所述计算机系统基于所更新的得分确定所述可用性查询过程的所述多个步骤中的一个或多个其他步骤将在不同于所述预计算数据库的另一数据源上执行；

由所述计算机系统执行所确定的一个或多个其他步骤，所述一个或多个其他步骤包括：

基于所述查询和应用需求来计算来自所述其他数据源的第二搜索结果，

所述第二搜索结果包括多个入住时长上的所述日期范围内的每一天的可用性，其中所述需求包括一个或多个基于酒店的需求和与处理时间周期相关联的一个或多个基于处理时间的需求，其中应用所述一个或多个基于酒店的需求的次序影响所述处理时间周期；

由所述计算机系统通过组合所述第一搜索结果和所述第二搜索结果来创建所述最终结果，所述最终结果包括针对所述多个入住时长上的所述日期范围内的每一天的最终可用性；以及

由所述计算机系统格式化所述最终结果，所述格式化包括针对所述多个入住时长上的所述日期范围内的每一天，格式化所述至少一家酒店的酒店房间的所述最终可用性。

2.如权利要求1所述的计算机实现的方法，其特征在于，创建所述最终结果包括使用位掩码来表示入住时长，每一位表示所述入住时长中的一天。

3.如权利要求1所述的计算机实现的方法，其特征在于，从旅行网站接收所述查询。

4.如权利要求1所述的计算机实现的方法，其特征在于，从客户机接收所述查询。

5.如权利要求1所述的计算机实现的方法，其特征在于，所述计算步骤用障碍规则执行。

6.如权利要求1所述的计算机实现的方法，其特征在于，所述计算步骤用限制规则执行。

7.如权利要求1所述的计算机实现的方法，其特征在于，还包括将酒店房间的所述最终可用性发送到旅行网站的步骤。

8.如权利要求1所述的计算机实现的方法，其特征在于，还包括将酒店房间的所述最终可用性发送到客户机的步骤。

9.如权利要求1所述的计算机实现的方法，其特征在于，针对所述日期范围内的每一天创建最终可用性包括组合与来自所述预计算数据库的所述第一可用性相关联的预计算数据和与来自所述第二搜索结果的所述可用性相关联的动态计算数据。

10.如权利要求9所述的计算机实现的方法，其特征在于，还包括调整所述预计算数据和所述动态计算数据之间的比率。

11.如权利要求10所述的计算机实现的方法，其特征在于，调整所述预计算数据和所述动态计算数据之间的比率至少部分地基于所述一个或多个基于处理时间的需求中的至少一个的处理时间。

12.一种用于通过在预计算数据和实时活动数据之间进行平衡来改进数据源的搜索的系统，所述系统包括：

一个或多个处理器；

存储器，包括可执行指令，当由所述一个或多个处理器执行所述可执行指令时使得所述系统至少：

将可用性查询过程的多个步骤分类为多个类别，其中所述可用性查询过程处理查询以便搜索所述数据源并输出指示酒店房间的可用性的最终结果；

将多个得分关联到所述多个类别，其中每个得分基于数据库调用量、发现逻辑和处理时间；

基于所述多个步骤的一个步骤的相应得分执行所述步骤，以便生成用于预计算数据库的数据；

基于执行所述步骤来更新所述预计算数据库，所述预计算数据库的数据包括与多个入住时长上的酒店房间相关联的实时可用性属性；

接收对所述预计算数据库的所述查询，所述查询包括针对一日期范围对至少一家酒店的请求，所述日期范围具有起始日期和结束日期；

基于所述查询确定来自所述预计算数据库的第一搜索结果，所述第一搜索结果包括与所述多个入住时长中至少一个入住时长相关联的第一可用性；

向所述可用性查询过程添加逻辑；

基于所添加的逻辑更新与所述多个类别相关联的所述得分；

基于所更新的得分确定所述可用性查询过程的所述多个步骤中的一个或多个其他步骤将在不同于所述预计算数据库的另一数据源上执行；

执行所确定的一个或多个其他步骤，所述一个或多个其他步骤包括：

基于所述查询和应用需求来计算来自所述其他数据源的第二搜索结果，

所述第二搜索结果包括多个入住时长上的所述日期范围内的每一天的可用性，其中所述需求包括一个或多个基于酒店的需求和与处理时间周期相关联的一个或多个基于处理时间的需求，其中应用所述一个或多个基于酒店的需求的次序影响所述处理时间周期；

通过组合所述第一搜索结果和所述第二搜索结果来创建所述最终结果，所述最终结果包括针对所述多个入住时长上的所述日期范围内的每一天的最终可用性；以及

格式化所述最终结果，所述格式化包括针对所述多个入住时长上的所述日期范围内的

每一天，格式化所述至少一家酒店的酒店房间的所述最终可用性。

13.如权利要求12所述的系统，其特征在于，更新所述预计算数据库包括使用位掩码来表示入住时长，每一位表示所述入住时长中的一天。

14.如权利要求12所述的系统，其特征在于，所述查询来自旅行网站。

15.如权利要求12所述的系统，其特征在于，所述查询来自客户机。

16.如权利要求12所述的系统，其特征在于，计算所述第二搜索结果包括使用障碍规则。

17.如权利要求12所述的系统，其特征在于，计算所述第二搜索结果包括使用限制规则。

18.如权利要求12所述的系统，其特征在于，所述指令还使得所述一个或多个处理器将酒店房间的所述最终可用性共同发送到旅行网站。

19.如权利要求12所述的系统，其特征在于，所述指令还使得所述一个或多个处理器将酒店房间的所述最终可用性共同发送到客户机。

20.一种包括非瞬态计算机可读介质的计算机程序产品，其中所述非瞬态计算机可读介质中存储有计算机可读程序代码，所述计算机可读程序代码适于被执行以便实现用于通过在预计算数据和实时活动数据之间进行平衡来改进数据源的搜索的方法，所述方法包括：

将可用性查询过程的多个步骤分类为多个类别，其中所述可用性查询过程处理查询以便搜索所述数据源并输出指示酒店房间的可用性的最终结果；

将多个得分关联到所述多个类别，其中每个得分基于数据库调用量、发现逻辑和处理时间；

基于所述多个步骤的一个步骤的相应得分执行所述步骤，以便生成用于预计算数据库的数据；

基于执行所述步骤来更新所述预计算数据库，所述预计算数据库的数据包括与多个入住时长上的酒店房间相关联的实时可用性属性；

接收对所述预计算数据库的所述查询，所述查询包括针对一日期范围对至少一家酒店的请求，所述日期范围具有起始日期和结束日期；

基于所述查询确定来自所述预计算数据库的第一搜索结果，所述第一搜索结果包括与所述多个入住时长中至少一个入住时长相关联的第一可用性；

向所述可用性查询过程添加逻辑；

基于所添加的逻辑更新与所述多个类别相关联的所述得分；

基于所更新的得分确定所述可用性查询过程的所述多个步骤中的一个或多个其他步骤将在不同于所述预计算数据库的另一数据源上执行；

执行所确定的一个或多个其他步骤，所述一个或多个其他步骤包括：

基于所述查询和应用需求来计算来自所述其他数据源的第二搜索结果，

所述第二搜索结果包括多个入住时长上的所述日期范围内的每一天的可用性，其中所述需求包括一个或多个基于酒店的需求和与处理时间周期相关联的一个或多个基于处理时间的需求，其中应用所述一个或多个基于酒店的需求的次序影响所述处理时间周期；

通过组合所述第一搜索结果和所述第二搜索结果来创建所述最终结果，所述最终结果

包括针对所述多个入住时长上的所述日期范围内的每一天的最终可用性;以及

格式化所述最终结果,所述格式化包括针对所述多个入住时长上的所述日期范围内的每一天,格式化所述至少一家酒店的酒店房间的所述最终可用性。

21.一种用于对数据库搜索酒店房间的可用性的计算机实现的方法,所述方法包括以下步骤:

由计算机系统生成预计算可用性关系数据库,其中所述预计算可用性关系数据库包括与多个入住时长上的酒店房间相关联的实时可用性属性;

由所述计算机系统从用户装置接收针对一日期范围对至少一家酒店的查询,所述日期范围具有起始日期和结束日期;

由所述计算机系统从所述预计算可用性关系数据库中确定与所述多个入住时长中至少一个入住时长相关联的酒店房间的可用性;

由所述计算机系统的可用性规则计算器引擎对所述酒店房间的所述可用性应用商业逻辑,其中应用所述商业逻辑包括应用商业规则,所述商业规则包括酒店能够设置的用来控制可用性的属性,并且如何应用所述商业规则的次序直接影响处理时间周期;

