Partitionierungsstrategien für Data Vault

Dani Schnider
Trivadis AG
Zürich/Glattbrugg, Schweiz

Einleitung

Während das Laden von Tabellen in Data Vault in der Regel nicht zeitkritisch ist, stellt uns das effiziente Extrahieren von Daten aus komplexen Data-Vault-Schemas immer wieder vor Herausforderungen. Abfragen auf zahlreiche Hubs, Links und Satellites sind notwendig, um die Informationen in einer Form zur Verfügung zu stellen, die beispielsweise das Laden einer Dimensions- oder einer Faktentabelle in einem dimensionalen Data Mart ermöglichen. Wie können die Partitionierungsmöglichkeiten von Oracle 12c genutzt werden, um das Lesen aus den Data-Vault-Tabellen so effizient wie möglich durchzuführen?

Im Gegensatz zur Partitionierung eines Star Schemas gibt es für Data Vault keine eindeutige Empfehlung, wie die Tabellen idealerweise aufgeteilt werden und welche Partitionierungsmethode verwendet werden soll. Es gibt verschiedene Ansätze, jede von ihnen hat Vor- und Nachteile. Nachfolgend werden drei mögliche Partitionierungsstrategien vorgestellt.

Strategie 1: Partitionierung nach Ladedatum

Jede Tabelle in Data Vault ist mit einem Ladedatum versehen. Mit diesem technischen Zeitstempel wird für jeden Datensatz festgehalten, wann er ins Data Warehouse geladen wurde. Eine naheliegende Partitionierungsstrategie besteht darin, das Ladedatum als Partitionierungsschlüssel zu verwenden. Nur bringt dies in den meisten Fällen keine Vorteile. Für spezifische Anwendungsfälle kann diese Strategie jedoch nützlich sein.

Bei der Data-Vault-Modellierung wird nicht unterschieden zwischen Stammdaten („Dimensionen“) und Bewegungsdaten („Fakten“). So werden beispielsweise Verkaufstransaktionen genau gleich behandelt wie Kunden oder Produkte. In allen Fällen wird ein Hub mit einem oder mehreren Satelliten erstellt. Für das physische Datenbankdesign ist es jedoch nützlich zu wissen, ob die Daten regelmäßig aktualisiert werden (typisch für Stammdaten), oder ob sie einmal geschrieben und danach nicht mehr verändert werden (typisch für Bewegungsdaten). Im ersten Fall wird für jede Änderung eine neue Version im entsprechenden Satelliten erstellt. Im zweiten Fall wird nur ein Eintrag in den Satelliten eingefügt, und zwar zur gleichen Zeit, in welcher der Hub-Eintrag geschrieben wird. Diese Eigenschaft können wir für die Partitionierung der Tabellen ausnützen.

Für Bewegungsdaten, die nur einmal geschrieben werden, können der Hub und die zugehörigen Satelliten und Links nach Ladedatum partitioniert werden (mittels RANGE- oder INTERVAL-Partitionierung). Dies hat den Vorteil, dass beim Extrahieren aller Bewegungsdaten (z.B. Verkaufstransaktionen) eines bestimmten Tages oder eines Datumintervalls nur die relevanten Tages- oder Monatspartitionen gelesen werden müssen (Partition Pruning). Außerdem ist es auch einfach möglich, historische Daten aus dem Data Vault zu löschen, wenn sie nicht mehr benötigt werden. Dazu werden einfach die alten Partitionen der Tabellen gelöscht. Auch andere Möglichkeiten von Information Lifecycle Management – beispielsweise die Verteilung der Daten auf unterschiedliche

Disksysteme oder partielle Backups der aktuellen Daten – lassen sich problemlos und effizient mit dieser Partitionierungsstrategie realisieren.

Im Beispiel in Abbildung 1 enthalten Hub, Link und Satellit auf der rechten Seite Bewegungsdaten, während im Hub und den zwei Satelliten links Stammdaten gespeichert sind. Die Tabellen mit Bewegungsdaten sind partitioniert nach Ladedatum.

