文章
姚 鑫 · 四月 15, 2021 阅读大约需0 分钟

第二章 定义和构建索引(三)

#
第二章 定义和构建索引(三)
位图索引

位图索引是一种特殊类型的索引，它使用一系列位串来表示与给定索引数据值相对应的一组ID值。

位图索引具有以下重要功能：

- 位图是高度压缩的：位图索引可以比标准索引小得多。这大大减少了磁盘和缓存的使用量。
- 位图操作针对事务处理进行了优化：与使用标准索引相比，可以在表中使用位图索引，而不会降低性能。
- 位图上的逻辑操作(counting、AND和OR)经过优化以获得高性能。
- SQL引擎包括许可以利用位图索引的特殊优化。

位图索引的创建取决于表的唯一标识字段的属性：

- 如果表的ID字段定义为具有正整数值的单个字段，则可以使用此ID字段为字段定义位图索引。此类型的表使用系统分配的唯一正整数ID，或使用IdKey定义自定义ID值，其中IdKey基类型为%Integer且MINVAL>的单个属性或类型%Numeric型且Scale=0且MINVA>0。
- 如果表的ID字段未定义为具有正整数值的单个字段(例如，子表)，则可以定义采用正整数的%BID(位图ID)字段作为代理ID字段；这允许为该表中的字段创建位图索引。

受列限制，位图索引的操作方式与标准索引相同。
索引值将被整理，可以在多个字段的组合上建立索引。

位图索引操作

位图索引的工作方式如下。
假设Person表，其中包含一些列:

Person					
RowID	Name	Age	State	Job	...
1	Smith	24	NY	Lawyer	...
2	Jones	35	NY	Doctor	...
3	Presley	48	CA	Farmer	...
4	Nixon	72	NY	Singer	...
...	...	...	...	...	...

此表中的每一行都有一个系统分配的RowID号(一组递增的整数值)。位图索引使用一组位字符串(包含1和0值的字符串)。在位串中，位的序号位置对应于索引表的RowID。对于给定值，假设State为"NY"，则有一个位串，每个位置对应一个包含"NY"的行，其他位置为0。

例如，State上的位图索引可能如所示：

StateIndex					
	Row 1	Row 2	Row 3	Row 4	...
CA	0	0	1	0	...
NY	1	1	0	1	...
WY	0	0	0	0	...
...	...	...	...	...	...