由所述计算机系统通过基于应用所述可用性规则计算器引擎的所述商业逻辑来限制来自所述预计算可用性关系数据库的所述可用性,从而创建针对所述多个入住时长上的所述日期范围内的每一天的最终可用性;以及

由所述计算机系统针对所述多个入住时长上的所述日期范围内的每一天,格式化所述至少一家酒店的酒店房间的所述最终可用性。

22.如权利要求21所述的方法,其特征在于,还包括:

由所述计算机系统基于所述最终可用性向所述用户装置提供对所述查询的响应,其中所述响应标识出对于所述日期范围内的日期有酒店房间可用,并且针对所述日期的所述酒店房间的所述可用性包括在基于所述可用性的所述限制的酒店房间的所述最终可用性中。

23.如权利要求21所述的方法,其特征在于,所述入住时长由位掩码表示,其中每一位表示所述入住时长中的一天。

24.如权利要求21所述的方法,其特征在于,从旅行网站接收所述查询。

25.如权利要求21所述的方法,其特征在于,从客户机接收所述查询。

26.如权利要求21所述的方法,其特征在于,用于应用商业规则的步骤包括应用障碍规则。

27.如权利要求21所述的方法,其特征在于,用于应用商业规则的步骤包括应用限制规则。

28.如权利要求21所述的方法,其特征在于,还包括将酒店房间的所述最终可用性发送到旅行网站的步骤。

29.如权利要求21所述的方法,其特征在于,还包括将酒店房间的所述最终可用性发送到客户机的步骤。

30.如权利要求21所述的方法,其特征在于,用于应用商业规则的步骤包括:

由所述可用性规则计算器引擎的产品项目收集器指定所述商业规则以及如何应用所述商业规则的次序;和

由所述可用性规则计算器引擎的多个可用性规则计算器中的每一个实现掩码计算器

界面并且提供所述产品项目收集器将调用的掩码获取方法。

31.一种用于对数据库搜索酒店房间的可用性的计算机实现的系统，所述系统包括：

用于生成预计算可用性关系数据库的装置，其中所述预计算可用性关系数据库包括与多个入住时长上的酒店房间相关联的实时可用性属性；

用于从用户装置接收针对一日期范围对至少一家酒店的查询的装置，其中所述日期范围具有起始日期和结束日期；

用于从所述预计算可用性关系数据库中确定与所述多个入住时长中至少一个入住时长相关联的酒店房间的可用性的装置；

用于对所述酒店房间的所述可用性应用商业逻辑的装置，其中用于应用所述商业逻辑的装置包括用于应用商业规则的装置，所述商业规则包括酒店能够设置用来控制可用性的属性，并且如何应用所述商业规则的次序直接影响处理时间周期；

用于通过基于应用所述可用性规则计算器引擎的所述商业逻辑来限制来自所述预计算可用性关系数据的所述可用性从而创建针对所述多个入住时长上的所述日期范围内的每一天的最终可用性的装置；以及

用于针对在所述多个入住时长上的所述日期范围内的每一天，格式化所述至少一家酒店的酒店房间的所述最终可用性的装置。

32.如权利要求31所述的系统，其特征在于，还包括：

用于基于所述最终可用性向所述用户装置提供对所述查询的响应的装置，其中所述响应标识出对于所述日期范围内的日期有酒店房间可用，并且针对所述日期的所述酒店房间的所述可用性包括在基于所述可用性的所述限制的酒店房间的所述最终可用性中。

33.如权利要求31所述的系统，其特征在于，所述入住时长由位掩码表示，其中每一位表示所述入住时长中的一天。

34.如权利要求31所述的系统，其特征在于，所述查询装置接收来自旅行网站的所述查询。

35.如权利要求31所述的系统，其特征在于，所述查询装置接收来自客户机的所述查询。

36.如权利要求31所述的系统，其特征在于，用于应用商业逻辑的装置包括用于应用障碍规则的装置。

37.如权利要求31所述的系统，其特征在于，用于应用商业逻辑的装置包括用于应用限制规则的装置。

38.如权利要求31所述的系统，其特征在于，还包括用于将酒店房间的所述最终可用性发送到旅行网站的装置。

39.如权利要求31所述的系统，其特征在于，还包括用于将酒店房间的所述最终可用性发送到客户机的装置。

40.如权利要求31所述的系统，其特征在于，用于对所述酒店房间的所述可用性应用商业逻辑的装置包括：

产品项目收集器，用于指定所述商业规则以及如何应用所述商业规则的次序；和

多个可用性规则计算器，每个可用性规则计算器用于实现掩码计算器界面并且提供所述产品项目收集器将调用的掩码获取方法。

自底向上优化的搜索系统和方法

[0001] 本发明申请是国际申请号为PCT/US2011/026966，国际申请日为2011年3月3日，进入中国国家阶段的申请号为201180022331.X，名称为“自底向上优化的搜索系统和方法”的发明专利申请的分案申请。

背景技术

[0002] 现有酒店和旅行的计算机化预定系统使用自顶向下的搜索方法，从而在从搜索请求中找出(discover)单一的可用性单元(cell)时创建这些可用性单元。例如，这些系统可允许用户输入诸如酒店的标识、登记日期和结账日期之类的搜索准则来找出所选数量的夜晚(即，单次入住时长)的可用酒店房间。仅当找到产品项目(即，酒店房间)之后，方应用商业逻辑以寻找对于该特定一组产品项目的和费率和可用性。

[0003] 现有技术的该方法具有若干缺点。由于该方法的自顶向下的性质，需要用逐一的搜索请求来搜索并找出产品项目(例如，可销售房间)。此外，搜索请求必须具有酒店名称/代码、登记日期和结账日期作为最低限。如果这些最少元素不在可用性请求中，则将拒绝该请求。

[0004] 此外，现有技术的方法的核心服务只允许一次核对一家酒店，并且假设在搜索中使用登记和结账日期，费率和可用性响应只反映特定数量的夜晚。基本搜索不允许使用替换的日期，并且由此无法找出替换的费率。

[0005] 如果进行城市搜索(即，搜索集中在一个城市)，则该请求分成对单家酒店的多个搜索，并且每一个这种核心酒店搜索一次执行一个。核心搜索服务无法识别多家酒店之间的关系。由于每一核心请求一次针对一家酒店执行(其中每一请求需要执行复杂的商业规则以找到费率和可用性)，因此CPU时间和资源的成本较高。另外，假设每一请求具有CPU和资源密集性质，响应时间可能相对较慢。针对每一请求找出和计算费率和可用性规则的要求花费时间。

[0006] 同样，在现有技术的系统中，该请求是非常用户专用的，并且不允许交易系统帮助找出替换类型的产品，诸如不同的房间类型等。

[0007] 现有技术的最大缺点可能是单一的可用性单元。

[0008] 单一的可用性单元从搜索请求中的定义好的入住时长(LOS)导出。LOS从搜索请求中的起始日期和结束日期导出。看不到可从搜索请求中的LOS导出的多个可能的组合。例如，如果请求者还想要评估关于LOS的不同选项，则对于每一LOS选项，它将需要若干单独的搜索请求。现有技术没有提供替换日期或产品项目的灵活性。建立具有全部灵活性的回复所需的成本和时间此时无法节约成本。例如，如果消费者想要核对针对五月第一周的可用性，则将需要49次单独的可用性调用。针对这一周的每一天，七个可用性调用(即，7天X每一天7个LOS值)。该交易将使成本增加48次的因数(即，1次调用和49次调用之间的差值)，并且将导致对消费者的很长的响应时间(即，差的用户体验)。

[0009] 因此，在现有技术中存在改进旅行和酒店交易系统的需要，如下文中更详细描述的。

附图说明

[0010] 结合在本说明书中且构成其一部分的附图示出了各个实施例，并且与描述一起用来解释方法和系统的原理：

[0011] 图1A示出酒店预定系统的结构。

[0012] 图1B示出一个实施例中的本发明的基础建筑块的结构。

[0013] 图2示出另一实施例中的本发明的结构。

[0014] 图3示出一个实施例中的描述本发明的高级序列图。

[0015] 图4是一个实施例中的与本发明一起使用的框架结构流程图。

[0016] 图5是一个实施例中的可与本发明一起使用各个计算器的示图。

[0017] 图6是可用于掩码(mask)计算器的样本UML。

[0018] 图7示出一个实施例中的可由本发明采用的步骤。

具体实施方式

[0019] 在公开和描述本方法和系统之后，应当理解本文中所使用的术语只是出于描述特定实施例的目的而不旨在是限制性的。

[0020] 如在说明书以及所附权利要求书中使用的，单数形式的“一”、“一个”以及“该”包括复数引用，除非上下文另外明确地指出。在本文中，范围可表达为从“约”一个特定值和/或到“约”另一特定值。当表达这种范围时，一个实施例只可以包括例如从一个特定值和/或到另一特定值。