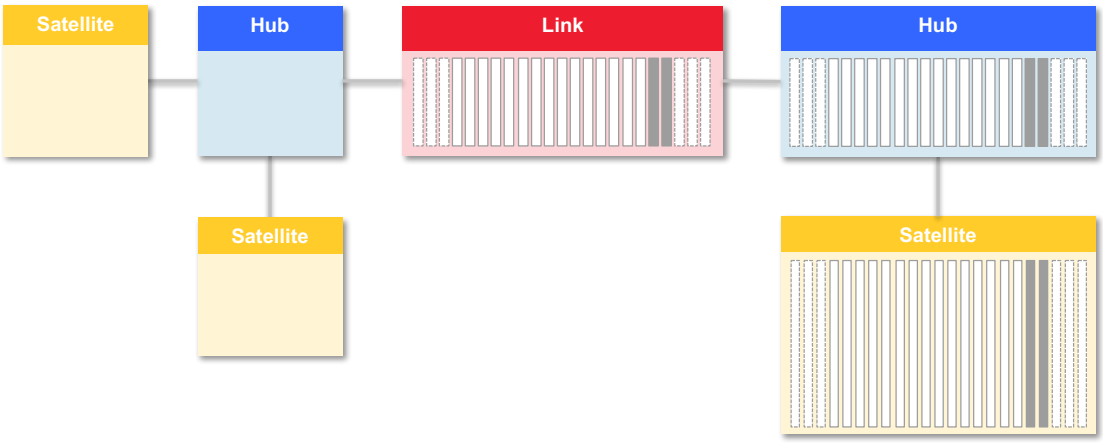

Abbildung 1: Hub, Link und Satellite für Bewegungsdaten sind RANGE-partitioniert nach Ladedatum

Die Vorteile dieser ersten Partitionierungsstrategie sind Partition Pruning und Information Lifecycle Management. Sie kommen jedoch nur bei Bewegungsdaten zum Einsatz. Für Stammdaten, die nachträglich geändert werden, kann das Ladedatum nicht als Filterkriterium für Extraktionen verwendet werden. Stattdessen wird dort üblicherweise die aktuelle Version aus jedem Satelliten abgefragt. Diese Version kann heute, aber auch vor mehreren Monaten oder Jahren eingefügt worden sein. Somit müssten immer alle Partitionen gelesen werden.

Strategie 2: Partitionierung nach Lade-Enddatum

Beim Extrahieren von Stammdaten wird üblicherweise die aktuelle Version jedes Datensatzes aus den Satelliten gelesen. Die aktuelle Version wird auch verwendet, um Unterschiede zwischen neu angelieferten und bereits im Data Vault vorhandenen Daten zu erkennen (Deltaermittlung). Deshalb ist es wichtig, die aktuellen Versionen in den Satelliten effizient ermitteln zu können. Aus diesem Grund wird oft ein Enddatum in den Satelliten gespeichert. Wird eine neue Version in einen Satelliten eingefügt, wird das Enddatum der vorherigen Version auf den Wert des Ladedatums der neuen Version gesetzt. Dadurch lässt sich für jeden Zeitpunkt einfach ermitteln, welche Versionen gültig waren. Das Enddatum der aktuellen Version ist entweder NULL oder enthält ein Datum in der Zukunft – oft der 31.12.9999.

Für Satelliten mit häufigen Änderungen kann es zweckmäßig sein, die aktuellen Versionen in einer separaten Partition zu speichern, während alle anderen (historischen) Versionen in einer anderen Partition abgelegt sind. So muss beim Extrahieren der aktuellen Version nur eine (kleine) Partition gelesen werden, nicht die gesamte Satelliten-Tabelle. Als Partitionierungsschlüssel wird das Lade-Enddatum verwendet, als Partitionierungsmethode kann entweder LIST- oder RANGE-Partitionierung verwendet werden.

Wird nun eine neue Version mit dem Lade-Enddatum 31.12.9999 eingefügt, wird sie automatisch in der aktuellen Partition gespeichert. Doch was passiert mit der vorherigen Version? Da dort das Lade-Enddatum nachträglich geändert wird, muss der Datensatz von der aktuellen in die historische Partition verschoben werden. Damit dies möglich ist, muss die partitionierte Tabelle mit der Option ENABLE ROW MOVEMENT angelegt werden.