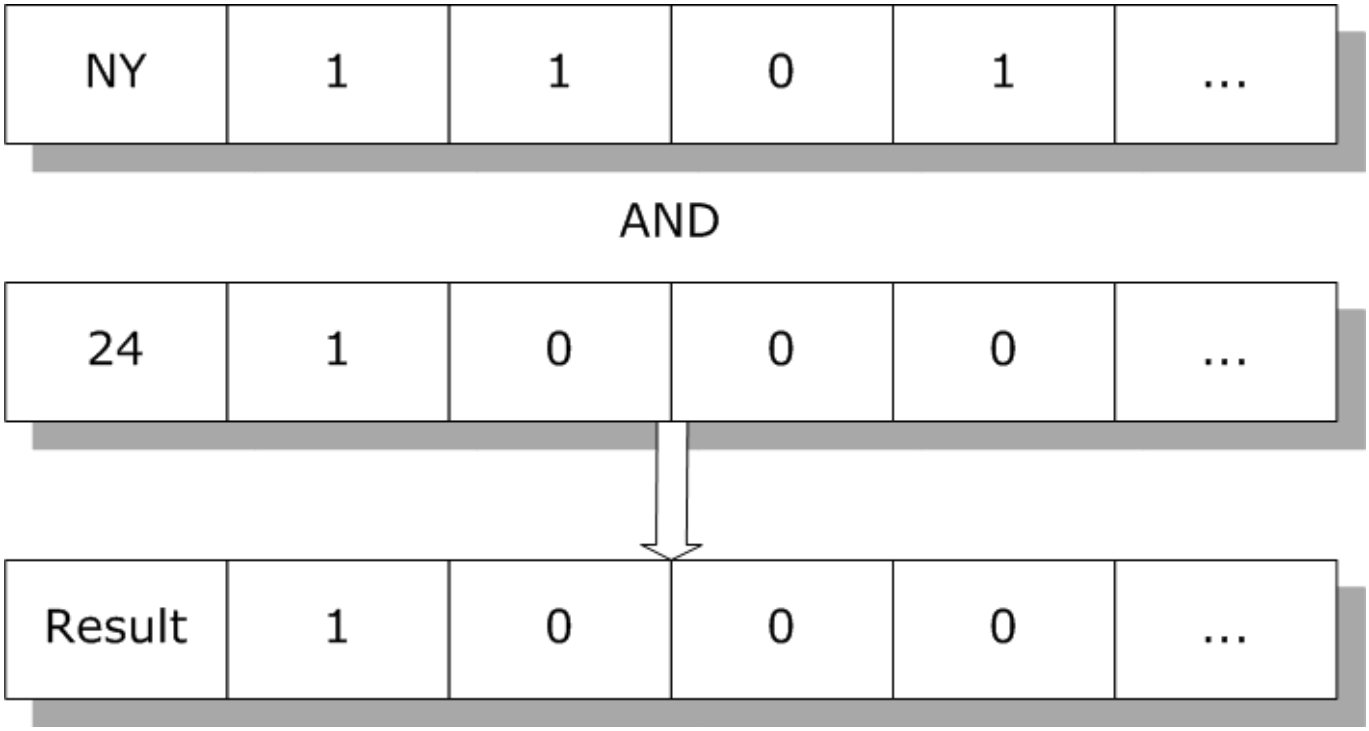
而关于Age 年龄的索引可能如下所示：

AgeIndex					
	Row 1	Row 2	Row 3	Row 4	...
24	1	0	0	0	...
35	0	1	0	0	...
48	0	0	1	0	...
...	...	...	...	...	...

注：此处显示的年龄字段可以是普通数据字段，也可以是其值可以可靠派生(Calculated 和SQLComputed)的字段。

除了将位图索引用于标准操作，SQL引擎还可以使用位图索引来使用多个索引的组合来高效地执行特殊的基集合

的操。例如，要查找居住在纽约的24岁Person的所有实例，SQL引擎只需执行Age和State索引的逻辑与：



生成的图包含匹配搜索条件的所有行的集合。SQL引擎使用它从这些行返回数据。

SQL引擎可以将位图索引用于以下操：
- 对给定表上的个条件进行AND运算。
- 对给定表上的个条件进行OR运算。
- 给定表上的RANGE范围条件。
- 对给定表上的操进行计数COUNT。

使用类定义定义IdKey位图索引

如果表的ID是值限制为唯一正整数的字段，则可以使用新建索引向导或通过与创建标准索引相同的方式编辑类定义的文本，将位图索引定义添加到类定义中。唯一的区别是需将索引类型指定为“位图”：

```
Class MyApp.SalesPerson Extends %Persistent [DdlAllowed]
{
 Property Name As %String;
 Property Region As %Integer;

 Index RegionIDX On Region [Type = bitmap];
}
```

使用类定义定义%BID位图索引

如果表的ID不限于正整数，则可以创建%BID属性用于创建位图索引定义。可以将此选项用于具有任何数据类型的ID字段的表，以及由多个字段组成的IDKEY(包括子表)。可以为以下任一数据存储类型创建%BID位图：默认结构表或%Storage.SQL表。此功能称为“任意表的位图”或BAT。

要在这样的表上启用位图索引，必须执行以下操：

1. 为类定义%BID属性字段。这可以是类的现有属性也可以是新属性它可以有任何名称。如果这是新属性

, 则必须为表中的所有现有行填充此属性字段。此%BID字段必须定义为将字段数据值限制为唯一正整数的数据类型。例如, 将MyBID属性设置为%Counter;
2. 定义新的类参数以定义哪个属性为%BID字段。此参数被命名为BIDField。此参数设置为%BID属性的SQLFieldName。例如, 参数BIDField="MyBID";
3. 定义%BID的索引。例如, MyBID上的Index BIDIdx[Type=Key, Unique];
4. 定义%BID定位器索引。
   这将%BID索引绑定到表的ID键字段。
   下面的例子是一个表的一个复合IDKey组成两个字段:

```
Index IDIdx On (IDfield1, IDfield2) [ IdKey, Unique ];
Index BIDLocIdx On (IDfield1, IDfield2, MyBID) [ Data = IdKey, Unique ];
```