[0021] 类似地，当值被表达为近似值时，应当理解，通过使用修饰语“约”，特定值构成另一实施例。还应理解，每一范围的端值无论是与另一个端值联系起来还是独立于另一个端值都是有意义的。

[0022] “任选的”或“任选地”表示随后描述的事件或情形可能或可能不发生，并且该描述包括所述事件或情形发生的实例以及它不发生的实例。

[0023] 在本说明书的具体实施方式和权利要求中，单词“包括”以及诸如“包含”和“具有”之类的该单词变体表示“包括但不限于”，而不旨在排除例如其他添加、组分、整体、或步骤。“示例性”表示优选或理想实施例的示例，而不旨在传递该实施例的指示。“诸如”不用于限制性的意义，而是出于解释的目的。

[0024] 公开的是可用于执行所公开的方法和系统的组件。在本文中公开了这些以及其他组件，并且应当理解，在公开这些组件的组合、子集、相互关系、组等而可能未明确地公开这些组件的每个不同的单独和集合的组合以及排列的具体参考时，对于所有方法和系统，在本文中具体考虑和描述它们中的每一个。这适用于本申请的所有方面，包括但不限于所公开的方法中的步骤。由此，如果存在可执行的各个附加步骤，则应当理解，可通过所公开的方法的任一特定实施例或者各个实施例的组合来执行这些附加步骤中的每一个。

[0025] 通过参考本文中所包括的优选实施例和示例的以下详细描述、以及附图及其先前和以下描述，可更容易地理解本方法和系统。

[0026] 本发明提供了经改进的酒店房间可用性交易系统。图1A示出用于六洲酒店(Six Continents Hotels)公司的全球连锁酒店的目前发展水平(state-of-the art)的酒店预

定系统(即,由六洲酒店公司操作的HOLBDEX(HDX)系统)的结构。本发明可结合该系统、或者市场上可用的其他合适的酒店预定系统使用。

[0027] 在图1A的预定系统中,HDX服务器102根据日期、费率以及关于酒店房间的其他详细信息来存储当前可用性。各个客户机102a、102b、102c和102d可向HDX服务器102提交请求。例如,客户机102a可包括使用例如OTA或XML协议与HDX服务器101通信的旅行网站,该旅行网站通过中间数据中心103耦合到HDX服务器101。客户机102b可包括使用例如Pegasus协议与HDX服务器101通信的全球分布系统(GDS),诸如Sabre、Galileo、WorldSpan、Amadeus或TravelWeb。酒店品牌的法人所有人可经由客户机102c通过例如3270或XML协议与HDX服务器通信。最后,各家酒店102d可使用例如XML和/或HMI协议与HDX服务器101通信。

[0028] 客户机(102a、102b、102c和/或102d)提交给HDX服务器101的可用性搜索请求可包括酒店代码、登记日期和结账日期作为最低限。由于搜索多家酒店、多个数据范围等可通常导致对HDX服务器101上的多个搜索的需要,因此本发明提供经改进的方式来进行搜索,如在下文中进一步描述的。每一主要数据中心中的当前环境(即,硬件和网络)中的最大挑战是每年60%的可用性业务量(traffic)的巨大增长。最有效的搜索处理是预计算数据和实时活动(on-the-go)数据之间的精确平衡。当每天数百万搜索请求到达预定系统时,已计算的可用性数据的节约部分编译到强大的可用性交易系统中。为了确定如何在HDX服务器101上最有效地进行搜索,标记化(tokenize)可用性处理以基于各种准则对每一步骤进行分类。当分类可采用许多形式时,在一个实施例中,可使用表格A的以下准则:

[0029]

准则	标记化
发现过程	基于数据库调用量以及找出用于该请求的产品项目所需的逻辑对可用性的每一元素进行分级。例如,在一个实施例中,分级可以是: 酒店=低 费率类别=低 产品项目=高

[0030]

处理周期	可基于每一商业逻辑组件需要的处理时间量对其进行分级（低到高）。例如，在一个实施例中，分级可以是： 抵达日不可入住= 低 入住天数 = 中 净收入 = 高
关联水平	每一商业逻辑组件可与执行特定商业逻辑的可用性的元素相关联。例如，在一个实施例中： 净收入 = 产品项目 入住天数 = 费率类别

[0031] 表格A

[0032] 如图1B所示，本发明包括预计算可用性数据库151、以及可用性规则计算器引擎152。在一个实施例中，预计算可用性数据库151可包括关系数据库，该关系数据库具有两个主要特性：

[0033] • 每一行表示一天最不可能销售的单元(产品项目)。不需要诸天找出一组产品项目的找出过程。逐天每一产品项目可从高级费率类别导出，该费率类别使用具有根据种类和日期范围可用性相关联的房间类型的费率代码限定。

[0034] • 表格的大小应当是可管理的。该表格可快速地更新和读出。每一产品项目的元素数量是非常有选择性的，以维持最少数量的元素供在每一行中预计算。

[0035] 数据库主关键字可由酒店代码、日期、房间类型和费率类别构成。在一个实施例中，每一行的其他关键字数据可以是：

[0036] • 物理入住时长(LOS)。该串的值表示对例如长达14天的入住(还可选择任何其他最大LOS)的每晚的位掩码。房间类型的物理可用性可使用分配、房间的总数、以及对房屋空间运用的类型等概念。这些规则可针对进行LOS掩码的每晚重复应用。

[0037] • 费率量。该费率量可从与产品项目的费率类别相关联的费率代码导出。该费率量的值可以是固定量或者百分比值。同时，该量可在费率代码水平或者从另一基础费率代码中找到。

[0038] • 收入LOS。该串的值可表示对例如长达14天的入住(或者任何其他最大LOS)的每一晚的位掩码。收益(yield)管理组件可使用基于LOS的净收入计算。正收入可用开(1)表示而负收入可用关(0)表示。可存在可指示使用基于其他费率或佣金的另一值的若干收益选

项。这些规则可针对进行LOS掩码的每晚重复应用。

[0039] • 费率代码。奇偶概念可使用费率代码来维持所有费率代码之间的关系的水平。

[0040] 该关系数据库的布局产生非常快的取出产品项目的方式。有可能将预计算可用性数据库151与属性表格结合，从而在该初始级实现过滤器。在一个实施例中，一些属性(比如床的类型、床的数量和/或最大占用率)可帮助预选产品项目。这不同于在可用性处理的结尾应用过滤器的当前技术，其导致关于不具资格的产品项目的额外处理周期。

[0041] LOS驱动的值可使用位掩码表示。每一位表示一晚，且这些值可以是开(1)或关(0)。对LOS可表示多长没有限制。适当的值可对应于适当的受众或酒店品牌。例如，在一个实施例中，加长入住品牌可使高达28天或更长的LOS被预计算。

[0042] 如图2所示，在一个实施例中，可用性规则计算器引擎151和预计算可用性数据库152可包括具有J2EE应用程序(application)201的预计算框架，该J2EE应用程序201消费来自全球网(Web Sphere)MQ 202的消息并处理这些消息，并且用所获取的数据来更新Oracle数据库203。框架可包括J2EE应用程序201，该J2EE应用程序201可包括核心Java应用程序204。图2示出根据本发明的一个实施例的具有每一应用程序中的主要组件的总体框架。也可使用其他合适的框架。

[0043] 图3是描述图1B和2的预计算引擎中的消息处理的一般流程的高级序列图。该序列图示出消息按顺序到达且立即处理时的情形。图4是一个实施例中示出本发明的整个流程的框架结构流程图。

[0044] 在下文中，将更详细地定义图2-4的结构组件。

[0045] • PreComputeMDBean 211-PreComputeMDBean 211是监听MQ Series 202的队列411的消息驱动的Bean(豆荚，一种类)，该队列411接收来自HOLEDEX 101的消息。PreComputeMDBean 211用从HOLIDEX 101接收到的XML消息来调用消息管理器(MessageManager)214。

[0046] • 无状态会话Bean 212:可使用以下无状态会话Bean 212:

[0047] • BootStrapBean:用于初始化应用程序。当应用程序启动时，在此处初始化Log4j以及其他计时器任务定向的Bean。

[0048] • MessageManagerBean:该EJB用于处理用于所有源和客户机的消息。基本上，该EJB可调用消息过滤器214或者扩展该消息过滤器214。同样，该Bean可由MDB 211调用。