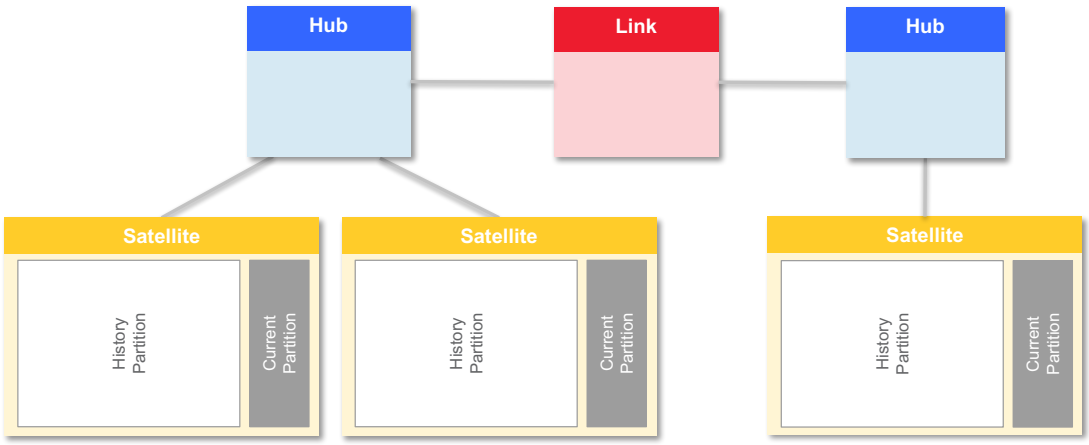

Abbildung 2: Alle Satelliten enthalten eine aktuelle und eine historische Partition

Diese Partitionierungsstrategie kann nur für Satelliten verwendet werden, und zwar nur dann, wenn diese ein Lade-Enddatum enthalten (in der „reinen Lehre“ von Data Vault ist dies nicht der Fall). Das redundante Speichern eines Enddatums ist auf jeden Fall zweckmäßig für die effiziente Ermittlung der aktuellen Versionen. Doch lohnt es sich, alle Satelliten nach diesem Attribut zu partitionieren? Der Nutzen ist vor allem dann groß, wenn in einer Satelliten-Tabelle viele Änderungen durchgeführt werden. Wenn beispielsweise pro Hub-Eintrag durchschnittlich 20 Versionen vorhanden sind, würde die aktuelle Partition nur ca. 5% der Datensätze enthalten. Das heißt, die Extraktionsjobs müssen nur einen kleinen Teil der Tabelle lesen und sind dadurch effizienter. Zu Beginn, wenn noch nicht viel historisiert ist, wird der Performancegewinn hingegen relativ klein sein. Ebenso bei Satelliten, in denen nur wenige Änderungen vorkommen.

Strategie 3: Partitionierung nach Hub-Schlüssel

Die dritte hier vorgestellte Partitionierungsstrategie verfolgt ein anderes Ziel. Hier geht es nicht darum, durch Partition Pruning die Datenmenge beim Extrahieren der Daten einzuschränken, sondern die zahlreichen Joins zu beschleunigen, die beim Lesen aus Data Vault ausgeführt werden müssen. Auch dazu ist Partitionierung ein geeignetes Mittel.

Sind zwei Tabellen auf die gleiche Weise partitioniert, so kann ein „Full Partition-wise Join“ ausgeführt werden. Das bedeutet, dass beim Join nur jeweils die Datensätze der korrespondierenden Partitionen in beiden Tabellen miteinander verglichen werden. Bei großen Datenmengen ergibt dies beträchtliche Performanceverbesserungen, insbesondere wenn die Ausführung parallel erfolgt. Voraussetzung dafür ist, dass beide Tabellen den gleichen Partitionierungsschlüssel verwenden und dass dieser als Join-Kriterium verwendet wird.

Werden in Data Vault Hubs und Satelliten nach dem Hub-Schlüssel partitioniert, so können die Tabellen mittels Full Partition-wise Join verknüpft werden. Der Hub-Schlüssel ist entweder ein Hashwert oder eine Sequenznummer, also ein nicht-sprechender Schlüsselwert. Er wird im Hub als Primärschlüssel und in den Satelliten als Fremdschlüssel verwendet. Als Partitionierungsmethode wird HASH-Partitionierung verwendet, die empfohlene Anzahl Partitionen pro Tabelle ist eine Zweierpotenz (2, 4, 8, 16, etc.). Dadurch werden die Daten gleichmäßig auf die vorhandenen Partitionen verteilt, sodass jeder (Teil-)Join zweier Partitionen etwa gleich viele Datensätze verarbeiten muss. Dies funktioniert sowohl mit numerischen Sequenznummern als auch mit Hashwerten.

Für die Link-Tabellen in Data Vault wird ein etwas anderer Ansatz verwendet. Idealerweise sollte ein Link nach jedem Hub-Schlüssel, den er referenziert, partitioniert werden. Bei Links mit Referenzen zu zwei Hubs ist dies möglich mittels Composite HASH-HASH-Partitionierung. Die Link-Tabelle wird HASH-partitioniert nach dem ersten Hub-Schlüssel. Für jede Partition werden zusätzlich HASH-Subpartitionen nach dem zweiten Hub-Schlüssel erstellt. Dadurch kann für die Joins zu beiden Hubs jeweils ein Full Partition-wise Join ausgeführt werden.