此表现在支持位图索引。可以使用标准语法根据需要定义位图索引。例如: Index RegionIDX On Region [Type = bitmap];

此表现在还支持位片索引。可以使用标准语法定义位片索引。

注意:要构建或重新生成BID位图索引, 必须使用%BuildIndices()。%BID位图索引不支持%ConstructIndicesParallel()方法。

使用DDL定义位图索引

如果使用DDL语句定义表, 还可以使用以下DDL命令为ID为正整数的表格创建和删除位图索引:

```
CREATE BITMAP INDEX RegionIDX ON TABLE MyApp.SalesPerson (Region)
```

生成位图范围索引

编译包含位图索引的类时, 如果类中存在任何位图索引, 并且没有为该类定义位图范围索引, 则类编译器会生成位图范围索引。如果位图范围索引存在(无论是定义的还是生成的), 该类从主超类继承位图范围索引。为类构建索引时, 如果要求构建位图范围索引, 或者正在构建另一个位图索引并且位图范围索引结构为空, 则会构建位图范围索引。

除非存在位图索引, 否则InterSystems IRIS不会生成位图范围索引。位图范围索引定义为:type = bitmap, extent = true。这意味着从主要超类继承的位图范围索引被认为是位图索引, 并且如果在该子类中没有显式定义位图范围索引, 则将触发在子类中生成位图范围索引。

InterSystems IRIS不会基于未来的可能性在超类中生成位图范围索引。这意味着, 除非存在type=bitmap的索引, 否则InterSystems
IRIS永远不会在持久类中生成位图范围索引。假设将来的某个子类可能引入type=bitmap的索引是不够的。

注意:在将位图索引添加到生产系统上的类的过程中要特别小心(在生产系统中, 用户正在使用特定的类, 编译所述类, 然后为其构建位图索引结构)。在这样的系统上, 位图范围索引可以在编译完成索引构建进行之间的过渡期间被填充。这可能导致索引构建过程未隐式构建位图范围索引, 这导致部分完整的位图范围索引。

选择索引类型

**下面是在位图和标准索引之间选择的一般准则。
一般来说, 所有类型的键和引用都要使用标准索引:**

- Primary key
- Foreign key
- Unique keys
- Relationships

- Simple object references

否则，位图索引通常更可取(假设表使用系统分配的数字ID号)。

其他因素：
-
每个属性单独位图索引通常比多个属性位图索引具有更好的性能。这是因为SQL引擎可以使用AND和OR操作有效地组合单独的位图索引。
-
如果一个属性确实需要一起编制索引的一组属性超过10,000-20,000个不同的值(或值组合)，请考虑标准索引。但是，如果这些值的分布非常不均匀，以至于很少的值只占行的很大一部分，那么位图索引可能会更好。一般来说，目标是减少索引所需的总体大小。

位图索引的限制

所有位图索引都有以下限制：

- 不能在唯一列上定义位图索引。
- 不能在位图索引中存储数据值。
- 除非字段的SqlCategory是 INTEGER, DATE, POSIXTIME, or NUMERIC(scale=0)，否则不能在字段上定义位图索引。
- **对于包含超过100万条记录的表，当唯一值的数量超过10,000时，位图索引的效率低于标准索引。因此，对于大型表，建议避免为任何包含(或可能包含)超过10,000个唯一值的字段使用位图索引；对于任意大小的表，避免对任何可能包含超过20,000个唯一值的字段使用位图索引。这些是一般的近似值，不是确切的数字。**

必须创建一个%BID属性来支持一个表上的位图索引：
- 使用非整数字段作为唯一的ID键。
- 使用一个多字段ID键。
- 是父子关系中的子表。

可以使用$SYSTEM.SQL.Util.SetOption()方法SET status=$SYSTEM.SQL.Util.SetOption("BitmapFriendlyCheck",1,.oldval)设置系统范围的配置参数，以便编译时检查此限制，从而确定%Storage.SQL类中是否允许定义的位图索引。此检查仅适用于使用%Storage.SQL的类。默认值为0可以使用$SYSTEM.SQL.Util.GetOption("BitmapFriendlyCheck")来确定此选项的当前配置。

应用程序逻辑限制

位图结构可以由位串数组表示，其中数组的每个元素表示具有固定位数的"chunk"。因为UNDEFINED等同于一个全为0位的块，所以该数组可以是稀疏的。表示全部0位的块的数组元素根本不需要存在。因此，应用程序逻辑应该避免依赖于0值位的$BITCOUNT(str, 0)计数。