[0049] • 计时器Beans 231-该EJB使用计时器服务。该EJB用于处理在待决消息表格中待决的消息。该EJB在预定时间运行并调用消息管理器，锁定具有待决消息的任一酒店，并且处理该消息。顺序地处理来自Holidex的事件是重要的。实现群集环境(即，不同的节点可同时对同一酒店进行更新)需要该更新处理方法。

[0050] • 度量管理器()232-该无状态会话Bean用于在预定义时间收集使用计时器服务(每四个小时说话)的度量。该无状态会话Bean实现TimedObject.该EJB将通过BootStrapBean初始化，并且将创建计时器。

[0051] • 数据客户机()233-数据客户机组件是在预计算引擎内存在的单个组件。其作用是双重的。首先，它提供关于如何代表推客户机建立产品项目的特定信息。其次，它提供一旦作出改变则向那些推客户机通知这些改变的手段。

[0052] • 登录234-Log4J将用作该应用程序的登录机制。登录的消息将被定向到在其上

运行应用程序服务器的服务器上的文件。应当通过使用引导EJB或者使用应用程序服务器中的启动类来加载Log4j的配置。存在不同水平的登录：调试、信息、警告和错误。

[0053] • 消息管理器214-消息管理器214可主要执行两个操作。首先，它可将XML消息转换成Java对象，并且随后它可基于消息类型调用特定预计算管理器216。它还基于酒店代码和消息序列号管理信息的顺序处理。如果这些消息不是按顺序的，则该消息将存储在单独的表格中以供稍后处理。

[0054] • 管理器制造厂(ManagerFactory)218-管理器制造厂218的类基于消息类动态地创建预计算管理器216。一旦该管理器制造厂218创建了管理器，它就可缓存该管理器以供稍后使用。

[0055] • XMLSchema Java类-这些XMLSchema Java类可以是消息管理器214的一部分，并且基于从HOLIDEX 101接收到的XML消息生成。这些类在其方法中保持XML元素和属性的值。这些XMLBean类用作数据载体，由此避免每一消息的单独传输对象。

[0056] • 预计算管理器() • 216-在一个实施例中，针对每一消息类型或者消息类型的组合，将存在单独的预计算管理器216。其基本功能是使用一些商业逻辑和预计算实用类将接收到的Java对象转换成与字段相关的期望数据库。大多数商业处理在此处以及实用类中进行。

[0057] • 锁定酒店直至(HotelLockUtil)220-该类用于锁定房屋消息表格中的酒店数据。一旦锁定记录，就处理该酒店的消息。有必要锁定该酒店以按照顺序次序处理消息并防止使用同一酒店的其他消息的竞争情形。如果消息被失序记录，则该类将使全部数据库无效。

[0058] • PendingMessageUtil-该类用于存储、检索和删除待决消息表格中的消息。当这些消息使用群集环境从MQ队列中读取时，有必要按顺序处理这些消息。每一酒店消息携带序列号以维持顺序次序处理。

[0059] • 验证类222-在一个实施例中，针对每一消息存在单独的验证类。这些验证类验证消息，并且如果它们找到错误，则它们抛出PreComputeException。

[0060] • 数据访问对象(DAO)226-数据访问对象226概括与Oracle数据库203相关的通信和CRUD操作。预计算管理器216调用适当的DAO 226来更新这些表格。DAO 226可基于单个表格或者基于相关的表格。基本上，该DAO 226从数据库连接器(DBConnector)类228获得数据库连接，准备对数据库的SQL调用，并且执行CRUD操作。一旦完成这些操作，它就释放到数据库203的连接。.

[0061] • 数据库连接器(DBConnector)228-数据库连接器228是概括数据库203的连接机制的Java类。在一个实施例中，该数据库连接器228将通过调用数据源对象来获得到数据库203的连接。该类将具有诸如释放连接、退回交易等各种方法。

[0062] • 预计算实用组件-可开发该应用程序所需的所有共用功能作为实用类，诸如如该应用程序所需的日期验证或特定数据转换。可存在消息类型专用的一些实用类。

[0063] 可用性规则计算器引擎152可包括归组为以下两类的逻辑：

[0064] • 基于搜索请求的登记日期值的商业逻辑，由此无法预计算该商业逻辑。这种逻辑的示例可以是预先购买限制。

[0065] • 在处理时分级为低(基于先前参考表格A描述的准则)的商业逻辑，该商业逻辑

与定义好的可用性元素相关联。

[0066] 总体目标是在预计算可用性数据库151和可用性归组计算器引擎152的逻辑之间推导出良好的平衡。可动态地调节该平衡以维持高性能。在向可用性模型添加新逻辑的情况下，这可修改一些可用性元素中的得分，其可在预计算可用性数据库151和可用性归组计算器引擎152的逻辑之间创建新的平衡(即，甜区(sweet spot))。

[0067] 可用性规则计算器152负责可用性处理的“实时活动”逻辑。商业规则可在不同的组件(即，可用性计算器)中分离，由此它们可独立地实现。可存在可与每一请求实时地应用的一些障碍(fence)和限制。例如，它们可表示可设置酒店以控制可用性的属性，例如预先购买、入住几晚、最短/最长入住、特别的需求。其他障碍和限制只可以在酒店水平、费率类别水平和费率代码水平找到。在处理时间周期中，如何应用商业规则的次序可具有直接的影响。

[0068] 可用性规则计算器引擎152可具有两种主要组件：

[0069] • 产品项目收集器，即可用性处理的指挥家(orchestrator)。该产品项目收集器使用收集选项来指定规则，并且将应用这些规则的次序。

[0070] • 可用性规则计算器。(在下文中参考图5进一步描述的)每一计算器实现掩码计算器507的界面，并且必须提供该计算器将调用的getMask()方法。相对于应用该规则的当前产品项目LOS掩码，连同其他掩码，所返回的掩码为ANDed。同样，它们必须通过实现getReasonCode()来提供原因代码。

[0071] 在一个实施例中，可建立和操作可用性规则计算器引擎152，如在下文中更详细描述的。

[0072] 类模型

[0073] 在内部，收集器502与若干“计算器”组件(507以及图6中的601-606，在下文中进一步描述的)合作，每一组件专属于入住时长(LOS)规则的特定类。一般而言，每一计算器返回作为ANDed的与产品项目LOS掩码一起导出整个LOS掩码的位掩码。

[0074] 收集上下文

[0075] 收集上下文对象503用于保持相对于收集周期的全局状态。当调用方调用收集器502上的collect()(经由产品项目收集器界面504)时，收集器502将创建收集上下文对象503。收集上下文503可具有以下特性：

[0076] • 酒店代码

[0077] • 费率类别代码

[0078] • 日期

[0079] • 数据库对象

[0080] • 收集选项

[0081] • 经缓存的事实

[0082] 收集上下文503被传递到每一计算器的构造器中。构造器初始化每一计算器的环境，并且它是该类的软件组件。计算器使用收集上下文503来获得数据库203以查询CRS表格。这些计算器可通过上下文对象使用该收集上下文503来相互共享事实。例如，如果发现酒店为HIRO酒店，则该事实对一个以上计算器可以是重要的。

[0083] 计算器

[0084] 计算器(601-606)是PACE引擎的关键商业逻辑组件。收集策略505基于在收集选项506中指定的是什么来确定使用哪些计算器。每一计算器实现掩码计算器507的界面，并且可提供收集器502将调用的getMask()方法。相对于应用该规则的当前产品项目LOS掩码，连同其他掩码，所返回的掩码为ANDed。还可通过实现例如getReasonCode()来供应原因代码。

[0085] 掩码计算器507的样本界面可以是：

[0086]
```
public interface MaskCalculator {
 int getMask(ProductItem pi) throws ServiceException;
 int getReasonCode();
}
```

[0087] 可存在任意数量的掩码计算器507。图6示出当前集合的示例UML。在一个实施例中，这些计算器可携带IHG可用性模型的非常具体的障碍或限制规则，诸如：

[0088] • 到达计算器601-该到达计算器601使用该请求的登记天数来验证是否在酒店或费率类别水平中设置了涂黑(black-out)日期或者到达日不可入住。• 最小最大计算器602-该最小最大计算器602使用登记日期和结账日期来导出LOS。如果存在入住限制的最小和最大的天数，则LOS值应当满足该入住限制的最小和最大的天数。