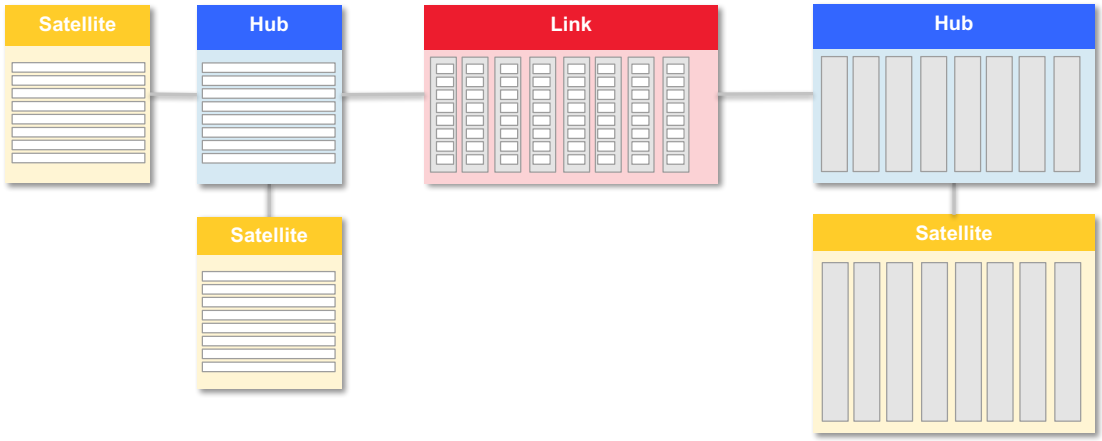

Abbildung 3: HASH-Partitionierung erlaubt Full Partition-wise Joins zwischen den Tabellen

Im Beispiel in Abbildung 3 sind die drei Tabellen links nach dem Hub-Schlüssel des linken Hubs partitioniert, mit jeweils 8 HASH-Partitionen. Für die zwei Tabellen rechts wird der Hub-Schlüssel des rechten Hubs verwendet. Die Link-Tabelle enthält 8 HASH-Partitionen, verteilt nach dem rechten Hub-Schlüssel. Jede der Partitionen umfasst 8 HASH-Subpartitionen, verteilt nach dem linken Hub-Schlüssel.

Composite HASH-HASH-Partitionierung wurde mit Oracle 12c eingeführt und ist die letzte bisher noch nicht unterstützte Kombination von Partitionen und Subpartitionen. Was es aber nach wie vor nicht gibt, ist Composite Partitioning über drei oder mehr Stufen. Für Links mit mehr als zwei Fremdschlüsseln können somit nicht alle Hub-Schlüssel zur Partitionierung verwendet werden.

Fazit

Die perfekte Partitionierungsstrategie für Data Vault gibt es leider nicht. Jede der hier vorgestellten Varianten hat Vorteile in bestimmten Situationen, aber auch Einschränkungen und Nachteile. Eine einheitliche Behandlung aller Hubs, Links und Satellites, wie dies in Data Vault idealerweise gewünscht wird, ist nicht möglich. Strategie 1 ist nur für Bewegungsdaten sinnvoll, während Strategie 2 ausschließlich für Satelliten-Tabellen möglich ist und sich nur lohnt, wenn viele Datenänderungen vorkommen. Strategie 3 schließlich kann in (fast) allen Fällen verwendet werden – mit gewissen Einschränkungen bei Links.

Ausführlichere Informationen zu den hier vorgestellten Strategien – beispielsweise ihre Auswirkungen auf die Indexierung der Tabellen – sowie Syntaxbeispiele der verschiedenen Varianten werden am Vortrag an der DOAG-Konferenz gezeigt. Sie sind außerdem in einem ausführlichen Trivadis White Paper beschrieben, das nach der DOAG-Konferenz unter https://danischnider.wordpress.com zur Verfügung stehen wird.

Kontaktadresse:

Dani Schnider
Trivadis AG
Sägereistrasse 29
CH-8152 Glattbrugg

Telefon:	+41 58 459 50 81
Fax:	+41 58 459 56 95
E-Mail	dani.schnider@trivadis.com
Internet:	www.trivadis.com