由于位串包含内部格式，因此应用程序逻辑不应依赖于位串的物理长度，也不应依赖于将具有相同位值的两个位串相等。在回滚操作之后，位串恢复到事务之前的位值。然而，由于内部格式化，回滚的位串可能不等于或不具有与事务之前的位串相同的物理长度。

维护位图索引

在易失性(执行许多插入和删除操作)中，位图索引的存储效率可能会逐渐降低。要维护位图索引，可以运行%SYS.Maint.Bitmap实用程序方法来压缩位图索引，使其恢复到最佳效率。可以使用OneClass()方法压缩单个类的位图索引。或者，可以使用Namespace()方法来压缩整个命名空间的位图索引。这些维护方法可以在带电系统上运行。

运行%SYS.Maint.Bitmap实用程序方法的结果将写入调用该方法的进程。这些结果还会写入%SYS.Maint.BitmapResults类。

位图块的SQL操

InterSystems SQL提供了以扩展来直接操作位图索引：

- %CHUNK函数
- %Bitpos函数
- %BITMAP聚合函数
- %BITMAPCHUNK聚合函数
- %SETINCHUNK谓词条件

所有这些扩展都遵循InterSystems
SQL位图表示约定，将一组正整数表示为一系列位图块，每个块最多包含64,000个整数。

这些扩展允许在查询和嵌入式SQL中更轻松、更高效地操作某些条件和筛选器。在嵌入式SQL中，它们支持位图的简单输入和输出，特别是在单个块级别。它们支持处理完整的位图，这些位图由%bitmap()和%SQL.Bitmap类处理。它们还支持非RowID值的位图处理，例如外键值、子表的父引用、关联的任一列等。

例如，要输出指定块的位图，请执行以下操作：

```
SELECT %BITMAPCHUNK(Home_Zip) FROM Sample.Person
WHERE %CHUNK(Home_Zip)=2
```

要输出整个表的所有块，请执行以下操作：

```
SELECT %CHUNK(Home_Zip),%BITMAPCHUNK(Home_Zip) FROM Sample.Person
GROUP BY %CHUNK(Home_Zip) ORDER BY 1
```

%CHUNK函数

%%CHUNK(F)返回位图索引字段f值的块分配。这被计算为f \ 64000+1.%%CHUNK(F)非位图索引字段的任何字段或值的%chunk(F)始终返回1。

%BITPOS函数

%Bitpos(F)返回分配给其区块内的位图索引字段f值的位位置。这被计算为f#64000+1。对于不是位图索引字段的任何字段或值f，%Bitpos(F)返回的值比其整数值少1。字符串的整数值为0。

%BITMAP聚合函数

聚合函数%bitmap(F)将许多f值组合到一个%SQL.Bitmap对象中，在该对象中，对于结果集中的每个值f，与适当块中的f相对应的位被设置为1。上述所有参数中的f通常是正整数字段(或表达式)，通常(但不一定)是RowID。

%BITMAPCHUNK聚合函数

聚合函数%BITMAPCHUNK(F)将字段f的许多值组合成64,000位的InterSystems SQL标准位图字符串，其中对于集合中的每个值f，位f#64000+1=%Bitpos(F)被设置为1。请 注 意，无论%chunk(F)的值是多少，都会在结果中设置该位。%BITMAPCHUNK()为空集生成NULL，并且与任何其他聚合一样，它忽略输入中的NULL值。

%SETINCHUNK谓词条件

当且仅当($BIT(BM，%Bitpos(F)=1时，条件(f%SETINCHUNK BM)为真。Bm可以是任何位图表达式字符串，例如输

入主机变量:bm, 或%BITMAPCHUNK()聚合函数的结果, 等等。请 注 意, 无论%chunk(F)的值是什么, 都会检查<
bm> 位。如果<bm> 不是位图或为NULL, 则条件返回FALSE。(F%SETINCHUNK NULL)生成FALSE(非未知)。

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E7%AB%A0-%E5%AE%9A%E4%B9%89%E5%92%8C%E6%9E%84%E5%BB%BA%E7%B4%A2%E5%BC%95%EF%BC%88%E4%B8%89%EF%BC%89