[0089] • 收入计算器603-该收入计算器603验证特定费率量的LOS产生正收入。

[0090] • 房间销售策略计算器604-该房间销售策略计算器604验证特定房间类型与酒店的房间销售策略相关联。酒店房间销售策略标识可看到销售的房间类型。

[0091] • 入住天数计算器605-该入住天数计算器605使用登记日期和结账日期来导出LOS。LOS值应当满足一周需要入住的天数。

[0092] • 有效天数计算器606-该有效天数计算器606使用登记日期和结账日期来导出LOS。LOS值应当满足一周中标记为有效的天数。

[0093] 酒店规则缓存

[0094] 具有尽可能高效的PACE引擎是有利的。参考图5和6，由于掩码计算器507包括在CRS数据库中编码的酒店规则，因此该PACE引擎的大部分落在该掩码计算器507的实现上。每当PACE引擎构建新的掩码计算器507时，计算器507可通过只需从数据库203中预先取出数据一次、并且随后针对其getMask()方法的每一调用使用该数据来优化它本身。即使这样，使所有计算器以几乎不变的规则查询数据库是对于对性能的重大影响，并且应当避免。

[0095] 为了解决该问题，数据上下文对象503(图5)可包括代表掩码计算器507对缓存规则的支持。一旦掩码计算器507已从数据库203建立其规则，它就可将这些规则缓存到数据上下文503上。进而，这表示所有掩码计算器507可首先检查以查看它们需要的规则在达到数据库203和重建这些规则之前是否仍然存在。该模型将有助于性能的增加，并且因此只要有可能就使用。

[0096] 此处是用于缓存规则的方法示例：

[0097]

```
public class CollectionContext {
 public Object getHotelRules(MaskCalculator c);
 public void putHotolRules (MaskCa1cu1atorc.Object rules);
}
```

[0098] 一旦掩码计算器507缓存规则，这些规则就可保持在存储器中，直至以下事件之一发生：

[0099] • 数据库203中的酒店规则变化

[0100] • LRU缓存清除用于资源管理的酒店规则

[0101] • 重启应用程序

[0102] 在一个实施例中，掩码计算器507可极力使用缓存方法并假定它们是可靠的。如果用于缓存变化管理的合适解决方案不可用，则PACE引擎可确保不激活缓存。仍然，掩码计算器507可假定缓存一直是有效的。

[0103] • 收集策略

[0104] 由于可出于不同的目的利用PACE引擎，因此用于所有情况的单一搜索算法是不适用的。例如，来自拉应用程序的可用性请求首先可看到更高级的限制、最后可看到产品项目，而推应用程序可能首先想要看到产品项目并可能地按照最有效的次序应用这些规则。

[0105] 为了解决该问题，PACE引擎可适用收集策略505的界面来表示实际上如何进行收集。这基本上是本领域技术人员已知的抽象算法设计图案的实现。对策略的选择可基于收集选项506中的设置。

[0106] 收集策略505的样本界面提供如下：

[0107]

```
public interface CollectionStrategy {


 void collect(CollectionContext ctx,
     Product ItemConsumer consumer) throws ServiceException;
}
```

[0108] 在一个实施例中，所支持的收集策略505可包括：

[0109]

类
基本策略
默认策略
产品项目ID策略

[0110]

描述

具有基本计算模板方法的抽象类
以用于背离类别代码的SQL准则来扩展基本策略
以用于产品项目ID的SQL来扩展基本策略

[0111] 查询

[0112] 当调用PACE引擎时，该PACE引擎最终可从产品项目表格中读取产品项目，并且应用掩码计算器507来导出最终LOS。用于选择产品项目的查询可考虑大量因素。这些因素包括：

[0113] • 选择准则

[0114] . • LOS历史

[0115] • 售罄禁止(Sell-Thru Exclusion)

[0116] 这些区段在下文中更详细地描述该查询的这些方面。

[0117] • 基本查询

[0118] • LOS历史

[0119] • 售罄禁止

[0120] (1)基本查询

[0121] 基本查询表示选择产品项目的最简单形式的SQL。实际上，这可能不是所使用的实际SQL。但是，由于该基本查询可用作另一全部形成的查询的基线，因此在此对其进行描述。换句话说，为了理解全部查询，对基本查询的理解是重要的。

[0122] 基本查询的准则包括以下方面：

[0123] • 酒店代码

[0124] • 数据范围

[0125] • 费率类别代码

[0126] • 有效状态

[0127] • 变化标记(任选的)

[0128] • 排除的房间代码(任选的)

[0129] • 产品项目ID(任选的)

[0130] “有效状态”只是指跳过无效产品项目的行。变化标记用于选择标记值大于或等于指定值的行。排除的房间类型用于过滤掉排除的房间。最后，可包括产品项目ID列表以代替变化标记，用于更精确地选择产品项目。

[0131] 标记值和产品项目ID用于帮助只过滤出实际上已通过预计算修改的那些产品项目。虽然存在可用于最好地指示什么时候和为什么为了预计算而改变产品项目的各种方式，在一个实施例中可使用标记数量加上产品项目ID。

[0132] 出于两个原因，产品项目传送给产品项目消费者的次序是重要的。首先，该次序允许计算器507关于上下文切换是高效的。其次，该次序允许产品项目消费者容易地识别它们建立的结果。

[0133] 为此，这些结果可如下地排序；

[0134] • 日期

[0135] • 费率类别代码

[0136] • 房间代码

[0137] 因此，基于以上教义且参考图7，在一个实施例中，本发明可执行以下步骤：

[0138] • 生成包括与多个入住时长的酒店房间相关联的实时可用性属性的预计算数据库(步骤701)；

[0139] • 接收针对一日期范围对至少一家酒店的查询，该日期范围具有起始日期和结束日期(步骤702)；

[0140] • 通过根据该查询应用商业需求来计算多个入住时长的日期范围内的每一天的可用性(步骤703)；

[0141] • 针对多个入住时长的日期范围内的每一天，通过组合来自预计算数据库的可用性和查询属性来创建最终可用性(步骤704)；以及

[0142] • 针对多个入住时长的日期范围内的每一天，格式化至少一家酒店的酒店房间的最终可用性(步骤705)。

[0143] 尽管已结合优选实施例和特定示例描述了方法和系统，但并不旨在将范围限制于所陈述的特定实施例，因为本文中的实施例在所有方面旨在是说明性的而不是限制性的。

[0144] 除非另外明确地说明，否则不旨在将本文中所陈述的任何方法解释为需要其步骤以特定的次序执行。因此，在方法权利要求实际上不叙述其步骤跟随的次序、或者在权利要求或具体实施方式中未以其他方式说明这些步骤限于特定次序的情况下，不旨在在任何方面推断次序。这在任何可能的未表达的基础上保留用于解释，包括：相对于步骤的排列或操作流程的逻辑内容；从语法组织或标点符号导出的普通含义；说明书中所描述的实施例的数量或类型。

[0145] 在不背离范围或精神的情况下可作出各种修改和变体对本领域技术人员而言是显而易见的。根据对本文中所公开的说明书和实践的思考，其他实施例对本领域技术人员而言是显而易见的。说明书和示例只是示例性的，其中真实范围和精神由所附权利要求书指明。

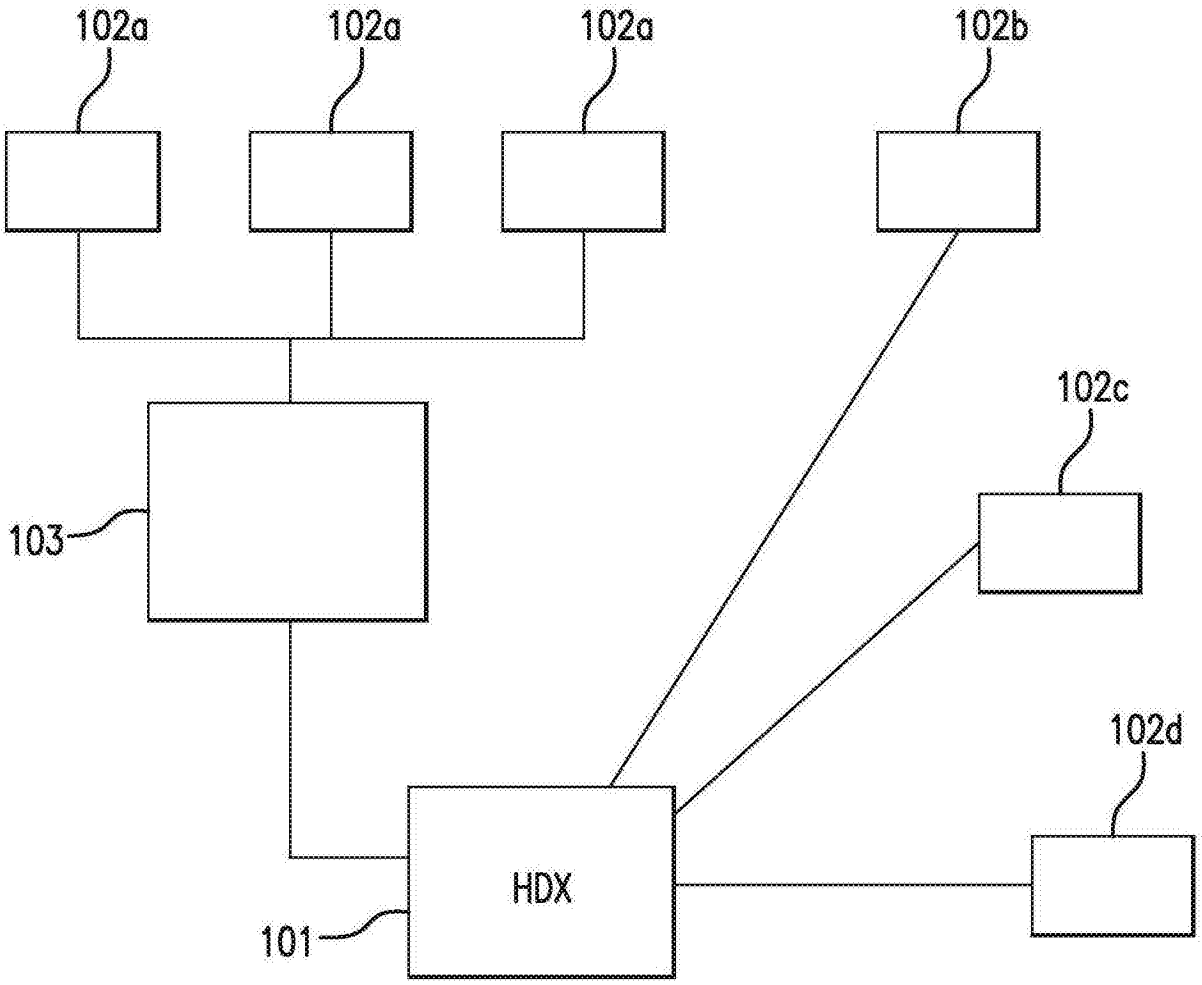

图1A

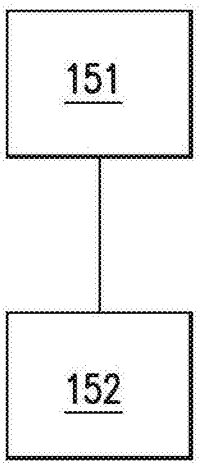

图1B

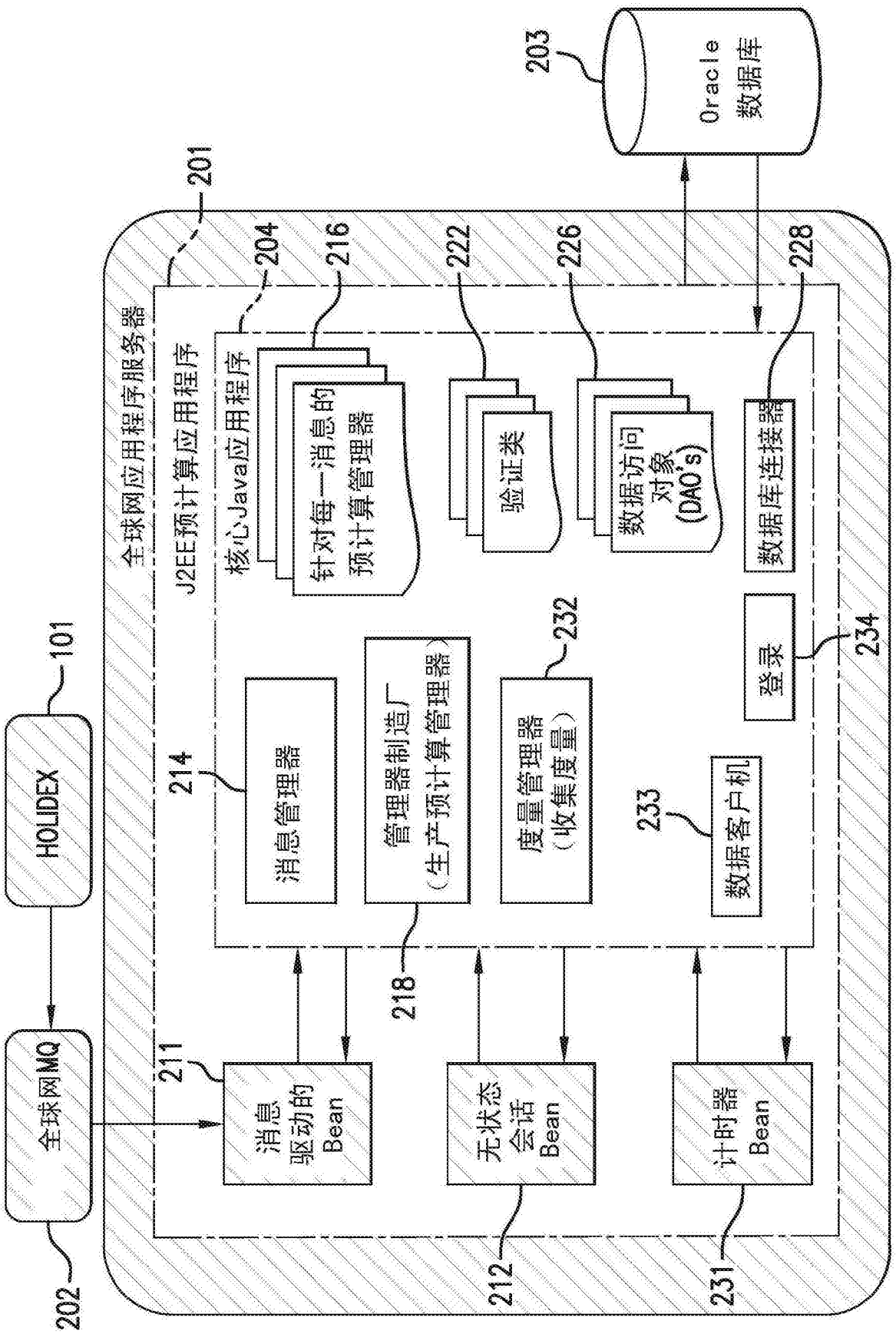
Oracle
数据库
203
201
全球网应用程序服务器
J2EE预计算应用程序
核心Java应用程序
204
216
针对每一消息的
预计算管理器
222
验证类
226
数据访问
对象
(DAO's)
228
数据库连接器
消息管理器
214
管理器制造厂
(生产预计算管理器)
218
232
度量管理器
(收集度量)
登录
234
233
数据客户机
HOLIDEX
101
全球网MQ
202
211
消息
驱动的
Bean
无状态
会话
Bean
212
计时器
Bean
231

图2

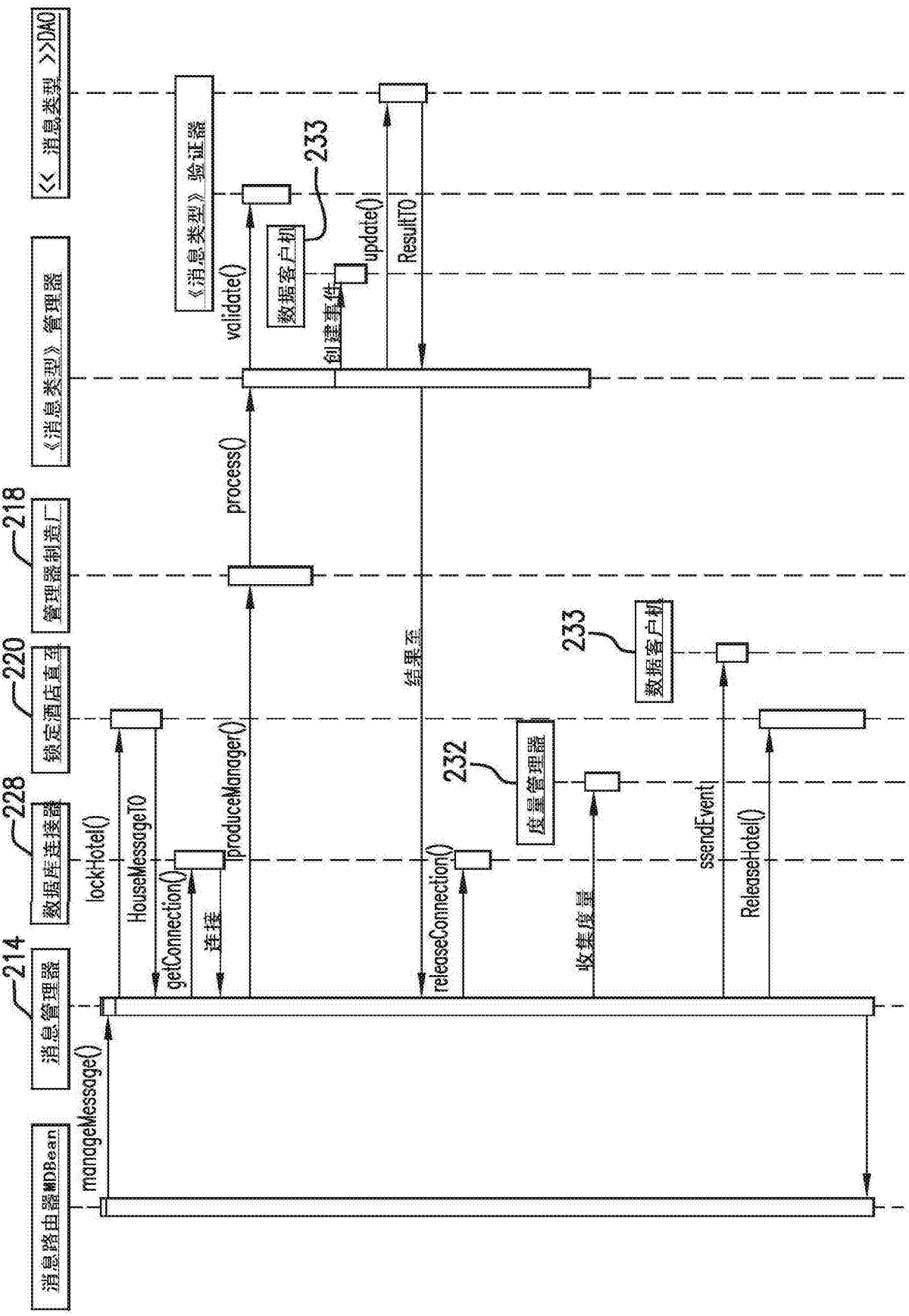
消息路由器MDBean
消息管理器
214
数据库连接器
228
锁定酒店直至
220
管理器制造厂
218
《消息类型》管理器
《消息类型》验证器
<< 消息类型 >>DAO
manageMessage()
lockHotel()
HouseMessageTO
getConnection()
连接
produceManager()
process()
validate()
233
数据客户机
创建事件
update()
ResultTO
结果至
releaseConnection()
232
度量管理器
收集度量
233
数据客户机
ssendEvent
ReleaseHotel()

图3

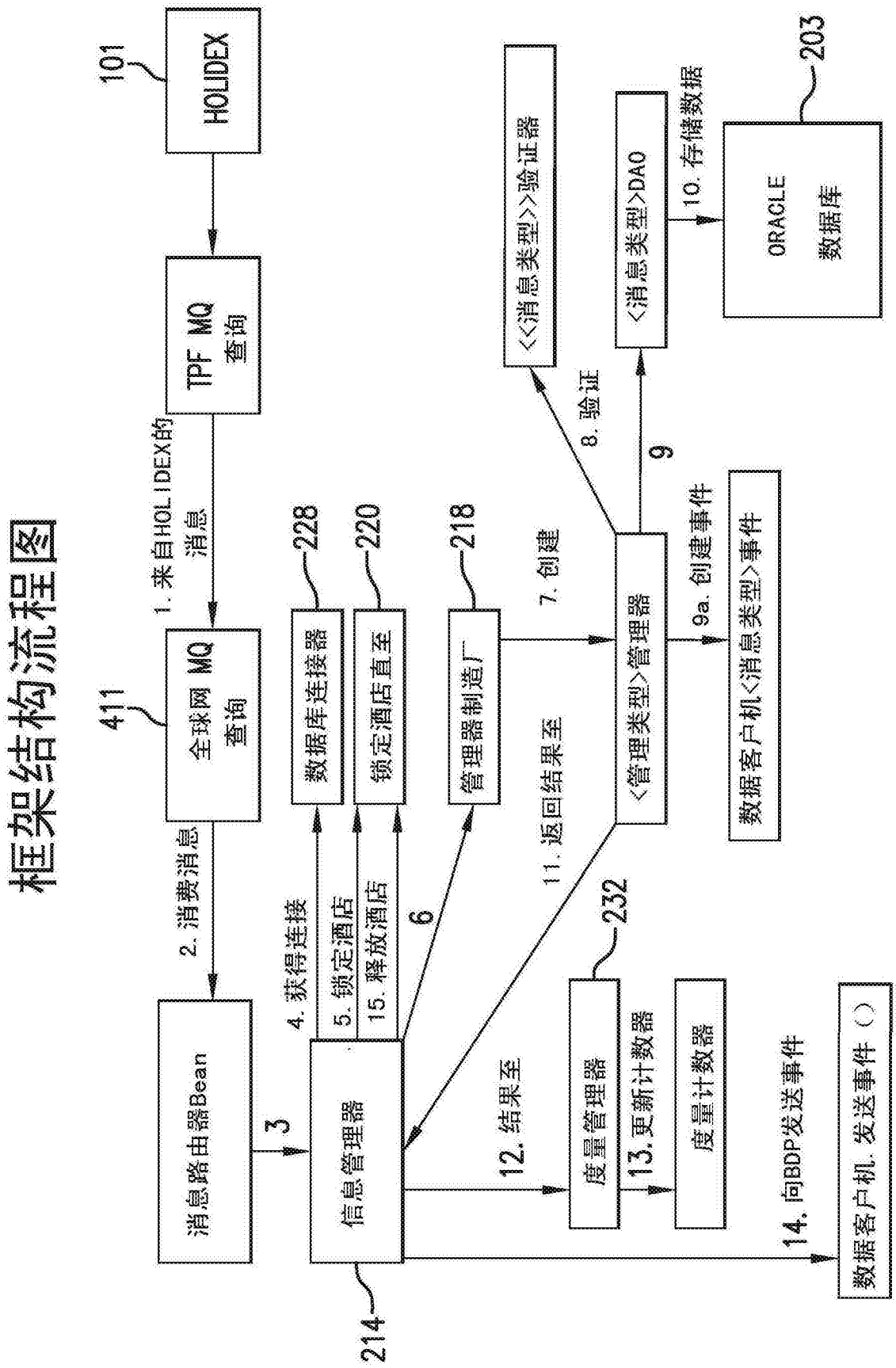
框架结构流程图
HOLIDEX
101
TPF MQ 查询
1. 来自HOLIDEX的消息
全球网 MQ 查询
411
2. 消费消息
消息路由器Bean
3
信息管理器
214
4. 获得连接
数据库连接器
228
5. 锁定酒店
锁定酒店直至
220
15. 释放酒店
6
管理器制造厂
218
7. 创建
11. 返回结果至
<管理类型>管理器
8. 验证
<<消息类型>>验证器
9
<消息类型>DAO
10. 存储数据
ORACLE 数据库
203
9a. 创建事件
数据客户机<消息类型>事件
12. 结果至
度量管理器
232
13. 更新计数器
度量计数器
14. 向BDP发送事件
数据客户机. 发送事件（）

图4

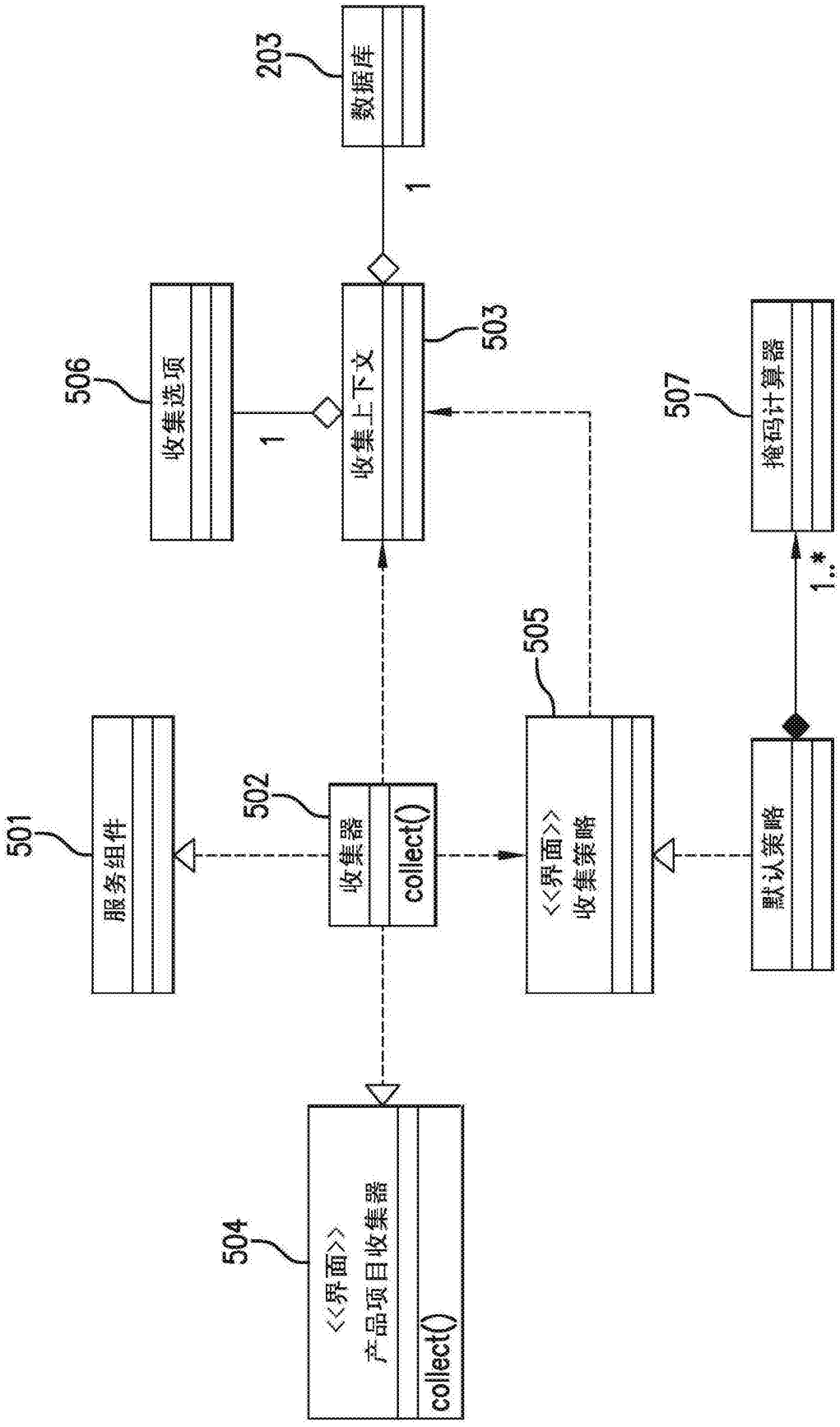
203
数据库
1
506
收集选项
1
收集上下文
503
507
掩码计算器
1..*
505
501
服务组件
502
收集器
collect()
<<界面>>
收集策略
默认策略
504
<<界面>>
产品项目收集器
collect()

图5

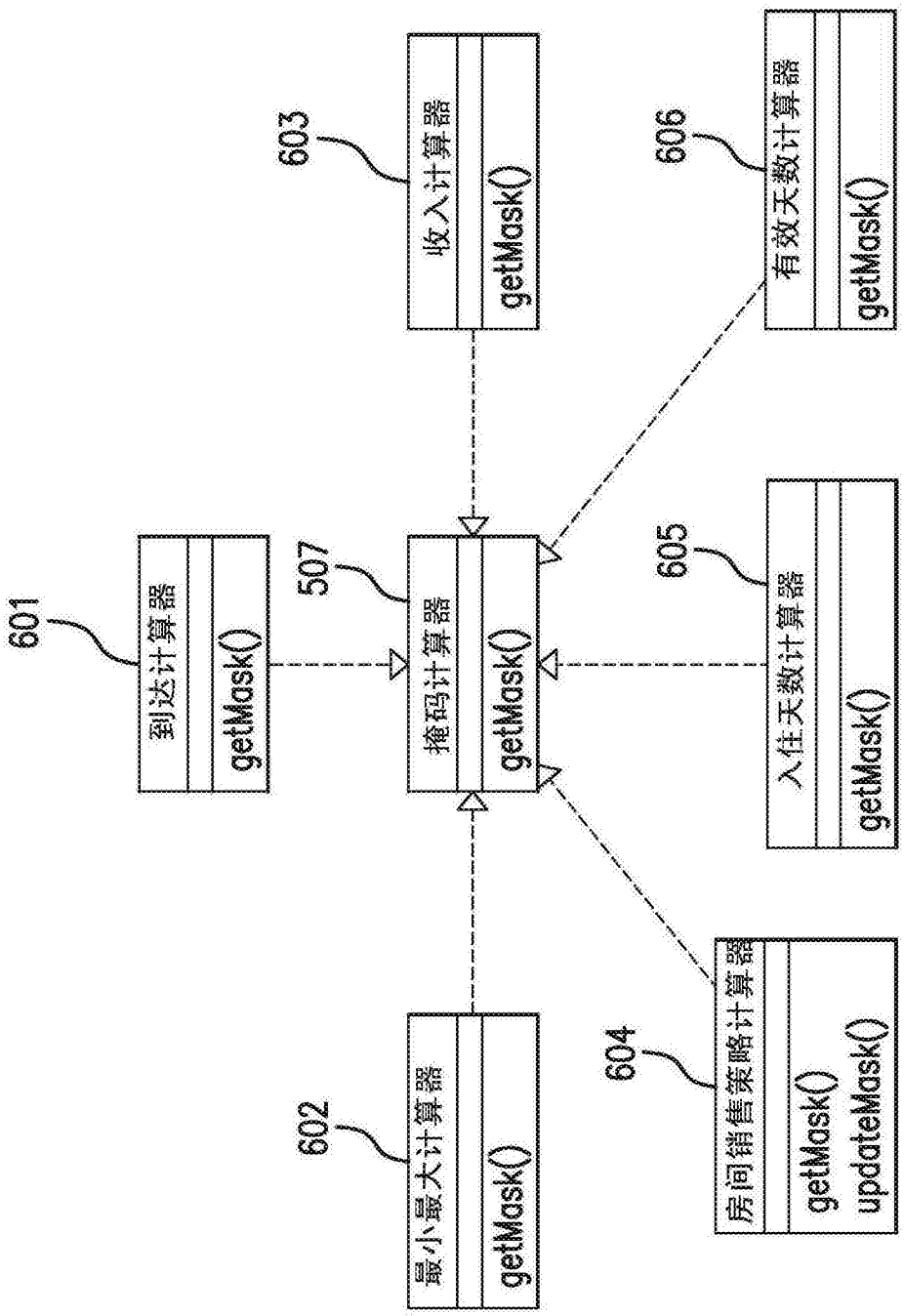
603
收入计算器
getMask()
606
有效天数计算器
getMask()
601
到达计算器
getMask()
507
掩码计算器
getMask()
605
入住天数计算器
getMask()
602
最小最大计算器
getMask()
604
房间销售策略计算器
getMask()
updateMask()

图6

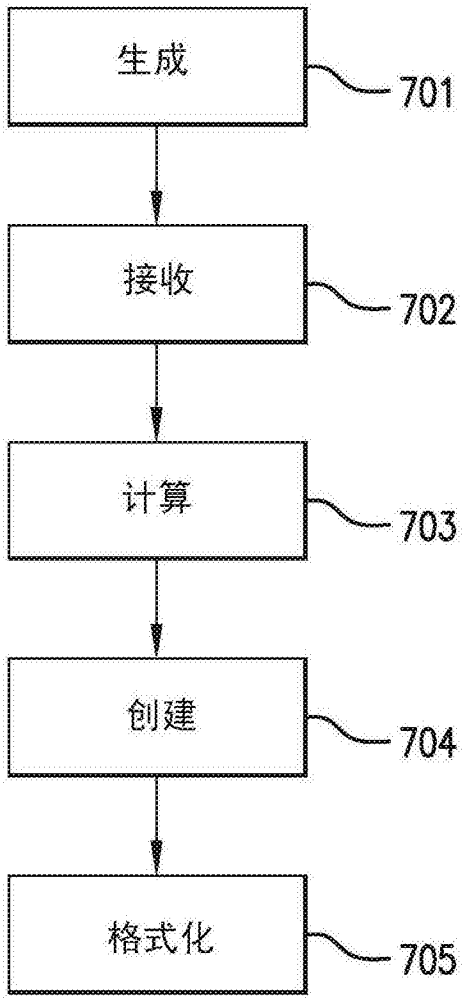

图7